Vector Derivatives, Gradients, and Generalized Gradient Descent Algorithms

ECE 275A – Statistical Parameter Estimation

Ken Kreutz-Delgado
ECE Department, UC San Diego

November 1, 2013

Objective Functions $\boldsymbol{\ell(x)}$

Let $\ell(x)$ be a real-valued function (aka *functional*) of an n-dimensional real vector $x \in \mathcal{X} = \mathbb{R}^n$

$$\ell(\cdot) : \mathcal{X} = \mathbb{R}^n \to \mathcal{V} = \mathbb{R}$$

where $\mathcal{X}$ is an n-dimensional real Hilbert space with metric matrix $\Omega = \Omega^T > 0$. Note that henceforth vectors in $\mathcal{X}$ are represented as column vectors in $\mathbb{R}^n$. Because here the *value space* $\mathcal{Y} = \mathbb{R}$ is one-dimensional, wlog we can take $\mathcal{V}$ to be Cartesian (because all (necessarily positive scalar) metric weightings yield inner products and norms that are equivalent up to an overall positive scaling).

We will call $\ell(x)$ an **objective function.** If $\ell(x)$ is a **cost, loss, or penalty function,** then we assume that it is bounded from below and we attempt to minimize it wrt x. If $\ell(x)$ is a **profit, gain, or reward function,** then we assume that it is bounded from above and we attempt to maximize it wrt x.

For example, suppose we wish to match a model pdf $p_x(y)$ to a true, but unknown, density $p_{x_0}(y)$ for an observed random vector, where **we assume $\mathbf{p_x(y) \le p_{x_0}(y), \forall x}$.** We can then use a penalty function of x to be given by a measure of (non-averaged or **instantaneous**) **divergence or discrepancy $\mathbf{D_I(x_0 \| x)}$** of the model pdf $p_x(y)$ from the true pdf $p_{x_0}(y)$ defined by

$$D_I(x_0 \| x) \triangleq \log\left(\frac{p_{x_0}(y)}{p_x(y)}\right) = \log p_{x_0}(y) - \log p_x(y)$$

Instantaneous Divergence & Negative Log-likelihood

Note that minimizing the instantaneous divergence $D_I(x_0\|x)$ is equivalent to maximizing the log-likelihood $p_x(y)$ or minimizing the negative log-likelihood

$$\ell(x) = -\log p_x(y)$$

A special case arises from use of the nonlinear Gaussian additive noise model with known noise covariance C and known mean function $h(\cdot)$

$$y = h(x) + n, \quad n \sim \mathrm{N}(0, C) \qquad \Longleftrightarrow \qquad y \sim \mathrm{N}(h(x), C)$$

which yields the nonlinear weighted least-squares problem

$$\ell(x) = -\log p_x(y) \doteq \|y - h(x)\|_W^2, \qquad W = C^{-1}$$

Further setting $h(x) = Ax$ is the linear weighted least-squares problem we have already discussed

$$\ell(x) = -\log p_x(y) \doteq \|y - Ax\|_W^2, \qquad W = C^{-1}$$

The symbol "$\doteq$" denotes that fact that we are ignoring additive terms and multiplicative factors which are irrelevant for the purposes of obtaining a extremum (here, a minimum) of a loss function. Of course we *cannot* ignore these terms if we are interested in the actual optimal value of the loss function itself.

Stationary Points and the Vector Partial Derivative

Henceforth let the real scalar function $\ell(x)$ be twice partial differentiable with respect to all components of $x \in \mathbb{R}^n$. A **necessary condition for x be be a local extremum (maximum or minimum) of $\ell(x)$** is that

$$\frac{\partial}{\partial x}\ell(x) \triangleq \underbrace{\begin{pmatrix} \frac{\partial \ell(x)}{\partial x_1} & \cdots & \frac{\partial \ell(x)}{\partial x_n} \end{pmatrix}}_{1 \times n} = 0$$

where the **vector partial derivative operator**

$$\boxed{\frac{\partial}{\partial \mathbf{x}} \triangleq \begin{pmatrix} \frac{\partial}{\partial \mathbf{x_1}} & \cdots & \frac{\partial}{\partial \mathbf{x_n}} \end{pmatrix}}$$

is defined as a *row operator.* (See the extensive discussion in the Lecture Supplement on Real Vector Derivatives.)

