STAT 802 Multivariate Analysis September 19, 2005

Today:
1. Review of matrix algebra we need for statistical distance, and easy calc method.
2. Multivariate Normal Distribution (Ch 4)

Statistical Distance

Heuristic Explanation – See Fig 1.20 p 31 and Fig 1.24 p 36.

Some details (Ch 2)

Vector Representation and Arithmetic.

L_x = Length of a vector x = $(x'x)^{1/2}$ (Note x'x is a scalar).
Angle between a vector x = (x_1,x_2) and the x_1 axis is $\cos^{-1}(x_1/L_x)$ (By definition of cos)
Inner Product of vectors x and y is denoted x'y and is scalar.
If θ is the angle between vectors x and y, then $\cos(\theta) = x'y/(L_xL_y)$
Generalizes to k dimensions. (not k variables!)

See example 2.1, p 54: angle between x and y where

```
x=c(1,3,2)
y=c(-2,1,-1)
costheta=t(x)%*%y/((t(x)%*%x)^.5*(t(y)%*%y)^.5)
theta=acos(costheta) # in radians
dtheta=(180/pi)*theta #in degrees
```

Recall definition of Linear Dependence
Vectors x_1, x_2, …., x_p are linearly dependent if you can express one as a linear function of the others. (See (2.7) p 54 bottom)

Projection of vector x on y – must be a multiple of the vector y – but what multiple?
Ans: x'y/(y'y)^.5 (see equation 2-8) so the projection is (x'y/(y'y))y. See Fig 2.5 p 55.
If we want to write the projection of x on y as a multiple of a unit vector, we re-scale y to be unit by y/L_y and multiply it by $|x'y|/L_y$
Can infer that length of this projection is $|x'y|/y'y = L_x \cos(\theta)$

Matrices (p 55 ff)
Arithmetic: Sums, Products, Inverses and the Identity Matrix

```
a= <- matrix(c(3,4,2,1), nrow = 2, ncol=2)
b=ginv(mdat)
>a%*%b
```

gives the identity matrix.

Symmetric Matrices are Important for Statistics (Why?)

Orthogonal (Square) Matrices: $A' = A^{-1}$ so $A'A = I$
(Note: Orthonormal might be better, with Orthogonal meaning A'A=Diagonal – but we will stick to the terminology of the book).

Eigenvalues and Eigenvectors

(Synonyms Latent and Characteristic have other meanings as well – confusing)

Square Matrix A – x is Eigen vector of A if $Ax=\lambda x$ where λ is a scalar. λ is called an eigenvalue of A corresponding to the eigenvector x. There are usually many such x and λ, but never more than the rank of A ($\leq$ the dimension of A).

Now concentrate on (square) symmetric matrices (like covariance matrices) – see theorem p 61. Eigenvectors are mutually perpendicular unless some eigenvalues are multiple.
By convention, usually specify eigenvectors to have unit length (wlog see definition).

Positive Definite Matrices

A symmetric matrix A is Positive Definite if $x'Ax > 0$ for all $x\neq 0$
Can show this is equivalent to all eigenvalues >0 (p 64)

Statistical Distance of vector x from vector 0 (origin) has the form x'Ax where A is pos definite.
From x to μ is $(x-\mu)'A(x-\mu)$. But we still have to make A specific to call it statistical distance.

To find appropriate A: See p 66 Fig 2.6 If the ellipse shown were a contour of the bivariate normal density, then we would find e_1, e_2, λ_1 and λ_2 from the eigenanalysis of the Σ^{-1} where Σ is the covariance matrix. So the distance of points from the origin would be $x'\Sigma^{-1} x$ (In the diagram, the variables are centered so the mean is the origin). This generalizes to p dimensions (See Theorem bottom p 81) . It implies that if you keep extracting maximal variance projections, each one perpendicular to the previous one, then the eigenvalues will be the maximal variances at each stage, and the eigenvectors will be the directions that the projections must be taken.

