present

Algorithms for Non Negative Matrix Factorization

Arvind Nayak, January 18, 2021

based on the paper by *Lee and Seung (2000)*

Seminar **Classical Topics for Machine Learning & Cognitive Algorithms, TU Berlin**

Introduction

A *non-negative* matrix (say, V) factorized into (usually) two **non-negative** matrices (W and H) is known as Non-Negative matrix factorization (NMF).

What numerical algorithms can be used for learning the optimal factors from data?

Roadmap

- Mathematical Description
- An example
- Cost functions
- Update rules
- Convergence
- Summary

Mathematical description

A matrix $V \in \mathbb{R}_+^{n,m}$, can be *approximately* factorized into two matrices $W \in \mathbb{R}_+^{n,r}$ and $H \in \mathbb{R}_+^{r,m}$, where $r = \min(n, m)$ such that,

$$V \approx WH$$

This statement can be posed as an optimization problem (*one of them!*).

Find $W \in \mathbb{R}_+^{n,r}$, $H \in \mathbb{R}_+^{r,m}$, where

$$\begin{aligned} \underset{W,H}{\text{argmin}} \quad & ||V - WH||_{Fr}^2 \\ \text{subject to} \quad & W \geq 0. \\ & H \geq 0. \end{aligned}$$

- Convex in either W or H, not both.
- Requires iterative methods *(Q's: Update rules, Initialization)*

Mathematical description

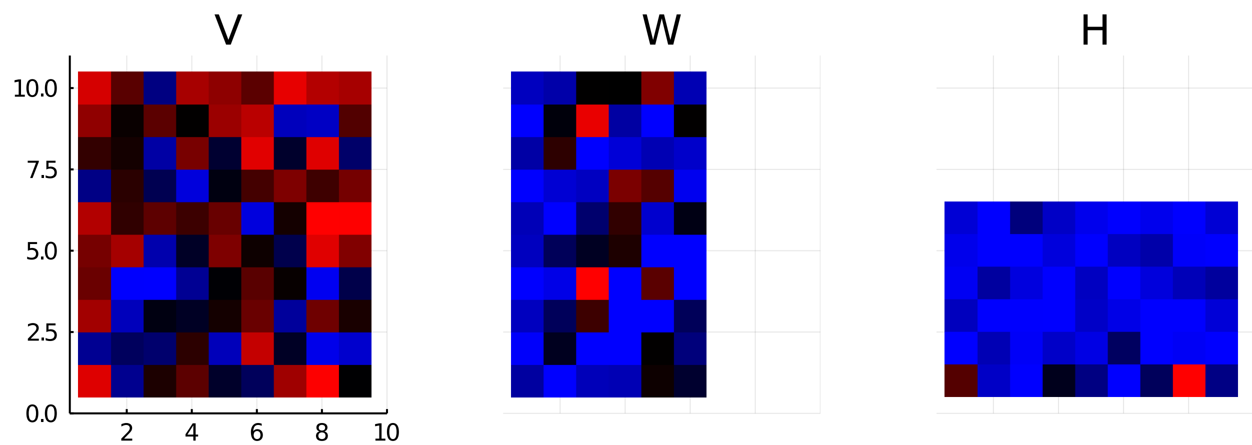
A matrix $V \in \mathbb{R}_+^{n,m}$, can be *approximately* factorized into two matrices $W \in \mathbb{R}_+^{n,r}$ and $H \in \mathbb{R}_+^{r,m}$, where $r = \min(n, m)$ such that,

$$V \approx WH$$

```
n=10;
```

```
m=9;
```

```
V = rand(n,m);
```



r = 1 50

A more conventional example

NMF done on the MNIST database

with 6 features(s)

