MULTIDIMENSIONAL SHARP QUADRATIC MAJORIZATION

JAN DE LEEUW

ABSTRACT. In a recent paper De Leeuw and Lange [2006] study sharp quadratic majorization for functions of a single variable. In this note we analyze a multivariate example and give some partial results.

1. INTRODUCTION

The problem we study in this paper is the minimization of a differentiable function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ using quadratic majorization. This means that in each iteration (k) we find a positive definite matrix $A(x^{(k)})$ such that

$$f(x) \leq g(x, x^{(k)}) = f(x^{(k)}) + (x - x^{(k)})'\mathscr{D}f(x^{(k)}) + \frac{1}{2}(x - x^{(k)})'A(x^{(k)})(x - x^{(k)}) \tag{1}$$

for all x. We then define the algorithmic map by

$$x^{(k+1)} = \mathop{\mathbf{argmin}}_x g(x, x^{(k)}) = x^{(k)} - A(x^{(k)})^{-1}\mathscr{D}f(x^{(k)}). \tag{2}$$

Since

$$f(x^{(k+1)}) \leq g(x^{(k+1)}, x^{(k)}) = \min_x g(x, x^{(k)}) < g(x^{(k)}, x^{(k)}) = f(x^{(k)}) \tag{3}$$

the algorithm generates a decreasing sequence of loss function values, and under the conditions given by Zangwill [1969] we have convergence to a local minimum at x_∞. The linear convergence rate of the algorithm is the spectral radius $\rho(I - A(x_\infty)^{-1}\mathscr{D}^2 f(x_\infty))$, and consequently it is of interest to choose matrices $A(x^{(k)})$ which are as large as possible.

This leads to the auxiliary problem of finding a "large" A satisfying the constraints

$$f(x) - f(x^{(k)}) - (x - x^{(k)})'\mathscr{D}f(x^{(k)}) - \frac{1}{2}(x - x^{(k)})'A(x - x^{(k)} \leq 0 \tag{4}$$

for all x. Equivalently (4) can be written as

$$\max_x f(x) - f(x^{(k)}) - (x - x^{(k)})'\mathscr{D}f(x^{(k)}) - \frac{1}{2}(x - x^{(k)})'A(x - x^{(k)} = 0. \tag{5}$$

Date: March 12, 2008 — 8h 26min — Typeset in TIMES ROMAN.

Note that (4) and (5) define a convex set of matrices A. Of course we need to have a specific definition of "large" to make this a well-defined optimization problem. One could use, for example, the trace, or the sum of squares, or the spectral radius.

2. Majorization of Multinomial Logistic

As an example consider the Multinomial Logistic with

$$\pi_j(x) = \frac{\exp(x_j)}{\sum_{\ell=1}^n \exp(x_\ell)}$$

and

$$f(x) = -\sum_{j=1}^m p_j \log \pi_j(x),$$

where p is a fixed probability vector.

Now

$$\mathscr{D}f(x) = \pi(x) - p,$$

and

$$\mathscr{D}^2 f(x) = \Pi(x) - \pi(x)\pi(x)'.$$

Since $\mathscr{D}^2 f(x) \lesssim \frac{1}{2}I$ uniform quadratic majorization is easy. The majorization function is just

$$g(x,y) = f(y) + (x-y)'\mathscr{D}f(y) + \frac{1}{4}(x-y)'(x-y),$$

leading to the algorithmic map

$$x^{(k+1)} = y - 2\mathscr{D}f(x^{(k)}),$$

which has convergence rate $\rho(I - 2\mathscr{D}^2 f(\hat{x}))$

In this note we investigate if locally sharper quadratic majorizations are possible, i.e. we look for a $\delta(y) \leq \frac{1}{2}$ such that

$$f(x) \leq f(y) + (x-y)'\mathscr{D}f(y) + \frac{1}{2}\delta(y)(x-y)'(x-y).$$

for all x and y. This will give a majorization algorithm with a faster convergence rate $\rho(I - \delta(y)^{-1}\mathscr{D}^2 f(\hat{x}))$.

