Singular Value Decomposition, Generalised Inverse, Principal Components

Consider a real $m \times n$ matrix $\boldsymbol{A}$, with $m \geq n$. Its **singular value decomposition** (SVD) takes the form

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top,$$

where $\boldsymbol{U}$ is $m \times n$, $\boldsymbol{D}$ is $n \times n$, and $\boldsymbol{V}$ is $n \times n$. Further, $\boldsymbol{U}$ has orthonormal columns, which means that $\boldsymbol{U}^\top\boldsymbol{U} = \mathbf{I}_n$, and $\boldsymbol{U}\boldsymbol{U}^\top$ is symmetric and idempotent, that is, an orthogonal projection matrix; $\boldsymbol{D}$ is diagonal, with non-negative diagonal elements arranged in decreasing order down the main diagonal; $\boldsymbol{V}$ is an orthogonal matrix, so that $\boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^\top = \mathbf{I}_n$, and $\boldsymbol{V}^\top = \boldsymbol{V}^{-1}$. The SVD always exists if $\boldsymbol{A}$ is real. The columns of $\boldsymbol{U}$ are the **left singular vectors**, those of $\boldsymbol{V}$ are the **right singular vectors**, and the elements of $\boldsymbol{D}$ are the **singular values**.

The elements of $\boldsymbol{D}$ are the square roots of the eigenvalues of $\boldsymbol{A}^\top\boldsymbol{A}$, and the right singular vectors are the eigenvectors:

$$\boldsymbol{A}^\top\boldsymbol{A} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top = \boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}^\top,$$

whence

$$(\boldsymbol{A}^\top\boldsymbol{A})\boldsymbol{V} = \boldsymbol{V}\boldsymbol{D}^2.$$

Similarly, the squares of the elements of $\boldsymbol{D}$ are some of the eigenvalues of $\boldsymbol{A}\boldsymbol{A}^\top$, with the left singular vectors as the corresponding eigenvectors:

$$(\boldsymbol{A}\boldsymbol{A}^\top)\boldsymbol{U} = \boldsymbol{U}\boldsymbol{D}^2.$$

The other eigenvalues of $\boldsymbol{A}\boldsymbol{A}^\top$ are all zero. Some of the bottom right diagonal elements of $\boldsymbol{D}$ may also be zero: this happens when $\boldsymbol{A}$ does not have full column rank. In this case, we have this setup:

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{D}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}. \tag{1}$$

If $\boldsymbol{A}$ has full column rank of n, then $\boldsymbol{D}$ is non-singular, and $\boldsymbol{D}^{-1}$ exists. Otherwise, make the definition

$$\boldsymbol{D}^+ = \begin{bmatrix} \boldsymbol{D}_1^{-1} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{bmatrix}.$$

We will see that $\boldsymbol{D}^+$ is a **generalised inverse**.

Any matrix $\boldsymbol{A}^+$ that satisfies the two requirements

$$\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{A}, \quad \text{and} \quad \boldsymbol{A}^+\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}^+$$

is a generalised inverse of $\boldsymbol{A}$. If we define $\boldsymbol{A}^+ = \boldsymbol{V}\boldsymbol{D}^+\boldsymbol{U}^\top$, then we can see at once that this $\boldsymbol{A}^+$ is indeed a generalised inverse. If in addition the matrices

$\boldsymbol{A}\boldsymbol{A}^+$ and $\boldsymbol{A}^+\boldsymbol{A}$ are symmetric, we obtain the **Moore-Penrose** generalised inverse. This property is also satisfied with our definition of $\boldsymbol{A}^+$.

Note that $\boldsymbol{A}\boldsymbol{A}^+$ and $\boldsymbol{A}^+\boldsymbol{A}$ are idempotent, where $\boldsymbol{A}^+$ is *any* generalised inverse of $\boldsymbol{A}$:

$$\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{A}\boldsymbol{A}^+ \quad \text{and} \quad \boldsymbol{A}^+\boldsymbol{A}\boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{A}^+\boldsymbol{A}.$$

This means that these two matrices are projection matrices; with the Moore-Penrose inverse, they are orthogonal projection matrices.

