NOTES ON LINEAR ALGEBRA AND NUMERICAL ANALYSIS.

By Javier Cabrera

1. VECTORS AND MATRICES:

- Matrix: a rectangular array of *rxc* elements.
- Dimension: *rxc* means *r* rows by *c* columns. A square matrix has equal number of rows and columns (*r*=*c*).
- Example: A = [a $_{ij}$], i=1,2,3; j=1,2,3 is a square matrix
- In general: A = [a $_{ij}$], i=1,…r; j=1,...c

Multivariate observations=vector: $\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_p \end{pmatrix}$ is a multivariate observation, $\boldsymbol{x} \in \Re^p$.

$\boldsymbol{x}_1,\dots,\boldsymbol{x}_n$ is a sample of multivariate observations.

A sample can be represented by a matrix:

$$\boldsymbol{X}=\begin{pmatrix} x_{11},\dots,x_{1p} \\ x_{21},\dots,x_{2p} \\ \dots \\ x_{n1},\dots,x_{np} \end{pmatrix}$$

Vector addition, transpose and multiplication:

$$\begin{pmatrix}1\\3\\4\end{pmatrix}+\begin{pmatrix}2\\2\\1\end{pmatrix}=\begin{pmatrix}3\\5\\5\end{pmatrix} \qquad \begin{pmatrix}1\\3\\4\end{pmatrix}^T=\begin{pmatrix}1&3&4\end{pmatrix} \qquad \begin{pmatrix}1\\3\\4\end{pmatrix}^T\begin{pmatrix}2\\2\\1\end{pmatrix}=1\times2+3\times2+4\times1=12$$

Norm of a vector: $|v| = (v^T v)^{1/2}$

Unit vector: $|v|=1$

Diagonal matrix: a square matrix with all off-diagonal elements equal to 0
Identity matrix I: a diagonal matrix with all diagonal elements equal to 1
Note that A I = I A = A
Scalar matrix: a diagonal matrix with all diagonal elements equal to a scalar

Examples:

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad 2I = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \qquad A = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Matrix multiplication:

$$\begin{pmatrix} 1 & 2 \\ 3 & 2 \\ 4 & 1 \end{pmatrix} \begin{pmatrix} 2 & 3 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 1\times 2 + 2\times 2 & 1\times 3 + 2\times 1 \\ 3\times 2 + 2\times 2 & 3\times 3 + 2\times 1 \\ 4\times 2 + 1\times 2 & 4\times 3 + 1\times 1 \end{pmatrix} = \begin{pmatrix} 6 & 5 \\ 10 & 11 \\ 10 & 13 \end{pmatrix}$$

Canonical basis $\{e_1, e_2, \ldots, e_p\} = \left\{ \begin{pmatrix} 1 \\ 0 \\ \ldots \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ \ldots \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} 0 \\ 0 \\ \ldots \\ 1 \end{pmatrix} \right\}$

Orthogonal basis, coordinate system, orthogonal system, orthonormal system=orthogonal system of unit vectors.

Gram-Schmidt: Given a linearly independent vector system convert it to an orthogonal or orthonormal system.

Unary vector: $\underset{n\times 1}{1} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ $\underset{n\times 1}{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ or **1** = $\begin{pmatrix} 1 \\ 1 \\ \ldots \\ 1 \end{pmatrix}$

Unary matrix:

$$\underset{n\times n}{J} = \underset{n\times 1}{1}\,\underset{1\times n}{1'} = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ 1 & 1 & \ldots & 1 \\ \ldots & \ldots & \ldots & \ldots \\ 1 & 1 & \ldots & 1 \end{bmatrix}$$

$$\underset{1\times n}{1'}\,\underset{n\times 1}{1} = n$$

For addition and subtraction number of rows of the matrices must be the same and number of columns must be the same

For multiplication number of columns of the first matrix (left multiplier) must be the same as number of rows of second matrix (right multiplier)

Note that A + B = B + A but A B ≠ B A in general

2. OPERATIONS ON MATRICES

Determinant of a square matrix:

$$\det\begin{pmatrix} 3 & 1 \\ 4 & 2 \end{pmatrix} = 3\times 2 - 4\times 1$$

$$\det\begin{pmatrix} 1 & 2 & 0 \\ 3 & 2 & 1 \\ 4 & 1 & 2 \end{pmatrix} = 1\det\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} - 2\det\begin{pmatrix} 3 & 1 \\ 4 & 2 \end{pmatrix} + 0\det\begin{pmatrix} 3 & 2 \\ 4 & 1 \end{pmatrix} = 3 - 4$$

Inverse Matrix:

for a **square** matrix A is A^{-1} such that:

$A\,A^{-1} = A^{-1}A = I$

A^{-1} is unique and has the same rank as A

To have an inverse a matrix needs to be **full rank** (or **nonsingular**)

To check where a matrix is of full rank we check whether its **determinant** is 0