A vector x_0 for which $\frac{\partial}{\partial x}\ell(x_0) = 0$ is known as a **stationary point** of $\ell(x)$. Stationary points are points at which $\ell(x)$ has a local maximum, minimum, or inflection.

Sufficient conditions for a stationary point to be a local extremum require that we develop a theory of vector differentiation that will allow us to clearly and succinctly discuss second-order derivative properties of objective functions.

Derivative of a Vector-Valued Function – The Jacobian

Let $f(x) \in \mathbb{R}^m$ have elements $f_i(x)$, $i = 1, \cdots, m$, which are all differentiable with respect to the components of $x \in \mathbb{R}^n$.

We define the vector partial derivative of the vector function $f(x)$ as

$$\mathbf{J_f(x)} \triangleq \frac{\partial}{\partial \mathbf{x}} \mathbf{f(x)} \triangleq \begin{pmatrix} \frac{\partial}{\partial \mathbf{x}} \mathbf{f_1(x)} \\ \vdots \\ \frac{\partial}{\partial \mathbf{x}} \mathbf{f_m(x)} \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{\partial \mathbf{f_1(x)}}{\partial \mathbf{x_1}} & \cdots & \frac{\partial \mathbf{f_1(x)}}{\partial \mathbf{x_n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{f_m(x)}}{\partial \mathbf{x_1}} & \cdots & \frac{\partial \mathbf{f_m(x)}}{\partial \mathbf{x_n}} \end{pmatrix}}_{\mathbf{m \times n}}$$

The matrix $J_f(x) = \frac{\partial}{\partial x} f(x)$ is known as the **Jacobian matrix (or operator) of the mapping f(x).** *It is the linearization of the nonlinear mapping $f(x)$ at the point x.* Often we write $y = f(x)$ and the corresponding Jacobian as $J_y(x)$.

If $m = n$ and $f(x)$ is invertible, then $y = f(x)$ can be viewed as a change of variables, in which case $\mathbf{det\, J_y(x)}$ is the **Jacobian of the transformation**. The Jacobian, $\det J_y(x)$, plays a fundamental role in the change of variable formulae of pdf's and multivariate integrals.

The Jacobian – Cont.

- A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is **locally one-to-one** in an open neighborhood of x if and only if its Jacobian (linearization) $J_f(\xi) = \frac{\partial}{\partial x} f(\xi)$ is a one-to-one matrix for all points $\xi \in \mathbb{R}^n$ in the open neighborhood of x.
- A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is **locally onto** an open neighborhood of $y = f(x)$ if and only if its Jacobian (linearization) $J_f(\xi) = \frac{\partial}{\partial x} f(\xi)$ is an onto matrix for all points $\xi \in \mathbb{R}^n$ in the corresponding neighborhood of x.
- A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is **locally invertible** in an open neighborhood of x if and only if it is locally one-to-one and onto in the open neighborhood of x which is true if and only if its Jacobian (linearization) $J_f(\xi) = \frac{\partial}{\partial x} f(\xi)$ is a one-to-one and onto (and hence invertible) matrix for all points $\xi \in \mathbb{R}^n$ in the open neighborhood of x. This is known as the **Inverse Function Theorem.**

Vector Derivative Identities

$$
\begin{aligned}
\frac{\partial\, c^T x}{\partial x} &= c^T \quad \text{for an arbitrary vector } c \\
\frac{\partial\, Ax}{\partial x} &= A \quad \text{for an arbitrary matrix } A \\
\frac{\partial\, g^T(x)h(x)}{\partial x} &= g^T(x)\frac{\partial\, h(x)}{\partial x} + h^T(x)\frac{\partial\, g(x)}{\partial x}, \quad g(x)^T h(x) \text{ scalar} \\
\frac{\partial x^T A x}{\partial x} &= x^T A + x^T A^T \quad \text{for an arbitrary matrix } A \\
\frac{\partial x^T \Omega x}{\partial x} &= 2x^T \Omega \quad \text{when } \Omega = \Omega^T \\
\frac{\partial h(g(x))}{\partial x} &= \frac{\partial h}{\partial g}\,\frac{\partial g}{\partial x} \qquad\qquad \text{(Chain Rule)}
\end{aligned}
$$

