Detailed derivation of multiplicative update rules for NMF

Juan José Burred
March 2014
Paris, France
jjburred@jjburred.com

1 Introduction

The goal of Non-negative Matrix Factorization (NMF) is to decompose a matrix of non-negative (i.e., zero or positive) elements into a product of two factor matrices, both of them containing also non-negative elements. It is common to use the notation $\mathbf{X}$ for the input matrix (of size $M \times N$), $\mathbf{W}$ for the first factor matrix (sometimes called *basis matrix* or *feature matrix*, of size $M \times K$) and $\mathbf{H}$ for the second factor matrix (sometimes called *coefficient matrix* or *activation matrix*, of size $K \times N$). The resulting factorization is often approximate:

$$\mathbf{X} \approx \mathbf{WH} \tag{1}$$

The problem can thus be formulated by defining a scalar error measure D (called *loss* or *cost function*) between the input matrix $\mathbf{X}$ and the output product $\mathbf{WH}$, and designing an algorithm to minimize it under the non-negativity constraint:

$$\{\hat{\mathbf{W}}, \hat{\mathbf{H}}\} = \underset{\mathbf{W},\mathbf{H}}{\operatorname{argmin}}\, D(\mathbf{X}, \mathbf{WH}) \quad \text{subject to} \quad w_{mk}, h_{kn} \geq 0, \tag{2}$$

where $\hat{\mathbf{W}}$ and $\hat{\mathbf{H}}$ are the obtained output factors, and w_{mk} and h_{kn} denote the entries of the factor matrices.

Most algorithms that tackle Eq. 2, including the multiplicative update rules described here, are based on *gradient descent*. In traditional gradient descent, local minima of the cost function are searched by moving in the direction of its gradient, i.e., its direction of steepest descent. At each iteration, we move along the independent variables of the cost function so that, after a given step length (called *learning rate*), the cost function has maximally decreased. Denoting the independent variables of the cost function collectively as the parameter vector $\boldsymbol{\theta}$, and using η for the learning rate, we have the following general update rule for gradient descent:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla D(\boldsymbol{\theta}_t), \tag{3}$$

where t is the iteration counter, and $\nabla D(\boldsymbol{\theta})$ is the gradient of the cost function, given by[1]

$$\nabla D(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial D(\boldsymbol{\theta})}{\partial \theta_1} \\ \frac{\partial D(\boldsymbol{\theta})}{\partial \theta_2} \\ \vdots \\ \frac{\partial D(\boldsymbol{\theta})}{\partial \theta_P} \end{pmatrix} \tag{4}$$

To avoid the iteration indices, Eq. 3 is often written as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla D(\boldsymbol{\theta}) \tag{5}$$

In this basic form of the update rule, the learning rate η is constant, but it can change along iterations (η_t), or have different values for each one of the parameters. In the latter case, we have a vector of learning rates $\boldsymbol{\eta}$, and the above update rule can be written in compact matrix form as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \boldsymbol{\eta} \circ \nabla D(\boldsymbol{\theta}), \tag{6}$$

where $\circ$ denotes the Hadamard (element-by-element) product. In this document, compact matrix notation will be favored, but one should keep in mind that the update rules happen always on an element-by-element basis, so the above equation is equivalent to a collection of update rules

$$\theta_p \leftarrow \theta_p - \eta_p \frac{\partial D(\boldsymbol{\theta})}{\partial \theta_p}, \tag{7}$$

[1] In this document, denominator-layout notation will be used. This means that we consider the gradient of a scalar function to be a column vector.

one for each individual parameter θ_p.

In the case of NMF, we have $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{H}\}$. Instead of jointly updating both matrices, the algorithms are much simpler if they are updated in turns: in each iteration, first one matrix is updated assuming that the other matrix is constant, and then the second. This scheme is called *block-coordinate descent*:

$$\mathbf{H} \leftarrow \mathbf{H} - \boldsymbol{\eta}_\mathbf{H} \circ \nabla_\mathbf{H} D(\mathbf{X}, \mathbf{WH}) \tag{8}$$
$$\mathbf{W} \leftarrow \mathbf{W} - \boldsymbol{\eta}_\mathbf{W} \circ \nabla_\mathbf{W} D(\mathbf{X}, \mathbf{WH}) \tag{9}$$

