Statistics, Data Analysis, and Machine Learning for Physicists
Timothy Brandt
Spring 2020

Lecture 7

The subject of today's lecture will be the singular value decomposition (SVD) from linear algebra. This turns out to be an astonishingly useful tool for two reasons:

- It minimizes χ^2 for a linear model even when two or more parameters are degenerate and the system of linear equations formed by differentiating χ^2 gives a singular matrix; and
- It gives the principal components and weights in principle component analysis (PCA).

For both of these reasons it is important to cover the SVD. You won't need to remember all of the underlying theory but we'll go over it here so that you have it for reference. You will generally use the SVD through three python functions:

- `numpy.linalg.lstsq` for the SVD least-squares solution to a system of equation (e.g. minimizing χ^2).
- `numpy.linalg.svd` the SVD itself, which gives the principal components of PCA.
- `numpy.linalg.pinv` the Moore-Penrose pseudo-inverse of a matrix, which has the benefits of always existing and often being useful. The pseudo-inverse may be obtained immediately from the SVD.

The SVD is a factoring of an $m \times n$ matrix $\mathbf{A}$. The matrix $\mathbf{A}$ itself multiples an n-dimensional vector $\mathbf{x}$ to give an m-dimensional vector $\mathbf{b}$, so $\mathbf{A}$: $\mathbb{R}^n \to \mathbb{R}^m$. The SVD factors $\mathbf{A}$ into three components: an $m \times m$ orthonormal matrix $\mathbf{U}$, an $m \times n$ diagonal matrix $\mathbf{W}$ with positive or zero elements on the diagonal, and the transpose of an $n \times n$ orthonormal matrix $\mathbf{V}$:

$$\mathbf{A} = \mathbf{U}\mathbf{W}\mathbf{V}^T. \tag{1}$$

The orthonormality of $\mathbf{U}$ and $\mathbf{V}^T$ mean that

$$\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I} \tag{2}$$

and

$$\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}. \tag{3}$$

The matrix $\mathbf{W}$ is diagonal but not necessarily square, so the most nonzero elements it can have is the lesser of m and n. There are several remarkable facts that I will state but not prove:

- This factorization is possible for any matrix.
- The factorization is unique up to a few conditions, e.g. permutations of the rows and columns of the matrices, and rotations when two or more singular values are equal. The singular values themselves are unique, and it is conventional to sort $\mathbf{U}$, $\mathbf{W}$, and $\mathbf{V}^T$ so that the singular values are in decreasing order along the diagonal.
- The columns i of $\mathbf{U}$ corresponding to nonzero w_i are an orthonormal basis for the *range* of $\mathbf{A}$, the space of vectors that can be written as the product of $\mathbf{A}$ and another vector. This is the basis for the use of SVD in PCA. The number of nonzero w_{ii} (equivalently, the dimension of the range) is the *rank* of $\mathbf{A}$. The rank can be at most the lesser of m and n.
- The columns j of $\mathbf{U}$ corresponding to $w_j = 0$ (plus those with $j > n$) are an orthonormal basis for the subspace of the full possible m-dimensional space that $\mathbf{A}$ cannot reach.

- The columns j of $\mathbf{V}$ for which $w_j = 0$ form an orthonormal basis of the null space, the space of vectors $\mathbf{x}$ for which $\mathbf{Ax} = \mathbf{0}$.

The SVD is useful in solving a system of equations given by the matrix equation

$$\mathbf{Ax} = \mathbf{b} \tag{4}$$

since I can use the SVD of $\mathbf{A}$ and the unitarity of $\mathbf{U}$ and $\mathbf{V}^T$ to write

$$\mathbf{UWV}^T\mathbf{x} = \mathbf{b} \tag{5}$$

$$\mathbf{x} = \mathbf{VW}^{-1}\mathbf{U}^T\mathbf{b}. \tag{6}$$

The second equation follows from the first only if the inverse of $\mathbf{W}$ exists. We'll now explore the usefulness of Equation (6) (and what exactly $\mathbf{W}^{-1}$ means) in the general case where $\mathbf{W}$ might not be invertible. Since $\mathbf{W}$ is diagonal, its inverse is trivial unless there are zeros along the diagonal, or unless it isn't square. In this latter case there are two possibilities: you are trying to make a square matrix out of the larger of the two dimensions of $\mathbf{W}$, in which you cannot possibly get the identity matrix, or you are trying to make a square matrix out of the smaller of the two dimensions of $\mathbf{W}$, in which case you can get the identity unless one or more $w_j = 0$ as before.