Define

$$\delta(x,y) = \frac{f(x) - f(y) - (x-y)'\mathscr{D}f(y)}{\frac{1}{2}(x-y)'(x-y)},$$

and

$$\hat{\delta}(y) = \max_x \delta(x,y).$$

We know that $\hat{\delta}(y) \leq \frac{1}{2}$, but computation suggests that $\hat{\delta}(y)$ can actually be much smaller than $\frac{1}{2}$, especially if n is large and/or y has large variation.

Example 1. For $n = 2$ and $y_1 \neq y_2$ we have

$$\hat{\delta}(y) = \frac{\pi_2(y) - \pi_1(y)}{y_2 - y_1}.$$

attained for $x_1 = y_2$ and $x_2 = y_1$. This is basically the same result as the sharp quadratic majorization of the logistic in De Leeuw and Lange [2006]. If $y_2 \rightarrow y_1$ then $\hat{\delta}(y) \rightarrow \frac{1}{2}$.

Example 2. If all y are equal (without loss of generality we can take them all to be zero) then the optimal x has $n-1$ elements equal to $-\frac{2}{n}\log(n-1)$ and one element equal to $2\frac{n-1}{n}\log(n-1)$. Also

$$\hat{\delta}(y) = \frac{1}{2}\frac{n-2}{(n-1)\log(n-1)}$$

By L'Hospital if $n \rightarrow 2$ we have $\hat{\delta}(y) \rightarrow \frac{1}{2}$.

Example 3. We can use the fact that for all permutations P we have $\pi(Px) = P(\pi(x))$. Thus if $\hat{x}$ is a permutation of the elements of y we must have $(P-I)\pi(y) = \hat{\delta}(y)(P-I)y$, i.e. there exist constants α and β such that $\pi_i(y) = \alpha + \beta y_i$. Clearly we can solve for α and β if y only takes two different values, generalizing Example 1. In this case $\hat{\delta}(y)$ does not depend on the number of times the two values occur in y.

3. The Sharp Upper Bound

For the computation $\hat{\delta}(y)$ we could use $\mathscr{D}_1\delta(x,y) = 0$ if $\pi(x) - \pi(y) = \delta(x,y)(x-y)$, which suggests the iterative algorithm

$$x^{(k+1)} = y + \frac{1}{\delta(x^{(k)},y)}(\pi(x^{(k)}) - \pi(y)).$$

This algorithm was discussed earlier in a unidimensional context by De Leeuw and Lange [2006]. Their proof that the algorithm is monotone and convergent for any convex function f with a bounded second derivative easily extends to the multivariate case. The `R` code for the function `deltaOpt()` is given in the Appendix.

At a stationary point $\hat{x}$ we have

$$\hat{x} = y + \frac{1}{\delta(y)}(\pi(\hat{x}) - \pi(y)),$$

which implies $\sum_{j=1}^{m}\hat{x}_j = \sum_{j=1}^{m} y_j$.

Instead, we propose an alternative algorithm which generalizes more easily to situations in which $\delta(y)$ is not necessarily a scalar. Let us start with some initial $\delta^{(0)} \leq \hat{\delta}(y)$. One easy initial estimate is

$$\delta^{(0)} = 2\max_{i=1}^{n} f(y+e_i) - f(y) - \mathscr{D}_i f(y).$$

In iteration k we compute

$$\max_{x}\{f(x) - f(y) - (x-y)'\mathscr{D}f(y) - \frac{1}{2}\delta^{(k)}(x-y)'(x-y)\}.$$

If the maximum value is positive, attained say at $x^{(k)}$, then we set

$$\delta^{(k+1)} = \frac{f(x^{(k)}) - f(y) - (x^{(k)}-y)'\mathscr{D}f(y)}{\frac{1}{2}(x^{(k)}-y)'(x^{(k)}-y)},$$

and we conclude that $\delta^{(k)} < \delta^{(k+1)} \leq \hat{\delta}(y)$. If the maximum value is zero, we stop, because that implies $\delta^{(k)} \geq \hat{\delta}(y)$, and thus actually $\delta^{(k)} = \hat{\delta}(y)$.