It is easy to check that $\boldsymbol{D}^+$ also satisfies the requirements for it to be the Moore-Penrose generalised inverse of $\boldsymbol{D}$. Of course, if $\boldsymbol{A}$ has full column rank, $\boldsymbol{D}^+ = \boldsymbol{D}^{-1}$.

The aim of **Principal Components Analysis** (PCA) is dimension reduction, with as little loss of information as possible. Think first of $\boldsymbol{A}$ as a matrix of data, where each row is an instance, or example, and each column corresponds to a feature. There are too many features, and the aim is to replace them by a smaller number of linear combinations, while still retaining as much of the information as possible.

The idea is to set equal to zero some of the smallest singular values. Even if $\boldsymbol{D}$ had no zero diagonal elements, this would replace the original $\boldsymbol{D}$ by a matrix like (1), where we suppose that $\boldsymbol{D}_1$ is a $k \times k$ matrix, with $k < n$. Thus causes the last $n - k$ columns of $\boldsymbol{U}$ to have no importance, as also the last $n - k$ rows of $\boldsymbol{V}$. Let

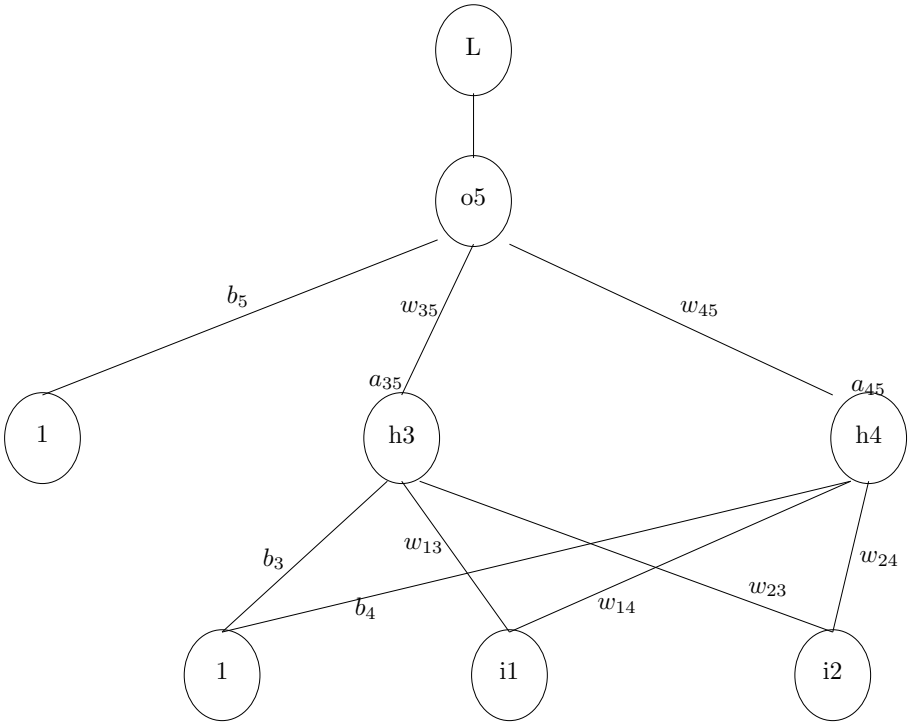
$$\boldsymbol{U} = [\,\boldsymbol{U}_1 \quad \mathbf{O}\,] \quad \text{and} \quad \boldsymbol{V}^\top = \begin{bmatrix} \boldsymbol{V}_1^\top \\ \mathbf{O} \end{bmatrix}.$$

Then $\boldsymbol{B} \equiv \boldsymbol{U}_1\boldsymbol{D}_1\boldsymbol{V}_1^\top$ is an $m \times k$ matrix, with $k < n$.

The name of principal components analysis could in principle be applied to the full matrix $\boldsymbol{U}$, which is of the same dimensions as $\boldsymbol{A}$ itself, with the columns of $\boldsymbol{U}$ being the principal components, or orthonormalised linear combinations of those of $\boldsymbol{A}$, ordered by how much of the total variation of $\boldsymbol{A}$ they account for, but normally the analysis is used to trim away the directions of only small variation.

