# DERIVATIVE OF FIXED-RANK APPROXIMATIONS

JAN DE LEEUW

ABSTRACT. In this paper we compute the matrix derivative of the fixed-rank least squares approximation. We describe the eigen structure of the derivative and apply the result to derive an expression for the speed of convergence of iterative nonlinear principal component algorithms.

## 1. INTRODUCTION

Suppose $X$ is an $n \times m$ matrix, with $n \geq m$. The singular value decomposition is $X = K\Lambda L'$, where the non-negative singular values are non-increasingly ordered along the diagonal of $\Lambda$ . The left singular vectors are in the $n \times m$ orthonormal matrix $K$ and the right singular vectors are in the $m \times m$ orthonormal matrix $L$. Thus $K'K = L'L = LL' = I$.

Suppose $\lambda_p > \lambda_{p+1}$. Define $K_p$ and $L_p$ as the $n \times p$ and $m \times p$ matrices with the left and right singular vectors correponding to the $p$ largest singular values. Also $\Lambda_p$ is the leading $p \times p$ principal submatrix of $\Lambda$. The unique least squares rank-p approximation to $X$ is

$$\Gamma_p(X) = K_p\Lambda_p L_p' = XL_pL_p'. \tag{1a}$$

This can also be written as

$$\Gamma_p(X) = \sum_{s=1}^{p} \lambda_s k_s l_s' = X\sum_{s=1}^{p} l_s l_s', \tag{1b}$$

where $k_s$ are $l_s$ are the columns of $K$ and $L$, i.e. the left and right singular vectors.

---

*Date*: September 14, 2008 — 9h 17min — Typeset in TIMES ROMAN.

2000 *Mathematics Subject Classification.* 00A00.

*Key words and phrases.* Singular Value Decomposition, Low-Rank Approximation, Jacobian.

## 2. DIFFERENTIATION

Now suppose $X$ depends on a parameter $\theta$. By the chain rule

$$\frac{\partial\Gamma_p(X)}{\partial\theta}=\frac{\partial X}{\partial\theta}\sum_{s=1}^p l_sl_s'+X\sum_{s=1}^p l_s\frac{\partial l_s'}{\partial\theta}+X\sum_{s=1}^p\frac{\partial l_s}{\partial\theta}l_s', \tag{2}$$

and by the formula for derivatives of eigenvectors [De Leeuw, 2007]

$$\frac{\partial l_s}{\partial\theta}=-(X'X-\lambda_s^2I)^+(X'\frac{\partial X}{\partial\theta}+\frac{\partial X'}{\partial\theta}X)l_s. \tag{3}$$

We write $\mathscr{D}\Gamma_p(X)$ for the derivative at $X$ of $\Gamma_p$, with respect to all coordinates. Thus $\mathscr{D}\Gamma_p(X)$ is a linear transformation from $\mathbb{R}^{n\times m}$ into $\mathbb{R}^{n\times m}$.

From Equation (2)

$$\mathscr{D}\Gamma_p(X)Z=ZL_pL_p'-X\{H_p(X,Z)+H_p'(X,Z)\}, \tag{4a}$$

where

$$H_p(X,Z)=\sum_{s=1}^p(X'X-\lambda_s^2I)^+(X'Z+Z'X)l_sl_s'. \tag{4b}$$

The derivative, and the partial derivatives with respect to any parameter, are computed by the `R` functions `lrPerturb()` and `lrDerive()` in the Appendix.

We now compute the eigenvalues and eigenvectors of the derivative operator $\mathscr{D}\Gamma_p(X)$. This allows us to give a complete description of the operator, and to easily compute its inverse, determinant, and so on.

Let us first look at matrices $Z$ of the form $k_\alpha l_\beta'+k_\beta l_\alpha'$ with $1\leq\alpha\leq p$ and $p<\beta\leq m$. There are $p(m-p)$ linearly independent ones. We have $Z\sum_{s=1}^p l_sl_s'=k_\beta l_\alpha'$ and $X'Z=\lambda_\alpha l_\alpha l_\beta'+\lambda_\beta l_\beta l_\alpha'$. It follows that $(X'Z+Z'X)l_sl_s'=(\lambda_\alpha+\lambda_\beta)l_\beta l_\alpha'\delta^{\alpha s}$ and $\sum_{s=1}^p(X'X-\lambda_s^2I)^+(X'Z+Z'X)l_sl_s'=\frac{1}{\lambda_\beta-\lambda_\alpha}l_\beta l_\alpha'$. Thus $\mathscr{D}\Gamma_p(X)Z=\frac{\lambda_\alpha}{\lambda_\alpha-\lambda_\beta}Z$.