Example: 2 by 2 matrices

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad A^{-1} = \frac{1}{D}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \quad D = ad - bc$$

$$A = \begin{bmatrix} 2 & 4 \\ 3 & 1 \end{bmatrix} \quad D = (2)(1) - (4)(3) = -10$$

$$A^{-1} = -\frac{1}{10}\begin{bmatrix} 1 & -4 \\ -3 & 2 \end{bmatrix} = \begin{bmatrix} -0.1 & 0.4 \\ 0.3 & -0.2 \end{bmatrix}$$

3. SPECTRAL DECOMPOSITION.

Covariance matrix: Symmetric positive definite

Eigenvalues and Eigenvectors of a covariance or correlation matrix

Multivariate Normal distribution has constant density on ellipsoids:

Σ is a *pxp* dimensional Covariance Matrix, Ellipsoids: $z^t\Sigma^{-1}z = c$

Ellipsoids are characterized also by the semi-axes direction and length.

Spectral decomposition of a covariance matrix:

$\Sigma = \sum_{i=1}^{p} \lambda_i v_i v_i^T$ where λ_i ,i=1,…,p is called the i-th eigenvalue and v_i , i=1,…,p is called the i-th eigenvector.

Synonyms: eigenvalue == semi-axis length == variance == characteristic root == latent root, eigenvector==semi-axis direction == principal component==characteristic vector==latent vector.

In order to find the eigenvalues and eigenvectors of Σ we use the equation

$$\Sigma v_i = \lambda_i v_i$$
$$(\Sigma - \lambda_i I) v_i = 0$$
$$\det(\Sigma - \lambda_i I) = 0$$

PROPERTIES

(i) $\Sigma = V \Lambda V^t$, where V is the matrix of eigenvectors and Λ the diagonal matrix of eigenvalues of Σ.

(ii) The eigenvectors are used to describe the orthogonal rotation from maximum variance. This type of rotation is often referred to as principal-axes rotation.

(iii) The trace of the matrix Σ is the sum of its eigenvalues $\{\lambda_i\}$.

(iv) The determinant of the matrix Σ is the product of the λ_i

(v) Two eigenvectors v_i and v_j associated with two distinct eigenvalues λ_i and λ_j of a symmetric matrix are mutually orthogonal, $v_i^t v_j = 0$.

(vi) Square Root Matrix. There are several matrices that are the square root of Σ.

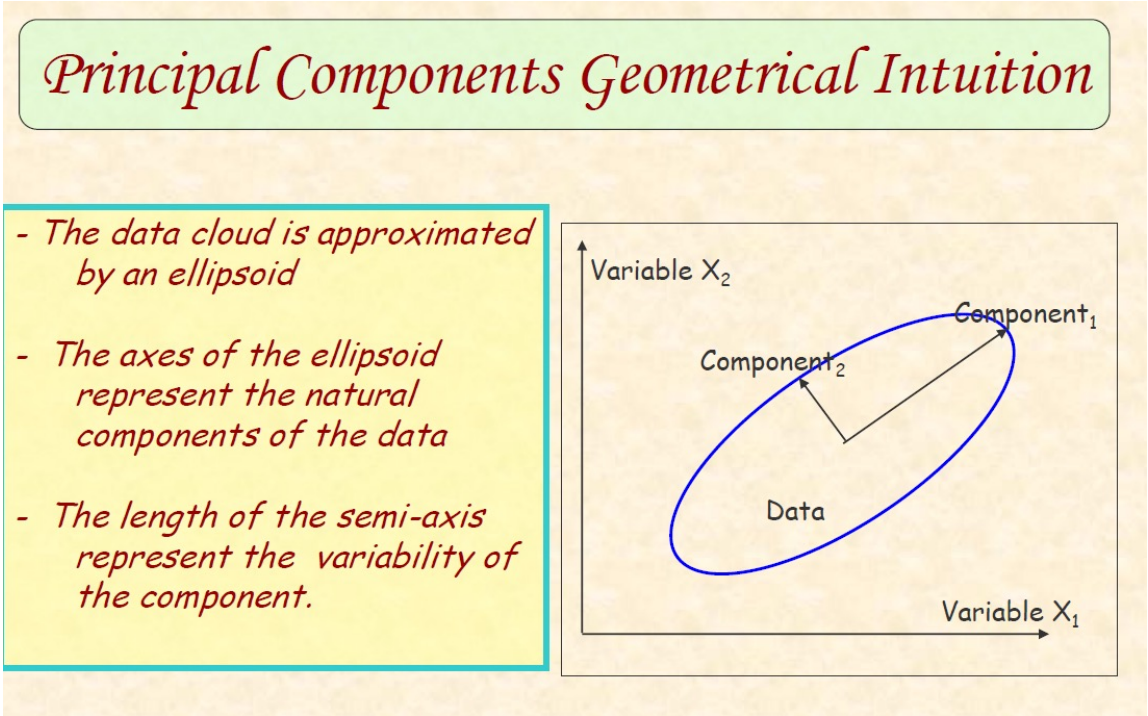
E.g. $\Sigma^{1/2} = V \Lambda^{1/2} V^t$, Choleski factorization: $\Sigma = R^T R$