Back-Propagation

Below is a representation of a feedforward network with an input layer containing two units and a constant (bias), a hidden layer with two units and another constant, an output layer with just one unit, and a loss function that depends on the content of that unit .



The inputs i1 and i2 are fed in, and then, for one iteration, the contents of the other units are computed as follows:

$$\begin{aligned} \text{h3} &= b_3 + w_{13}\text{i1} + w_{23}\text{i2} \\ \text{h4} &= b_4 + w_{14}\text{i1} + w_{24}\text{i2} \\ \text{o5} &= b_5 + w_{35}a_{35}(\text{h3}) + w_{45}a_{45}(\text{h4}) \\ \text{loss} &= L(\text{o5}) \end{aligned} \tag{2}$$

The **parameters** are the biases b_3, b_4, and b_5, and the weights w_{13}, w_{14}, w_{23}, w_{24}, w_{35}, and w_{45}; a_{35} and a_{45} are the **activation functions**.

The feedforward step involves the computations (2), the results of which are stored, as also the values of the activation functions $a_{35}(\text{h3})$ and $a_{45}(\text{h4})$. Also stored are the derivatives $a'_{35}(\text{h3})$, $a'_{45}(\text{h4})$, and $L'(\text{o5})$, which are also stored in the corresponding units.

Next comes back-propagation, which involves computing the partial derivatives of the loss function with respects to all of the parameters. This starts with the parameters that take us from the hidden layer to the output layer, namely b_5, w_{35}, and w_{45}. We have

$$\frac{\partial L}{\partial b_5} = L'(\text{o5})\frac{\partial\,\text{o5}}{\partial b_5} = L'(\text{o5}); \quad \frac{\partial L}{\partial w_{35}} = L'(\text{o5})\frac{\partial\,\text{o5}}{\partial w_{35}} = L'(\text{o5})\,a_{35}(\text{h3}); \quad \text{and}$$
$$\frac{\partial L}{\partial w_{45}} = L'(\text{o5})\frac{\partial\,\text{o5}}{\partial w_{45}} = L'(\text{o5})\,a_{45}(\text{h4})$$

Notice that everything needed for computing these derivatives has been stored. To proceed further, we need the derivatives with respect to h3 and h4. These are

$$\frac{\partial L}{\partial\,\text{h3}} = L'(\text{o5})\,w_{35}\,a'_{35}(\text{h3}) \quad \text{and} \quad \frac{\partial L}{\partial\,\text{h4}} = L'(\text{o5})\,w_{45}\,a'_{45}(\text{h4})$$

But these are just intermediary calculations for the derivatives with respect to the parameters that take us from the input layer to the hidden layer. For these, we have

$$\begin{aligned} \frac{\partial L}{\partial b_3} &= \frac{\partial L}{\partial\,\text{h3}}\,\frac{\partial\,\text{h3}}{\partial b_3} = L'(\text{o5})\,w_{35}\,a'_{35}(\text{h3}); \\ \frac{\partial L}{\partial w_{13}} &= \frac{\partial L}{\partial\,\text{h3}}\,\frac{\partial\,\text{h3}}{\partial w_{13}} = L'(\text{o5})\,w_{35}\,a'_{35}(\text{h3})\ \text{i1} \\ \frac{\partial L}{\partial w_{14}} &= \frac{\partial L}{\partial\,\text{h3}}\,\frac{\partial\,\text{h3}}{\partial w_{14}} = L'(\text{o5})\,w_{35}\,a'_{35}(\text{h3})\ \text{i2} \end{aligned}$$

Again, everything needed was stored during the feedforward step. The derivatives with respect to b_4, w_{14}, and w_{24} are computed in exactly the same way.

We see that what we have done is indeed *back*-propagation. We start with the derivative of the loss function with respect to the outputs of the output layer (here there is only one, o5), and go back down, systematically applying the chain rule, for which everything is available when needed.

What we have computed is the **gradient** of the loss function, and so we can move to the next iteration by updating all the parameters by, for instance, stochastic gradient descent.