The task of deriving the update rules amounts, then, to compute the partial gradients $\nabla_\mathbf{W} D$ and $\nabla_\mathbf{H} D$, and to choose appropriate learning rates $\boldsymbol{\eta}_\mathbf{W}$ and $\boldsymbol{\eta}_\mathbf{H}$. Now the gradients:

$$\nabla_\mathbf{W} D = \begin{pmatrix} \frac{\partial D}{\partial w_{11}} & \frac{\partial D}{\partial w_{12}} & \cdots & \frac{\partial D}{\partial w_{1K}} \\ \frac{\partial D}{\partial w_{21}} & \frac{\partial D}{\partial w_{22}} & \cdots & \frac{\partial D}{\partial w_{2K}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial D}{\partial w_{M1}} & \frac{\partial D}{\partial w_{M2}} & \cdots & \frac{\partial D}{\partial w_{MK}} \end{pmatrix} \qquad \nabla_\mathbf{H} D = \begin{pmatrix} \frac{\partial D}{\partial h_{11}} & \frac{\partial D}{\partial h_{12}} & \cdots & \frac{\partial D}{\partial h_{1N}} \\ \frac{\partial D}{\partial h_{21}} & \frac{\partial D}{\partial h_{22}} & \cdots & \frac{\partial D}{\partial h_{2N}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial D}{\partial h_{K1}} & \frac{\partial D}{\partial h_{K2}} & \cdots & \frac{\partial D}{\partial h_{KN}} \end{pmatrix} \tag{10}$$

and the learning rates $\boldsymbol{\eta}_\mathbf{W}$ and $\boldsymbol{\eta}_\mathbf{H}$ are full matrices of sizes $M \times K$ and $K \times N$, respectively. The final form of the update rules will thus depend on the chosen cost function D. This document will focus on three of the most popular choices: the Euclidean distance, the Kullback-Leibler divergence and the Itakura-Saito divergence.

2 Update rules based on the Euclidean distance

Let's start by computing the gradients $\nabla_\mathbf{W} D$ and $\nabla_\mathbf{H} D$ for the Euclidean distance. The familiar form of the Euclidean distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ of N elements is

$$d_{\text{EUC}}(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_N - y_N)^2} = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \tag{11}$$

Alternatively, we can consider the Euclidean norm:

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \ldots + x_N^2} = \sqrt{\sum_{i=1}^{N} x_i^2} \tag{12}$$

and interpret Eq. 11 as the norm of the difference vector:

$$d_{\text{EUC}}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\| \tag{13}$$

For matrices, the equivalent of the Euclidean norm is the Frobenius norm, given by

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} x_{ij}^2}, \tag{14}$$

and in analogy we can define an Euclidean-like distance between matrices, which is really the Frobenius norm of their difference:

$$D_{\text{EUC}}(\mathbf{X}, \mathbf{Y}) = \|\mathbf{X} - \mathbf{Y}\|_F = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} (x_{ij} - y_{ij})^2} \tag{15}$$

To simplify the computation of the gradients, what is actually used in the context of NMF is the square of the Frobenius norm. This gives finally the following cost function:

$$D_{\text{EUC}}(\mathbf{X}, \mathbf{WH}) = \|\mathbf{X} - \mathbf{WH}\|_F^2 = \sum_m \sum_n (x_{mn} - \mathbf{WH}|_{mn})^2, \tag{16}$$

where $\mathbf{WH}|_{mn}$ denotes the mn-th element of the result of the matrix product $\mathbf{WH}$. To compute the gradients of this equation, we can proceed in two possible ways: we can either keep using compact matrix notation and resort to matrix calculus, or we can look into element-by-element derivations. The two routes will be detailed here.

Euclidean gradient computation using matrix calculus

We will start from one of the properties of the trace of the matrix. Recall that the *trace* is the sum of the elements on the main diagonal of a square matrix. The property that will interest us here is the following:

$$\operatorname{tr}(\mathbf{X}^T\mathbf{Y}) = \sum_{i=1}^{M}\sum_{j=1}^{N} x_{ij}y_{ij} \tag{17}$$

In other words, the trace of a product of matrices equals the sum of their element-by-element (Hadamard) products (both matrices must have the same size). From this, it follows that the Frobenius norm (Eq. 14) can be rewritten as

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N} x_{ij}^2} = \sqrt{\operatorname{tr}(\mathbf{X}^T\mathbf{X})}, \tag{18}$$

and the Euclidean NMF cost function becomes:

$$D_{\text{EUC}}(\mathbf{X},\mathbf{WH}) = \|\mathbf{X}-\mathbf{WH}\|_F^2 = \operatorname{tr}\left[(\mathbf{X}-\mathbf{WH})^T(\mathbf{X}-\mathbf{WH})\right]. \tag{19}$$