(vii) Given a set of variables $X_1, X_2, ..., X_p$, with nonsingular covariance matrix Σ, we can always derive a set of uncorrelated variables $Y_1, Y_2, ..., Y_p$ by a set of linear transformations corresponding to the principal-axes rotation. The covariance matrix of this new set of variables is the diagonal matrix $\Lambda = V^t \Sigma V$

(viii) Given a set of variables $X_1, X_2, ..., X_p$, with nonsingular covariance matrix Σ_x, a new set of variables $Y_1, Y_2, ..., Y_p$ is defined by the transformation $Y' = X'V$, where V is an orthogonal matrix. If the covariance matrix of the Y's is Σ_y, then the following relation holds:

$y^T \Sigma_y^{-1} y = x^T \Sigma_x^{-1} x$ In other words, the quadratic form is invariant under rigid rotation.

(ix) The principal components of a dataset are the eigenvectors of the covariance matrix



Principal components are useful for dimension reduction. The covariance matrix of a data matrix X is $S = V\Lambda V^T$
The columns of V are the principal components, the diagonal elements of Λ are the variance estimates of the principal components variables. The scores XV are the representation of the data in the principal components coordinate system.

4. Singular value decomposition for big and small data

$X = UDV^T$ where U is (n by p) orthogonal, D is diagonal, V is (p by p) orthogonal.
If X is centered (zero column mean) then S=Cov(X) = $X^TX/(n\text{-}1)= VDU^TUDV^T/(n\text{-}1)= VD^2V^T/(n\text{-}1)=$
Then V is the matrix of eigenvectors of X and the diagonal of $D^2/(n\text{-}1)=$ are the eigenvalues.

This is convenient as long as $n > p$. In big data (Megavariate) datasets like microarray data the dimension of S is $p\times p$, which is not feasible computationally (p>10000). Also V is $p\times p$.
In these cases we want to avoid working with such large matrices but if n < 10000 then we use SVD not in X but in X^T resulting in : $X^T=U^* D^* V^{*T}$. X^T is now $p\times n$, U^* is $p\times n$, D^* and V^* are both $n\times n$, therefore all the matrices are small and are computable. Hence it follows that

S=Cov(X) = $X^TX/(n\text{-}1)= U^*D^* V^{*T}V^* D^*U^{*T}/(n\text{-}1)= U^*D^{*2}U^{*T}/(n\text{-}1)$

where $D^{*2}/(n\text{-}1)$ and U^* are now the eigenvalue and eigenvector matrices of S respectively.
At this point you should be able to answer questions 1-7 from the Midterm set.

Example in R:

```
S = cov(stack.x)
w = eigen(S)

Sminushalf= w$vectors %*% diag(w$values^(-1/2))%*%t(w$vectors)
round(cov(stack.x %*% Sminushalf),13)
```

```
Shalf <- w$vectors %*% diag(w$values^(1/2)) %*% t(w$vectors)
round(Shalf%*%Shalf - S,13)
```

5. Linear regression problem (and solution):

$\underset{n\times 1}{Y} = \underset{n\times p}{X}\ \underset{p\times 1}{\beta} + \underset{n\times 1}{\varepsilon}$ (Model)

Find b such that $\min_{\beta} |Y - X\beta|^2 = |Y - Xb|^2$

$\underset{p\times p}{X'X}\ \underset{p\times 1}{b} = \underset{p\times 1}{X'Y}$ (Normal Equations)

$b = (X'X)^{-1}X'Y$ (Least Squares Estimate)

Q is orthonormal if $Q'Q = I$,

Rotations are orthonormal transformations. $R = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$

Reflexions are orthonormal transformations: $I - 2uu'$

Sometimes people use orthogonal as orthonormal but technically orthogonal means $Q'Q = diag\{d\}$
Suppose we apply an orthogonal transformation Q to X so that $X^* = QX$ is upper triangular.
This is $QY = QX\beta + Q\varepsilon$ or $Y^* = X^*\beta + \varepsilon^*$, where X^* is upper triangular. X^* has all zeroes bellow the diagonal and bellow the pth row.

$$X^* = \begin{pmatrix} x^*_{11}, \ldots\ldots, x^*_{1p} \\ 0, x^*_{22}, \ldots, x^*_{2p} \\ \ldots\ldots\ldots\ldots \\ 0, \ldots\ldots, 0, x^*_{pp} \\ 0, \ldots\ldots\ldots, 0 \\ 0, \ldots\ldots\ldots, 0 \end{pmatrix} = \begin{pmatrix} X^*_1 \\ X^*_2 \end{pmatrix} \quad Y^* = \begin{pmatrix} Y^*_1 \\ Y^*_2 \end{pmatrix} \quad Q = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}$$

Assignment 2:

1. Assuming the framework and notation of section 5 on linear regression please show that

$$b = (X'X)^{-1}X'Y = X_1^{*-1}Y_1^* \quad \text{or} \quad X_1^* b = Y_1^*$$

Since X_1^* is upper triangular this equation can be easily solved by back substitution.

