**The Multiplicative Update Rule**

$$\min_{W,H>0} f(W,H) = \min_{W,H>0} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} (A_{ij} - (WH)_{ij})^2 \qquad (1)$$

The most used approach to minimize (1) is a simple multiplicative update method proposed by Lee and Seung (2001):
This algorithm is just a special case of the Gradient Descent with a step size

$$\epsilon(W_{ia}^t) = \frac{w_{ia}^t}{[W^t H^t (H^t)^T]_{ia}} \quad \forall i, a$$

$$\epsilon(H_{bj}^t) = \frac{h_{bj}^t}{[(W^t)^T W^t H^t]_{bj}} \forall b, j$$

**Algorithm 1:** The Multiplicative Update Rule

**Initialization:** $w_{ia}^1, h_{bj}^1 > 0 \quad \forall i, j, a, b;$

**for** $t \leftarrow 1, 2 \ldots\ldots$ **do**

$$w_{ia}^{t+1} = w_{ia}^t \frac{(A(H^t)^T)_{ia}}{[W^t H^t (H^t)^T]_{ia}} \quad \forall i, a;$$

$$h_{bj}^{t+1} = h_{bj}^t \frac{((W^{t+1})^T A)_{ia}}{[(W^{t+1})^T W^{t+1} H^t]_{bj}} \quad \forall j, b;$$

**end**

This algorithm is a fixed-point type method, meaning that If $[(H^t)^T H^t W^t]_{ia} \neq 0$ and $w_{ia}^{t+1} = w_{ia}^t$, then $(A(H^t)^T)_{ia} = [W^t H^t (H^t)^T]_{ia}$, implies $\nabla_W f(W^t, H^t)_{ia} = 0$. Which is part of the KKT condition.

# – Proof of convergence of the Multiplicative Update Rule Algorithm

**Theorem (Lee and Seung, 2001)**

*The Euclidean distance* $||A - WH||$ *is non-increasing under the update rules*

$$w_{ia} \leftarrow w_{ia} \frac{(AH^t)_{ia}}{[WH(H)^T]_{ia}}, \qquad h_{bj} \leftarrow h_{bj} \frac{(W^T A)_{ia}}{[(W)^T WH]_{bj}}$$

*The Euclidean distance is invariant under these updates if and only if* $W$ *and* $H$ *are at a stationary point of the distance.*

# – Proof

**Definition (Auxilary Function)**

A function $G(h, k)$ is called an auxiliary function for $f(h)$if:
1. $G(h, h) = f(h)$
2. $G(h, k) \geq f(h)$ for all $k$.

The following lemma illustrates why the concept of auxiliary function can be useful to minimize $F(h)$ and to find a local minimum.

**Lemma (Iterative minimization, Lee and Seung, 2001))**

*If $G$ is an auxiliary function, then $F$ is nonincreasing under the update*

$$F(h^{t+1}) = \underset{h}{\operatorname{argmin}} G(h, h^t) \tag{2}$$

**Proof**:
$F(h^{t+1}) \leq G(h^{t+1}, h^t) \leq G(h^t, h^t) = F(h^t)$
Let us now construct an Auxilary Function for our Loss function

**Lemma**

*If $K(h^t)$ is the diagonal matrix*

$$K_{ij} = \delta_{ij} \frac{(W^T W h^t)_i}{h_i^t}$$

*then*

$$G(h, h^t) = F(h^t) + (h + h^t)^T \nabla F(h^t) + \frac{1}{2}(h - h^t)^T K(h^t)(h - h^t) \tag{3}$$

*is an auxilary function for*

$$F(h) = \frac{1}{2} \sum_k (m_k - (Wh)_k) \tag{4}$$

**Proof:**

$G(h, h) = F(h)$, follows from the difinition of $G$, now we need $G(h, k) \leq F(h) \ \ \forall k$

see Here for the proof.

With all this preliminary work the proof of our Theorem can be demonstrated.

**Proof:**
Using the auxiliary function, which was defined in the previous Lemma and solving (2) by setting the gradient to zero leads to the update rule:

$$h^{t+1} = h^t - K(h^t)^{-1} \nabla F(h^t)$$

According to the Iterative minimization Lemma, F is non-increasing under this update rule. It can be obtained

$$h_i^{t+1} = h_i^t \frac{(W^T m)_i}{(W^t W h^t)_i}$$

by applying the explicit structure of $K(h^t)^{-1}$ and $\nabla F(h^t)$.
Rewritten in matrix form this is equivalent to the update rule.

**Weaknesses and modifications:**

Lee and Seung claim that the limit of the sequence $\{W^t, H^t\}$ is a stationary point (i.e., a point satisfying the KKT condition).

However, Gonzales and Zhang (2005) indicate that this claim is wrong as having the cost function non increasing under the update may not imply the convergence.

Therefore, this multiplicative update method still lacks optimization properties.

It has been repeatedly shown that the convergence is notoriously slow.

A number of modifications to the original Lee and Seung algorithms have been introduced with the objective to overcome these shortcomings.

For instance, Lin in 2007 proposed a modification that is guaranteed to converge to a stationary point, this algorithm requires more work per iteration than the already slow Lee and Seung algorithm.

