ECE 174 – Lecture Supplement – SPRING 2009

Introduction to Nonlinear Least-Squares

Kenneth Kreutz-Delgado
Electrical and Computer Engineering
Jacobs School of Engineering
University of California, San Diego

VERSION ECE174NLS-S2009v1.0

May 20, 2009

Nonlinear Mappings and Vector Derivatives. Let the function

$$h(\cdot) : \mathcal{X} = \mathbb{R}^n \to \mathcal{Y} = \mathbb{R}^m$$

be a nonlinear mapping between two finite-dimensional real vector inner-product (vector) spaces.[1] We assume that the space $\mathcal{X}$ has the standard (Cartesian) inner product weighting matrix $\Omega = I$ while the space $\mathcal{Y}$ has a general inner product weighting matrix W.

The *image* of $h(\cdot)$ is defined to be the image of the domain $\mathcal{X}$ under the action of the mapping,[2]

$$\mathrm{Im}(h) = h(\mathcal{X}) = \{ y \,|\, y = h(x),\ x \in \mathcal{X} \} \subset \mathcal{Y}\,.$$

If $h(\cdot)$ happens to be linear, then the image of h is also called the range of h.

Let $x = (x_1 \cdots x_n)^T \in \mathcal{X} = \mathbb{R}^n$. We denote the $1 \times n$ *vector derivative* operator[3] by

$$\frac{\partial}{\partial x} = \left(\frac{\partial}{\partial x_1} \ \cdots \ \frac{\partial}{\partial x_n} \right).$$

[1] Henceforth all vector spaces are assumed to be real.
[2] When $h(\cdot)$ is linear, the image is equal to *the range of h.*
[3] Also know as the *row gradient* or *covariant gradient.*

If $f(x)$ is a scalar–valued function we define $\frac{\partial}{\partial x} f(x)$ to be equal to

$$\frac{\partial}{\partial x} f(x) \triangleq \left(\frac{\partial}{\partial x_1} f(x) \ \cdots \ \frac{\partial}{\partial x_n} f(x) \right).$$

For a general m–dimensional nonlinear function $h(x)$ we define

$$\frac{\partial}{\partial x} h(x) \triangleq \begin{pmatrix} \frac{\partial}{\partial x} h_1(x) \\ \vdots \\ \frac{\partial}{\partial x} h_m(x) \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial x_1} h_1(x) & \cdots & \frac{\partial}{\partial x_n} h_1(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_1} h_n(x) & \cdots & \frac{\partial}{\partial x_n} h_n(x) \end{pmatrix}.$$

This is also known as the *Jacobian matrix*[4] and if $m = n$ the *Jacobian of the transformation*[5] $y = h(x)$ is given by the scalar–valued function $\left| \det \frac{\partial}{\partial x} h(x) \right|$.

By component–level considerations, the following useful identities can be readily shown to be true:

$$\frac{\partial}{\partial x} c^T x = c^T \quad \text{for an arbitrary vector } c \tag{1}$$

$$\frac{\partial}{\partial x} Ax = A \quad \text{for an arbitrary matrix } A \tag{2}$$

$$\frac{\partial}{\partial x} x^T \Omega x = 2x^T \Omega \quad \text{when } \Omega = \Omega^T \tag{3}$$

$$\frac{\partial}{\partial x} h(g(x)) = \frac{\partial h}{\partial g} \frac{\partial g}{\partial x} \quad \text{(Chain Rule)}. \tag{4}$$

Various "product rule" identities can also be proved, but they will not be needed here.

As an *important example* of the usefulness of these identities, consider the derivation of the vector derivative of the *nonlinear weighted least-squares loss function*[6]

$$\ell(x) = \frac{1}{2} \|e(x)\|_W^2, \quad e = y - h(x). \tag{5}$$

We have,

$$\frac{\partial}{\partial x} \ell(x) = \frac{\partial \ell}{\partial e} \frac{\partial e}{\partial x} = e^T W \frac{\partial e}{\partial x} = -e^T W \frac{\partial}{\partial x} h(x) = -\left(y - h(x)\right)^T W \frac{\partial}{\partial x} h(x). \tag{6}$$

The *gradient*[7] of a scalar–valued function $f(x)$ is defined by[8]

$$\nabla_x f(x) \triangleq \left(\frac{\partial}{\partial x} f(x) \right)^T. \tag{7}$$

---

[4] Discussed at great length in robotics textbooks.