2. Show that the transformation $HX = H_p \ldots H_2 H_1 X$ is upper triangular by showing that each individual component makes zeros bellow the diagonal of X.

OVERALL PICTURE:

- In order to solve the regression problem we use the QR decomposition.
- In order to compute the QR decomposition we apply Householder transformations or Given's rotations.
- Another way to do it is by Gram –Schmidt

***QR* decomposition:**

Notice also that $X_1^* = Q_1 X$ so $X = Q_1^T X_1^*$ This is called the QR decomposition of X.

R Code:

```
> A <- matrix(runif(12), 4)
```

```
> A                                                    [,1]       [,2]        [,3]
          [,1]        [,2]        [,3]  [1,] -0.8875021 -0.8958916 -0.76105747
[1,] 0.4702383 0.486914919 0.461374841  [2,]  0.0000000 -0.5396342  0.62811379
[2,] 0.5928044 0.884345026 0.125048376  [3,]  0.0000000  0.0000000 -0.09710094
[3,] 0.1413287 0.273164872 0.004273825  > Q%*%R
[4,] 0.4417522 0.007447518 0.868701271            [,1]        [,2]        [,3]
                                        [1,] 0.4702383 0.486914919 0.461374841
> (Q <- qr.Q(qr(A)))                    [2,] 0.5928044 0.884345026 0.125048376
           [,1]        [,2]       [,3]  [3,] 0.1413287 0.273164872 0.004273825
[1,] -0.5298448 -0.02266620 -0.7453015  [4,] 0.4417522 0.007447518 0.868701271
[2,] -0.6679470 -0.52987173  0.5198505  > y = 1:4
[3,] -0.1592432 -0.24183094 -0.3602226  > lsfit(A,y,int=F)$coef
[4,] -0.4977478  0.81255153  0.2109995          X1          X2          X3
                                         8.7902727 -3.3647536 -0.5945298
                                        > c(solve( R, t(Q)%*%y))
                                         [1]  8.7902727 -3.3647536 -0.5945298

>(R <- qr.R(qr(A)))
```

```
Code summary:
(A <- matrix(runif(12), 4))
(Q <- qr.Q(qr(A)))
(R <- qr.R(qr(A)))
 Q%*%R
 y = 1:4
 lsfit(A,y,int=F)$coef
 c(solve( R, t(Q)%*%y))
```

Householder Transformations:

To construct Q we use Householder transformations

Suppose we have a nx1 vector x, we define an orthonormal transformation H_i that transforms x into a vector x^* that has zeroes at positions $(i+1),\ldots,n$

$H_i = (I - 2\, u_i u_i'/|u_i|^2)$ where

$$u_i = \begin{pmatrix} 0 \\ .... \\ x_i \pm s \\ x_{i+1} \\ .... \\ x_n \end{pmatrix} \text{ and } s^2 = \sum_{j=i}^{n} x_j^2$$

Homework (cont): Show that the transformation $HX = H_p \ldots H_2 H_1 X$ is upper triangular by showing that each individual component makes zeros bellow the diagonal of X.

Gram-Schmidt: Given a linearly independent vector system convert it to an orthogonal system.

Let $U = \{ \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p\}$ be a vector system that is independent but not orthogonal.

Step 1 Let $\mathbf{v}_1 = \mathbf{u}_1$.

Step 2 Let $\mathbf{v}_2 = \mathbf{u}_2 - \text{Proj}_{W_1} \mathbf{u}_2$ (W_1 is the subspace spanned by $\mathbf{v}_1$) $= \mathbf{u}_2 - \langle \mathbf{u}_2, \mathbf{v}_1 \rangle / |\mathbf{v}_1|^2 \ \mathbf{v}_1$

Step 3 $\mathbf{v}_3=\mathbf{u}_3-\text{Proj}_{W_2}\,\mathbf{u}_3$ (W_2 is subspace spanned by $\mathbf{v}_1\,\mathbf{v}_2$)$= \mathbf{u}_3-\langle\mathbf{u}_3,\mathbf{v}_1\rangle/|\mathbf{v}_1|^2\,\mathbf{v}_1-\langle\mathbf{u}_3,\mathbf{v}_2\rangle/|\mathbf{v}_2|^2\,\mathbf{v}_2$ and so on…
This process can be also adapted to produce the QR decomposition.
The problem with Gram-Schmidt is that it becomes unstable when U is nearly singular.

