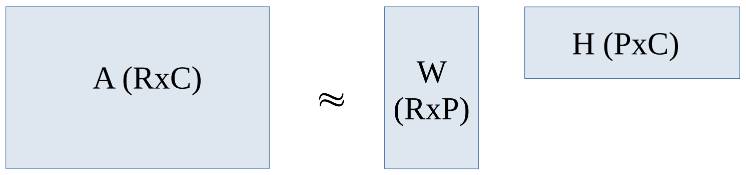
Notes on NMF using projected gradient descent

June 2024 – SDB

We are given a matrix A whose elements are all positive. We want to factor it into the product of two matrices, W, H, whose elements are also all positive. The product of the two matrices might not give exactly A, but we want it to be as close as possible. That is,

$$A \approx W H$$

The idea is that we can make W and H smaller than A so that they will capture some of the structure of A without all the unnecessary details (i.e. in statistical language, we hope that the smaller matrices will capture "latent factor" information from A). That is, we want



As an optimization we can formulate the problem as

$$\underset{w_{mn} \in W, h_{mn} \in H}{\text{argmin}} \; f(W,H) = \frac{1}{2} \sum_{i=1}^{R} \sum_{j=1}^{C} \left(A_{ij} - (W H)_{ij} \right)^2 \quad \text{s.t. } w_{mn} > 0, h_{mn} > 0 \qquad (1)$$

or, written in matrix format

$$\underset{w_{mn} \in W, h_{mn} \in H}{\text{argmin}} \; f(W,H) = \frac{1}{2} \| A - W H \|_{Fro}^2 \quad \text{s.t. } w_{mn} > 0, h_{mn} > 0 \qquad (2)$$

where $\| * \|_{Fro}^2$ denotes the (square of the) Frobenius norm, which is just the sum of the squares of all the elements inside the norm symbol. The lower-case letters w_{mn} and h_{mn} denote the individual elements of the W and H matrices. Therefore, equations (1) or, equivalently (2) define the nonnegative matrix factorization problem.

There are several algorithms which find the W and H matrices [1], [2]. A simple algorithm is "alternating projected gradient descent". It looks like this:

1. Choose starting guess matrices W and H
2. Freeze H and vary W to minimize equation (2).
3. Freeze W and vary H to further minimize equation (2).
4. Check for convergence.
   - If not converged, go back to 2.
   - Otherwise, we're done.

We will use projected gradient descent in both steps 2 and 3 to minimize (2). To use gradient descent, we need the gradients of the objective function. But gradients with respect to what? The objective function is a function of all elements of W and H. That is, we can write $f(W,H)=f(w_{11},w_{12},\dots w_{21},w_{22}\dots h_{11},h_{12},\dots h_{21},h_{22}\dots)$. Therefore, when taking the gradient of $f(W,H)$, we take the partial derivatives w.r.t. each of $w_{11},w_{12},\dots w_{21},w_{22}\dots h_{11},h_{12},\dots h_{21},h_{22}\dots$ and stack them all up to create a tall vector. We can see how this works by considering how a 3x3 matrix may be made from two 2x3 matrices. For the decomposition we approximate

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \approx \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix} \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \end{pmatrix} = \begin{pmatrix} w_{11}h_{11}+w_{12}h_{21} & w_{11}h_{12}+w_{12}h_{22} & w_{11}h_{13}+w_{12}h_{23} \\ w_{21}h_{11}+w_{22}h_{21} & w_{21}h_{12}+w_{22}h_{22} & w_{21}h_{13}+w_{22}h_{23} \\ w_{31}h_{11}+w_{32}h_{21} & w_{31}h_{12}+w_{32}h_{22} & w_{31}h_{13}+w_{32}h_{23} \end{pmatrix}$$

