9.4 Singular Value Decomposition

The Singular Value Decomposition (SVD) is one of the most important matrix decompositions used in computer vision. As you might recall from your undergraduate studies a square matrix $\mathbf{A}$ can be diagonalized where the diagonal entries are the eigenvalues of $\mathbf{A}$. This can be written as

$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^T, \qquad (9.16)$$

where $\mathbf{V}$ is an orthogonal matrix (i.e. a matrix with all comumn (or row) vectors having norm 1 and spanning a basis (are orthogonal to each other)), and $\mathbf{D}$ is a diagonal matrix whose diagonal entries $(d_1, \ldots, d_n)$ are the eigenvalues of $\mathbf{A}$.

The most important issue here is that an eigenvalue decomposition just exists for a *square* matrix, i.e. $\mathbf{A} \in \mathbb{R}^{n \times n}$! However, we want to know whether a similar decomposition exists for non-square matrices, i.e. if $\mathbf{A} \in \mathbb{R}^{m \times n}$. And it indeed does! This is the SVD.

In the following we will give a formal definition for the SVD, explain it more informally (but hopefully intuitively), enumerate its properties, and finally explain how and why it can be used for linear optimization, hence parameter estimation in computer vision.

9.4.1 Definition of the SVD

For any given matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ there exists a decomposition

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T,$$

such that

- U is an $m \times n$ matrix with orthogonal columns

- D is a $n \times n$ diagonal matrix with non-negative entries

- V^T is an $n \times n$ orthogonal matrix

We can visualize this decomposition:

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} = \begin{bmatrix} & & \\ & U & \\ & & \end{bmatrix} \begin{bmatrix} & & \\ & D & \\ & & \end{bmatrix} \begin{bmatrix} & & \\ & V^T & \\ & & \end{bmatrix}$$

The diagonal values of D are called *Singular Values* of $\mathbf{A}$.
The column vectors of U are the *Left Singular Vectors* of $\mathbf{A}$.
The column vectors of V are the *Right Singular Vectors* of $\mathbf{A}$.

9.4.2 Properties of the SVD

The SVD can be performed on matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$, where $m \geq n$[3]. In the case that $m = n$ there will be only non-zero positive diagonal elements. In the case that $m > n$ $d_1, \ldots, d_n$ are non-zero positive, $d_{n+1}, \ldots, d_m$ are zero. The SVD can be performed such that the diagonal values of D are descending i.e. $d_1 \geq d_2 \geq \ldots \geq d_n \geq 0$.
We will assume that the SVD is always performed in that way.

The diagonal values of D are the square roots of the eigenvalues of $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ (hence the non-negativity of the elements of $\mathbf{D}$). Why is that so? Have a look at the SVD of $\mathbf{A}^T\mathbf{A}$:

$$
\begin{aligned}
\mathbf{A}^T\mathbf{A} \quad &= \\
(\mathbf{U}\mathbf{D}\mathbf{V}^T)^T\mathbf{U}\mathbf{D}\mathbf{V}^T \quad &= \\
\mathbf{V}\mathbf{D}\mathbf{U}^T\mathbf{U}\mathbf{D}\mathbf{V}^T \quad &\overset{orth.columns}{=} \\
\mathbf{V}\mathbf{D}\mathbf{D}\mathbf{V}^T \quad &= \\
\mathbf{V}\mathbf{D}^2\mathbf{V}^T \quad &
\end{aligned}
$$

It holds for the left singular vectors $\mathbf{u}_i$:

$$\mathbf{A}^T\mathbf{A}\mathbf{u}_i = d_i^2\mathbf{u}_i$$

It holds for the right singular vectors $\mathbf{v}_i$:

$$\mathbf{A}\mathbf{A}^T\mathbf{v}_i = d_i^2\mathbf{v}_i$$

It holds:

$$\mathbf{A}\mathbf{u}_i = d_i\mathbf{v}_i \text{ and } \mathbf{A}^T\mathbf{v}_i = d_i\mathbf{u}_i$$

The left singular vectors $\mathbf{u}_i$ are eigenvectors of $\mathbf{A}^T\mathbf{A}$
The right singular vectors $\mathbf{v}_i$ are eigenvectors of $\mathbf{A}\mathbf{A}^T$

For those who want to know even more properties of the SVD we will shortly introduce them. These are, however, not necessarily relevant to the application in 3D Computer Vision and thus not explained in detail.

The SVD explicitly constructs orthonormal bases for the null-space and the range of a matrix.
The columns of $\mathbf{U}$ corresponding to non-zero elements of $\mathbf{D}$ span the range.
The columns of $\mathbf{V}$ corresponding to zero elements of $\mathbf{D}$ span the null-space.
The SVD allows a rank decision:

---

[3]It can also be performed if $m < n$, but this is not interesting in the context of 3D Computer Vision

$$rank(\mathbf{A}) \text{ is the largest } r \text{ s.t. } d_r > 0.$$

There are $m - r$ left singular vectors corresponding to the singular value 0.
There are $n - r$ right singular vectors corresponding to the singular value 0.

