Example 127. How is the rank of A reflected in its singular value decomposition $A=U\Sigma V^T$?

Solution. The rank of A is equal to the number of nonzero singular values.

Theorem 128. (matrix approximation lemma) Suppose A is a $m\times n$ matrix, and we want to approximate A using a matrix B of rank s (smaller than the rank of A).

Let $A=U\Sigma V^T$ be the SVD of A.

Then, the best such approximation is $B=U\Sigma_s V^T$, where Σ_s is obtained from Σ by setting all but the largest s singular values to 0.

In other words, Σ_s has the values $\sigma_1,\sigma_2,...,\sigma_s$ on its diagonal (assuming the singular values are in decreasing order, as usual), followed by zeros.

Comment. Equivalently, $B=U_s\Sigma_s V_s^T$, where Σ_s is the $s\times s$ diagonal matrix with entries $\sigma_1, \sigma_2, ..., \sigma_s$ and U_s, V_s are obtained from the corresponding matrices in the SVD $A=U\Sigma V^T$ by only taking the first s columns (do you see it?! check out the comment following the example below). That is, by choosing s small compared to r, we can store A using much less data.

Comment. This approximation will be good if the omitted singular values $\sigma_{s+1},\sigma_{s+2},...,\sigma_r$ are all "small".

In other words. Here is another common way to say the same thing:

- Observe that $A=U\Sigma V^T$ is equivalent to $A=\sum_{i=1}^{r}\sigma_i \boldsymbol{u}_i\boldsymbol{v}_i^T$.
- Each matrix $\boldsymbol{u}_i\boldsymbol{v}_i^T$ has rank 1.
- The best rank s approximation to A is $B=\sum_{i=1}^{s}\sigma_i \boldsymbol{u}_i\boldsymbol{v}_i^T$.

Example 129. Consider $A=\begin{bmatrix}1&1&1\\1&0&-1\end{bmatrix}$. What is the best approximation to A using a 2×3 matrix with rank 1?

Solution. In the practice problems for the midterm, we determined that A has the SVD

$$A=\begin{bmatrix}1&0\\0&-1\end{bmatrix}\begin{bmatrix}\sqrt{3}&0&0\\0&\sqrt{2}&0\end{bmatrix}\begin{bmatrix}1/\sqrt{3}&-1/\sqrt{2}&1/\sqrt{6}\\1/\sqrt{3}&0&-2/\sqrt{6}\\1/\sqrt{3}&1/\sqrt{2}&1/\sqrt{6}\end{bmatrix}^T=\begin{bmatrix}1&1&1\\1&0&-1\end{bmatrix}$$

Hence, the best approximation to A using a 2×3 matrix with rank 1 (that is, we keep 1 singular value) is

$$\begin{bmatrix}1&0\\0&-1\end{bmatrix}\begin{bmatrix}\sqrt{3}&0&0\\0&0&0\end{bmatrix}\begin{bmatrix}1/\sqrt{3}&-1/\sqrt{2}&1/\sqrt{6}\\1/\sqrt{3}&0&-2/\sqrt{6}\\1/\sqrt{3}&1/\sqrt{2}&1/\sqrt{6}\end{bmatrix}^T=\begin{bmatrix}1&1&1\\0&0&0\end{bmatrix}.$$

Comment. Note how we can throw away all but 1 columns of U and V:

$$\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}\sqrt{3}\end{bmatrix}\begin{bmatrix}1/\sqrt{3}\\1/\sqrt{3}\\1/\sqrt{3}\end{bmatrix}^T=\begin{bmatrix}1&1&1\\0&0&0\end{bmatrix}.$$

Armin Straub
straub@southalabama.edu

Example 130. (image compression) Let us load a 341x512 grayscale photo and store it as a matrix A. Each entry of the matrix is a value between 0 (black) and 1 (white).