We can now use the addition property of the transpose, $(\mathbf{X}+\mathbf{Y})^T = \mathbf{X}^T + \mathbf{Y}^T$, and its multiplication property, $(\mathbf{XY})^T = \mathbf{Y}^T\mathbf{X}^T$, to expand this further as

$$D_{\text{EUC}}(\mathbf{X},\mathbf{WH}) = \operatorname{tr}\left[(\mathbf{X}^T-\mathbf{H}^T\mathbf{W}^T)(\mathbf{X}-\mathbf{WH})\right] = \operatorname{tr}(\mathbf{X}^T\mathbf{X}-\mathbf{X}^T\mathbf{WH}-\mathbf{H}^T\mathbf{W}^T\mathbf{X}+\mathbf{H}^T\mathbf{W}^T\mathbf{WH}) \tag{20}$$

For computing the gradient of this equation, we will make use of these further properties of the trace:

- Trace of a sum:
$$\operatorname{tr}(\mathbf{A}+\mathbf{B}) = \operatorname{tr}(\mathbf{A})+\operatorname{tr}(\mathbf{B}) \tag{21}$$
- Cyclic permutation:
$$\operatorname{tr}(\mathbf{ABC}) = \operatorname{tr}(\mathbf{CAB}) = \operatorname{tr}(\mathbf{BCA}) \tag{22}$$
- Gradient of traces of product with constant matrix $\mathbf{A}$:

$$\nabla_{\mathbf{X}}\operatorname{tr}(\mathbf{AX}) = \mathbf{A}^T \tag{23}$$
$$\nabla_{\mathbf{X}}\operatorname{tr}(\mathbf{X}^T\mathbf{A}) = \mathbf{A} \tag{24}$$
$$\nabla_{\mathbf{X}}\operatorname{tr}(\mathbf{X}^T\mathbf{AX}) = (\mathbf{A}+\mathbf{A}^T)\mathbf{X} \tag{25}$$
$$\nabla_{\mathbf{X}}\operatorname{tr}(\mathbf{XAX}^T) = \mathbf{X}(\mathbf{A}^T+\mathbf{A}) \tag{26}$$

Let's begin with the computation of $\nabla_{\mathbf{H}}D$. Eq. 21 allows us to compute the gradient of Eq. 20 term by term. For the first term, we immediately see that $\nabla_{\mathbf{H}}\operatorname{tr}(\mathbf{X}^T\mathbf{X}) = 0$, since the term does not depend on $\mathbf{H}$. For the second term, we use Eq. 23 to obtain:

$$\nabla_{\mathbf{H}}\operatorname{tr}(\mathbf{X}^T\mathbf{WH}) = (\mathbf{X}^T\mathbf{W})^T = \mathbf{W}^T\mathbf{X} \tag{27}$$

For the third term, we use Eq. 24 to obtain:

$$\nabla_{\mathbf{H}}\operatorname{tr}(\mathbf{H}^T\mathbf{W}^T\mathbf{X}) = \mathbf{W}^T\mathbf{X} \tag{28}$$

For the last term, we can set $\mathbf{A} = \mathbf{W}^T\mathbf{W}$ and use Eq. 25:

$$\nabla_{\mathbf{H}}\operatorname{tr}(\mathbf{H}^T\mathbf{W}^T\mathbf{WH}) = \left[\mathbf{W}^T\mathbf{W}+(\mathbf{W}^T\mathbf{W})^T\right]\mathbf{H} = 2\mathbf{W}^T\mathbf{WH} \tag{29}$$

Putting all the terms together with the appropriate signs of Eq. 20, we finally get

$$\boxed{\nabla_{\mathbf{H}}D_{\text{EUC}}(\mathbf{X},\mathbf{WH}) = -2\mathbf{W}^T\mathbf{X}+2\mathbf{W}^T\mathbf{WH}} \tag{30}$$

For $\nabla_{\mathbf{W}}D$, we proceed analogously. First, we also have $\nabla_{\mathbf{W}}\operatorname{tr}(\mathbf{X}^T\mathbf{X}) = 0$. For the second term, we use the cyclic permutation property and Eq. 23:

$$\nabla_{\mathbf{W}}\operatorname{tr}(\mathbf{X}^T\mathbf{WH}) = \nabla_{\mathbf{W}}\operatorname{tr}(\mathbf{HX}^T\mathbf{W}) = (\mathbf{HX}^T)^T = \mathbf{XH}^T \tag{31}$$