The first nice thing about SVD is that it immediately tells you if there is a problem with matrix inversion (assuming an inverse isn't precluded by dimensional considerations). If all of the matrices are square and there are singular values (zeros along the diagonal of $\mathbf{W}$), $\mathbf{A}$ is not invertible and the set of equations has no solution. If $\mathbf{A}$ is not singular, its *condition number* is the ratio of its largest to its smallest singular values. If this condition number approaches 10^{15} (for double-precision floating point arithmetic), then roundoff error makes it nearly impossible to numerically invert $\mathbf{A}$ and simple methods to solve the system of equations will fail, either by returning a meaningless result or by throwing a singular matrix error. In practice, singular values smaller than about 10^{-14} or 10^{-15} of the maximum singular value are unreliable due to roundoff error and are set to zero.

What about the case where the matrix $\mathbf{W}$ is singular (it does have zeros along the diagonal), or where $\mathbf{W}$ is not square? In that case $\mathbf{A}$ is not invertible, and Equation (4) either has no solution or (less commonly) many solutions. I can still apply Equation (6) to get *some* solution, as long as I decide what to do about the inverses of the singular values $1/w_j$. Now, there is a remarkable fact: if I set $1/w_j = 0$ for $w_j = 0$ (in other words, $1/0 = 0$!), I get the "solution" vector $\mathbf{x}$ that minimizes the Frobenius norm of Equation (4), or the square root of the sum of the squares of the residual:

$$\big|\big|\mathbf{Ax} - \mathbf{b}\big|\big|_F. \tag{7}$$

While this $\mathbf{x}$ is not, in general, a solution to the equation, it is the best solution in the least squares sense. *Note that finding this solution is the exact goal of χ^2 minimization.*

To prove this, let's plug in the solution given by Equation (6) and compute the residual $\mathbf{Ax} - \mathbf{b}$:

$$\mathbf{R} = \mathbf{AVW}^{-1}\mathbf{U}^T\mathbf{b} - \mathbf{b} \tag{8}$$

$$= \mathbf{UWV}^T\mathbf{VW}^{-1}\mathbf{U}^T\mathbf{b} - \mathbf{b} \tag{9}$$

$$= \mathbf{UWW}^{-1}\mathbf{U}^T\mathbf{b} - \mathbf{b}. \tag{10}$$

Where I have written $\mathbf{W}^{-1}$, I really mean the matrix with $1/w_i$ on the diagonal where $w_i \neq 0$ and zero on the diagonal otherwise (it may or may not be the actual inverse of $\mathbf{W}$). If I add an arbitrary vector $\mathbf{x}'$ to my solution, it is equivalent to adding $\mathbf{b}' = \mathbf{Ax}'$ to the vector $\mathbf{b}$, and my residual vector $\mathbf{R}$ becomes

$$\mathbf{R}' = \mathbf{R} + \mathbf{b}'. \tag{11}$$

Now I want to show that

$$\big|\big|\mathbf{R}'\big|\big|_F \geq \big|\big|\mathbf{R}\big|\big|_F \tag{12}$$

for any $\mathbf{x}'$. Using the fact that $\mathbf{U}$ is unitary and then taking the Frobenius norm, I have

$$\left|\left|\mathbf{R}'\right|\right|_F = \left|\left|\mathbf{U}\mathbf{W}\mathbf{W}^{-1}\mathbf{U}^T\mathbf{b} - \mathbf{b} + \mathbf{b}'\right|\right|_F \tag{13}$$

$$= \left|\left|\mathbf{U}\mathbf{U}^T\left(\mathbf{U}\mathbf{W}\mathbf{W}^{-1}\mathbf{U}^T\mathbf{b} - \mathbf{b} + \mathbf{b}'\right)\right|\right|_F \tag{14}$$

$$= \left|\left|\mathbf{U}\left(\mathbf{U}^T\mathbf{U}\mathbf{W}\mathbf{W}^{-1}\mathbf{U}^T\mathbf{b} - \mathbf{U}^T\mathbf{b} + \mathbf{U}^T\mathbf{b}'\right)\right|\right|_F. \tag{15}$$