The interpretation depends on the multivariate normality, but the procedure does not. The means and covariances of the data will produce statistical distances from the centroid for every "case".

Σ can be expressed as $E[(x-\mu)'(x-\mu)]$ and is estimated by the equivalent sample formula (p 125). Because Σ is positive definite usually, there is a spectral representation based on the eigenanalysis (see p 67 , (2-21) The A in this formula can be replaced by Σ or its sample estimate.). This representation shows that the statistical distance is indeed found by an orthogonal rotation of the axes followed by the Euclidean distance in the standardized new coordinates. ($(\lambda_\iota)^{1/2}$ is the sd of the projections of the data in the direction of the ith eigenvector).

Statistical distances the easy way.

Let x' be a row vector of a data matrix. Then
$(x-\hat{\mu})'\,\hat{\Sigma}^{-1}(x-\hat{\mu})$
is the statistical distance from x to the centroid $\hat{\mu}$.

Note $\hat{\Sigma}^{-1}$ is $\sum_{\iota-1}^{\pi}$ $(e_i e_i')/\lambda_\iota$

Here is an R program to compute the statistical distances for an n by p data matrix.

(Note: I used my.dist instead of just dist as my function name since R already has a function called "dist")

Note the loop for the computing the distance for each p-tuple.

We can do this matrix-wise as well

```
function (data)
{
        n=length(data[,1])
        p=length(data[1,])
        diff=matrix(nrow=n,ncol=p)
        distance=1:n
diff=as.matrix(data-mean(data))
distance=diag((diff%*%ginv(cov(data))%*%t(diff))^.5)
return(distance)
}
```

Note:

I generate some multivariate normal data with a specific sample mean and covariance matrix. Then I compute the eigenanalysis from the covariance matrix, and use this to compute the new uncorrelated coordinates. Rescaling these with their sd, then using Euclidean distance, gives me the statistical distances of the original data.

Here are the programs: mat.ex generates the data and does the eigen approach my.dist uses the inverse covariance matrix approach. If you execute my.dist(mat.ex()) you get the two identical vectors of statistical distance.

```
mat.ex
function (n=25,p=2,print=F,corr=.5)
{
        cov=matrix(ncol=p,nrow=p)
        for (i in 1:p){
                for (j in 1:p){
                        cov[i,j]=corr}
                        cov[i,i]=1
                        }
        data=as.data.frame(mvrnorm(n,c(rep(0,p)),cov))
        #plot(data)
        cova=cov(data)
        eigen=eigen(cova)
        if (print==T){print(cov);print(data);print(cova);print(eigen)}
        data=as.matrix(data)
        ev=as.matrix(eigen$vectors)
        ndata=data%*%ev
        v1=(ndata[,1]-mean(ndata[,1]))/sd(ndata[,1])
        v2=(ndata[,2]-mean(ndata[,2]))/sd(ndata[,2])
        d=(v1^2+v2^2)^.5
        print(d)
        return(data)
}
my.dist
function (data)
{
        n=length(data[,1])
        p=length(data[1,])
        diff=matrix(nrow=n,ncol=p)
        distance=1:n
diff=as.matrix(data-mean(data))
distance=diag((diff%*%ginv(cov(data))%*%t(diff))^.5)
return(distance)
}
```

Ch 3: We skip most of it except

Sample Mean and Sample Variance
Generalized variance

Sample Variance – see p 124 for sample variance S and generalized sample variance |S|. Interpretation of scalar |S| see equation 3-15 on p 126.   Use |R| when appropriate.

Ch 4: Multivariate Normal Distribution

See density 4-4 p 150

Bivariate case useful by analogy for understanding p-variate case.

Connection with statistical distance Eqn 4-7 p 153.

Distribution of squared statistical distance Eqn 4-8 p 155. Chi Sq (df=p when Σ known).

See Remark p 164.   Statistical Distance is Euclidean Distance in the Transformed Variables.

Exercise: Analyze the dat in Table 4.4 using the techniques of Section 4.7. Hand in Monday Sept 26.   Show code for whatever software you use.