The third term gives, from a permutation of Eq.24:

$$\nabla_{\mathbf{W}} \operatorname{tr}(\mathbf{H}^T\mathbf{W}^T\mathbf{X}) = \nabla_{\mathbf{W}} \operatorname{tr}(\mathbf{X}\mathbf{H}^T\mathbf{W}^T) = \mathbf{X}\mathbf{H}^T \tag{32}$$

For the last term, we permute twice, set $\mathbf{A} = \mathbf{H}\mathbf{H}^T$ and use Eq. 26:

$$\nabla_{\mathbf{W}} \operatorname{tr}(\mathbf{H}^T\mathbf{W}^T\mathbf{W}\mathbf{H}) = \nabla_{\mathbf{W}} \operatorname{tr}(\mathbf{W}\mathbf{H}\mathbf{H}^T\mathbf{W}^T) = \mathbf{W}\left[(\mathbf{H}\mathbf{H}^T)^T + \mathbf{H}\mathbf{H}^T\right] = 2\mathbf{W}\mathbf{H}\mathbf{H}^T \tag{33}$$

And putting the terms together:

$$\boxed{\nabla_{\mathbf{W}} D_{\mathrm{EUC}}(\mathbf{X},\mathbf{W}\mathbf{H}) = -2\mathbf{X}\mathbf{H}^T + 2\mathbf{W}\mathbf{H}\mathbf{H}^T} \tag{34}$$

Euclidean gradient computation using elementwise derivations

Alternatively, we can compute the gradients using traditional element-wise derivatives. We will first use the definition of matrix multiplication:

$$\mathbf{W}\mathbf{H}|_{mn} = \sum_k w_{mk}h_{kn} \tag{35}$$

to expand the cost function of Eq.36 to:

$$D_{\mathrm{EUC}}(\mathbf{X},\mathbf{W}\mathbf{H}) = \sum_m\sum_n (x_{mn} - \mathbf{W}\mathbf{H}|_{mn})^2 = \sum_m\sum_n (x_{mn} - \sum_k w_{mk}h_{kn})^2 \tag{36}$$

We wish to obtain the matrix gradients of Eq. 10 element-by-element by computing the partial derivatives with respect to particular elements at indices i and j. We will start with the gradient for $\mathbf{W}$:

$$\frac{\partial D_{\mathrm{EUC}}(\mathbf{X},\mathbf{W}\mathbf{H})}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_m\sum_n (x_{mn} - \sum_k w_{mk}h_{kn})^2 \tag{37}$$

Using the chain rule, we have

$$\frac{\partial}{\partial w_{ij}} \sum_m\sum_n (x_{mn} - \sum_k w_{mk}h_{kn})^2 = 2\sum_m\sum_n \left[(x_{mn} - \sum_k w_{mk}h_{kn})\frac{\partial}{\partial w_{ij}}(x_{mn} - \sum_k w_{mk}h_{kn})\right] \tag{38}$$

Because $\partial x_{mn}/\partial w_{ij} = 0$, the derivative that needs to be computed is

$$\frac{\partial}{\partial w_{ij}} \sum_k w_{mk}h_{kn} = \frac{\partial}{\partial w_{ij}}(w_{m1}h_{1n} + w_{m2}h_{2n} + \ldots + w_{mK}h_{Kn}) \tag{39}$$

From this expansion, we see that the derivative will only be non-zero if $m = i$, in which case its value will be h_{jn}:

$$\frac{\partial}{\partial w_{ij}} \sum_k w_{mk}h_{kn} = \frac{\partial}{\partial w_{ij}}\mathbf{W}\mathbf{H}|_{mn} = \begin{cases} h_{jn} & \text{if} \quad m = i \\ 0 & \text{if} \quad m \neq i \end{cases} \tag{40}$$

This result allows us to simplify Eq. 38 by eliminating all zero terms, i.e., by eliminating the summation over m by setting $m = i$, finally obtaining the derivative:

$$\frac{\partial D_{\mathrm{EUC}}(\mathbf{X},\mathbf{W}\mathbf{H})}{\partial w_{ij}} = 2\sum_n \left[(x_{in} - \sum_k w_{ik}h_{kn})(-h_{jn})\right] = -2\sum_n x_{in}h_{jn} + 2\sum_n\sum_k w_{ik}h_{kn}h_{jn} \tag{41}$$