[5] Used to obtain the probability density function of transformed random variables as discussed in courses on probability theory.

[6] The factor $\frac{1}{2}$ is added merely for convenience. Recall that W must be symmetric, positive–definite and corresponds to the use of a weighted inner–product on $\mathcal{Y} = \mathbb{R}^m$.

[7] Also known as the *contravariant derivative, contravariant gradient,* or *column derivative.*

[8] For a general $\mathcal{X}$-space metric weighting matrix Ω the gradient is more generally given by $\nabla_x f(x) = \Omega^{-1} \left(\frac{\partial}{\partial x} f(x) \right)^T$.

The gradient gives the local *direction of steepest ascent* (increase in value) in the space $\mathcal{X}$ of the scalar function $f(x)$.

Note, in particular, that the gradient of the weighted least-squares loss function is given by

$$\nabla_x \ell(x) = -\left(\frac{\partial h(x)}{\partial x}\right)^T W\,(y - h(x)) = -\left(\frac{\partial h(x)}{\partial x}\right)^* (y - h(x))\,, \tag{8}$$

where

$$\left(\frac{\partial h(x)}{\partial x}\right)^* = \left(\frac{\partial h(x)}{\partial x}\right)^T W$$

is the adjoint operator of the Jacobian matrix $\frac{\partial h(x)}{\partial x}$ with respect to the weighted inner product on $\mathcal{Y} = \mathbb{R}^m$.

Multivariate Taylor Series Expansion. Using the vector derivative notation developed above, we can denote the Taylor series expansion of a *vector–valued* function $h(x)$ about a point x_0 as

$$h(x) = h(x_0 + \Delta x) = h(x_0) + \frac{\partial h(x_0)}{\partial x}\Delta x + \text{higher order terms}\,, \tag{9}$$

where $\Delta x \triangleq x - x_0$. Note in particular that if we set

$$\Delta y \triangleq h(x) - h(x_0) = h(x_0 + \Delta x) - h(x_0)$$

we have

$$\Delta y = \frac{\partial h(x_0)}{\partial x}\,\Delta x + \text{higher order terms}\,,$$

showing that the Jacobian matrix,

$$H(x) \triangleq \frac{\partial h(x)}{\partial x}\,, \tag{10}$$

provides the *linearization* of $h(x)$,

$$\Delta y \approx H(x_0)\Delta x\,.$$

The differential statement is, of course, exact

$$dy = \frac{\partial h(x_0)}{\partial x}\,dx = H(x_0)\,dx\,.$$

The Taylor series expansion of a *scalar–valued* function $f(x)$ can be written as

$$f(x_0 + \Delta x) = f(x_0) + \frac{\partial f(x_0)}{\partial x}\Delta x + \frac{1}{2}\Delta x^T \mathcal{H}(x_0)\Delta x + \text{h.o.t.}\,, \tag{11}$$

where $\mathcal{H}(x)$ denotes the *hessian matrix* of second–order partial derivatives,

$$\mathcal{H}(x) \triangleq \frac{\partial^2 f(x)}{\partial x^2} \triangleq \frac{\partial}{\partial x}\nabla_x f(x) = \left(\frac{\partial^2 f(x)}{\partial x_i x_j}\right) .$$

Note that we can approximate the scalar–function $f(x)$ to quadratic order about a point x_0 as

$$\begin{aligned} f(x) &\approx f(x_0) + \frac{\partial f(x_0)}{\partial x}\Delta x + \frac{1}{2}\Delta x^T \mathcal{H}(x_0)\Delta x \qquad (12)\\ &= f(x_0) + \left(\nabla_x f(x_0)\right)^T \Delta x + \frac{1}{2}\Delta x^T \mathcal{H}(x_0)\Delta x\,, \qquad (13)\end{aligned}$$

where the last step follows from the gradient definition given in Equation (7).

The Nonlinear Least–Squares Problem. Suppose we want to solve the nonlinear inverse problem

$$y \approx h(x)$$

for a given nonlinear function $h(\cdot) : \mathcal{X} \to \mathcal{Y}$. We assume that $h(\cdot)$ is (locally) one–to–one[9] but generally not onto, $\mathrm{Im}(h) = h(\mathcal{X}) \neq \mathcal{Y}$.[10] The inner–product weighting matrix on the domain $\mathcal{X}$ is taken to be $\Omega = I$. On the codomain $\mathcal{Y}$ the inner–product weighting matrix is taken to be an arbitrary symmetric positive–definite $m \times m$ matrix W. Defining the discrepancy between y and $h(x)$ by the error $e(x) = y - h(x)$, a rational way to proceed is to find a value for x which minimizes the nonlinear least–squares loss function defined above in Equation (5).