Now, using the matrix elements in the definition of the objective function defined in equation (1) we have

$$\begin{aligned} 2f() = {} & (a_{11}-w_{11}h_{11}-w_{12}h_{21})^2+(a_{12}-w_{11}h_{12}-w_{12}h_{22})^2+(a_{13}-w_{11}h_{13}-w_{12}h_{23})^2 \\ & +(a_{21}-w_{21}h_{11}-w_{22}h_{21})^2+(a_{22}-w_{21}h_{12}-w_{22}h_{22})^2+(a_{23}-w_{21}h_{13}-w_{12}h_{23})^2 \\ & +(a_{31}-w_{31}h_{11}-w_{32}h_{21})^2+(a_{32}-w_{31}h_{12}-w_{32}h_{22})^2+(a_{33}-w_{31}h_{13}-w_{32}h_{23})^2 \end{aligned}$$

Note I moved the factor 2 to the LHS for ease of manipulation.

Now, taking the gradient w.r.t. the elements of W, we get

$$2\begin{pmatrix} \partial f/\partial w_{11} \\ \partial f/\partial w_{12} \\ \partial f/\partial w_{21} \\ \partial f/\partial w_{22} \\ \partial f/\partial w_{31} \\ \partial f/\partial w_{32} \end{pmatrix} = \begin{pmatrix} 2(a_{11}-w_{11}h_{11}-w_{12}h_{21})h_{11}+2(a_{12}-w_{11}h_{12}-w_{12}h_{22})h_{12}+2(a_{13}-w_{11}h_{13}-w_{12}h_{23})h_{13} \\ 2(a_{11}-w_{11}h_{11}-w_{12}h_{21})h_{21}+2(a_{12}-w_{11}h_{12}-w_{12}h_{22})h_{22}+2(a_{13}-w_{11}h_{13}-w_{12}h_{23})h_{23} \\ 2(a_{21}-w_{21}h_{11}-w_{22}h_{21})h_{11}+2(a_{22}-w_{21}h_{12}-w_{22}h_{22})h_{12}+2(a_{23}-w_{21}h_{13}-w_{12}h_{23})h_{13} \\ 2(a_{21}-w_{21}h_{11}-w_{22}h_{21})h_{21}+2(a_{22}-w_{21}h_{12}-w_{22}h_{22})h_{22}+2(a_{23}-w_{21}h_{13}-w_{12}h_{23})h_{23} \\ 2(a_{31}-w_{31}h_{11}-w_{32}h_{21})h_{11}+2(a_{32}-w_{31}h_{12}-w_{32}h_{22})h_{12}+2(a_{33}-w_{31}h_{13}-w_{32}h_{23})h_{13} \\ 2(a_{31}-w_{31}h_{11}-w_{32}h_{21})h_{21}+2(a_{32}-w_{31}h_{12}-w_{32}h_{22})h_{22}+2(a_{33}-w_{31}h_{13}-w_{32}h_{23})h_{23} \end{pmatrix} \quad (3)$$

The gradient w.r.t. $h_{11},h_{12},\dots h_{21},h_{22}\dots$ is similar.

As an aid to manipulations I will do later, I will separate the RHS of equation (3) as follows:

$$\begin{pmatrix} 2a_{11}h_{11}+2a_{12}h_{12}+2a_{13}h_{13} \\ 2a_{11}h_{21}+2a_{12}h_{22}+2a_{13}h_{23} \\ 2a_{21}h_{11}+2a_{22}h_{12}+2a_{23}h_{13} \\ 2a_{21}h_{21}+2a_{22}h_{22}+2a_{23}h_{23} \\ 2a_{31}h_{11}+2a_{32}h_{12}+2a_{33}h_{13} \\ 2a_{31}h_{21}+2a_{32}h_{22}+2a_{33}h_{23} \end{pmatrix} - \begin{pmatrix} 2(w_{11}h_{11}+w_{12}h_{21})h_{11}+2(w_{11}h_{12}+w_{12}h_{22})h_{12}+2(w_{11}h_{13}+w_{12}h_{23})h_{13} \\ 2(w_{11}h_{11}+w_{12}h_{21})h_{21}+2(w_{11}h_{12}+w_{12}h_{22})h_{22}+2(w_{11}h_{13}+w_{12}h_{23})h_{23} \\ 2(w_{21}h_{11}+w_{22}h_{21})h_{11}+2(w_{21}h_{12}+w_{22}h_{22})h_{12}+2(w_{21}h_{13}+w_{12}h_{23})h_{13} \\ 2(w_{21}h_{11}+w_{22}h_{21})h_{21}+2(w_{21}h_{12}+w_{22}h_{22})h_{22}+2(w_{21}h_{13}+w_{12}h_{23})h_{23} \\ 2(w_{31}h_{11}+w_{32}h_{21})h_{11}+2(w_{31}h_{12}+w_{32}h_{22})h_{12}+2(w_{31}h_{13}+w_{32}h_{23})h_{13} \\ 2(w_{31}h_{11}+w_{32}h_{21})h_{21}+2(w_{31}h_{12}+w_{32}h_{22})h_{22}+2(w_{31}h_{13}+w_{32}h_{23})h_{23} \end{pmatrix} \quad (4)$$

This expression could be used in gradient descent as-is, but it is messy. Here is a trick which will make evaluation of the above matrix expression easier. The elements of the gradient are formed into a column vector due to tradition. However, we could rearrange the elements into a 3x2 table and then only use operations appropriate for vector algebra (i.e. element-wise summing) when handling it for gradient descent. (I won't dwell on it here, but a theorem from linear algebra states that a set of matrices form a vector space. Therefore, one could show an isomorphism between the usual column vector representation of this object, and one in which the elements have been folded over into a matrix. Therefore, a tabular representation of the gradient is non-traditional, but perfectly valid.) We will fold the elements of the column vector as follows:

$$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

As an example, here is the folded representation of the left-hand term of equation (4):

$$(4\ \text{LHS}) = \begin{pmatrix} 2a_{11}h_{11}+2a_{12}h_{12}+2a_{13}h_{13} & 2a_{11}h_{21}+2a_{12}h_{22}+2a_{13}h_{23} \\ 2a_{21}h_{11}+2a_{22}h_{12}+2a_{23}h_{13} & 2a_{21}h_{21}+2a_{22}h_{22}+2a_{23}h_{23} \\ 2a_{31}h_{11}+2a_{32}h_{12}+2a_{33}h_{13} & 2a_{31}h_{21}+2a_{32}h_{22}+2a_{33}h_{23} \end{pmatrix} \quad (5)$$

Now observe that this matrix is a product of two matrices,

$$\begin{aligned}(5) &= 2\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}\begin{pmatrix} h_{11} & h_{21} \\ h_{12} & h_{22} \\ h_{13} & h_{23} \end{pmatrix} \\ &= 2AH^T\end{aligned}$$

It turns out that the left-hand term of (4) can also be decomposed in a similar way. I won't show the decomposition here since it's just a lot of tedious algebra. At the end of the algebraic mess is an expression for the gradient,

$$\nabla_W f(W,H) = (WH - A)H^T$$

where we have written the gradient as a table of numbers instead of a tall column vector. Note that I finally canceled out the factor of 2 which I had to carry around.

A similar expression exists for $\nabla_H f$. Both expressions for the gradients are are presented in reference [1]. They are

$$\begin{aligned}\nabla_W f(W,H) &= (WH - A)H^T \\ \nabla_H f(W,H) &= W^T(WH - A)\end{aligned}$$

where again we have elected to write the gradient as a table (matrix) instead of a column vector. These are the expressions we will use for stepping in gradient descent.

Next, we also need a step size to implement gradient descent. In my demo code I tried two step sizes. One was an ad-hoc value chosen by playing around. That value is $\alpha = 5\text{e-}5$. The other step sizes are given in reference [1] where they are motivated. There is one step size for each of the two stages of the iteration.