# – Fast Multiplicative Update Rule Algorithm by Li-Xin Li, Lin Wu, Hui-Sheng Zhang, and Fang-Xiang Wu (2014)

**Algorithm 8:** Fast Multiplicative Update Rule Algorithm

**Initialize:** $w_{ia}^1, h_{bj}^1 > 0 \quad \forall i, j, a, b$;

**for** $t \leftarrow 1, 2 \ldots .$ **do**

  **if** $w_b^t = 0$ **then**

    $h_{bj}^{t+1} = 0$, *for all* $j$;

    *Where;*

    $w_b^t$ *is the b-th column vector of* $W^t$;

  **else**

    $h_{bj}^{t+1} = max(0, \bar{h}_{bj}^{t+1})$;

    *Where ;*

    $\bar{h}_{bj}^{t+1} = h_{bj}^t - \frac{1}{||w_b^t||^2} \frac{\partial f(W^t, H^t)}{\partial h_{bj}}$;

  **end**

  **if** $h_a^t = 0$ **then**

    $w_{ia}^{t+1} = 0$, *for all* $i$;

    *Where;*

    $h_a^t$ *is the b-th column vector of* $H^t$;

  **else**

    $w_{ia}^{t+1} = max(0, \bar{w}_{ia}^{t+1})$;

    *Where ;*

    $\bar{w}_{ia}^{t+1} = w_{ia}^t - \frac{1}{||h_a^t||^2} \frac{\partial f(W^t, H^t)}{\partial w_{ia}}$;

  **end**

  **end**

**Theorem**

*The presented Algorithm is (in most cases, strictly) faster than the Multiplicative Update Rule Algorithm.*

Li-Xin Li, Lin Wu, Hui-Sheng Zhang, and Fang-Xiang Wu
Presented a comparison between the three Algorithms in there paper in 2014.
Three algorithms areprogrammed in MATLAB R2013a and run on a computer with the following specifications: a processor of Intel Core 2 Quad CPU Q9450 at 2.66 GHZ 2.67 GHZ and a RAM of 4 GB (3.72 GB usable) **Note:**
- Algo 2 = Multiplicative Update Rule
- Algo 3 = modified Multiplicative Update Rule
- Algo 4 = Algorithm 2

TABLE I

RESULTS OF THE MEDIATE-SIZE SYNTHETIC DATA SET

| | Algorithm2 | Algorithm3 | Algorithm4 |
|---|---|---|---|
| initial values in case I | | | |
| CPUTime(s) | 0.70 | 1.05 | **0.25** |
| Iteration | 1519.83 | 1814.57 | **303.06** |
| OBJ.ave | 1114.33 | 1113.00 | **1110.05** |
| OBJ.std | 2.29 | 2.31 | **0.71** |
| initial values in case II | | | |
| CPUTime(s) | 0. 71 | 1.35 | **0.32** |
| Iteration | 1595.67 | 1338.90 | **399.26** |
| OBJ.ave | 1165.19 | 1150.37 | **1147.83** |
| OBJ.std | 3.85 | 1.90 | **1.07** |

TABLE II

RESULTS OF THE LARGE-SIZE SYNTHETIC DATA SET

| | Algorithm2 | Algorithm3 | Algorithm4 |
|---|---|---|---|
| initial values in case I | | | |
| CPUTime(s) | 110.14 | 129.80 | **106.60** |
| Iteration | 2730.37 | 2978.53 | **776.80** |
| OBJ.ave | 149450.1 | 149354.3 | **148681.0** |
| OBJ.std | 35.39 | 43.57 | **27.79** |
| initial values in case II | | | |
| CPUTime(s) | 91.66 | 130.12 | **88.18** |
| Iteration | 2518.97 | 3330.40 | **741.97** |
| OBJ.ave | 149914.2 | 149290.5 | **148639.6** |
| OBJ.std | 34.86 | 48.45 | **22.33** |

TABLE III

RESULTS OF THE REAL-LIFE IMAGE DATA SET

| | Algorithm2 | Algorithm3 | Algorithm4 |
|---|---|---|---|
| initial values in case I | | | |
| CPUTime(s) | 716.89 | 860.59 | **280.97** |
| Iteration | 5751.8 | 6529.97 | **1288.70** |
| OBJ.ave | 13.22 | 13.19 | **13.08** |
| OBJ.std | 0.034 | 0.033 | **0.016** |
| initial values in case II | | | |
| CPUTime(s) | 706.72 | 1053.66 | **322.02** |
| Iteration | 5173.73 | 7295.63 | **1032.73** |
| OBJ.ave | 16.74 | 13.20 | **13.08** |
| OBJ.std | 0.048 | 0.028 | **0.016** |

## – Alternating Non-negative Least Squares (ANLS)

The Alternating Least Squares - ALS algorithms were first introduced by Paatero 1994 , who initially invented the whole NMF theory.

**Algorithm 9:** Basic ALS for NMF

**Initialization:** $W > 0$;
**for** $t \leftarrow 1, 2 \ldots$ **do**
    *Solve for H the LS equation* $W^T WH = W^T A$;
    *Set all negative elements of in H to 0;*
    *Solve for W the LS equation* $WHH^T = AH^T$;
    *Set all negative elements of in W to 0;*
**end**

**Theorem**

*Any limit point of the sequence* $\{W^t, H^t\}$ *generated by ALS Algorithm is a stationary point of (??).*