Number of examples in the dataset are (m) = 1000

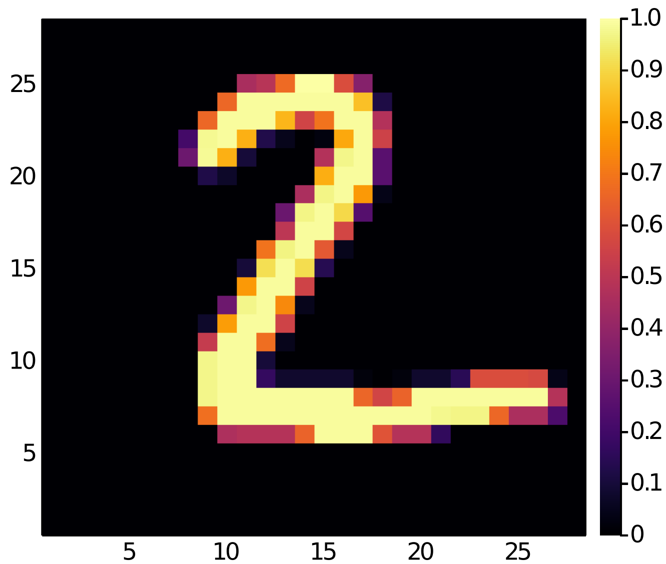
Dimensions of the image (n) = 28 x28 = 784

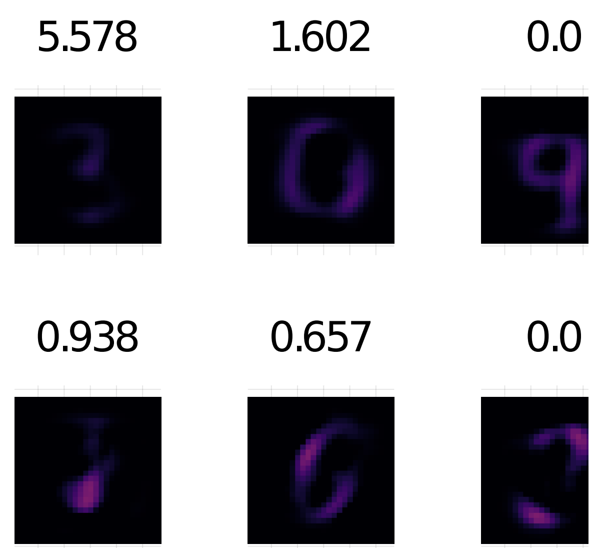
```
df=CSV.read("mnist_test.csv",DataFrame);
```

```
result_mnist_eg = solve(dataset,r,:multmse);
```

number

features (r) = 1 50





Show basis and weights

```
samples=1000;
```

```
dataset=Matrix(df[1:samples,2:end])' /255;
```

```
solve (generic function with 2 methods)
```

Cost functions

Problem 1 Find $W \in \mathbb{R}_+^{n,r}$, $H \in \mathbb{R}_+^{r,m}$, where

$$\begin{aligned} \underset{W,H}{\operatorname{argmin}} \quad & ||V - WH||_{Fr}^2 \\ \text{subject to} \quad & W \geq 0. \\ & H \geq 0. \end{aligned}$$

Problem 2 Find $W \in \mathbb{R}_+^{n,r}$, $H \in \mathbb{R}_+^{r,m}$, where

$$\begin{aligned} \underset{W,H}{\operatorname{argmin}} \quad & D(V||WH) \\ \text{subject to} \quad & W \geq 0. \\ & H \geq 0. \end{aligned}$$

$$D(V||WH) = \sum_{ij} \left(V_{ij} \log \frac{V_{ij}}{(WH)_{ij}} - V_{ij} + (WH)_{ij} \right)$$

known as the **Kulback-Leibler (KL) divergence** when, $\sum_{ij} V_{ij} = \sum_{ij} (WH)_{ij} = 1$

Update Rules

Algorithm : NMF ($V \approx WH$) under Frobenius norm measure.