It is well-known that a *necessary* condition for x_0 to be a local minimum point (or a local maximum point, or an inflection point) for a differentiable loss function $\ell(x)$ is that the gradient of the loss function evaluated at x_0 vanish

$$\nabla_x \ell(x_0) = 0\,.$$

From Equation (8) above, this condition is equivalent to the *nonlinear normal equations*

$$H^*(x_0)\left(y - h(x_0)\right) = 0 \qquad (14)$$

where $H(x) = \frac{\partial h(x)}{\partial x}$ is the Jacobian matrix of $h(x)$ and $H^*(x) = H^T(x)W$ is the adjoint operator to the matrix $H(x)$ assuming the inner–product weighting matrices $\Omega = I$ and W on the domain and codomain respectively.

The condition (14) is interpeted geometrically as indicating that the error $e(x) = y - h(x)$ must be orthogonal to the tangent hyperplane spanned by the columns of $H(x)$ at an optimal

[9] That is, $h(x)$ is one–to–one in a neighborhood of the point x.

[10] Note that "generally not onto" means that we *might* have $h(\cdot)$ onto. We want to have a generally enough development that we can handle both the case where $h(\cdot)$ is onto as well as the case where it is not onto.

point x_0. Under the assumed condition that $h(x)$ is locally one-to-one at the point x it must be the case that the jacobian matrix $H(x)$ is one-to-one (has full column rank)[11] and therefore the $n \times n$ matrices $H(x)^T H(x)$ and $H^*(x)H(x) = H^T(x)WH(x)$ are invertible.

Note that *if $h(x)$ happens to be linear, $h(x) = Ax$*, then $H(x) = A$, $H^*(x) = A^T W$ and the condition (14) becomes the standard linear least–squares normal equations,

$$A^T W A\, x_0 = A^T W y\,.$$

It is well-known that a *sufficient* condition for x_0 to be a local minimum for the loss function $\ell(x)$ is that the condition (14) holds at x_0 *and* the hessian matrix of $\ell(x)$ be positive definite at x_0

$$\mathcal{H}(x_0) = \frac{\partial}{\partial x}\nabla_x \ell(x_0) = \left(\frac{\partial^2 \ell(x_0)}{\partial x_i \partial x_j}\right) > 0\,. \tag{15}$$

It can be shown that the hessian matrix of $\ell(x)$ is given by

$$\mathcal{H}(x) = H^T(x)WH(x) + \begin{pmatrix} (h(x)-y)^T W \frac{\partial H_1(x)}{\partial x} \\ \vdots \\ (h(x)-y)^T W \frac{\partial H_n(x)}{\partial x} \end{pmatrix}\,, \tag{16}$$

where $H_i(x)$ denotes the i–th column of the Jacobian matrix $H(x)$. In the special case when $h(x)$ is *linear*, $h(x) = Ax$, we have that $H(x) = A$ and $\frac{\partial H_i(x)}{\partial x} = 0$, $i = 1, \cdots n$, yielding,

$$\mathcal{H}(x) = H^T(x)WH(x) = A^T W A\,.$$

The conditions (14) and (15) tell us when we have found a locally optimal (i.e., minimizing) solution. The question still remains as to how we actually find one. Below we discuss iterative techniques for finding an optimum. In particular we present the Newton Method and the Gauss–Newton Method and discuss how they can be interpreted as special cases of Generalized Gradient Descent, where the Generalized Gradient Descent methods are a general family of methods which generalize the standard gradient descent method of nonlinear optimization.