Now, I use the fact that for a unitary matrix $\mathbf{U}$ and an arbitrary matrix $\mathbf{A}$,

$$\left|\left|\mathbf{U}\mathbf{A}\right|\right|_F = \sqrt{\text{Trace}\left(\left(\mathbf{U}\mathbf{A}\right)^T\mathbf{U}\mathbf{A}\right)} \tag{16}$$

$$= \sqrt{\text{Trace}\left(\mathbf{A}^T\mathbf{U}^T\mathbf{U}\mathbf{A}\right)} \tag{17}$$

$$= \sqrt{\text{Trace}\left(\mathbf{A}^T\mathbf{A}\right)} \tag{18}$$

$$= \left|\left|\mathbf{A}\right|\right|_F \tag{19}$$

to factor out the unitary matrix $\mathbf{U}$ and write

$$\left|\left|\mathbf{R}'\right|\right|_F = \left|\left|\mathbf{W}\mathbf{W}^{-1}\mathbf{U}^T\mathbf{b} - \mathbf{U}^T\mathbf{b} + \mathbf{U}^T\mathbf{b}'\right|\right|_F \tag{20}$$

$$\left|\left|\mathbf{R}'\right|\right|_F = \left|\left|\left(\mathbf{W}\mathbf{W}^{-1} - \mathbf{I}\right)\mathbf{U}^T\mathbf{b} + \mathbf{U}^T\mathbf{b}'\right|\right|_F \tag{21}$$

The first term in $\mathbf{R}'$ is zero everywhere the diagonal of $\mathbf{W}$ does not vanish. This means that in the product $\mathbf{U}^T\mathbf{b}$, all of the entries j corresponding to $w_j \neq 0$ are set equal to zero when we multiply by $\mathbf{W}\mathbf{W}^{-1}$. In the second term, element j corresponds to the product of column j of $\mathbf{U}$ (row j of $\mathbf{U}^T$) with the vector $\mathbf{b}'$, and $\mathbf{b}'$ is in the range of the original matrix $\mathbf{A}$. So, the product of $\mathbf{b}'$ with a row of $\mathbf{U}^T$ in the range of $\mathbf{A}$ might be nonzero; this would correspond to a j for which $w_j \neq 0$. Everywhere $w_j = 0$ corresponds to a row of $\mathbf{U}^T$ in the null space of $\mathbf{A}$, and therefore that entry in the second term is zero. As an alternative way of thinking about this, the vector $\mathbf{b}'$ can be expressed as a linear combination of the rows of $\mathbf{U}^T$ for which $w_j \neq 0$ (those in the range of $\mathbf{A}$); it is orthogonal to all of the other rows of $\mathbf{U}^T$. The first term in Equation (21), then, can only have nonzero elements where $w_j = 0$, and the second term can only have nonzero elements where $w_j \neq 0$.

We'll look at this one other way before completing the proof. The quantity $\mathbf{U}^T\mathbf{b}$ forms a vector of length m (recall that the original matrix $\mathbf{A}$ is $m \times n$). The product $\mathbf{W}\mathbf{W}^{-1}$ is an $m \times m$ that is almost the identity. For illustration, I will assume $m > n$ (the result is the same otherwise) and that the rank of $\mathbf{A}$ is k, with $k \leq n$. Note that the nonzero diagonal terms could go all the way to the end of the matrix in one of the dimensions.

$$\mathbf{W}\mathbf{W}^{-1} = \begin{bmatrix} w_{11} & 0 & 0 & 0 & \dots \\ 0 & \ddots & 0 & 0 & \dots \\ 0 & 0 & w_{kk} & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} 1/w_{11} & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & \ddots & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 1/w_{kk} & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots \\ 0 & \ddots & 0 & 0 & \dots \\ 0 & 0 & 1 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{22}$$

I have also assumed that the matrix $\mathbf{W}$ is arranged with the w_{ii} in decreasing order along the diagonal. This is conventional, and I am actually free to rearrange them however I like (this part of SVD is not unique). The product is $\mathbf{W}\mathbf{W}^{-1}$ is $m \times m$ with $k \leq \min(m, n)$ ones along the diagonal and the rest zeros, where k is the rank of $\mathbf{A}$. So, the matrix $\mathbf{W}\mathbf{W}^{-1} - \mathbf{I}$ looks like

$$\mathbf{W}\mathbf{W}^{-1} - \mathbf{I} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & \ddots & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & -1 & 0 & \dots \\ 0 & 0 & 0 & 0 & -1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{23}$$