> **Input**: $V \in \mathbb{R}_+^{n,m}$, rank parameter $r \in \mathbb{N}$, Stopping criterion ϵ
>
> **Output**: $W \in \mathbb{R}_+^{n,r}$ and $H \in \mathbb{R}_+^{r,m}$
>
> **Procedure**: Define $W^{(0)}$ and $H^{(0)}$ by random or informed initialization. Set $i = 0$. Apply the following update rules:
>
> (1) Compute: $H^{(i+1)} = H^{(i)} \odot \left(((W^{(i)})^T V)/((W^{((i)})^T W^{(i)} H^{(i)})\right)$
>
> (2) Compute $W^{(i+1)} = W^{(i)} \odot \left((V(H^{(i+1)})^T)/((W^{((i)}) H^{(i+1)} H^{(i+1)})^T\right)$

(3) i = i+1

Repeat the steps (1) to (3) until $||H^{(i)} - H^{(i-1)}|| \leq \epsilon$ and $||W^{(i)} - W^{(i-1)}|| \leq \epsilon$ (or some other stop criterion is fulfilled).

Set $H = H^{(i)}$ and $W = W^{(i)}$

End

Update Rules

Algorithm : NMF ($V \approx WH$) under divergence $D(V||WH)$ measure.

Input: $V \in \mathbb{R}_+^{n,m}$, rank parameter $r \in \mathbb{N}$, Stopping criterion ϵ

Output: $W \in \mathbb{R}_+^{n,r}$ and $H \in \mathbb{R}_+^{r,m}$

Procedure: Define $W^{(0)}$ and $H^{(0)}$ by random or informed initialization. Set $i = 0$. Apply the following update rules:

(1) Compute: $H^{(i+1)} = H^{(i)} \odot \frac{(W^{(i)})^T \frac{V^{(i)}}{W^{(i)}H^{(i)}}}{(W^{(i)})^T \mathbf{1}}$

(2) Compute $W^{(i+1)} = W^{(i)} \odot \frac{\frac{V^{(i)}}{W^{(i)}H^{(i+1)}}(H^{(i+1)})^T}{\mathbf{1}(H^{(i+1)})^T}$

(3) i = i+1

Repeat the steps (1) to (3) until $||H^{(i)} - H^{(i-1)}|| \leq \epsilon$ and $||W^{(i)} - W^{(i-1)}|| \leq \epsilon$ (or some other stop criterion is fulfilled).

Set $H = H^{(i)}$ and $W = W^{(i)}$

End

A note on Initialization

- Since we search for a local minima, initialization does affect our final approximation
- Most common method is **Random initialization**
- Many improvements have been suggested such as SVD based initialization. See [3]