To put this back into matrix form, we use the fact that $\mathbf{A}\mathbf{B}^T|_{ij} = \sum_k a_{ik}b_{jk}$:

$$\frac{\partial D_{\mathrm{EUC}}(\mathbf{X},\mathbf{W}\mathbf{H})}{\partial w_{ij}} = -2\mathbf{X}\mathbf{H}^T|_{ij} + 2\sum_n (\mathbf{W}\mathbf{H}|_{in})h_{jn} = -2\mathbf{X}\mathbf{H}^T|_{ij} + 2\mathbf{W}\mathbf{H}\mathbf{H}^T|_{ij} \tag{42}$$

We have obtained the same gradient as before (Eq. 34). Let's now derive wrt h_{ij}. Using the same logic, we now find:

$$\frac{\partial}{\partial h_{ij}} \sum_k w_{mk}h_{kn} = \frac{\partial}{\partial h_{ij}}\mathbf{W}\mathbf{H}|_{mn} = \begin{cases} w_{mi} & \text{if} \quad n = j \\ 0 & \text{if} \quad n \neq j \end{cases} \tag{43}$$

and now the derivative is

$$\frac{\partial D_{\text{EUC}}(\mathbf{X},\mathbf{WH})}{\partial h_{ij}} = -2\sum_m x_{mj}w_{mi} + 2\sum_m\sum_k w_{mk}h_{kj}w_{mi} = -2\sum_m w_{mi}x_{mj} + 2\sum_m w_{mi}\sum_k w_{mk}h_{kj} \tag{44}$$

and using $\mathbf{A}^T\mathbf{B}|_{ij} = \sum_k a_{ki}b_{jk}$:

$$\frac{\partial D_{\text{EUC}}(\mathbf{X},\mathbf{WH})}{\partial h_{ij}} = -2\mathbf{W}^T\mathbf{X}|_{ij} + 2\sum_m w_{mi}\mathbf{WH}|_{mj} = -2\mathbf{W}^T\mathbf{X}|_{ij} + 2\mathbf{W}^T\mathbf{WH}|_{ij} \tag{45}$$

From additive to multiplicative update rules

Having obtained both gradients $\nabla_{\mathbf{H}}D$ (Eq. 30) and $\nabla_{\mathbf{W}}D$ (Eq. 34), we can now put them into Eqs. 9 and 8 to obtain the block-coordinate gradient descent algorithm for Euclidean NMF:

$$\mathbf{H} \leftarrow \mathbf{H} + \boldsymbol{\eta}_{\mathbf{H}} \circ (\mathbf{W}^T\mathbf{X} - \mathbf{W}^T\mathbf{WH}) \tag{46}$$

$$\mathbf{W} \leftarrow \mathbf{W} + \boldsymbol{\eta}_{\mathbf{W}} \circ (\mathbf{XH}^T - \mathbf{WHH}^T) \tag{47}$$

Note that the scalar 2 can be ignored, since we can consider that it is absorbed into the learning rates.

In conventional gradient descent, all the elements of the learning rates $\boldsymbol{\eta}_{\mathbf{H}}$ and $\boldsymbol{\eta}_{\mathbf{W}}$ are positive. Due to the subtraction present in both update rules, this can produce negative elements, violating the non-negativity constraint. To avoid this, Lee and Seung proposed in 2001 [3] to use data-adaptive learning rates that would make it impossible to obtain negative elements. The trick is to define the learning rates so that any subtraction dissapears from the update rule. For $\mathbf{H}$, if we set:

$$\boldsymbol{\eta}_{\mathbf{H}} = \frac{\mathbf{H}}{\mathbf{W}^T\mathbf{WH}} \tag{48}$$

(the fraction line is used here to denote element-by-element division) then the first update rule becomes:

$$\mathbf{H} \leftarrow \mathbf{H} + \frac{\mathbf{H}}{\mathbf{W}^T\mathbf{WH}} \circ (\mathbf{W}^T\mathbf{X} - \mathbf{W}^T\mathbf{WH}) = \mathbf{H} + \mathbf{H} \circ \frac{\mathbf{W}^T\mathbf{X}}{\mathbf{W}^T\mathbf{WH}} - \mathbf{H} \circ \frac{\mathbf{W}^T\mathbf{WH}}{\mathbf{W}^T\mathbf{WH}} = \mathbf{H} \circ \frac{\mathbf{W}^T\mathbf{X}}{\mathbf{W}^T\mathbf{WH}} \tag{49}$$