The Newton Method. The *Newton method* is based on reiteratively minimizing the quadratic approximation (13) of the loss function $\ell(x)$ evaluated at a current estimate, $\hat{x}_k$ of the unknown optimal solution x_0,

$$\ell(x) = \ell(\hat{x}_k + \Delta x_k) \approx \hat{\ell}^{\text{Newton}}(\Delta x_k) \triangleq \ell(\hat{x}_k) + (\nabla_x \ell(\hat{x}_k))^T \Delta x_k + \frac{1}{2}\Delta x_k^T \mathcal{H}(\hat{x}_k)\Delta x_k\,, \tag{17}$$

[11] In fact, the linearization $H(x)$ being one-to-one at the point x is a necessary and sufficient condition for the nonlinear function $h(x)$ to be locally one-to-one at the point x.

where $\Delta x_k = x - \hat{x}_k$.[12] To minimize the approximation (17) with respect to Δx_k, we solve the necessary condition

$$\frac{\partial}{\partial \Delta x_k} \hat{\ell}^{\,\text{Newton}}(\Delta x_k) = 0\,, \tag{18}$$

for an optimum "update" Δx_k. Using the vector derivative identities given above and assuming the invertibility of the matrix $\mathcal{H}(\hat{x}_k)$, the necessary condition (18) yields,

$$\Delta x_k = -\left(\mathcal{H}(\hat{x}_k)\right)^{-1} \nabla_x \ell(\hat{x}_k) = \left(\mathcal{H}(\hat{x}_k)\right)^{-1} H^T(\hat{x}_k) W \left(y - h(\hat{x}_k)\right)\,, \tag{19}$$

where the last step follows from Equations (8) and (10). Note that the hessian matrix $\frac{\partial^2}{\partial(\Delta x_k)^2}\hat{\ell}^{\,\text{Newton}}(\Delta x_k)$ of the approximation (17) is equal to $\mathcal{H}(\hat{x}_k)$.

Once we have determined the correction, we can obtain an improved estimate of the optimal solution as

$$\hat{x}_{k+1} = \hat{x}_k + \Delta x_k = \hat{x}_k + \left(\mathcal{H}(\hat{x}_k)\right)^{-1} H^T(\hat{x}_k) W \left(y - h(\hat{x}_k)\right)\,. \tag{20}$$

Note that if $\hat{x}_k$ is already an optimum solution, then $\nabla_x \ell(\hat{x}_k) = 0$ yielding $\Delta x_k = 0$.[13]

Equation (20) provides an iterative method for generating estimates of an unknown optimum solution x_0. As discussed below, the iterative algorithm (20) is usually slightly generalized by adding a positive, real–valued step–size parameter α_k to the correction term,

$$\Delta x_k = -\alpha_k \left(\mathcal{H}(\hat{x}_k)\right)^{-1} \nabla_x \ell(\hat{x}_k) = -\alpha_k \left(\mathcal{H}(\hat{x}_k)\right)^{-1} H^T(\hat{x}_k) W \left(y - h(\hat{x}_k)\right)\,, \tag{21}$$

yielding the *Newton Algorithm:*

$$\hat{x}_{k+1} = \hat{x}_k + \Delta x_k = \hat{x}_k + \alpha_k \left(\mathcal{H}(\hat{x}_k)\right)^{-1} H^T(\hat{x}_k) W \left(y - h(\hat{x}_k)\right)\,. \tag{22}$$

Often the step size is taken to be constant, $\alpha_k = \alpha \leq 1$. The simple choice $\alpha = 1$ is called the *Newton step.*

With a appropriate choice of step–sizes α_k,[14] the algorithm (22) can usually be stabilized so that the iterative estimates converge to a locally optimal solution x_0,

$$\hat{x}_\infty \triangleq \lim_{k\to\infty} \hat{x}_k = x_0\,.$$

However, the initial condition $\hat{x}_0$ used in (22) often results in different locally optimal solutions, so that a variety of initializations are usually tried. The resulting locally optimal solutions are then compared and the most optimal of them is kept as the final solution.[15]

As discussed in textbooks on optimization theory, the Newton method usually has robust and fast convergence properties. Unfortunately, the method is also usually difficult to implement as the construction of the Hessian via equation (16) and its subsequent inversion at each iteration–step is usually difficult and time consuming. For these reasons other methods, such as the Gauss–Newton method discussed next, are more often used.

[12] Which is equivalent to $x = \hat{x}_k + \Delta x_k$.

[13] Thus, if $\hat{x}_k$ is an optimum solution, there is no need to correct it.

[14] Quite often the Newton step $\alpha = 1$ will suffice.

[15] The question of if and when the true globally optimal solution has been found is generally a difficult one to answer.