In the same way for $Z$ of the form $k_\alpha l_\beta'-k_\beta l_\alpha'$ with $1\leq\alpha\leq p$ and $p<\beta\leq m$ we have $\mathscr{D}\Gamma_p(X)Z=\frac{\lambda_\alpha}{\lambda_\alpha+\lambda_\beta}Z$. This defines another $p(m-p)$ eigenvectors.

Now consider $Z=\sum_{\alpha=p+1}^n\sum_{\beta=p+1}^m c_{\alpha\beta}k_\alpha l_\beta'$. This defines $(n-p)(m-p)$ linearly independent eigen vectors. Note that this involves the left singular vectors $k_\alpha$ with $\alpha>m$, corresponding with the null space of $X$. Both $Z\sum_{s=1}^p l_sl_s'=0$ and $(X'Z+Z'X)l_sl_s'=0$ and thus $\mathscr{D}\Gamma_p(X)Z=0$.

Finally we have $(n-m)p+p^2$ linearly independent eigenvectors of the form $Z = \sum_{\alpha=1}^{p}\sum_{\beta=1}^{p} c_{\alpha\beta}k_\alpha l'_\beta + \sum_{\alpha=(n-m)+1}^{n}\sum_{\beta=1}^{p} d_{\alpha\beta}k_\alpha l'_\beta$. These correspond with eigenvalues equal to one, i.e. $\mathscr{D}\Gamma_p(X)Z = ZL_pL_p' = Z$.

It follows that all eigenvalues are non-negative, and the largest eigenvalue, i.e. the sup-norm, of the derivative is $\|\mathscr{D}\Gamma_p(X)\|_\infty = \frac{\lambda_p}{\lambda_p-\lambda_{p+1}}$.

## 3. APPLICATIONS

There are various forms of nonlinear principal component analysis, or NLPCA, discussed in the literature. We look at two popular ones, discussed and compared in De Leeuw [2006a].

In logistic NLPCA [De Leeuw, 2006b] the iteration function, which updates from one iteration to the next, is $\Phi(X) = \Gamma_p(X - 4(Y - \Pi(X)))$. Here $Y$ is the binary data matrix, which is fixed, and $\Pi(X)$ is the logistic transformation, i.e.

$$\pi_{ij}(X) = \pi(x_{ij}) = \frac{1}{1+\exp(-x_{ij})}.$$

This can be written as $\Phi(X) = \Gamma_p(\mathscr{P}(X))$, where $\mathscr{P}(X)$ is short for $X - 4(Y - \Pi(X))$. Note that $\mathscr{D}\mathscr{P}(X)$ is a diagonal operator with elements $1 - 4\pi(x_{ij})(1 - \pi(x_{ij}))$.

In least squares NLPCA [Gifi, 1990, Chapter 4] we alternate low-rank approximation with normalized cone regression of the columns of the data matrix. Thus the iteration is again of the form $\Phi(X) = \Gamma_p(\mathscr{P}(X))$, where now the cone projection defines $\mathscr{P}$. For most cones we use in practice the projection $\mathscr{P}$ is at least locally linear, although the normalization destroys the linearity again.

In all these cases we can write

$$\mathscr{D}\Phi(X) = \mathscr{D}\Gamma_p(\mathscr{P}(X))\mathscr{D}\mathscr{P}(X), \tag{5}$$

and the largest eigenvalue of this linear operator will give the linear convergence rate of the iterative algorithm.

As an example, consider least squares NLPCA in which the cones of transformations of the variables are centered third degree polynomial functions of the data. The orthogonal projectors $P_1, \cdots, P_m$ project on the three-dimensional subspaces of centered cubics. One iteration is

$$X^{(k+1)} = \Gamma_p(Y^{(k)}), \tag{6a}$$

where

$$Y^{(k)}=\begin{bmatrix}\frac{P_1x_1^{(k)}}{\|P_1x_1^{(k)}\|}&\cdots&\frac{P_mx_m^{(k)}}{\|P_mx_m^{(k)}\|}\end{bmatrix}\tag{6b}$$

The code in the appendix generates a $10\times 4$ example with standard normals, and fits the NLPCA in one, two, and three dimensions. If $p=1$ the observed convergence rate after 462 iterations is 0.9566, if $p=2$ it is 0.8735 after 141 iterations, and if $p=3$ it is 0.9959 after 4088 iterations. In all three cases the observed rate corresponds with the largest eigenvalue of the operator in Equation (5). If $p=3$ the iterations converge to a perfect fit, and the convergence will eventually be sublinear.