Analogously, for $\mathbf{W}$ we can set:

$$\boldsymbol{\eta}_{\mathbf{W}} = \frac{\mathbf{W}}{\mathbf{WHH}^T} \tag{50}$$

to turn the second update rule into:

$$\mathbf{W} \leftarrow \mathbf{W} + \frac{\mathbf{W}}{\mathbf{WHH}^T} \circ (\mathbf{XH}^T - \mathbf{WHH}^T) = \mathbf{W} + \mathbf{W} \circ \frac{\mathbf{XH}^T}{\mathbf{WHH}^T} - \mathbf{W} \circ \frac{\mathbf{WHH}^T}{\mathbf{WHH}^T} = \mathbf{W} \circ \frac{\mathbf{XH}^T}{\mathbf{WHH}^T} \tag{51}$$

The additive update rules have thus been transformed into multiplicative update rules, which cannot generate negative elements since all values are positive and only multiplications and divisions are involved at each update:

$$\boxed{\mathbf{H} \leftarrow \mathbf{H} \circ \frac{\mathbf{W}^T\mathbf{X}}{\mathbf{W}^T\mathbf{WH}}} \tag{52}$$

$$\boxed{\mathbf{W} \leftarrow \mathbf{W} \circ \frac{\mathbf{XH}^T}{\mathbf{WHH}^T}} \tag{53}$$

Choosing the adaptive learning rates in the form of Eqs. 48 and 50 to avoid subtraction is equivalent to deriving a multiplicative learning rule of the generic form

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} \circ \frac{\nabla^-_{\boldsymbol{\theta}} D(\boldsymbol{\theta})}{\nabla^+_{\boldsymbol{\theta}} D(\boldsymbol{\theta})} \tag{54}$$

where we put all the negative terms of the gradient in the numerator ($\nabla^-_{\boldsymbol{\theta}} D(\boldsymbol{\theta})$) and all the positive terms in the denominator ($\nabla^+_{\boldsymbol{\theta}} D(\boldsymbol{\theta})$).

3 Update rules based on the Kullback-Leibler divergence

Another commonly used cost function for NMF is based on the Kullback-Leibler (KL) divergence, whose original definition as a measure of similarity between (discrete) probability distributions p and q is:

$$D_{KL}(p,q) = \sum_i p(i) \log \frac{p(i)}{q(i)} \tag{55}$$

In the context of NMF, what is actually used is the *generalized* KL divergence [3]:

$$D_{KL}(p,q) = \sum_i \left[p(i) \log \frac{p(i)}{q(i)} - p(i) + q(i) \right] \tag{56}$$

In matrix form, and adapted to our NMF problem, this becomes:

$$D_{KL}(\mathbf{X}, \mathbf{WH}) = \sum_m \sum_n \left(x_{mn} \log \frac{x_{mn}}{\mathbf{WH}|_{mn}} - x_{mn} + \mathbf{WH}|_{mn} \right) \tag{57}$$

We first compute its derivative wrt w_{ij}:

$$\frac{\partial D_{KL}(\mathbf{X}, \mathbf{WH})}{\partial w_{ij}} = \sum_m \sum_n \frac{\partial}{\partial w_{ij}} \left(x_{mn} \log \frac{x_{mn}}{\mathbf{WH}|_{mn}} \right) + \sum_m \sum_n \frac{\partial}{\partial w_{ij}} \mathbf{WH}|_{mn} \tag{58}$$

We already encountered the derivative of the last term during the derivation of the Euclidean update rules (Eq. 40), so the last term becomes

$$\sum_m \sum_n \frac{\partial}{\partial w_{ij}} \mathbf{WH}|_{mn} = \sum_n h_{jn} \tag{59}$$

For the first term, we can use $\log(a/b) = \log(a) - \log(b)$:

$$\frac{\partial}{\partial w_{ij}} \left(x_{mn} \log \frac{x_{mn}}{\mathbf{WH}|_{mn}} \right) = x_{mn} \frac{\partial}{\partial w_{ij}} \left(\log x_{mn} - \log \mathbf{WH}|_{mn} \right) = -x_{mn} \frac{\partial}{\partial w_{ij}} \log \mathbf{WH}|_{mn} \tag{60}$$

Using the chain rule for logarithms of functions:

$$\frac{\partial}{\partial x} \log(f(x)) = \frac{\partial f(x)/\partial x}{f(x)} \tag{61}$$

we obtain

$$-x_{mn} \frac{\partial}{\partial w_{ij}} \log \mathbf{WH}|_{mn} = -\frac{x_{mn}}{\mathbf{WH}|_{mn}} \frac{\partial}{\partial w_{ij}} \mathbf{WH}|_{mn} \tag{62}$$