Note that the last identity) is a statement about Jacobians and can be restated in an illuminating manner as

$$\boxed{\mathbf{J_{h\circ g} = J_h\, J_g}} \tag{1}$$

I.e., **“the linearization of a composition is the composition of the linearizations.”**

Application to Linear Gaussian Model

Stationary points of

$$\ell(x) = -\log p_x(y) \doteq \|e(x)\|_W^2 \quad \text{where} \quad e(x) = y - Ax \quad \text{and} \quad W = C^{-1}$$

satisfy

$$0 = \frac{\partial \ell}{\partial x} = \frac{\partial \ell}{\partial e}\frac{\partial e}{\partial x} = (2e^T W)(-A) = -2(y - Ax)^T WA$$

or

$$A^T W(y - Ax) = 0 \quad \Longleftrightarrow \quad e(x) = y - Ax \in \mathcal{N}(A^*) \quad \text{with} \quad A^* = \Omega^{-1} A^T W$$

Therefore stationary points satisfy the **Normal Equation**

$$A^T WAx = A^T Wy \quad \Longleftrightarrow \quad A^* Ax = A^* y$$

The Hessian of an Objective Function

The **Hessian,** or matrix of second partial derivatives of $\ell(x)$, is defined by

$$\mathcal{H}(\mathbf{x}) = \frac{\partial^2 \ell(\mathbf{x})}{\partial^2 \mathbf{x}} \triangleq \frac{\partial}{\partial \mathbf{x}} \left(\frac{\partial}{\partial \mathbf{x}} \ell(\mathbf{x}) \right)^{\mathbf{T}} = \underbrace{\begin{pmatrix} \frac{\partial \ell(\mathbf{x})}{\partial \mathbf{x}_1 \partial \mathbf{x}_1} & \cdots & \frac{\partial \ell(\mathbf{x})}{\partial \mathbf{x}_n \partial \mathbf{x}_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial \ell(\mathbf{x})}{\partial \mathbf{x}_1 \partial \mathbf{x}_n} & \cdots & \frac{\partial \ell(\mathbf{x})}{\partial \mathbf{x}_n \partial \mathbf{x}_n} \end{pmatrix}}_{\mathbf{n} \times \mathbf{n}}$$

As a consequence of the fact that

$$\frac{\partial \ell(x)}{\partial x_i \partial x_j} = \frac{\partial \ell(x)}{\partial x_j \partial x_i}$$

the Hessian is obviously symmetric

$$\mathcal{H}(x) = \mathcal{H}^T(x)$$

Vector Taylor Series Expansion

Taylor series expansion of a scalar-valued function $\ell(\mathbf{x})$ about a point x_0 to second-order in $\Delta x = x - x_0$:

$$\boldsymbol{\ell(\mathbf{x_0} + \Delta\mathbf{x}) = \ell(\mathbf{x_0}) + \frac{\partial \ell(\mathbf{x_0})}{\partial \mathbf{x}} \Delta\mathbf{x} + \frac{1}{2} \Delta\mathbf{x}^T \mathcal{H}(\mathbf{x_0}) \Delta\mathbf{x} + \text{h.o.t.}}$$

where $\mathcal{H}$ is the Hessian of $\ell(x)$.

Taylor series expansion of a vector-valued function $\mathbf{h(x)}$ about a point x_0 to first-order in $\Delta x = x - x_0$:

$$\boldsymbol{\mathbf{h(x)} = \mathbf{h}(\mathbf{x_0} + \Delta\mathbf{x}) = \mathbf{h}(\mathbf{x_0}) + \frac{\partial \mathbf{h}(\mathbf{x_0})}{\partial \mathbf{x}} \Delta\mathbf{x} + \text{h.o.t.}}$$

To obtain notationally uncluttered expressions for higher order expansions, one switches to the use of tensor notation.

