

# Finding Basis Vectors Using the SVD function in IDL

Emmett Ientilucci 1-22-05

For this discussion we assume ROW major notation, which is what the majority of the math community adheres to. IDL, however, is column major. We will make the distinction where necessary.

## 1 Data Vectors are in Rows

Assume we have an  $M \times N$  test data set with  $N=3$  bands (sometimes called variables) and  $M=10$  pixels (or observations). That is

```
IDL> print, A
  3  4  1
  8  2  9
  8  7  7
  8  2  2
  4  6  2
  9  1  6
  9  6  9
  2  4  8
  8  6  3
  8  2  5
```

Here we have the case where the number of rows  $M$  is greater than or equal to the number of columns  $N$ . When this is the case,  $\mathbf{A}$  can be written as the product of an  $M \times N$  matrix  $\mathbf{U}$ , an  $N \times N$  diagonal matrix  $\mathbf{W}$ , and the transpose of an  $N \times N$  matrix  $\mathbf{V}$ .

The “data vectors” we are interested in make up the 10 vectors in the *rows* of  $\mathbf{A}$ . We then use the SVD function in IDL to decompose matrix  $\mathbf{A}$ . That is

```
SVDC, A, w, U, V, /DOUBLE
```

This produces arrays of size (in COL major notation):

```
U          DOUBLE   = Array[3, 10]
V          DOUBLE   = Array[3, 3]
W          DOUBLE   = Array[3]
```

where we have 3 singular values, from a diagonal matrix, which are

```
IDL> print, w
    30.926379      7.2824447      7.5845280
```

Note that these need to be ordered from largest to smallest. We will touch on this later when selecting basis vectors. The  $\mathbf{U}$  matrix is

```

IDL> print, U
-0.13971142    0.047959336    0.35400407
-0.37767509    0.018512151    -0.46720402
-0.40550304    0.20201472     0.21118254
-0.24741061   -0.49108387    0.10747525
-0.20736089    0.15635893    0.48583307
-0.33185989   -0.36174684   -0.31550236
-0.45273389    0.18575288   -0.047600994
-0.24695420    0.65359577   -0.22892424
-0.31805692   -0.15500329    0.43673220
-0.30323825   -0.27268557   -0.13881586

```

and the  $\mathbf{V}$  matrix is

```

IDL> print, V
-0.71197800   -0.69940899    0.062565093
-0.40232973    0.47933259    0.77998145
-0.57551553    0.53015783   -0.62266729

```

At this point we are ready to select a set of orthonormal basis vectors. Based on the convention set forth thus far, our basis vectors form the columns of  $\mathbf{V}$ . Take note that if we only want “2” basis vectors in this example, they would correspond to columns 1 and 3 not 1 and 2. This is because of the ordering issue stated earlier.

Orthonormality can be checked by multiplying a column by itself and then with any other column. One should obtain, using columns one and two, for example

$$\mathbf{V}_1^T \mathbf{V}_1 = 1$$

and

$$\mathbf{V}_1^T \mathbf{V}_2 = 0.$$

The  $\mathbf{U}$  matrix, in other packages, may be of size  $M \times M$ . However, IDL only takes the columns associated with the first  $N$  singular values. Therefore, we have a matrix that is of size  $M \times N$ , in row major notation, which is what one would manually generate if given  $\mathbf{U}$  as  $M \times M$ .

We can easily show that  $\mathbf{U}$  is not a row-orthonormal matrix by checking orthonormality across the rows.

## 2 Data Vectors are in Columns

Assume we have an  $M \times N$  test data set with  $M=3$  bands and  $N=10$  pixels.

```

IDL> print, A
  3      8      8      8      4      9      9      2      8      8
  4      2      7      2      6      1      6      4      6      2
  1      9      7      2      2      6      9      8      3      5

```

The “data vectors” of interests are the 10 vectors in the *columns*. Here the rows of **A** are NOT greater than the number of columns. That is  $M \leq N$ . This produces slightly different result than found in Section (1), which we will explain later.

We then use the SVD function in IDL to decompose matrix **A**. That is

```
SVDC, A, w, U, V, /DOUBLE
```

This produces arrays of size (in COL major notation):

```
U          DOUBLE   = Array[10, 3]
V          DOUBLE   = Array[10, 10]
W          DOUBLE   = Array[10]
```

where we have 3 singular values, from a diagonal matrix, which are

```
IDL> print, w
 30.926379  2.4325160e-015  7.5845280  7.2824447  0.0  0.0  0.0  0.0  0.0  0.0
```

Again, note that these need to be sorted, after which the first “3” are valid, as per our example.