Using again Eq. 40 we finally get

$$\frac{\partial D_{KL}(\mathbf{X}, \mathbf{WH})}{\partial w_{ij}} = -\sum_n \left(\frac{x_{in} h_{jn}}{\mathbf{WH}|_{in}} \right) + \sum_n h_{jn} \tag{63}$$

To put this into compact matrix notation, we can imagine a matrix $\mathbf{A}$ whose elements are $a_{ij} = x_{ij}/\mathbf{WH}|_{ij}$, so that we can write the first term as a matrix product:

$$-\sum_n \left(\frac{x_{in} h_{jn}}{\mathbf{WH}|_{in}} \right) = -\sum_n a_{in} h_{jn} = -\mathbf{AH}^T|_{ij} \tag{64}$$

Since $\mathbf{A}$ has been constructed on an element-by-element basis, we can write

$$\mathbf{A} = \frac{\mathbf{X}}{\mathbf{WH}} \tag{65}$$

We can finally put the remaining term $\sum_n h_{jn}$ into matrix form by observing that it builds a matrix whose elements are the sum of the current row of $\mathbf{H}$ or, alternatively, the current column of $\mathbf{H}^T$. This column-summing result can be obtained by a matrix multiplication with a matrix of ones:

$$\sum_n h_{jn} = \mathbf{1H}^T|_{ij} \tag{66}$$

Putting all this together we get the matrix gradient:

$$\boxed{\nabla_{\mathbf{W}} D_{KL}(\mathbf{X}, \mathbf{WH}) = -\frac{\mathbf{X}}{\mathbf{WH}}\mathbf{H}^T + \mathbf{1}\mathbf{H}^T} \tag{67}$$

Proceeding likewise for h_{ij}, we get

$$\frac{\partial D_{KL}(\mathbf{X}, \mathbf{WH})}{\partial h_{ij}} = -\sum_m \frac{x_{mj} w_{mi}}{\mathbf{WH}|_{mj}} + \sum_m w_{mi} = -\mathbf{W}^T\mathbf{A}|_{ij} + \mathbf{W}^T\mathbf{1}|_{ij} \tag{68}$$

and thus:

$$\boxed{\nabla_{\mathbf{H}} D_{KL}(\mathbf{X}, \mathbf{WH}) = -\mathbf{W}^T\frac{\mathbf{X}}{\mathbf{WH}} + \mathbf{W}^T\mathbf{1}} \tag{69}$$

Setting the adaptive learning rates of Eqs. 46 and 47 to avoid subtractions, or alternatively using the heuristic of Eq. 54, we obtain the following update rules for NMF based on the Kullback-Leibler divergence:

$$\boxed{\mathbf{H} \leftarrow \mathbf{H} \circ \frac{\mathbf{W}^T \frac{\mathbf{X}}{\mathbf{WH}}}{\mathbf{W}^T\mathbf{1}}} \tag{70}$$

$$\boxed{\mathbf{W} \leftarrow \mathbf{W} \circ \frac{\frac{\mathbf{X}}{\mathbf{WH}}\mathbf{H}^T}{\mathbf{1}\mathbf{H}^T}} \tag{71}$$

4 Update rules based on the Itakura-Saito divergence

The Itakura-Saito divergence was originally proposed in 1968 [2] as a measure of the similarity between two spectra p and q:

$$D_{IS}(p, q) = \sum_i \left[\frac{p(i)}{q(i)} - \log\frac{p(i)}{q(i)} - 1\right] \tag{72}$$

It has been adopted as a cost function for NMF, especially for audio applications [1]. The cost function becomes:

$$D_{IS}(\mathbf{X}, \mathbf{WH}) = \sum_m \sum_n \left(\frac{x_{mn}}{\mathbf{WH}|_{mn}} - \log\frac{x_{mn}}{\mathbf{WH}|_{mn}} - 1\right) \tag{73}$$

Deriving wrt w_{ij}:

$$\frac{\partial D_{IS}(\mathbf{X}, \mathbf{WH})}{\partial w_{ij}} = \sum_m \sum_n x_{mn} \frac{\partial}{\partial w_{ij}} \frac{1}{\mathbf{WH}|_{mn}} + \sum_m \sum_n \frac{\partial}{\partial w_{ij}} \log \mathbf{WH}|_{mn} \tag{74}$$