The Gauss–Newton Method. Whereas the Newton method is based on reiteratively approximating the loss function about a current estimate $\hat{x}_k$ to quadratic order, the *Gauss–Newton method* is based on *reiteratively linearizing the inverse problem* about the current estimate. Note from equation (9) that an expansion of $h(x)$ about a current estimate $\hat{x}_k$ yields

$$e(x) = y - h(x) \approx y - h(\hat{x}_k) - H(\hat{x}_k)\Delta x_k = \Delta y_k - H(\hat{x}_k)\Delta x_k \,,$$

and therefore

$$\ell(x) = \ell(\hat{x}_k + \Delta x_k) = \frac{1}{2}\|e(x)\|_W^2 \approx \hat{\ell}^{\,\mathrm{Gauss}}(\Delta x_k) \triangleq \frac{1}{2}\|\Delta y_k - H(\hat{x}_k)\Delta x_k\|_W^2 \,. \tag{23}$$

Note that this loss function provides a weighted least–squares solution to the *linearized inverse problem*

$$\Delta y_k \approx H(\hat{x}_k)\Delta x_k \,. \tag{24}$$

Recalling that $h(x)$ is locally one–to–one if and only $H(x)$ is one–to–one, the loss–function approximation (23) can be minimized with respect to Δx_k yielding the unique correction

$$\begin{aligned} \Delta x_k &= \left(H^T(\hat{x}_k) W H(\hat{x}_k)\right)^{-1} H^T(\hat{x}_k) W \Delta y_k \\ &= \left(H^T(\hat{x}_k) W H(\hat{x}_k)\right)^{-1} H^T(\hat{x}_k) W \left(y - h(\hat{x}_k)\right) \\ &= -\left(H^T(\hat{x}_k) W H(\hat{x}_k)\right)^{-1} \nabla_x \ell(\hat{x}_k) \,, \end{aligned} \tag{25}$$

where the last step follows from Equations (8) and (10). The hessian matrix $\frac{\partial^2}{\partial(\Delta x_k)^2}\hat{\ell}^{\,\mathrm{Gauss}}(\Delta x_k)$ of the loss–function approximation (23) is equal to $H^T(\hat{x}_k) W H(\hat{x}_k)$.

As expected the correction (25) is equivalent to

$$\Delta x_k = H^+(\hat{x}_k)\, \Delta y_k \tag{26}$$

and provides the solution to the linearized inverse problem (24), where

$$H^+(\hat{x}_k) = \left(H^*(\hat{x}_k) H(\hat{x}_k)\right)^{-1} H^*(\hat{x}_k) = \left(H^T(\hat{x}_k) W H(\hat{x}_k)\right)^{-1} H^T(\hat{x}_k) W$$

is the pseudoinverse of $H(\hat{x}_k)$ with respect to the W–weighted inner–product on $\mathcal{Y} = \mathbb{R}^m$.

Once we have determined the correction (25)–(26), we can obtain an improved estimate of the optimal solution as

$$\hat{x}_{k+1} = \hat{x}_k + \Delta x_k = \hat{x}_k + H(\hat{x}_k)^+ \Delta y_k \tag{27}$$

$$= \hat{x}_k + \left(H^T(\hat{x}_k) W H(\hat{x}_k)\right)^{-1} H^T(\hat{x}_k) W \left(y - h(\hat{x}_k)\right) \,. \tag{28}$$

Note that if $\hat{x}_k$ is already an optimum solution, then $\nabla_x \ell(\hat{x}_k) = 0$ yielding $\Delta x_k = 0$ as a consequence of Equations (8) and (10). Equation (28) provides an iterative method for generating estimates of an unknown optimum solution x_0.

As justified below, the iterative algorithm (28) is usually slightly generalized by adding a positive, real–valued step–size parameter α_k,

$$\Delta x_k = \alpha_k \, H^+(\hat{x}_k)\Delta y_k = \alpha_k \, \left(H^T(\hat{x}_k)WH(\hat{x}_k)\right)^{-1} H^T(\hat{x}_k)W\Delta y_k \,, \tag{29}$$

yielding the *Gauss–Newton Algorithm:*

$$\begin{aligned} \hat{x}_{k+1} &= \hat{x}_k + \Delta x_k = \hat{x}_k + \alpha_k \, H(\hat{x}_k)^+ \Delta y_k & (30)\\ &= \hat{x}_k + \alpha_k \, \left(H^T(\hat{x}_k)WH(\hat{x}_k)\right)^{-1} H^T(\hat{x}_k)W\left(y - h(\hat{x}_k)\right) \,. & (31) \end{aligned}$$