Sufficient Condition for an Extremum

Let x_0 be a stationary point of $\ell(x)$, $\frac{\partial \ell(x_0)}{\partial x} = 0$. Then from the second-order expansion of $\ell(x)$ about x_0 we have

$$\Delta\ell(x) = \ell(x) - \ell(x_0) \approx \frac{1}{2}(x - x_0)^T \mathcal{H}(x_0)(x - x_0) = \frac{1}{2}\Delta x^T \mathcal{H}(x_0)\Delta x$$

assuming that $\Delta x = x - x_0$ is small enough in norm so that higher order terms in the expansion can be neglected. (That is, we consider only *local excursions* away from x_0.)

We see that If the Hessian is *positive definite,* then *all* *local excursions* of x away from x_0 *increase* the value of $\ell(x)$ and thus

Suff. Cond. for Stationary Point $\mathbf{x_0}$ to be a Unique Local Min: $\boldsymbol{\mathcal{H}(x_0) > 0}$

Contrawise, if the Hessian is *negative definite,* then *all* *local excursions* of x away from x_0 *decrease* the value of $\ell(x)$ and thus

Suff. Cond. for Stationary Point $\mathbf{x_0}$ to be a Unique Local Max: $\boldsymbol{\mathcal{H}(x_0) < 0}$

If the Hessian $\mathcal{H}(x_0)$ is full rank and indefinite at a stationary point x_0, then x_0 is a saddle point. If $\mathcal{H}(x_0) \geq 0$ then x_0 is a non-unique local minimum. If $\mathcal{H}(x_0) \leq 0$ then x_0 is a non-unique local maximum.

Application to Linear Gaussian Model – Cont.

Note that the scalar loss function

$$\ell(x) \doteq \|y - Ax\|_w^2 = (y - Ax)^T W (y - Ax) = y^T W y - 2y^T W A x + x^T A^T W A x$$

has an *exact* quadratic expansion. Thus the arguments given in the previous slide hold for arbitrarily large (global) excursions away from a stationary point x_0. In particular, if $\mathcal{H}$ is positive definite, the stationary point x_0 must be a unique *global minimum.*

Having shown that

$$\frac{\partial \ell(x)}{\partial x} = -2(y - Ax)^T W A = 2x^T A^T W A - 2y^T W A$$

we determine the Hessian to be

$$\mathcal{H}(x) = \frac{\partial}{\partial x}\left(\frac{\partial \ell}{\partial x}\right)^T = 2\,A^T W A \geq 0$$

Therefore stationary points (which, as we have seen, necessarily satisfy the normal equation) are global minima of $\ell(x)$. Furthermore, if A is one-to-one (has full column rank), then $\mathcal{H}(x) = 2\,A^T W A > 0$ and there is only one unique stationary point (i.e., weighted least-squares solution) which minimizes $\ell(x)$. Of course this could not be otherwise, as we know from our previous analysis of the weighted least-squares problem using Hilbert space theory.

The Gradient of an Objective Function $\ell(\mathbf{x})$

Note that $d\ell(x) = \frac{\partial \ell(x)}{\partial x} dx$ or, setting $\dot{\ell} = d\ell/dt$ and $v = \dot{x} = dx/dt$,

$$\dot{\ell}(v;x) = \frac{\partial \ell(x)}{\partial x} v = \frac{\partial \ell(x)}{\partial x} \Omega_x^{-1} \Omega_x v = \left[\Omega_x^{-1} \left(\frac{\partial \ell(x)}{\partial x} \right)^T \right]^T \Omega_x v = \langle \nabla_x \ell(x), v \rangle$$

with the **Gradient of $\ell(\mathbf{x})$ defined by**

$$\boxed{\boldsymbol{\nabla_x \ell(x) \triangleq \Omega_x^{-1} \left(\frac{\partial \ell(x)}{\partial x} \right)^T = \Omega_x^{-1} \begin{pmatrix} \frac{\partial \ell(x)}{\partial x_1} \\ \vdots \\ \frac{\partial \ell(x)}{\partial x_n} \end{pmatrix}}}$$

with Ω_x a local Riemannian metric. **Note that $\dot{\ell}(\mathbf{v};\mathbf{x}) = \langle \nabla_\mathbf{x} \ell(\mathbf{x}), \mathbf{v} \rangle$ is a linear functional of the velocity vector $\mathbf{v} = \dot{\mathbf{x}}$.**

In the vector space structure we have seen to date Ω_x is independent of x, $\Omega_x = \Omega$, and represents the metric of the (globally) defined metric space containing x. Furthermore, if $\Omega = I$, then the space is a (globally) Cartesian vector space. When considering spaces of smoothly parameterized “regular family” probability density functions, a natural Riemannian (local, non-Cartesian) metric is provided by the Fisher Information Matrix to be discussed later in this course.