The gradient and the step are enough to implement gradient descent, where we update W and H iteratively until we reach convergence. However, there is one final piece to NMF: We need to make sure the elements of both W and H are non-negative. We accomplish this using "projected gradient descent", which is a fancy way to say that if we find any negative elements in these matrices, then we just set them to zero. This is easy because the constraint on this optimization problem is trivial: All matrix elements must be non-zero. If the constraint was not trivial, then it would be much more difficult – or even impossible – to implement NMF.

With these moving pieces, the overall NMF iteration is this:

1. Start with initial guess matrices W_n , H_n where $n=0$.
2. Loop:
3. We first update W . Get gradient $g_n^W = \nabla_W f(W, H) = (WH - A)H^T$
4. Compute W step: $t^W = 1/\|H_n H_n^T\|$
5. Now take W step to get new W : $W_{n+1} = W_n - t^W g_n^W$
6. Now look for any negative elements in W . If any negative elements are present, then push them back into the feasible region – i.e. set them to zero.
7. Next update H . Get gradient $g_n^H = \nabla_H f(W, H) = W^T(WH - A)$
8. Compute H step: $t^H = 1/\|W_n^T W_n\|$
9. Now take H step: $H_{n+1} = H_n - t^H g_n^H$
10. Now look for any negative elements in H_{n+1} . If any negative elements are present, then push them back into the feasible region – i.e. set them to zero.
11. Check for convergence, and if not converged then loop back to step 3.

This is the algorithm I implemented in the Matlab program available on Canvas. Screenshots showing how the program handles decomposition of a simple image are shown in Figures 1 and 2.

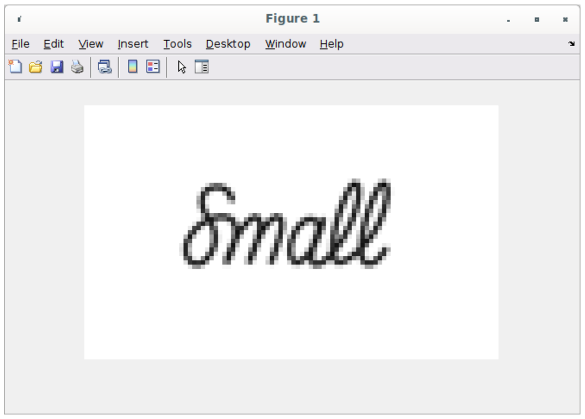


Figure 1: Original image.

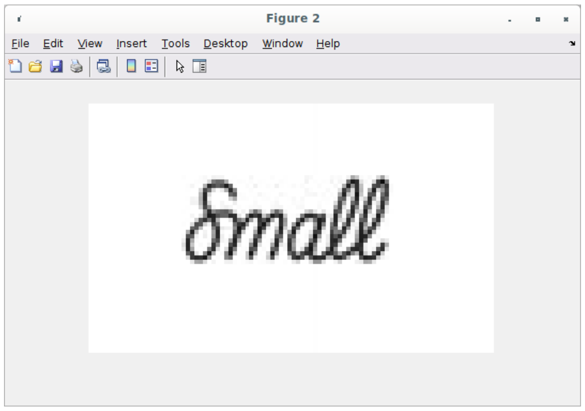


Figure 2: First the input image A is decomposed into W and H using NMF. Then the image is reconstructed A' = WH. This is the reconstructed image.

Regarding compression, a typical run looks like this:

```
>> run_nonnegative_matrix_factorization_image
Image size = [59,96].
Converged after 53673 iterations.
W size = [59,15].
H size = [15,96].
```

The original matrix holds 5664 elements, after the NMF the W and H matrices together hold 2325 elements, a reduction of a little over a factor of 2.

References

1. Nonnegative Matrix Factorization via (alternating) Projected Gradient Descent”, Andersen Ang. https://angms.science/doc/NMF/nmf_pgd.pdf.
2. "Projected Gradient Methods for Nonnegative Matrix Factorization”, C. Lin, Neural Computation, vol. 19, no. 10, pp. 2756-2779, Oct. 2007.