Often the step size is taken to be constant, $\alpha_k = \alpha \leq 1$ with the value $\alpha = 1$ called the *Gauss–Newton step.*

With a appropriate choice of step–sizes α_k, the algorithm (31) can usually be stabilized[16] so that the iterative estimates converge to a locally optimal solution x_0,

$$\hat{x}_\infty \triangleq \lim_{k\to\infty} \hat{x}_k = x_0 \,.$$

However, the initial condition $\hat{x}_0$ used in (31) often results in different locally optimal solutions, so that a variety of initializations are usually tried. The resulting locally optimal solutions are then compared and the most optimal of them is kept as the final solution.[17]

A comparison of Equations (16), (19) and (25) indicates that the Gauss–Newton method can be viewed as an approximation to the Newton method corresponding to the assumption that the second term on the right–hand–side of (16) evaluated at $\hat{x}_k$ is negligible,

$$\begin{pmatrix} \left(h(\hat{x}_k) - y\right)^T W \frac{\partial H_1(\hat{x}_k)}{\partial x} \\ \vdots \\ \left(h(\hat{x}_k) - y\right)^T W \frac{\partial H_n(\hat{x}_k)}{\partial x} \end{pmatrix} \approx 0 \tag{32}$$

so that

$$\mathcal{H}(\hat{x}_k) \approx H^T(\hat{x}_k)WH(\hat{x}_k) = H^*(\hat{x}_k)H(\hat{x}_k) \,. \tag{33}$$

Another way to see this is to expand the definition of $\hat{\ell}^{\,\text{Gauss}}(\Delta x_k)$ given on the right–hand–side of (23) and then compare the result to the definition of $\hat{\ell}^{\,\text{Newton}}(\Delta x_k)$ given in (17).

Consideration of the condition (32) allows us to conclude that if $h(\cdot)$ is onto, or if we have other reasons to believe that the approximation error $e(\hat{x}_k) = y - h(\hat{x}_k)$ can be made small, then the difference between the Newton and Gauss–Newton algorithms will become negligible as $e(\hat{x}_k) \to 0$. Not surprisingly then, the simpler Gauss–Newton method is more often implemented than the Newton method. However the Gauss–Newton method still requires a matrix inverse at each iteration step, which can be computationally prohibitive. As discussed below, an even simpler algorithm is provided by the gradient descent algorithm.[18]

[16] Quite often the Gauss–Newton step $\alpha = 1$ will suffice.

[17] As with the Newton method, the question of if and when the true globally optimal solution has been found is generally a difficult one to answer.

[18] However the increase in simplicity usually comes at the price of a significant degradation in the rate of convergence.

Generalized Gradient Descent. Consider the nonlinear least–squares loss function (5). We have that

$$d\ell = \frac{\partial \ell(x)}{\partial x} dx \,,$$

so that if we linearize about a current estimate $\hat{x}_k$ we can claim that

$$\Delta\ell(\Delta x_k) \triangleq \ell(x) - \ell(\hat{x}_k) = \ell(\hat{x}_k + \Delta x_k) - \ell(\hat{x}_k) \approx \frac{\partial \ell(\hat{x}_k)}{\partial x} \Delta x_k = \left(\nabla_x \ell(\hat{x}_k)\right)^T \Delta x_k \qquad (34)$$

to a high degree of accuracy *provided that* $\Delta x_k = x - \hat{x}_k$ *is "small enough."* It is the requirement that this approximation be valid which leads us to introduce the step–size parameter α_k. As we show below, the step–size α_k is chosen to have a small value *precisely* in order to keep the correction Δx_k "small enough" to preserve the validity of Equation (34) and this is done in order to guarantee stability and convergence of the resulting algorithm.