The Gradient as the Direction of Steepest Ascent

What unit velocities, $\|v\| = 1$, in the domain space $\mathcal{X}$ result in the fastest rate of change of $\ell(x)$ at the point x as measured by $|\dot{\ell}(v; x)|$?

Equivalently, What unit velocity *directions* v in the domain space $\mathcal{X}$ result in the fastest rate of change of $\ell(x)$ as measured by $|\dot{\ell}(v; x)|$?

From the Cauchy-Schwarz inequality, we have

$$|\dot{\ell}(v; x)| = |\langle \nabla_x \ell(x), v \rangle| \le \|\nabla_x \ell(x)\| \, \|v\| = \|\nabla_x \ell(x)\|$$

or

$$-\|\nabla_x \ell(x)\| \le \dot{\ell}(v; x) \le \|\nabla_x \ell(x)\|$$

Note that

$$v = c\, \nabla_x \ell(x) \quad \text{with} \quad c = \|\nabla_x \ell(x)\|^{-1} \qquad \Longleftrightarrow \qquad \dot{\ell}(v) = \|\nabla_x \ell(x)\|$$

$$\boldsymbol{\nabla_x \ell(x) = \text{direction of steepest ascent.}}$$

$$v = -c\, \nabla_x \ell(x) \quad \text{with} \quad c = \|\nabla_x \ell(x)\|^{-1} \qquad \Longleftrightarrow \qquad \dot{\ell}(v) = -\|\nabla_x \ell(x)\|$$

$$\boldsymbol{-\nabla_x \ell(x) = \text{direction of steepest descent.}}$$

The Cartesian or "Standard" Gradient

In a Cartesian vector space the gradient of a cost function $\ell(x)$ corresponds to taking $\Omega = I$, in which case we have the **Cartesian gradient**

$$\nabla_x^c \ell(x) = \left(\frac{\partial \ell(x)}{\partial x} \right)^T = \begin{pmatrix} \frac{\partial \ell(x)}{\partial x_1} \\ \vdots \\ \frac{\partial \ell(x)}{\partial x_n} \end{pmatrix}$$

Often one naively assumes that the gradient takes this form even if it is not evident that the space is, in fact, Cartesian. In this case one might more accurately refer to the gradient shown as the **standard gradient**. Because the Cartesian gradient is the standard form assumed in many applications, it is common to just refer to it as *the* gradient, even if it is *not* the the correct, true gradient. (Assuming that we agree that the true gradient must give the direction of steepest descent and therefore depends on the metric Ω_x.)

This is the terminology adhered to by Amari and his colleagues, who then refer to the true Riemannian metric-dependent gradient as the **natural gradient.** As mentioned earlier, when considering spaces of smoothly parameterized "regular family" probability density functions, a natural Riemannian metric is provided by the Fisher Information Matrix. Amari is one of the first researchers to consider parametric estimation from this **Information Geometry** perspective. He has argued that the use of the natural (true) gradient can significantly improve the performance of statistical learning algorithms.

Continuous Dynamic Minimization of $\boldsymbol{\ell}(\mathbf{x})$

Let $\ell(\cdot) : \mathcal{X} = \mathbb{R}^n \to \mathbb{R}$ be twice differentiable with respect to $x \in \mathbb{R}^n$ and bounded from below by 0, $\ell(x) \geq 0$ for all x. Recalling that $\dot{\boldsymbol{\ell}}(\mathbf{x}) = \frac{\partial \boldsymbol{\ell}(\mathbf{x})}{\partial \mathbf{x}} \dot{\mathbf{x}}$, set