4. Sharp Majorization Matrix

If we don't want to compute a sharp majorization constant, but a sharp majorization matrix, then that matrix Δ should satisfy

$$h(x,y) = f(x) - f(y) - (x-y)'\mathscr{D}f(y) - \frac{1}{2}(x-y)'\Delta(x-y) \leq 0 \tag{6}$$

for all x. Or, equivalently, $\max_x h(x,y) = 0$, with the maximum attained at $x = y$. We usually also require positive semi-definiteness, i.e. $\Delta \gtrsim 0$. This defines an infinite system of linear inequalities in Δ, which means that for each y there is a convex set of matrices $\Delta(y)$ satisfying these inequalities. In general this set could be empty, but we know that in our example every matrix $\Delta \gtrsim \frac{1}{2}I$ will be in $\Delta(y)$.

If f is convex, then we can use majorization to check if $\Delta \in \Delta(y)$. The algorithm is

$$x^{(k+1)} = y + \Delta^{-1}(\mathscr{D}f(x^{(k)}) - \mathscr{D}f(y)) = y + \Delta^{-1}(\pi(x^{(k)}) - \pi(y)). \tag{7}$$

Thus will increase $h(x,y)$ and as soon as $h(x^{(k)},y)$ becomes larger than zero, we know that $\Delta \notin \Delta(y)$. In fact, either $h(x^{(k)},y)$ converges to zero, and $x^{(k)}$ converges to y, or $h(x^{(k)},y)$ becomes positive and then diverges. Note that p does not play a role in this algorithm. Again the `R` code to test a particular candidate matrix Δ is in the Appendix.

Suppose ϕ measures how large Δ is. For instance, ϕ could be the trace, or the sum-of-squares, or the sup-norm. Start with a Δ which is not in $\Delta(y)$. Compute the x maximizing $h(x,y)$, say $\hat{x}$, and add the linear constraint $h(\hat{x},y) \leq 0$ to the constraint set. Find Δ minimizing $\phi(\Delta)$ over the linear constraints accumulated so far. Find a new $\hat{x}$, and add the new constraint, and so on. This is a variation of the cutting plane method, and it will find the majorization matrix with the smallest value of ϕ, in other words the majorization that is ϕ-sharp.

To start we can use (6) to derive simple lower bounds for the diagonal elements of Δ. We have

