10-725: Optimization **Fall 2012**

Lecture 11: October 2

Lecturer: Geoff Gordon/Ryan Tibshirani *Scribes: Tongbo Huang, Shoou-I Yu*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

This lecture's notes illustrate some uses of various LaTeX macros. Take a look at this and imitate.

11.1 Matrix Differential Calculus

11.1.1 Review of Previous Class

Matrix differential is a solution to matrix calculation pain, it can be understood by either of the following:

- A compact way of writing Taylor expansion.
- Definition:
  $df = a(x; dx)[r(dx)]$ where r is the residual term
  $a(x, .)$ is linear in the second argument
  $\frac{r(dx)}{||dx||\to 0}$ as $dx \to 0$

The derivative is linear, so it passes through addition and scalar multiplication.
It also generalizes Jacobian, Hessian, gradient and velocity.
Other topics covered: chain rule, product rule, bilinear functions, identities and identities theorems. Please refer to previous scribed notes for details.

11.1.2 Finding a maximum, minimum or saddle points

ID for df(x)	scalar x	vector $\boldsymbol{x}$	matrix X
scalar f	df = a dx	df = $\boldsymbol{a}^T d\boldsymbol{x}$	df = tr(A^TdX)
vector $\boldsymbol{f}$	d$\boldsymbol{f}$ = $\boldsymbol{a}$ dx	d$\boldsymbol{f}$ = A d$\boldsymbol{x}$	
matrix F	dF = A dx		

The principle: set coefficient of dX to 0 to find min, max, or saddle point:

- if $df = c(A; dX)[r(dX)]$ then $dX = tA$, $df = c(A; tA) = tc(A; A)$
- function is at min/max/saddle point iff $c(A; A) = 0$
- if c is any product, then $A = 0$

11.1.3 Infomax ICA

Suppose we have n training examples $x_i \in R^d$ and a scalar-valued, component-wise function g. We would like to find the $d \times d$ matrix W that maximizes the entropy of $y_i = g(Wx_i)$.

Detour: volume rule:

$$vol(AS) = |det(A)|vol(S)$$

Interpretation: small determinant value means the existence of small eigenvalue, thus squash the volume flat, vice versa.

Back to infomax ICA. We have $y_i = g(Wx_i)$ where $dy_i = J(x;W)dx_i = J_i dx_i$. We want to maximize the entropy over the distribution of y:

$$max_W \Sigma_i(-ln(P(y_i))),\ P(y_i) = \frac{P(x_i)}{|detJ(x_i;W)|}$$

And from

$$maxH(P(y)) = -\int P(y)ln(P(y))dy = -E(ln(P(y)))$$

it is equivalent to maximizing

$$max_W \Sigma_i ln(|detJ(x_i;W)|)$$

since $P(x)$ is independent to W.

11.1.4 Solving ICA Gradient

Define $u_i = g'(Wx_i)$, $v_i = g''(Wx_i)$.

For gradient of $y_i = g(Wx_i)$:

$$\begin{aligned} dy_i &= g'(Wx_i) \circ d(Wx_i) \\ &= u_i \circ (Wdx_i) \\ &= diag(u_i)Wdx_i \end{aligned}$$

For gradient of $J_i = diag(u_i)W$:

$$\begin{aligned} dJ_i &= d(diag(u_i))W + diag(u_i)dW \\ &= diag(v_i \circ d(Wx_i))W + diag(u_i)dW \\ &= diag(u_i)dW + diag(v_i)diag(d(Wx_i))W \end{aligned}$$

Finally, define $diag(\alpha_i) = diag(u_i)^{-1} diag(v_i)$ solving the gradient of $\Sigma_i ln(|detJ(x_i; W)|)$:

$$\begin{aligned} dL &= \Sigma_i d(ln|detJ(x_i; W)|) \\ &= \Sigma_i tr(J_i^{-1} dJ_i) \\ &= \Sigma_i tr(W^{-1} dW + W^{-1} diag(u_i)^{-1} diag(v_i) diag(d(Wx_i)))W \\ &= \Sigma_i tr(W^{-1} dW) + tr(diag(\alpha_i) diag(d(Wx_i))) \\ &= n tr(W^{-1} dW) + \Sigma_i tr(\alpha_i^T d(Wx_i)) \\ &= n tr(W^{-1} dW) + tr(\Sigma_i x_i \alpha_i^T d(Wx_i)) \\ &= n W^{-T} + \Sigma_i \alpha_i x_i^T \\ &= n W^{-T} + C \end{aligned}$$