We've already encountered the derivative of the second term in Eq. 62:

$$\sum_m \sum_n \frac{\partial}{\partial w_{ij}} \log \mathbf{WH}|_{mn} = \sum_n \frac{h_{jn}}{\mathbf{WH}|_{in}} \tag{75}$$

For the first term, we use the reciprocal rule of derivation:

$$\frac{\partial}{\partial x}\frac{1}{f(x)} = \frac{-\partial f(x)/\partial x}{f(x)^2} \tag{76}$$

to get

$$\frac{\partial}{\partial w_{ij}} \frac{1}{\mathbf{WH}|_{mn}} = \frac{-1}{(\mathbf{WH}|_{mn})^2} \frac{\partial}{\partial w_{ij}} \mathbf{WH}|_{mn} \tag{77}$$

and again using Eq. 40 and putting the previous equations together:

$$\frac{\partial D_{IS}(\mathbf{X}, \mathbf{WH})}{\partial w_{ij}} = -\sum_n \frac{x_{in} h_{jn}}{(\mathbf{WH}|_{in})^2} + \sum_n \frac{h_{jn}}{\mathbf{WH}|_{in}} \tag{78}$$

Introducing the matrices $a_{ij} = x_{ij}/(\mathbf{WH}|_{ij})^2$ and $b_{ij} = 1/\mathbf{WH}|_{ij}$ this simplifies to

$$\frac{\partial D_{IS}(\mathbf{X}, \mathbf{WH})}{\partial w_{ij}} = -\sum_n a_{in} h_{jn} + \sum_n b_{in} h_{jn} = -\mathbf{A}\mathbf{H}^T|_{ij} + \mathbf{B}\mathbf{H}^T|_{ij} \tag{79}$$

And in matrix form we have

$$\mathbf{A} = \frac{\mathbf{X}}{(\mathbf{WH})^{\circ 2}} \qquad \mathbf{B} = \frac{\mathbf{1}}{\mathbf{WH}} \tag{80}$$

where $^{\circ 2}$ means element-wise power and $\mathbf{1}$ is a matrix of ones, thus getting the gradient

$$\boxed{\nabla_{\mathbf{W}} D_{IS}(\mathbf{X}, \mathbf{WH}) = -\frac{\mathbf{X}}{(\mathbf{WH})^{\circ 2}}\mathbf{H}^T + \frac{\mathbf{1}}{\mathbf{WH}}\mathbf{H}^T} \tag{81}$$

Similarly, for h_{ij} we have

$$\frac{\partial D_{IS}(\mathbf{X}, \mathbf{WH})}{\partial h_{ij}} = -\sum_m \frac{x_{mj} w_{mi}}{(\mathbf{WH}|_{mj})^2} + \sum_m \frac{w_{mi}}{\mathbf{WH}|_{mj}} = -\mathbf{W}^T\mathbf{A}|_{ij} + \mathbf{W}^T\mathbf{B}|_{ij} \tag{82}$$

and finally

$$\boxed{\nabla_{\mathbf{H}} D_{IS}(\mathbf{X}, \mathbf{WH}) = -\mathbf{W}^T \frac{\mathbf{X}}{(\mathbf{WH})^{\circ 2}} + \mathbf{W}^T \frac{\mathbf{1}}{\mathbf{WH}}} \tag{83}$$

Using the same procedure as before, we get these multiplicative update rules:

$$\boxed{\mathbf{H} \leftarrow \mathbf{H} \circ \frac{\mathbf{W}^T \frac{\mathbf{X}}{(\mathbf{WH})^{\circ 2}}}{\mathbf{W}^T \frac{\mathbf{1}}{\mathbf{WH}}}} \tag{84}$$

$$\boxed{\mathbf{W} \leftarrow \mathbf{W} \circ \frac{\frac{\mathbf{X}}{(\mathbf{WH})^{\circ 2}}\mathbf{H}^T}{\frac{\mathbf{1}}{\mathbf{WH}}\mathbf{H}^T}} \tag{85}$$

References

[1] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. Nonnegative Matrix Factorization with the Itakura-Saito Divergence. With Application to Music Analysis. *Neural Computation*, 21:793–830, 2009.

[2] F. Itakura and S. Saito. Analysis synthesis telephony based on the maximum likelihood method. In *Proc. International Congress on Acoustics*, Los Alamitos, USA, 1968.

[3] D. Lee and H. Seung. Algorithms for non-negative matrix factorization. In *Proc. Neural Information Processing Systems (NIPS)*, Denver, USA, 2001.