We also provide code to fit a logistic NLPCA. For a coin toss $50\times 6$ example the NLPCA in two dimensions converges sublinearly, and takes more than 10,000 iterations to meet the convergence criteria. The largest eigenvalue of (5) is larger than 0.9999, and more than 50 eigenvalues are larger than 0.99. From the point of view of convergence the algorithm is hopelessly slow, and definitely beyond salvation. The 1,000 iteration movie below, showing the left singular vectors scaled to the length of the singular values, indicates that maybe not all is lost.

(Loading logit.mov)

FIGURE 1. Logistic NLPCA Example

## REFERENCES

J. De Leeuw. Nonlinear Principal Component Analysis and Related Techniques. In M. Greenacre and J. Blasius, editors, *Multiple Correspondence Analysis and Related Methods*. Chapman and Hall, 2006a.

J. De Leeuw. Principal Component Analysis of Binary Data by Iterated Singular Value Decomposition. *Computational Statistics and Data Analysis*, 50(1):21–39, 2006b.

J. De Leeuw. Derivatives of Generalized Eigen Systems with Applications. Preprint Series 528, UCLA Department of Statistics, 2007.

A. Gifi. *Nonlinear Multivariate Analysis*. Wiley, Chichester, England, 1990.

# APPENDIX A. CODE

## A.1. **Derivatives.**

```
require(MASS)

lrPerturb<-function(x,dx,p) {
n<-nrow(x); m<-ncol(x)
c<-crossprod(x); e<-eigen(c)
l<-as.matrix(e$vectors[,1:p]); v<-e$values[1:p]
dl<-matrix(0,m,p)
for (s in 1:p) {
    cinv<-ginv(c-v[s]*diag(m))
    dl[,s]<--cinv%*%(crossprod(x,dx)+crossprod(dx,x))%*%l[,s]
    }
dz<-dx%*%tcrossprod(l)+x%*%(tcrossprod(l,dl)+tcrossprod(dl,l))
return(list(dz=dz,dl=dl))
}

lrDerive<-function(x,p) {
n<-nrow(x); m<-ncol(x); k<-1
dz<-matrix(0,n*m,n*m); dx<-matrix(0,n,m)
for (j in 1:m) for (i in 1:n) {
    dx<-matrix(0,n,m); dx[i,j]<-1
    dz[,k]<-as.vector(lrPerturb(x,dx,p)$dz)
    k<-k+1
    }
return(dz)
}
```

## A.2. **Least Squares NLPCA.**

```
rankfit<-function(x,p) {
s<-svd(x)
u<-s$u[,1:p]; v<-s$v[,1:p]; d<-s$d[1:p]
return(u%*%(d*t(v)))
}

polfit<-function(x) {
y<-cbind(p1%*%x[,1],p2%*%x[,2],p3%*%x[,3],p4%*%x[,4])
return(apply(y,2,function(z) z/sqrt(sum(z^2))))
}

go_for_ls<-function(p=2,itmax=1000,eps=1e-10,verbose=TRUE) {
zold<-rankfit(x,p); itel<-1; odif<-Inf
repeat {
   y<-polfit(zold); znew<-rankfit(y,p)
   fun<-sqrt(sum((y-znew)^2)); ndif<-sqrt(sum((zold-znew)^2))
   rat<-ndif/odif
   if (verbose)
       cat("Iteration: ",formatC(itel,digits=5,width=5,format="d"),
           "fun: ",formatC(fun,digits=15,width=20,format="f"),
```

```
            "dif: ",formatC(ndif,digits=15,width=20,format="f"),
            "rat: ",formatC(rat,digits=15,width=20,format="f"),
            "\n")
   if ((itel == itmax) | (ndif < eps)) break()
   itel<-itel+1; zold<-znew; odif<-ndif
   }
return(list(y=y,z=znew,f=fun,itel=itel,rat=rat))
}
```