11.1.5 Natural Gradient

Define $L(W)$ as a function from $R^{d \times d}$ to R, then $dL = tr(G^T dW)$. So step S is:

$$S = argmax_S M(S), \; M(S) = tr(G^T S) - \frac{||SW^{-1}||_F^2}{2}$$

which, in scaler case:

$$M = gS - \frac{S^2}{2W^2}$$

So, to find the max/min/saddle point:

$$M = tr(G^T S) - \frac{1}{2} tr(SW^{-1} W^{-T} S^T)$$

$$dM = tr(G^T dS) - \frac{1}{2} tr(dSW^{-1} W^{-T} S^T)$$

So, natural gradient becomes $G = W^{-1} W^{-T}$, and thus $GW^T W = S$. Using the gradient previously derived, $[W^{-T} + C]W^T W = W + CW^T W$.

11.1.6 More Info

- Minkas cheat sheet:
  http://research.microsoft.com/en-us/um/people/minka/papers/matrix/
- Magnus & Neudecker. Matrix Differential Calculus. Wiley, 1999. 2nd ed.
  http://www.amazon.com/Differential-Calculus- Applications-Statistics-Econometrics/dp/047198633X
- Bell & Sejnowski. An information-maximization approach to blind separation and blind deconvolution. Neural Computation, v7, 1995.

11.2 Newton's Method

Newton's method have two main applications: solving nonlinear equations and finding minima/maxima/saddles.

11.2.1 Solving Nonlinear Equations

For $x \in \mathbb{R}^d$ and $f : \mathbb{R}^d \to \mathbb{R}^d$ which is differentiable, we want to solve $f(x) = 0$. We perform first order Taylor approximation on $f(x)$,

$$f(y) \approx f(x) + J(x)(y - x) = \hat{f}(y)$$

where $J(x)$ is the Jacobian. We now try to solve for $\hat{f}(y) = 0$.

$$f(x) + J(x)(y - x) = 0$$

$$dx = y - x = -J(x)^{-1} f(x) \tag{11.1}$$

dx represents the update step for Newton's method.

We now work on the example of approximating the reciprocal of the Golden Number ϕ. The function and the derivative is as follows.

$$f(x) = \frac{1}{x} - \phi, \quad f'(x) = -\frac{1}{x^2}$$

And dx becomes the following.

$$dx = -J(x)^{-1} f(x) = x^2(\frac{1}{x} - \phi) = x - x^2\phi$$

The update rule is $x^+ = x + dx$. Figure 11.1 shows an iteration of Newton's method when $x = 1$.

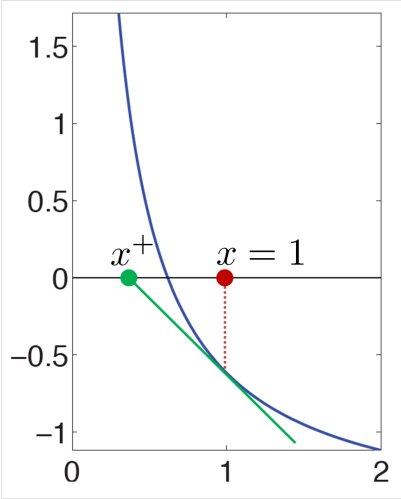

Figure 11.1: Example of one iteration of Newton's Method in solving non-linear equations.

We now perform error analysis of Newton's method. For value x, the error is $\epsilon = x\phi - 1$. For value x^+, the error ϵ^+ is the following.

$$\begin{aligned}
\epsilon^+ &= x^+\phi - 1 \\
&= \left(x + x - x^2\phi\right)\phi - 1 \\
&= \left(x + x\left(1 - x\phi\right)\right)\phi - 1 \\
&= \left(x - x\epsilon\right)\phi - 1 \\
&= x\phi - 1 - x\epsilon\phi \\
&= \epsilon - x\epsilon\phi \\
&= \epsilon\left(1 - x\phi\right) \\
&= -\epsilon^2
\end{aligned}$$

This shows that if $\epsilon < 1$, then Newton's method has quadratic convergence. However, if $\epsilon > 1$, then Newton's method will diverge.