Now, when I multiply $\mathbf{W}\mathbf{W}^{-1} - \mathbf{I}$ by any vector of length m, the first k elements are guaranteed to be zero:

$$\left(\mathbf{W}\mathbf{W}^{-1} - \mathbf{I}\right)\mathbf{U}^T\mathbf{b} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ c_{k+1} \\ \vdots \\ c_m \end{bmatrix}. \tag{24}$$

The second term in Equation (21) is $\mathbf{U}^T\mathbf{b}'$, with $\mathbf{b}'$ in the range of $\mathbf{A}$. Remember that, by the definition of the SVD, the columns of the matrix $\mathbf{U}$ (and therefore the rows of $\mathbf{U}^T$) are in the range of $\mathbf{A}$ for $i \leq k$, with k being the rank of $\mathbf{A}$. The product of any one of these rows with $\mathbf{b}'$ might be nonzero. However, if I multiply $\mathbf{b}'$ by one of the other rows of $\mathbf{U}^T$, I am guaranteed to get zero. The vector $\mathbf{b}'$ is a linear combination of the first k rows of $\mathbf{U}^T$, and each of the subsequent rows $k+1, \ldots, m$ is perpendicular to each of the first k since $\mathbf{U}$ (and $\mathbf{U}^T$) are orthogonal matrices. So, $\mathbf{U}^T\mathbf{b}'$ looks something like

$$\mathbf{U}^T\mathbf{b}' = \begin{bmatrix} c_1 \\ \vdots \\ c_k \\ 0 \\ \vdots \end{bmatrix}. \tag{25}$$

The vectors in equations (24) and (25) are orthogonal: their dot product is zero because every element $1, \ldots, m$ is zero in at least one of the two vectors. The orthogonality of these two vectors lets me write

$$\left|\left|\mathbf{R}'\right|\right|_F = \left|\left|\mathbf{R}\right|\right|_F + \left|\left|\mathbf{U}^T\mathbf{b}'\right|\right|_F \geq \left|\left|\mathbf{R}\right|\right|_F, \tag{26}$$

thus completing the proof. You can also see directly from Equation (24) that making one or more of the first k elements nonzero (which is all $\mathbf{U}^T\mathbf{b}'$ can do according to Equation (25)) will only increase the norm of the residual. This proof also works for a system of equations with more equations than unknowns–notice that I didn't assume square matrices anywhere. In that case, I usually can't find an exact solution, but I can still use SVD to find the best solution in the least-squares sense. Here, then, is how you can recast the χ^2 problem for independent measurements so that you can solve it with SVD.

Remember that in a linear χ^2 model you are trying to minimize

$$\chi^2 = \sum_{i=1}^{n} \frac{(y_i - \sum_{j=1}^{m} \alpha_j x_{ij})^2}{\sigma_i^2} \tag{27}$$

where the α_j are the coefficients of your linear model and the x_{ij} are the model parameters for each measurement y_i. If I have m model parameters ($\boldsymbol{\alpha}$ is an array of length m) and n measurements, $n > m$, then $\mathbf{y}$ is an array of length n). In that case I can rewrite χ^2 as

$$\chi^2 = \sum_{i=1}^{n} \left(\frac{y_i}{\sigma_i} - \left(\sum_{j=1}^{m} \alpha_j \frac{x_{ij}}{\sigma_i} \right) \right)^2 \tag{28}$$

$$= \left(\left|\left|\mathbf{y}' - \mathbf{X}\boldsymbol{\alpha}\right|\right|_F\right)^2 \tag{29}$$

where

$$y_i' = \frac{y_i}{\sigma_i} \tag{30}$$

and

$$X_{ij} = \frac{x_{ij}}{\sigma_i}. \tag{31}$$

You then call SVD on $\mathbf{X}$ and use the SVD to obtain the best-fit parameters $\boldsymbol{\alpha}$. Or, you can use the pre-packaged python routine `numpy.linalg.lstsq`, which does all of this for you. You just form the matrix $\mathbf{X}$ and the array $\mathbf{y}'$ and then call

$$\boldsymbol{\alpha} = \texttt{numpy.linalg.lstsq}(\mathbf{X}, \mathbf{y}')[0] \tag{32}$$

where the use of the index 0 at the end is because `numpy.linalg.lstsq` returns more than just the best-fit coefficients (it also returns the residuals, the rank of $\mathbf{X}$, and the singular values).