The beautiful picture is taken from: http://www.southalabama.edu/departments/publicrelations/brand/photography.html

[The same approach works with color pictures. These are often represented by three matrices: one for the red component of the pixel, one for the green and for the blue component (RGB color scheme).]

```
Sage] import pylab
```

```
Sage] A = matrix(pylab.imread('/home/armin/photo.png'))
```

```
Sage] A.dimensions()
```

$(341, 512)$

```
Sage] A[0,0]
```

0.137254908681

```
Sage] matrix_plot(A, cmap='gray')
```

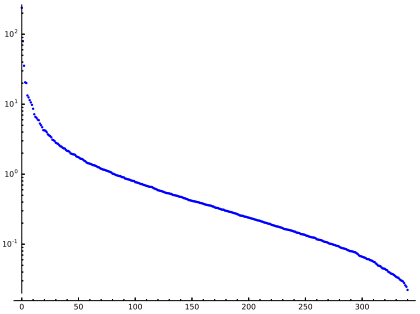
Next, we compute the SVD of A. Despite the size of A that takes the computer only a fraction of a second:

```
Sage] U,S,V = A.SVD()
```

```
Sage] S.diagonal()[:6]
```

$[238.443435709, 79.4429775448, 35.4540786319, 20.5662302846, 20.0697710337, 13.3421216529]$

```
Sage] list_plot(S.diagonal(), scale='semilogy')
```



As we can see, the magnitude of the singular values drops off quickly. We get a good approximation to A (our original photo) by computing a best rank s approximation to A by computing $U_s \Sigma_s V_s$ where Σ_s is the $s \times s$ diagonal matrix with entries $\sigma_1, \sigma_2, ..., \sigma_s$ and U_s, V_s are obtained from the corresponding matrices in the SVD $A = U \Sigma V^T$ by only taking the first s columns.

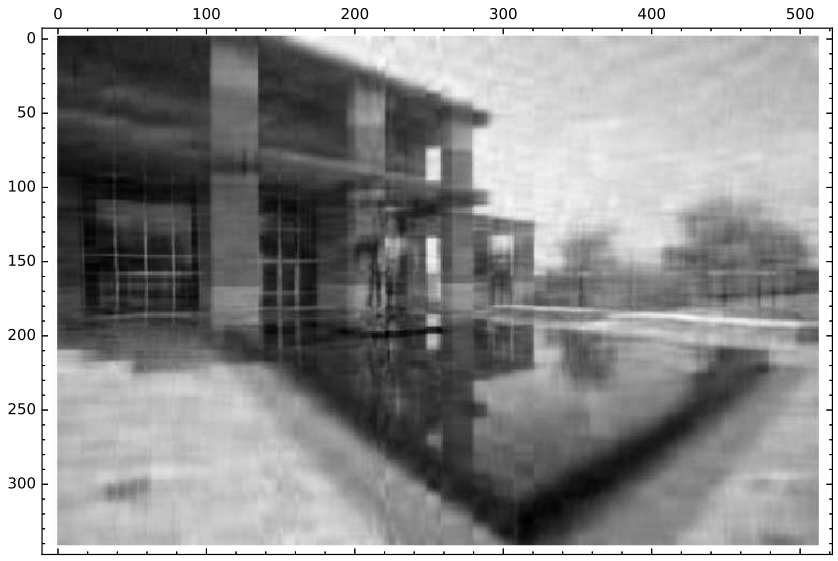
```
Sage] def A_approx(s):
          U0 = U.matrix_from_columns([0..s-1])
          S0 = diagonal_matrix(S.diagonal()[:s])
          V0 = V.matrix_from_columns([0..s-1])
          return U0*S0*V0.transpose()
```

Taking only 100 of the 341 singular values, we get an approximation, which is almost as good as the original:

```
Sage] matrix_plot(A_approx(100), cmap='gray')
```

But notice the development of artifacts. Taking only 20 of the 341 singular values, a lot is lost:

```
Sage] matrix_plot(A_approx(20), cmap='gray')
```



Comment. Image compression is just one (nice visual) example of the power of SVD. A variation of this approach can, for instance, also be used for image denoising. Much more generally, the SVD is able to extract the most important features of any sort of data!