$$\boxed{\dot{\mathbf{x}} = -\mathbf{Q}(\mathbf{x}) \nabla_{\mathbf{x}}^{\mathbf{c}} \boldsymbol{\ell}(\mathbf{x}) \quad \textbf{with} \quad \nabla_{\mathbf{x}}^{\mathbf{c}} \boldsymbol{\ell}(\mathbf{x}) = \left(\frac{\partial \boldsymbol{\ell}(\mathbf{x})}{\partial \mathbf{x}} \right)^{\mathbf{T}} = \textbf{Cartesian Gradient of } \boldsymbol{\ell}(\mathbf{x})}$$

with $Q(x) = Q(x)^T > 0$ for all x. This yields

$$\dot{\boldsymbol{\ell}}(\mathbf{x}) = -\| \nabla_{\mathbf{x}}^{\mathbf{c}} \boldsymbol{\ell}(\mathbf{x}) \|_{\mathbf{Q}(\mathbf{x})}^{2} \leq \mathbf{0}$$

with

$$\dot{\boldsymbol{\ell}}(\mathbf{x}) = \mathbf{0} \quad \text{if and only if} \quad \nabla_{\mathbf{x}}^{\mathbf{c}} \boldsymbol{\ell}(\mathbf{x}) = \mathbf{0}$$

Thus continuously evolving the value of x according to $\dot{x} = -Q(x) \nabla_x^c \ell(x)$ ensures that the cost $\ell(x)$ dynamically decreases in value until a stationary point turns off learning.

Generalized Gradient Descent Algorithm

A family of algorithms for *discrete-step* dynamic minimization of $\ell(x)$ can be developed as follows.

Setting $x_k = x(t_k)$, approximate $\dot{x}_k = \dot{x}(t_k)$ by the first-order forward difference

$$\dot{x}_k \approx \frac{x_{k+1} - x_k}{t_{k+1} - t_k}$$

This yields

$$x_{k+1} \approx x_k + \alpha_k \dot{x}_k \quad \text{with} \quad \alpha_k = t_{k+1} - t_k$$

which suggests the **Generalized Gradient Descent Algorithm**

$$\boxed{\hat{\mathbf{x}}_{\mathbf{k+1}} = \hat{\mathbf{x}}_{\mathbf{k}} - \boldsymbol{\alpha}_{\mathbf{k}} \mathbf{Q}(\hat{\mathbf{x}}_{\mathbf{k}}) \nabla_{\mathbf{x}}^{\mathbf{c}} \boldsymbol{\ell}(\hat{\mathbf{x}}_{\mathbf{k}})}$$

A vast body of literature in Mathematical Optimization Theory (aka Mathematical Programming) exists which gives conditions on step size α_k to guarantee that a generalized gradient descent algorithm will converge to a stationary value of $\ell(x)$ for various choices of $Q(x) = Q(x)^T > 0$.

Note that a Generalized Gradient Algorithm turns off once the sequence of estimates has converged to a stationary point of $\ell(x)$: $\hat{x}_k \to \hat{x}$ with $\nabla_x^c \ell(\hat{x}) = 0$.

Generalized Gradient Descent Algorithm – Cont.

IMPORTANT SPECIAL CASES:

$$\begin{aligned} \mathbf{Q(x)} &= \mathbf{I} & &\textbf{Gradient Descent Algorithm} \\ \mathbf{Q(x)} &= \boldsymbol{\mathcal{H}^{-1}(\mathbf{x})} & &\textbf{Newton Algorithm} \\ \mathbf{Q(x)} &= \boldsymbol{\Omega_{\mathbf{x}}^{-1}} & &\textbf{General Gradient Algorithm} \end{aligned}$$

- The (Cartesian or “naive”) Gradient Descent Algorithm is simplest to implement, but slowest to converge.

- The Newton Algorithm is most difficult to implement, due to the difficulty in constructing and inverting the Hessian, but fastest to converge.

- The General (or true) Gradient Descent Algorithm provides improved convergence speed over (naive) gradient descent when $\Omega_x \neq I$. It is also known as the **Natural Gradient Algorithm,** after Amari.