Just for reference, if you did want to use the SVD itself to compute the least-squares solution $\boldsymbol{\alpha}$, it might look something like this, after first forming $\mathbf{X}$ and $\mathbf{y}'$:

```
U, W, VT = numpy.linalg.svd(X)
Winv = 1/W
threshold = 1e-14
Winv[np.where(W < threshold*np.amax(W))] = 0
alpha = numpy.multi_dot(VT.T, Winv, U.T, yp)
```

It really isn't that bad. (though for readability I strongly suggest using `numpy.linalg.lstsq`). As one last aside, the matrix $\mathbf{V}\mathbf{W}^{-1}\mathbf{U}^T$ with $1/w_i$ set to zero for $w_i = 0$ has a special name: it is called the *Moore-Penrose psuedoinverse*. So, yet another way of calculating the least-squares solution with numpy would be:

```
Xinv = numpy.linalg.pinv(X)
alpha = numpy.dot(Xinv, yp)
```

Again, readability (and possibly speed) favors the use of `numpy.linalg.lstsq`, but it's good to know that all of these routines are basically wrappers for the same matrix decomposition.

Numerical Recipes discusses the SVD in Chapter 2 Section 6, and states that $\mathbf{U}$ is $m \times n$ and that $\mathbf{W}$ is $n \times n$ square. You can verify that the product of $\mathbf{U}$ and $\mathbf{W}$, and the products of $\mathbf{U}^T$ and $\mathbf{W}^{-1}$, are the same under this definition as under the canonical definition of SVD. The extra columns of $\mathbf{U}$/rows of $\mathbf{U}^T$ all multiply out to zeros, and the extra elements of $\mathbf{W}$ and $\mathbf{W}^{-1}$ are zeros. If $m \gg n$, truncating the SVD in this way can save a significant amount of memory and computational cost and includes all of the necessary information for a least-squares solution and for PCA.

One final note about SVD and confidence intervals: we have discussed confidence intervals a lot, and SVD actually gives you the full covariance matrix for the fitted parameters. If you recast your χ^2 problem as

$$\chi^2 = \sum_{i=1}^{n} \left(\frac{y_i}{\sigma_i} - \left(\sum_{j=1}^{m} \alpha_j \frac{x_{ij}}{\sigma_i} \right) \right)^2 \tag{33}$$

$$= \left(\left|\left| \mathbf{y}' - \mathbf{X}\boldsymbol{\alpha} \right|\right|_F \right)^2, \tag{34}$$

then performing SVD on $\mathbf{X}$ will give arrays $\mathbf{U}$, $\mathbf{W}$, and $\mathbf{V}^T$. It turns out that if all of the Gaussianity assumptions are satisfied then the standard error on parameter j is given by

$$\sigma^2(\alpha_j) = \sum_{i=1}^{m} \left(\frac{V_{ji}}{w_i} \right)^2, \tag{35}$$

i.e., the weighted sum of the columns i of $\mathbf{V}$ with the weights being the inverses of the diagonal elements of $\mathbf{W}$. The *covariance* between two parameters α_j and α_k is given by

$$\mathrm{Cov}(\alpha_j, \alpha_k) = \sum_{i=1}^{m} \left(\frac{V_{ji} V_{ki}}{w_i^2} \right). \tag{36}$$

If you want a one-liner for this in python, try

```
Cov = np.sum(V[np.newaxis, :, :]*V[:, np.newaxis, :]*Winv**2, axis=-1)
```

If any of these errors or covariances encounter division by zero or by a number close to zero, it is an indication that either a parameter isn't constraining your fit, or that varying some combination of multiple parameters has a negligible effect on χ^2. If this happens, you should think about recasting your problem to solve this issue. You can still get a sensible result by setting $1/w_i = 0$ if $w_i = 0$: this gives zero error for a quantity that is not constrained in the optimization.