Givens Rotations

Notice the simple linear transformation knocks out the second component of the vector [a,b]'

$$R=\begin{pmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{pmatrix}=\begin{pmatrix}c & s\\ -s & c\end{pmatrix}\quad R\begin{bmatrix}a\\ b\end{bmatrix}=\begin{bmatrix}r\\ 0\end{bmatrix}\quad\begin{cases}r=\sqrt{a^2+b^2}\\ \cos\theta=a/r\\ \sin\theta=b/r\end{cases}$$

Givens rotations are matrices of the form

$$G(i,j,\theta)=\begin{bmatrix}1 & \cdots & 0 & \cdots & 0 & \cdots & 0\\ \vdots & \ddots & \vdots & & \vdots & & \vdots\\ 0 & \cdots & c & \cdots & s & \cdots & 0\\ \vdots & & \vdots & \ddots & \vdots & & \vdots\\ 0 & \cdots & -s & \cdots & c & \cdots & 0\\ \vdots & & \vdots & & \vdots & \ddots & \vdots\\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1\end{bmatrix}$$

As in the simple case $G(i,j,\theta)$ could be use to knock out the *jth* component of the *ith* column of the X matrix (X_i).
Notice that in order to make X upper triangular we would need as many Givens rotations as elements bellow the diagonal are made zero.

Other methods that could be computed with Householder and Given's:

- Choleski:
- SVD:

6. Generalization of Least Squares

The center piece of this part is the Iterative Reweighted Least Squares (IRLS) Algorithm which is fundamental for both GLM and M-estimation
GLM General Linear models

- X is generated from one or more factors by using dummy variables.
- Each factor is represented by k-1 dummies.
- Column of 1's maybe dropped
- Some columns of X must be dropped in order to avoid singularity

- X is likely to be very sparse
- X maybe high dimensional

The computations with GLM need to deal with these issues.
SAS and R both do a good job with these problems.

Weighted Least Squares (WLS)

$$\underset{n\times 1}{Y} = \underset{n\times p}{X}\ \underset{p\times 1}{\beta} + \underset{n\times 1}{\varepsilon} \qquad \varepsilon:\ F(0,\sigma^2 V) \quad V = \mathrm{diag}\{1/w_i\} \quad V_{ii} = 1/w_i$$

$$Y^* = V^{-1/2}Y = V^{-1/2}X\beta + V^{-1/2}\varepsilon = X^*\beta + \varepsilon^* \qquad \varepsilon^*:\ F^*(0,\sigma^2 I)$$

The computations change very little since we just need to replace X and Y by their X^* and Y^*

Generalized Linear Interactive models GLIM

$E(Y) = \mu$;
Link function: $f(\mu)$ is linear in X $f(\mu) = X\beta$
Variance function: $Var(Y) = v(\mu)$

Family	Link	Function	Variance function
Binomial	Logit	$\log(\mu/(1-\mu))$	$\mu(1-\mu)$
Poisson	Log	$\log(\mu)$	μ
Gaussian	Identity	μ	1

Weights are inverse to the variance: $w_i = 1/v(\mu_i)$

Iterative Reweighted Least Squares (IRLS) Algorithm

$\hat{\mu}^{(1)} = X\hat{\beta}^{(1)}$, $w_i^{(1)} = 1/v(\hat{\mu}_i^{(1)})$;

Use these weights in WLS to calculate $\hat{\beta}^{(2)}$

iterate: $\hat{\mu}^{(k)} = X\hat{\beta}^{(k)}$, $w_i^{(k)} = 1/v(\hat{\mu}_i^{(k)})$;

Use these weights in WLS to calculate $\hat{\beta}^{(k+1)}$

until it conversges .

This algorithm is very general and can be used for many problems

If $\hat{V}$ is the variance function estimate for the last iteration then $\hat{\sigma}^2(X'\hat{V}^{-1}X)^{-1}$ is an estimate of the covariance of $\hat{\beta}$.

7. Robust regression – M- estimators

$$S(\beta)= \sum \phi(\frac{y_i - x_i\beta}{s}) = \sum \phi(\frac{r_i}{s})$$

Huber's ϕ: $\phi(z) = \begin{cases} z^2/2 & \text{if } |z| \le c \\ c|z| - c^2/2 & \text{if } |z| > c \end{cases}$

Taking derivatives with r/s to β_i

$$-\frac{s}{2}\frac{\partial S(\beta)}{\partial \beta_j} = 0 = \sum x_{ij}\phi'(\frac{r_i}{s}) = \sum x_{ij}\psi(\frac{r_i}{s})$$

Huber's ψ: $\psi(z) = \begin{cases} z & \text{if } |z| \le c \\ sign(z)c & \text{if } |z| > c \end{cases}$

The idea is to write the system of equations as follows:

$$\sum x_{ij}\psi(\frac{r_i}{s}) = \sum x_{ij} r_i \frac{\psi(\frac{r_i}{s})}{r_i} = \sum x_{ij} r_i w_i = \mathbf{0}$$

In matrix terms this gives $X'Wr=0$ or $X'WY=X'W\beta$

The solution is $\hat{\beta} = (X'WX)^{-1} X'WY$

But of course the matrix W does depend on β so we apply here the same iterative reweighted least squares algorithm of the previous section.

At convergence we obtain our estimate $\hat{\beta}$ of β.