Derivation of the Newton Algorithm

A Taylor series expansion of $\ell(x)$ about a current estimate of a minimizing point $\hat{x}_k$ to second-order in $\Delta x = x - \hat{x}_k$ yields

$$\boldsymbol{\ell(\hat{x}_k + \Delta x) \approx \ell(\hat{x}_k) + \frac{\partial \ell(\hat{x}_k)}{\partial x} \Delta x + \frac{1}{2} \Delta x^T \mathcal{H}(x_0) \Delta x}$$

Minimizing the above wrt Δx results in

$$\widehat{\Delta x}_k = -\mathcal{H}^{-1}(\hat{x}_k) \nabla_x^c \ell(\hat{x}_k)$$

Finally, updating the estimate of the minimizing point via

$$\hat{x}_{k+1} = \hat{x}_k + \alpha_k \widehat{\Delta x}_k = \hat{x}_{k+1} = \hat{x}_k - \alpha_k \mathcal{H}^{-1}(\hat{x}_k) \nabla_x^c \ell(\hat{x}_k)$$

yields the Newton Algorithm.

As mentioned, the Newton Algorithm generally yields fast convergence. This is particularly true if it can be stabilized using the so-called **Newton step-size $\boldsymbol{\alpha_k = 1}$.**

Nonlinear Least-Squares

For the Linear Gaussian model, with known covariance matrix $C = W^{-1}$, the negative log-likelihood, $\ell(x) = -\log p_x(y)$, becomes equivalent to the (weighted)

Nonlinear Least-Squares Loss Function:

$$\boxed{\boldsymbol{\ell(\mathbf{x}) \doteq \|\mathbf{y} - \mathbf{h}(\mathbf{x})\|_{\mathbf{W}}^2}}$$

The (Cartesian) gradient is given by $\nabla_x^c \ell(x) = \left(\frac{\partial \ell(x)}{\partial x}\right)^T$, or

$$\boxed{\boldsymbol{\nabla_{\mathbf{x}}^{\mathbf{c}} \ell(\mathbf{x}) = -\mathbf{H}^{\mathbf{T}}(\mathbf{x})\mathbf{W}\left(\mathbf{y} - \mathbf{h}(\mathbf{x})\right)}}$$

where $H(x)$ is the Jacobian (linearization) of $h(x)$

$$\boxed{\boldsymbol{\mathbf{H}(\mathbf{x}) = \frac{\partial \mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}}}$$

This yields the **Nonlinear Least-Squares Generalized Gradient Descent Algorithm:**

$$\boxed{\boldsymbol{\hat{\mathbf{x}}_{\mathbf{k}+1} = \hat{\mathbf{x}}_{\mathbf{k}} + \alpha_{\mathbf{k}} \mathbf{Q}(\hat{\mathbf{x}}_{\mathbf{k}})\mathbf{H}^{\mathbf{T}}(\hat{\mathbf{x}}_{\mathbf{k}})\left(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}_{\mathbf{k}})\right)}}$$

Nonlinear Least-Squares – Cont.

Note that the Natural (true) gradient of $\ell(x)$ is given by

$$\boxed{\nabla_x \ell(x) = -\Omega_x^{-1} H^T(x) W \left(y - h(x)\right) = -H^*(x)\left(y - h(x)\right)}$$

where $H^*(x) = \Omega_x^{-1} H^T(x) W$ is the adjoint of the matrix $H(x)$.

A stationary point x, $\nabla_x \ell(x) = 0$, must satisfy the **Nonlinear Normal Equation**

$$\boxed{H^*(x) h(x) = H^*(x) y}$$

and the prediction error $e(x) = y - h(x)$ must be in $\mathcal{N}(H^*(x)) = \mathcal{R}(H(x))^\perp$

$$\boxed{H^*(x) e(x) = H^*(x)\left(y - h(x)\right) = 0}$$

Provided that the step size α_k is chosen to stabilize the nonlinear least-squares generalized gradient descent algorithm, the sequence of estimates $\hat{x}_k$ will converge to a stationary point of $\ell(x)$, $\hat{x}_k \to \hat{x}$ with $\nabla_x \ell(\hat{x}) = 0 = \nabla_x^c \ell(\hat{x})$.

Nonlinear Least-Squares – Cont.