11.2.2 Finding Minima/Maxima/Saddles

For $x \in \mathbb{R}^d$ and $f : \mathbb{R}^d \to \mathbb{R}^d$ which is twice differentiable, we want to find $\min_x f(x)$. In the example, we only focus on minimizing f, but finding the maxima and saddle points are the same. We first define $g = f'$. Minimizing f is the same as finding x such that $g = f' = 0$. From Equation 11.1, the Newton's update is the following,

$$\begin{aligned}
d &= -J^{-1}g \\
&= -\left(g'\right)^{-1} g \\
&= -\left(f''\right)^{-1} f' \\
&= -H^{-1}g
\end{aligned}$$

where H is the Hessian. We now show that Newton's method is a descent method if $H \succ 0$. We set $dx = td$ for $t > 0$. $r(dx)$ is the residual. Using first order Taylor expansion, we get the following.

$$\begin{aligned}
df &= g^T dx + r(dx) \\
&= g^T t\left(-\left(f''\right)^{-1} f'\right) + r(dx) \\
&= -tf'^T\left(f''\right)^{-1} f' + r(dx) \\
&= -tf'^T H^{-1} f' + r(dx)
\end{aligned}$$

If $H \succ 0$, then $H^{-1} \succ 0$, which makes the first term always negative, thus making Newton's method a descent method.

11.2.3 Newton's Method and Steepest Descent

Newton's method is a special case of steepest descent when the norm used is the Hessian norm. To find the step for steepest descent, we minimize the following.

$$\min_d g^T d + \frac{1}{2}||d||_H^2,\ ||d||_H^2 = \sqrt{d^T H d} \tag{11.2}$$

The solution to this minimization is $d = -H^{-1}g$. Steepest descent with a constraint or a penalty in the objective is equivalent. The equivalence will be covered when duality is covered in class. Figure 11.2 shows the difference between the direction of steps for gradient descent and Newton's method.

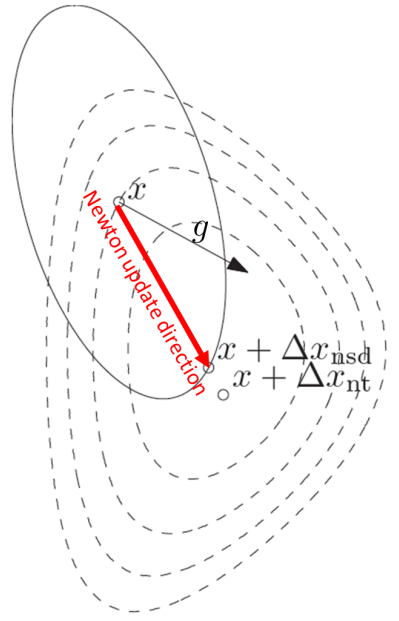

Figure 11.2: Direction of step for gradient descent and Newton's method.

11.2.4 Damped Newton

Damped Newton is combining the Newton's method with backtracking line search to make sure that the objective value does not increase.

Initialize x_1
for $k = 1, 2, \ldots$
- $g_k = f'(x_k)$; — gradient
- $H_k = f''(x_k)$; — Hessian
- $d_k = -H_k \backslash g_k$; — Newton direction
- $t_k = 1$; — backtracking line search
- **while** $f(x_k + t_k d_k) > f(x_k) + t g_k^T d_k / 3$ — divide by 3 to make sure $< \frac{1}{2}$ for future proofs
  - $t_k = \beta t_k$ — $\beta < 1$
- $x_{k+1} = x_k + t_k d_k$ — step

Damped Newton is affine invariant, meaning that suppose $g(x) = f(Ax + b)$, and we get Newton's updates $x_1, x_2, \ldots$ from $g(x)$ and $y_1, y_2, \ldots$ from $f(y)$, and if $y_1 = Ax_1 + b$, then $y_i = Ax_i + b \; \forall i$.

For damped Newton, if f is bounded below, then $f(x_k)$ converges. If f is strictly convex with bounded level sets, then x_k converges. Finally, damped Newton typically converges at quadratic rate in the neighborhood of x^*.