9.4.3 SVD for Optimization

We can use the SVD to solve a system of linear equations $\mathbf{Ax} = \mathbf{b}$. This is equivalent with minimizing the squared norm $||\mathbf{Ax} - \mathbf{b}||^2$, which is a linear least-squares optimization problem.

The SVD of A gives us $svd(\mathbf{A}) = \mathbf{UDV}^T$, where $\mathbf{U} \in \mathbb{R}^{n \times m}$ with its columns being orthogonal, i.e. $\mathbf{U}^T\mathbf{U} = \mathbf{I}_{n \times n}$, $\mathbf{D}$ is a diagonal matrix with positive or zero elements on its diagonal, and $\mathbf{V}$ is an orthogonal matrix, i.e. $\mathbf{V}^T\mathbf{V} = \mathbf{VV}^T = \mathbf{I}_{n \times n}$

An orthogonal matrix $\mathbf{M}$ has a norm-preserving property, i.e. for any vector $\mathbf{v}$

$$||\mathbf{Mv}|| = ||\mathbf{v}|| \tag{9.17}$$

holds [4], p.94.
Putting it all together we have

$$\begin{aligned} min(||\mathbf{Ax} - \mathbf{b}||^2) &\Leftrightarrow \\ min(||\mathbf{Ax} - \mathbf{b}||) &= \\ min(||\mathbf{UDV}^T\mathbf{x} - \mathbf{b}||) &\overset{9.17}{=} \\ min(||\mathbf{DV}^T\mathbf{x} - \mathbf{U}^T\mathbf{b}||). \end{aligned}$$

Substituting $\mathbf{y} = \mathbf{V}^T\mathbf{x}$ and $\mathbf{b}' = \mathbf{U}^T\mathbf{b}$ gives us $\mathbf{Dy} = \mathbf{b}'$ with $\mathbf{D}$ a diagonal matrix:

$$\left(\begin{array}{cccc} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \\ \hline & & 0 & \end{array} \right) \left(\begin{array}{c} y_1 \\ y_2 \\ \vdots \\ y_n \end{array} \right) = \left(\begin{array}{c} b'_1 \\ b'_2 \\ \vdots \\ b'_n \\ \hline b'_{n+1} \\ \vdots \\ b'_m \end{array} \right)$$

Now, the solution to this system of linear equations is simply $y_i = b'_i/d_i$[4].
With $\mathbf{y}$ we can easily get $\mathbf{x} = \mathbf{Vy}$.

9.4.3.1 Solving the Homogeneous Case

Sometimes the system of linear equations is not $\mathbf{Ax} = \mathbf{b}$, but $\mathbf{Ax} = \mathbf{0}$, i.e. $\mathbf{b} = \mathbf{0}$. Thus, our minimization problem is not $||\mathbf{Ax} - \mathbf{b}||$ but $||\mathbf{Ax}||$. Here we have to pay attention that we do not use the trivial solution, i.e. $x_i = 0$, so we have to impose a constraint on $\mathbf{x}$ to avoid

---

[4]for $d_i \neq 0$ the rank of matrix $\mathbf{A}$ has to be at least n.

it to become **0** and our problem can be stated as

Find **x** *that minimizes* $||\mathbf{Ax}||^2$ *subject to* $||\mathbf{x}||^2 = 1$[5], or

Find **x** *that minimizes* $||\mathbf{Ax}||$ *subject to* $||\mathbf{x}|| = 1$.

Again, we are using the norm preserving property of orthogonal matrices and the SVD - $||\mathbf{Ax}|| = ||\mathbf{UDV}^T\mathbf{x}|| = ||\mathbf{DV}^T\mathbf{x}||$ and $||\mathbf{x}|| = ||\mathbf{V}^T\mathbf{x}||$.

Now, we substitute $\mathbf{y} = \mathbf{V}^T\mathbf{x}$, thus our problem becomes: minimize $||\mathbf{Dy}||$ subject to $||\mathbf{y}|| = 1$; D is a diagonal matrix with its entries in descending order, so the minimal solution is $\mathbf{y} = (0, 0, \ldots, 1)^T$, and since $\mathbf{x} = \mathbf{Vy}$, **x** is the last column of **V**.

You can see, it is rather easy to solve a linear least squares problem once reformulated as $\mathbf{Ax} = \mathbf{b}$, or $\mathbf{Ax} = \mathbf{0}$.

9.4.4 SVD Computation

The only thing left is to show how an SVD of a matrix A can be computed; but we won't do that - we will use SVD as a black box. The original algorithm has been implemented by Golub and Reinsch, for an C-implementation of this algorithm have a look at [5].
The computational complexity is important however. First we want to mention that the algorithm is extremely stable. The computation time for an SVD of a $m \times n$ matrix **A** is:

- Computation of U, V and D: $4m^2n + 8mn^2 + 9n^3$

- Computation of V and D: $4mn^2 + 8n^3$

Keep in mind that in most of our cases $m >> n$.

Implementation in Matlab   Matlab has the command

```
[U,S,V] = svd(X)
```

Attention: The command returns **V** and not $\mathbf{V}^T$, hence it holds that X = U*S*V'

---

[5]to avoid the trivial solution