IMPLEMENTING THE NEWTON ALGORITHM

One computes the Hessian from

$$\mathcal{H}(x) = \frac{\partial}{\partial x}\left(\frac{\partial \ell(x)}{\partial x}\right)^T = \frac{\partial}{\partial x}\nabla_x^c \ell(x)$$

This yields the **Hessian for the Weighted Least-Squares Loss Function:**

$$\boxed{\boldsymbol{\mathcal{H}(x) = H^T(x) W H(x) - \sum_{i=1}^{m} \mathcal{H}^i(x)\,[W\,(y - h(x))]_i}}$$

where

$$\boxed{\boldsymbol{\mathcal{H}^i(x) \triangleq \frac{\partial}{\partial x}\left(\frac{\partial h_i(x)}{\partial x}\right)^T}}$$

denotes the Hessian of the the i-th scalar-valued component of the vector function $h(x)$.

Note that all terms on the right-hand-side of of the Hessian expression are symmetric, as required if $\mathcal{H}(x)$ is to be symmetric.

Nonlinear Least-Squares – Cont.

- Evidently, *the Hessian matrix of the least-squares loss function* $\ell(x)$ *can be quite complex.* Also note that because of the second term on the right-hand-side of the Hessian expression, $\mathcal{H}(x)$ can become singular or indefinite.

- In the special case when $h(x)$ is *linear*, $h(x) = Hx$, we have that $H(x) = H$ and $\frac{\partial H_i(x)}{\partial x} = 0$, $i = 1, \cdots n$, yielding,

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

  which for full column-rank A and positive definite W is always symmetric and invertible.

- Also note that if we have a good model and a value $\hat{x}$ such that the prediction error $e(\hat{x}) = y - h(\hat{x}) \approx 0$, then

  $$\mathcal{H}(\hat{x}) \approx H^T(\hat{x})WH(\hat{x})$$

  where the right hand side is positive definite for positive definite W if $h(x)$ is locally one-to-one about the point $\hat{x}$.

Nonlinear Least-Squares – Gauss-Newton Algorithm

Linearizing $h(x)$ about a current estimate $\hat{x}_k$ with $\Delta x = x - \hat{x}_k$ we have

$$h(x) = h(\hat{x}_k + \Delta x) \approx h(\hat{x}_k) + \frac{\partial}{\partial x} h(\hat{x}_k)\Delta x = h(\hat{x}_k) + H(\hat{x}_k)\Delta x$$

This yields the loss-function approximation

$$\ell(x) = \ell(\hat{x}_k + \Delta x) \approx \frac{1}{2}\| (y - h(\hat{x}_k)) - H(\hat{x}_k)\Delta x\|_W^2$$

Assuming that $H(\hat{x}_k)$ has full column rank (which is guaranteed if $h(x)$ is one-to-one in a neighborhood of $\hat{x}_k$), then we can uniquely minimize $\ell(\hat{x}_k + \Delta x)$ wrt Δx to obtain

$$\widehat{\Delta x}_k = \left(H^T(\hat{x}_k) W H(\hat{x}_k)\right)^{-1} H^T(\hat{x}_k) W \left(y - h(\hat{x}_k)\right)$$

The update rule $\hat{x}_{k+1} = \hat{x}_k + \alpha_k \widehat{\Delta x}_k$ yields the a nonlinear least-squares generalized gradient descent algorithm known as the **Gauss-Newton Algorithm.**

Gauss-Newton Algorithm – Cont.

The Gauss-Newton algorithm corresponds to taking the weighting matrix $Q(x) = Q^T(x) > 0$ in the nonlinear least-squares generalized gradient descent algorithm to be (under the assumption that $h(x)$ is locally one-to-one)

$$\mathbf{Q(x)} = \left(\mathbf{H^T(x)WH(x)}\right)^{-1} \qquad \textbf{Gauss-Newton Algorithm}$$

- Note that when $e(x) = y - h(x) \approx 0$, the Newton and Gauss-Newton algorithms become essentially equivalent.
- Thus it is not surprising that the Gauss-Newton algorithm can result in very fast convergence, assuming that $e(\hat{x}_k)$ becomes asymptotically small.
- This is particularly true if the Gauss-Newton algorithm can be stabilized using the Newton step-size $\alpha_k = 1$.
- Straightforward modifications of $Q(x)$ yield the **Levenberg Algorithm** and the **Levenberg–Marquardt Algorithm**, both of which can have improved convergence relative to the Gauss-Newton Algorithm.