Assuming the validity of Equation (34), let the correction be given by

$$\Delta x_k = -\alpha_k \, Q(\hat{x}_k) \, \nabla_x \ell(\hat{x}_k) \; = \alpha_k \, Q(\hat{x}_k) \, H^T(\hat{x}_k) W \left(y - h(\hat{x}_k)\right) , \qquad (35)$$

where $\alpha_k > 0$ and $Q(x) = Q^T(x) > 0$ is an arbitrary positive–definite, symmetric matrix–valued function of x.[19] We call the term $Q(\hat{x}_k) \, \nabla_x \ell(\hat{x}_k)$ a *generalized gradient* and the resulting correction (35) a *generalized gradient–descent correction.*

Recalling that the gradient $\nabla_x \ell(\hat{x}_k)$ defines the direction of *steepest ascent* of the function $\ell(x)$ at the point $x = \hat{x}_k$, it is evident that the negative gradient, $-\nabla_x \ell(\hat{x}_k)$, gives the direction of *steepest descent* at the point $\hat{x}_k$. For the case $Q(x) = I$, the correction is directly along the direction of steepest descent and the resulting algorithm is known as a *gradient descent algorithm.* For the case of a general positive–definite matrix–valued function $Q(x) \neq I$, we can potentially improve the descent direction by using the negative of the *generalized* gradient as a descent direction, and the resulting algorithm for an arbitrary $Q(x)$ is called a *generalized gradient descent algorithm.* Thus the generalized gradient descent algorithms include the standard gradient descent algorithm as a special case.

With the correction (35) we obtain the *Generalized Gradient Descent Algorithm:*

$$\hat{x}_{k+1} = \hat{x}_k + \Delta x_k = \hat{x}_k + \alpha_k \, Q(\hat{x}_k) \, H^T(\hat{x}_k) W \left(y - h(\hat{x}_k)\right) . \qquad (36)$$

Note the following important special cases:

$$Q(x) \;=\; I \qquad \text{Gradient Descent Method} \qquad (37)$$

$$Q(x) \;=\; \left(H(x)^T W H(x)\right)^{-1} \qquad \text{Gauss–Newton Method} \qquad (38)$$

$$Q(x) \;=\; \left(\mathcal{H}(x)\right)^{-1} \qquad \text{Newton Method} \qquad (39)$$

[19] Note that if the approximation (34) is not valid for this particular choice of Δx_k, we can reduce the size of α_k until it is.

In particular, note that

$$\hat{x}_{k+1} = \hat{x}_k + \Delta x_k = \hat{x}_k + \alpha_k \, H^*(\hat{x}_k) \, (y - h(\hat{x}_k)) \quad \text{Gradient Descent Method} \tag{40}$$

$$\hat{x}_{k+1} = \hat{x}_k + \Delta x_k = \hat{x}_k + \alpha_k \, H^+(\hat{x}_k) \, (y - h(\hat{x}_k)) \quad \text{Gauss–Newton Method} \tag{41}$$

The increase in complexity involved in moving from a gradient descent algorithm to a Gauss–Newton algorithm (or a Newton algorithm) is quite significant, however this move also generally results in a significant improvement in performance. Although the classical Gradient Descent method is the easiest of the methods to implement, it tends to have very slow convergence compared to the Gauss–Newton and Newton methods.

What is the importance of the step–size parameter α_k? From Equations (35) and (36), and *assuming the validity of* (34), we obtain

$$\Delta\ell(\Delta x_k) = \ell(\hat{x}_{k+1}) - \ell(\hat{x}_k) \approx -\alpha_k \left(\nabla_x \ell(\hat{x}_k)\right)^T Q(\hat{x}_k) \nabla_x \ell(\hat{x}_k) = -\alpha_k \|\nabla_x \ell(\hat{x}_k)\|^2_{Q(\hat{x}_k)} < 0 \,,$$

whenever $\nabla_x \ell(\hat{x}_k) \neq 0$,[20], so that we expect that the algorithm (36) will result in a strictly decreasing loss function,

$$\ell(\hat{x}_{k+1}) < \ell(\hat{x}_k) \,.$$

This will not be the case if the step–size α_k is chosen too large so that the key approximation (34) is invalid. Thus we see that a proper choice of step–size is key to ensuring good performance and, indeed, the step–size issue is usually discussed at great length in course on mathematical optimization.