Package robust: huberM covRob lmRob glmRob

Package MASS: cov.rob rlm lmsreg(lqs)

Example:

```
x=rnorm(20)
x =rbind( cbind(x, x + rnorm(20)/2),c(-4,4))
plot(x)
(H = covRob(x, estim="donostah")$cov)
(G = covRob(x, estim="mcd")$cov)
 (S = cov(x))
ISqrtM = function(x) {e = eigen(x);
e$vec%*%diag(1/sqrt(e$val))%*%t(e$vec)}
```

```
plot( rs <-x%*%ISqrtM(S) )
plot( rs <-x%*%ISqrtM(G) )
plot( rs <-x%*%ISqrtM(H) )
```

8. Nonlinear Methods. Optimization methods.

Maximum Likelihood estimation.

$L_x(\theta) = P(X|\theta)$ $l_x(\theta) = ln(P(X|\theta))$

$\hat{\theta}$ maximizes $L_x(\theta)$, or $l_x(\theta)$. The information matrix is defined as

$I(\theta) = [-E_\theta \ddot{l}_X(\cdot)]\Big|_\theta$ and asymptotically converges to the inverse covariance matrix of $\hat{\theta}$

$asy\ \text{var}(\hat{\theta}) = I(\theta_0)^{-1}$ where θ_0 is the true value of the parameter.

Optimization methods

Newton-Raphson:
Solve f(x) = 0 starting at an initial value close to the solution.
$x_{i+1} = x_i - f(x_i) / f'(x_i)$

Newton-Raphson usually converges very fast
Multivariate version f(x) = 0
$0 = f(x_k) + f'(x_k)\ (s - x_k) + \ldots$
$x_{k+1} \sim x_k - [f'(x_k)]^{-1} f(x_k)$
px1 px1 pxp px1

f: R^p -> R
Do a search along the direction of *d* and find a minimum.
Steepest descent: $d_S = -F'(x)$ (notice that d_S is a *p*-vector)

Newton Steps: $d_N = -[F''(x)]^{-1}F'(x)$

Levenberg-Marquardt : $d_{LM} = -[F''(x)+\lambda I]^{-1}F'(x)$ for $\lambda > 0$

Simplex Method (Nelder Mead): Construct simplex around initial value. Then use one of 3 moves {reflexion, contraction, expansion}. Repeat until convergence. This method is very robust and usually works in difficult cases, but is also slow.

Use functions optimize and optim in R.
Example:

```
f <- function(x) ifelse(x > -1, ifelse(x < 4, exp(-1/abs(x-1)),10),10)
fp <- function(x) { cat(x, w<-f(x),"\n"); w };
plot(f, -2,5, ylim = 0:1, col = 2)
```

```
optimize(fp, c(-4, 20))# doesn't see the minimum
optimize(fp, c(-7, 20))# ok
```

Example:

```
fr <- function(x) {   ## Rosenbrock Banana function
        x1 <- x[1]
        x2 <- x[2]
        100 * (x2 - x1 * x1)^2 + (1 - x1)^2
    }
    grr <- function(x) { ## Gradient of 'fr'
        x1 <- x[1]
        x2 <- x[2]
        c(-400 * x1 * (x2 - x1 * x1) - 2 * (1 - x1),
           200 *      (x2 - x1 * x1))
    }
    optim(c(-1.2,1), fr, control = list(trace = TRUE))
)
    optim(c(-1.2,1), fr, grr, method = "BFGS")
    optim(c(-1.2,1), fr, NULL, method = "BFGS", hessian = TRUE)
    optim(c(-1.2,1), fr, grr, method = "CG")
    optim(c(-1.2,1), fr, grr, method = "CG", control=list(type=2))
    optim(c(-1.2,1), fr, grr, method = "L-BFGS-B")
```

9. Non Linear Least squares:

Remember that when the error term is additive and *iid* normal then NLS is the same as maximum likelihood. These are the steps of the Gauss-Newton algorithm.

$$y_i = g(x_i,\theta)+\varepsilon_i \equiv y_i = g_i(\theta)+\varepsilon_i \approx g_i(\theta_0)+(\theta-\theta_0)^t g_i'(\theta_0)+\varepsilon_i \quad \varepsilon_i : F(0,\sigma^2)$$

$$y_i - g_i(\theta_0) \approx (\theta-\theta_0)^t g_i'(\theta_0)+\varepsilon_i \Rightarrow Y_i = X_i^t(\theta-\theta_0)+\varepsilon_i$$

Gauss-Newton Method:

$$(\theta-\theta_0) = (X^tX)^{-1}X^tY = [\sum (g_i')(g_i')^t]^{-1}\sum g_i'(y_i-g_i) = -G^{-1}F'$$

Iterate this equality until convergence. To do that at each iteration replace θ_0 with the new θ.

Example: Emax model

$$g(D) = E(Y|D) = E_0 + \frac{E_{\max} D}{ED_{50} + D},$$

where E_0 is the response Y at baseline (absence of dose), E_{max} is the asymptotic maximum dose effect and ED_{50} is the dose which produces 50% of the maximal effect. A generalization of equation above is the 4-parameter EMAX model for

$$g(D) = E\left(Y \middle| D\right) = E_0 + \frac{E_{\max} D^{\lambda}}{ED_{50}^{\lambda} + D^{\lambda}},$$

where λ is the 4th parameter which is sometimes called the Hill parameter (Holford and Sheiner). The Hill parameter affects the shape of the curve and is in some cases very difficult to estimate. In Figure 1 we show curves for a range of Hill parameter values.