$$\delta_{ii} \geq 2\{f(y+e_i) - f(y) - \mathscr{D}_i f(y)\}$$

Appendix A. Code

```
delta<-function(x,y,p) {
pix<-exp(x)/sum(exp(x))
piy<-exp(y)/sum(exp(y))
fx<--sum(p*log(pix))
fy<--sum(p*log(piy))
gy<-piy-p
return((fx-fy-sum(gy*(x-y)))/(.5*sum((x-y)^2)))
}

deltaOpt<-function(y,p=rep(1/length(y),length(y)),itmax=100,eps=1e-10,verbose=TRUE) {
        n<-length(y)
        p<-runif(n)
        p<-p/sum(p)
        b<-bnds(y,p)
        x<-y
        x[which.max(b)]<-x[which.max(b)]+1
        piy<-exp(y)/sum(exp(y))
        fy<--sum(p*log(piy))
        gy<-piy-p
        itel<-1
        repeat {
        pix<-exp(x)/sum(exp(x))
        fx<--sum(p*log(pix))
        gx<-pix-p
        lb<-(fx-fy-sum(gy*(x-y)))/(0.5*sum((x-y)^2))
        if (verbose) cat("D-Iteration: ",formatC(itel,digits=3,width=3),
                "lb: ",formatC(lb,digits=6,width=10,format="f"),
                "\n")
        xnew<-y+(gx-gy)/lb
        if ((max(abs(x-xnew)) < eps) | (itel == itmax)) break()
        x<-xnew
```

```
        itel<-itel+1
        }
return(list(y=y,p=p,x=x,pix=pix,piy=piy,d=lb))
}

deltaMat<-function(y,x=sample(y,length(y)),d,itmax=1000,
    eps=1e-10,ops=Inf,verbose=TRUE) {
        n<-length(y)
        p<-runif(n)
        p<-p/sum(p)
        piy<-exp(y)/sum(exp(y))
        fy<--sum(p*log(piy))
        gy<-piy-p
        itel<-1
        repeat {
        pix<-exp(x)/sum(exp(x))
        fx<--sum(p*log(pix))
        gx<-pix-p
        hxy<-fx-(fy+sum((x-y)*gy)+sum((x-y)*colSums((x-y)
            *d))/2)
        if (verbose) cat("H-Iteration: ",formatC(itel,
            digits=3,width=3),
                "hxy: ",formatC(hxy,digits=6,width=10,
                    format="f"),
                "\n")
        xnew<-y+solve(d,(gx-gy))
        if ((max(abs(x-xnew)) < eps) | (itel == itmax))
            break()
        if (hxy > ops) break()
        x<-xnew
        itel<-itel+1
        }
return(list(h=hxy,x=x,cf=0.5*(x-y)^2,bf=fx-(fy+sum((x-y)
    *gy))))
}
```

```
bnds<-function(y,p) {
n<-length(y)
b<-rep(0,n)
piy<-exp(y)/sum(exp(y))
fy<--sum(p*log(piy))
gy<-piy-p
for (i in 1:n){
        x<-y
        x[i]<-x[i]+1
        pix<-exp(x)/sum(exp(x))
        fx<--sum(p*log(pix))
        gx<-pix-p
        b[i]<-2*(fx-(fy+gy[i]))
        }
return(b)
}

altIter<-function(y,p=rep(1/length(y),length(y)),itmax
    =1000,eps=1e-10,verbose=FALSE){
        n<-length(y)
        piy<-exp(y)/sum(exp(y))
        fy<--sum(p*log(piy))
        gy<-piy-p
        b<-bnds(y,p)
        iold<-which.max(b)
        bold<-b[iold]
        xold<-y
        xold[iold]<-xold[iold]+1
        itel<-1
        repeat {
        xupd<-deltaMat(y,x=xold,d=bold*diag(n),verbose=
            verbose)
        xnew<-xupd$x
        hval<-xupd$h
        if (hval < eps) break()
        ddst<-sum((xnew-y)^2)/2
```

```
        pix<-exp(xnew)/sum(exp(xnew))
        fx<--sum(p*log(pix))
        bnew<-(fx-(fy+sum((xnew-y)*gy)))/ddst
        cat("B-Iteration: ",formatC(itel,digits=3,width
            =3),
                "bold: ",formatC(bold,digits=6,width=10,
                    format="f"),
                "bnew: ",formatC(bnew,digits=6,width=10,
                    format="f"),
                "hval: ",formatC(hval,digits=6,width=10,
                    format="f"),
                "ddst: ",formatC(ddst,digits=6,width=10,
                    format="f"),
                "\n")
        if (((bnew-bold) < eps) | (itel == itmax))
            break()
        bold<-bnew
        itel<-itel+1
        }
}
```

References

J. De Leeuw and K. Lange. Sharp Quadratic Majorization in One Dimension. Preprint 503, UCLA Department of Statistics, 2006. URL http://preprints.stat.ucla.edu/503/sharp_third.pdf.

W. I. Zangwill. Convergence Conditions for Nonlinear Programming Algorithms. *Management Science*, 16:1–13, 1969.

Department of Statistics, University of California, Los Angeles, CA 90095-1554

E-mail address, Jan de Leeuw: deleeuw@stat.ucla.edu

URL, Jan de Leeuw: http://gifi.stat.ucla.edu