Constrained Optimization: The Method of Lagrange Multipliers. Assume, for simplicity, that only the standard inner product holds on all spaces of interest. If there are p independent equality constraint conditions,[21] they can be represented as

$$g(x) \equiv 0 \,, \tag{42}$$

where $g(\cdot) : \mathbb{R}^n \to \mathbb{R}^p$. The constraint condition (42) defines a locally p–dimensional smooth surface in $\mathcal{X} = \mathbb{R}^n$ which we call the *constraint manifold.* Because the constraint condition (42) holds identically, it must be the case that admissible differential variations satisfy the condition

$$\frac{\partial g(x)}{\partial x} \, dx = 0 \,.$$

Thus, admissible variations dx must be in the nullspace of $\frac{\partial g(x)}{\partial x}$, $dx \in \mathcal{N}\left(\frac{\partial g(x)}{\partial x}\right)$.

[20] If $\nabla_x \ell(\hat{x}_k) = 0$ then $\hat{x}_k$ satisfies the necessary condition for a local optimum and the algorithm has converge to a solution.

[21] The p constraints $g(x) = 0$ are linearly independent at the point x if and only if the Jacobian matrix of $g(x)$, $\frac{\partial g(x)}{\partial x}$, has full row–rank at the point x.

Let $\ell(x)$ be a general loss function.[22] As discussed in class, a necessary condition to have minimized the loss function $\ell(x)$ on the constraint manifold is that the projection of its gradient $\nabla_x \ell(x)$ into the nullspace of $\frac{\partial g(x)}{\partial x}$ is zero. Equivalently, when $\nabla_x \ell(x) \perp \mathcal{N}(\frac{\partial g(x)}{\partial x})$. Thus

$$\nabla_x \ell(x) \in \mathcal{N}\left(\frac{\partial g(x)}{\partial x}\right)^{\perp} = \mathcal{R}\left(\frac{\partial g(x)}{\partial x}^T\right),$$

and therefore

$$\left(\frac{\partial}{\partial x}\ell(x)\right)^T = \nabla_x \ell(x) = -\frac{\partial g(x)}{\partial x}^T \lambda \tag{43}$$

for some vector $\lambda \in \mathbb{R}^p$.[23] Equation (43) is a necessary condition for the point x to minimize $\ell(x)$ on the constraint manifold.

The necessary optimality condition (43) can be rewritten as the equivalent necessary condition,

$$0 = \frac{\partial}{\partial x}\left(\ell(x) + \lambda^T g(x)\right) = \frac{\partial}{\partial x}\mathcal{L}(x;\lambda), \tag{44}$$

where

$$\mathcal{L}(x;\lambda) = \ell(x) + \lambda^T g(x) \tag{45}$$

is the *lagrangian function.* Notice that the stationarity condition

$$\frac{\partial}{\partial \lambda}\mathcal{L}(x;\lambda) = 0$$

retrieves the equality constraint condition (42), while the stationarity condition

$$\frac{\partial}{\partial x}\mathcal{L}(x;\lambda) = 0$$

retrieves the necessary condition (43).

The two necessary conditions (42) and (43) are together to be solved for the optimal point x. *Thus, the lagrangian (45) provides a complete encoding of the information needed to solve for a solution to the necessary conditions (42) and (43).*

As an example, consider the constrained quadratic optimization problem

$$\min_x \frac{1}{2}\|x\|_\Omega^2 \quad \text{subject to } Ax = y$$

where the $m \times n$ matrix A is onto. The constraint condition can be written as

$$g(x) = y - Ax = 0,$$

[22] Not limited only to the least squares loss function described above.

[23] The elements of the vector λ are called *lagrange multipliers.*

and the lagrangian as

$$\mathcal{L} = \frac{1}{2}\|x\|_\Omega^2 + \lambda^T (y - Ax) \ .$$

We have that

$$0 = \frac{\partial}{\partial x}\mathcal{L} = x^T\Omega - \lambda^T A \,,$$

which can be solved to yield the condition[24]

$$\hat{x} = \Omega^{-1}A^T\lambda \,.$$

Applying the condition $Ax = y$ to this equation allows us to solve for the lagrange multipliers as

$$\lambda = \left(A\Omega^{-1}A^T\right)^{-1} y$$

so that finally

$$\hat{x} = \Omega^{-1}A^T\left(A\Omega^{-1}A^T\right)^{-1} y = A^+ y \,,$$

which is equivalent to the solution we found earlier using geometric methods.

[24]Note that this condition is equivalent to the geometric condition that $\hat{x} = A^*\lambda$, i.e. the condition that $\hat{x} \in \mathcal{R}(A^*)$. This justifies the claim made in lecture earlier in the quarter that the vector λ can be interpreted as a vector of lagrange multipliers.