In fitting models above the error distribution is assumed to be iid normal with zero mean and equal variance σ^2. This assumption is not in any way a restriction of the methods that we discuss here but a convenient assumption that simplifies the issues that we address. Our model becomes:

$$Y = g_{\theta}(D) + \varepsilon\,, \qquad (1.3)$$

where $\theta = (E_0, ED_{50}, EMAX, \lambda)$ and $\varepsilon \sim iid\ N(0, \sigma^2)$.

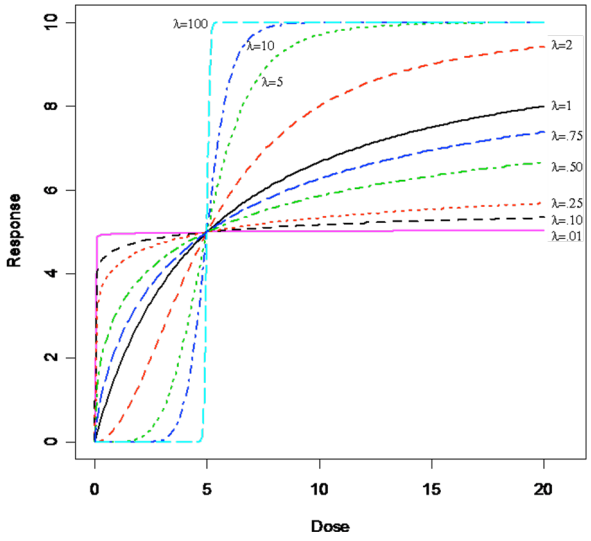


Figure: Curve Shapes for Different Values of Hill Parameter .

Figure above illustrates the rich variety of shapes generated by changing the Hill parameter λ. For

$\lambda \leq 1$the curves have concave downward shapes whereas for $\lambda > 1$ the curves have a sigmoidal shape.

For very small λ the curve represents a flat response for any dose greater than zero, whereas for very large λ the curve approaches a step function at ED_{50}.

2.2. Maximum Likelihood Estimation for the EMAX model.

Under the Emax model the MLE estimator of θ is the least squares estimator $\hat{\theta}$. One way to calculate $\hat{\theta}$ is by non-linear least squares minimization (NLS). Given the data $\{(D_i, y_i), i=1,\ldots,N\}$ the NLS estimator minimizes (in θ) the following quantity:

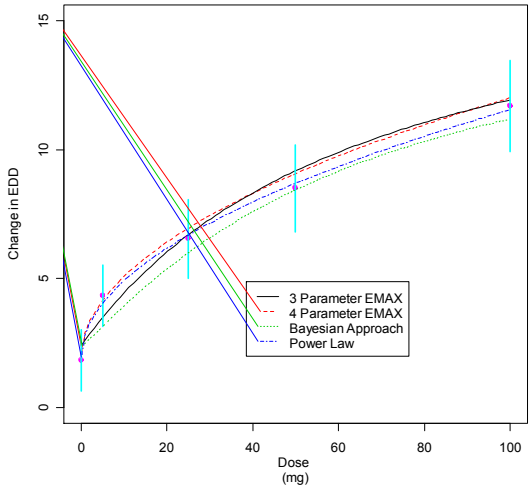
$$\text{SSE}(\theta)= \sum_{i=1}^{N} (y_i - g_\theta(D_i))^2$$

The algorithm to minimize this quantity is a special case of Newton-Raphson specialized for minimizing sums of squares. It is implemented in many standard software packages such as *R*, *S-plus* and others. The NLS implementation of the MLE's is convenient because it allows the computation of confidence intervals for the model parameters and p-values for testing the significance of the parameters.

Example. Dose response estimation in a clinical trial.

In order to illustrate the issues that we address in this paper we use data from an unpublished phase II clinical trial. The study had 5 doses (0, 5, 25, 50 and 100 mg) and corresponding group sizes (n=78, 81, 81, 81, 77). Unlike most dose-response trials this study has about 80 patients per arm and a high signal to noise ratio. The Figure below plots the response against the doses with 90% confidence intervals around the mean. The graph shows that it is a potentially good candidate for the EMAX model. We first fit 3-Paramater EMAX model to the data. The maximum likelihood estimators (MLEs) converge quickly and have estimates of $\hat{E}_{\max}$ = 15.13, $\hat{E}_0$= 2.47, and ED_{50}=67.49.

Figure: Clinical trial example with 5 doses. Red stars represent the response at a given dose and 90% CI. Solid line is the 3-parameter fit. Dashed line refers to the 4-parameter fit (using the estimates after 100 iterations.)



One concern about the fit in the figure is that the 3 parameter EMAX fit lacks the curvature necessary to fit well with the observed relationship and the residuals show an inverted U pattern.