## A.3. **Logistic NLPCA.**

```
rankfit<-function(x,p) {
s<-svd(x)
u<-s$u[,1:p]; v<-s$v[,1:p]; d<-s$d[1:p]
return(u%*%(d*t(v)))
}

logitfit<-function(x) {
    p<-1/(1+exp(-x))
    return(x+4*(y-p))
}

go_for_lg<-function(p=2,itmax=1000,eps=1e-10,verbose=TRUE) {
xold<-rankfit(y,p); itel<-1; odif<-Inf
repeat {
   z<-logitfit(xold); xnew<-rankfit(z,p)
pdf(paste("a",formatC(itel,width=4,flag="0",format="d"),".pdf",sep=""))
s<-svd(z)
plot(s$u[,1:2]%*%diag(s$d[1:2]),xlab="",ylab="")
dev.off()
   fun<-sum((1-y)*xnew)+sum(log(1+exp(-xnew)))
   ndif<-sqrt(sum((xold-xnew)^2))
   rat<-ndif/odif
   if (verbose)
       cat("Iteration: ",formatC(itel,digits=5,width=5,format="d"),
           "fun: ",formatC(fun,digits=15,width=20,format="f"),
           "dif: ",formatC(ndif,digits=15,width=20,format="f"),
           "rat: ",formatC(rat,digits=15,width=20,format="f"),
           "\n")
   if ((itel == itmax) | (ndif < eps)) break()
   itel<-itel+1; xold<-xnew; odif<-ndif
   }
pp<-1/(1+exp(-xnew)); vv<-1-4*pp*(1-pp)
return(list(x=xnew,z=z,p=pp,v=vv,f=fun,itel=itel,rat=rat))
}
```

## A.4. **Least Squares NLPCA Example.**

```
set.seed(12345)
x<-matrix(rnorm(40),10,4)
q<-qr.Q(qr(outer(x[,1],0:3,"^")))[,-1]
```

```
p1<-tcrossprod(q)
q<-qr.Q(qr(outer(x[,2],0:3,"^")))[,-1]
p2<-tcrossprod(q)
q<-qr.Q(qr(outer(x[,3],0:3,"^")))[,-1]
p3<-tcrossprod(q)
q<-qr.Q(qr(outer(x[,4],0:3,"^")))[,-1]
p4<-tcrossprod(q)
for (p in 3:1) {
    gfi<-go_for_ls(p=p,itmax=10000,verbose=FALSE)
    cat("itel: ",formatC(gfi$itel,digits=3,width=3,format="f"),
        "fun: ",formatC(gfi$f,digits=15,width=20,format="f"),
        "rat: ",formatC(gfi$rat,digits=15,width=20,format="f"),
        "\n")
    y<-gfi$y; z<-gfi$z
    gg<-lrDerive(y,p)
    hh<-matrix(0,40,40)
    r<-drop(p1%*%z[,1]); ss<-sum(r^2); s<-sqrt(ss)
    hh[1:10,1:10]<-(p1-outer(r,r)/ss)/s
    r<-drop(p2%*%z[,2]); ss<-sum(r^2); s<-sqrt(ss)
    hh[11:20,11:20]<-(p2-outer(r,r)/ss)/s
    r<-drop(p3%*%z[,3]); ss<-sum(r^2); s<-sqrt(ss)
    hh[21:30,21:30]<-(p3-outer(r,r)/ss)/s
    r<-drop(p4%*%z[,4]); ss<-sum(r^2); s<-sqrt(ss)
    hh[31:40,31:40]<-(p4-outer(r,r)/ss)/s
    cat(formatC(Re(eigen(gg%*%hh)$values),digits=6,width=10,format="f"))
    }
```

## A.5. **Logistic NLPCA Example.**

```
set.seed(12345)
y<-matrix(rbinom(300,1,.5),50,6)
gfi<-go_for_lg(itmax=10000)
cat("itel: ",formatC(gfi$itel,digits=3,width=3,format="f"),
    "fun: ",formatC(gfi$f,digits=15,width=20,format="f"),
    "rat: ",formatC(gfi$rat,digits=15,width=20,format="f"),
    "\n")
z<-gfi$z; v<-gfi$v
gg<-lrDerive(z,2)
matrix(Re(eigen(gg%*%diag(as.vector(v)))$values),50,6)
```

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