The **U** matrix is

```
IDL> print, U
-0.71197800  4.7968655e-021  -0.06256509  0.6994089  0.0  0.0  0.0  0.0  0.0  0.0
-0.40232973  2.7106250e-021  -0.77998145  -0.4793325  0.0  0.0  0.0  0.0  0.0  0.0
-0.57551553  3.8774868e-021  0.62266729  -0.5301578  0.0  0.0  0.0  0.0  0.0  0.0
```

and the **V** matrix is

```
IDL> print, V[0:4,*]
-0.13971142  0.92350514  -0.35400407  -0.047959336  0.00000000
-0.37767509  0.12099421  0.46720402  -0.018512151  0.14927865
-0.40550304  -0.15278895  -0.21118254  -0.20201472  -0.57914450
-0.24741061  -0.053124458  -0.10747525  0.49108387  -0.017249444
-0.20736089  -0.22572310  -0.48583307  -0.15635893  0.74808664
-0.33185989  0.089521581  0.31550236  0.36174684  0.16949035
-0.45273389  -0.059891097  0.047600994  -0.18575288  -0.0012264659
-0.24695420  0.016450123  0.22892424  -0.65359577  0.059991245
-0.31805692  -0.20747833  -0.43673220  0.15500329  -0.20955446
-0.30323825  0.021497826  0.13881586  0.27268557  0.078331755
```

```
IDL> print, V[5:9,*]
0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
-0.43875110  -0.39698376  -0.39360077  0.054820352  -0.30625160
0.20478311  -0.27636992  -0.16893127  -0.49280129  -0.041220350
-0.35743094  0.061226097  0.59281186  -0.32656961  -0.30557293
0.066124175  -0.15192916  -0.10713470  -0.21679946  0.00056564712
0.76996827  -0.023928431  0.057609025  0.063886783  -0.14799117
-0.092333639  0.83613641  -0.20578072  0.0058455310  -0.068147782
0.036021519  -0.12665435  0.63788821  0.16911731  0.052052096
-0.042975066  -0.14515911  -0.0021474762  0.75350773  -0.080136775
-0.17003242  -0.048819116  0.032656107  -0.0012856318  0.88060143
```

### 3 Note about Comments in, “Numerical Recipes in C”

At this point we make reference to a section the in *Numerical Recipes in C* book about “Constructing an Orthonormal Basis”. Here we are told that a basis set of vectors can be formed from an  $M \times N$  matrix  $\mathbf{A}$  whose  $N$ -COLUMNS are the data vectors. The desired orthonormal basis vectors will then form the columns of the matrix  $\mathbf{U}$ . We can see that the first few columns of  $\mathbf{U}$  above are exactly the same as the columns in  $\mathbf{V}$  when we formed matrix  $\mathbf{A}$  with the data vectors in ROWS.

In other words, if the data vectors form columns in  $\mathbf{A}$ , then we extract the basis vectors from  $\mathbf{U}$ . If the data vectors form rows in  $\mathbf{A}$ , then we extract the basis vectors from  $\mathbf{V}$ . So which one should we use?

The choice becomes clear when we look at the type of data matrices we have and run times. For the IDL SVD routine (which is taken from Numerical Recipes in C) we need  $\mathbf{A}$  to have more rows than columns ( $M \geq N$ ). Since hyperspectral data always has more pixels than bands, we form a matrix  $\mathbf{A}$  with  $M$  pixels (rows) and  $N$  bands (columns). But by doing this the data vectors in  $\mathbf{A}$  now form *rows* not columns, like the algorithm expects. Therefore, we now have to obtain the basis vectors, not from  $\mathbf{U}$ , but from  $\mathbf{V}$  because of the transpose we had to setup in matrix  $\mathbf{A}$ .

Setting up the data with ( $M \geq N$ ) is much more computationally efficient as well, as per the algorithm. If we actually set up  $\mathbf{A}$  with the data vectors in columns ( $M=3$  bands and  $N=10$  pixels), then we get an extremely large  $\mathbf{V}$  matrix ( $N \times N$ ), where  $N$  is associated with the number of image pixels and can take on values of many thousands.

### 4 Concluding Remarks

Set up your data (as input to this algorithm) with rows as pixels and columns as bands. For example, if your data is 100 pixels by 100 pixels by 224 bands, matrix  $\mathbf{A}$  would be 100000 rows by 224 columns. Your data vectors would make up the *rows* of  $\mathbf{A}$ . In IDL format this would be [224 x 100000]

Order the singular values and corresponding column-orthonormal vectors of  $\mathbf{V}$  from largest to smallest.

Choose your basis vectors from matrix  $\mathbf{V}$ , as per the discussion above.