To address the above lack of fit we next fit a 4-parameter EMAX model , but in this case we encounter non-convergence of the MLEs. Estimates of ED_{50}, and $E_{\max}$ do not converge. The Hill

parameter, λ, trends toward a value less than 1. If we trace the values of the MLEs after a few hundred iterations of the NLS algorithm we observe $\hat{E}_{max}$ = 2787, $\hat{E}_0$= 2.0, $\hat{\lambda}$=0.52 and ED_{50}=4.4×10^6 with large standard errors. The sequence of ED_{50}'s and E_{max}'s are monotonically increasing at a faster speed at each iteration until the computation becomes unstable (algorithm fails). At the same time the mean squared error (MSE) decreases very slowly towards an asymptote. This suggests that the likelihood, although bounded, converges to the maximum as both ED_{50} and E_{max} go to ∞.

R CODE:

```
m = nls(ch8~e0+(emax*dose^lambda)/(dose^lambda+ed50^lambda),start=
c(ed50=25,e0=5,emax=20,lambda=1),
control=nls.control(maxiter=100),trace=T,na.action=na.omit,data=edd)
f1 = fitted(m)
sig = summary(m)$sigma
```

10. Stochastic Approximation.

Solve an equation f(x) = 0 but instead of observing f(x) we observe g(x) = f(x) + error

The algorithm is due to Robins-Monroe(1955) $x_{i+1} = x_i - c\ g(x_i)/i$

Example: Logistic regression

$$E_\theta(\hat{\theta}) = \hat{\theta}_0 \quad \text{or} \quad E_\theta(\hat{\theta}) - \hat{\theta}_0 = 0$$

$$\hat{\theta}_{i+1} = \hat{\theta}_i - \frac{1}{i}(E_{\hat{\theta}_i}(\hat{\theta}) - \hat{\theta}_0)$$

$$Median_\theta(\hat{\theta}) = \hat{\theta}_0 \quad \text{or} \quad \text{Median}_\theta(\hat{\theta}) - \hat{\theta}_0 = 0$$

$$\hat{\theta}_{i+1} = \hat{\theta}_i - \frac{1}{i}(Median_{\hat{\theta}_i}(\hat{\theta}) - \hat{\theta}_0)$$

Homework:

1. Given the following data x= x = c(-10,(-3):5,10) ; y = c(0,0,1,0,0,1,0,1,1,1,1) calculate the stochastic approximation estimator of beta.

Expected value function

```
ef =function(b,z=x,ns=1000) {
n = length(x); px = 1/(1+ exp(-b*z) )
y = array( (runif(n*ns) <=px)*1, c(n,ns) )
for(i in 1:ns) n[i] = glm(y[,i]~z-1,family=binomial)$coef
mean(n)
}
```

```
Ef = function(b,z=x,ns=100) {
n = nrow(z); px = c(1/(1+ exp(-b%*%t(z)) ))
y = array( (runif(n*ns) <=px)*1, c(n,ns) )
w = array(NA,dim=c(ns,length(b)))
for(i in 1:ns) w[i,] = glm(y[,i]~.-1,family=binomial,data=z)$coef
apply(w,2,mean)
}

z = NULL
z[1] =  glm(y~x-1,family="binomial")$coef
for(i in 1:20) z[i+1] = z[i] - 1/i*(ef(z[i]) - z[1])
```

11. Other Topics of Interest

11.1 Random number generation

- Linear an multiplicative congruential
- Marsaglia Super-Duper: combination of 2 generators: Linear congruential: x=69069x+odd mod 2^32 and shift register: y=y(I+L^17)(I+R^15) where y is 32-bit sequence binary, L and R are left and right shift matrices, Period ~ 2^63-1.
- Mersenne-Twister: 624-dimensional set of 32-bit integers. Has an almost infinity cycle

Simulations can benefit form large cycle

11.2 Sampling from known CDF's.

Sampling from generic F

Let $F(x)$ be a cdf and let U be a uniform(0,1) random variable, then $F(x)$ is the cdf of the random variable $X = F^{-1}(U)$. (show this)

Sampling from Normal distribution:

Let X_1 and X_2 be iid standard normal random variables,

$Y = X_1^2 + X_2^2$ has a chis-square distribution with 2 degrees of freedom.

X_1 and X_2 can be generated from U and V two iid Uniform(0,1) random variables

$$X_1 = \sqrt{-2\ln U}\cos(2\pi V) \text{ and } X_2 = \sqrt{-2\ln U}\sin(2\pi V),$$

11.3 EM algorithm

Expectation step + maximization step

Useful for missing data problem

$$p(D_i) = \sum_{k=1}^{K} p(D_i \mid c_k)\,\alpha_k$$

\\\\\\\\\\\

Generative Model

- **select a component c_k for individual i**
- **generate data according to $p(D_i \mid c_k)$**
  - **$p(D_i \mid c_k)$ can be very general**
  - **e.g., sets of sequences, spatial patterns, etc**
- **given $p(D_i \mid c_k)$, we can define an EM algorithm**

- Converges to the MLE sometimes

MCMC
Penalized est
imation and optimization