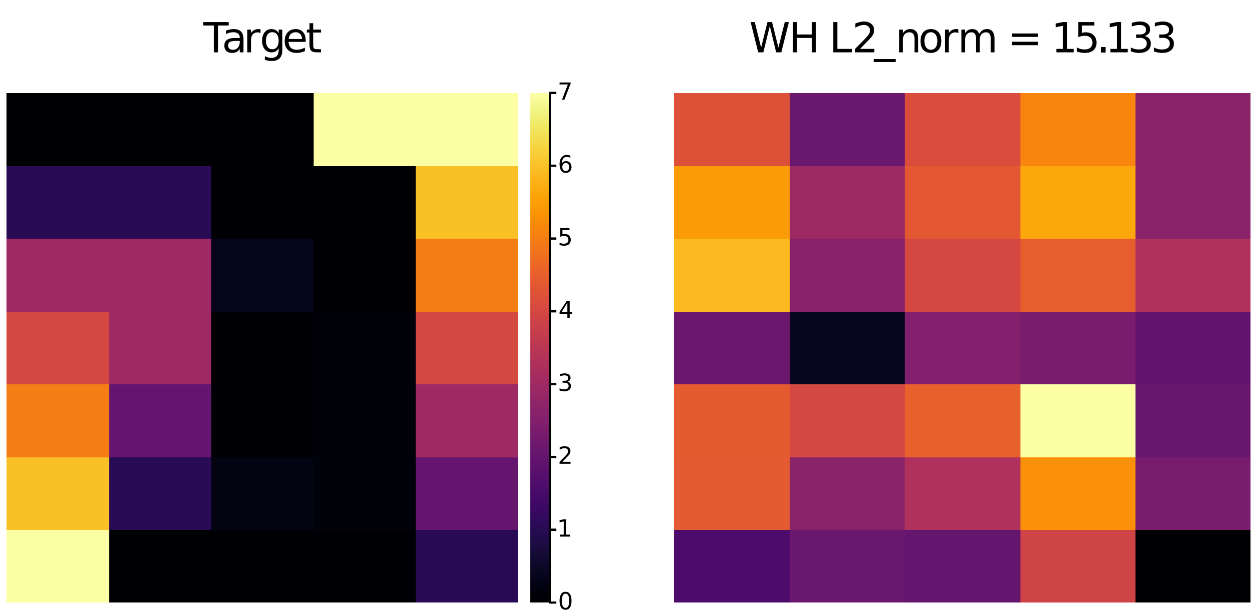
Convergence Results

features (r) = 1 50

dataset_conv_eg =

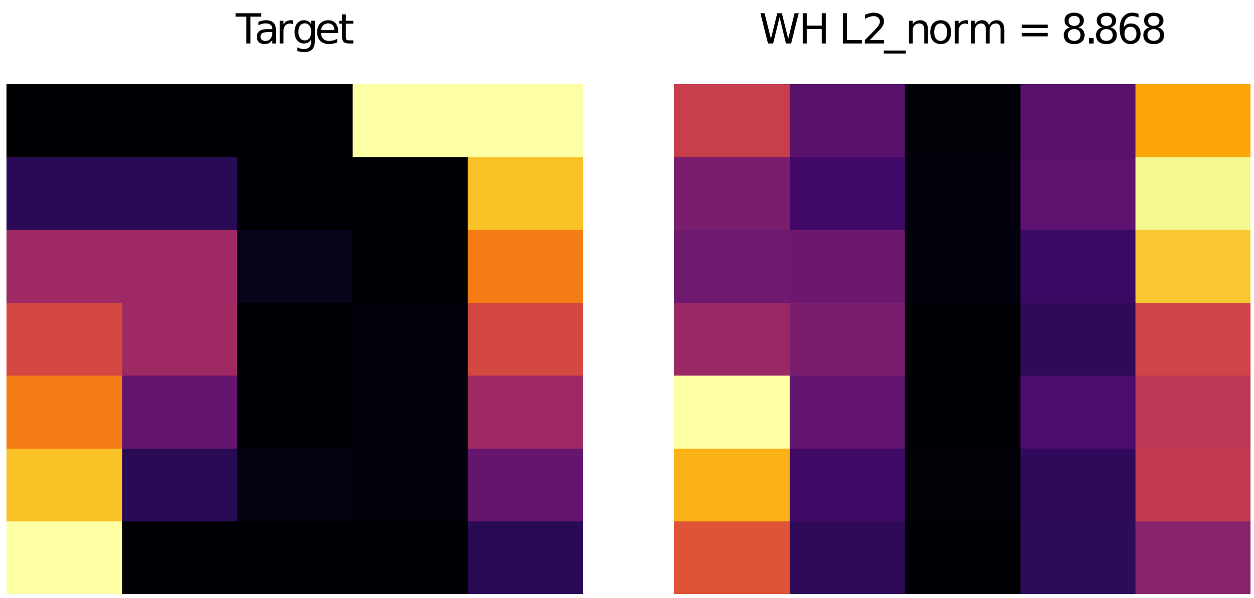
```
dataset_conv_eg = [0. 1 3 4 5 6 7;
                   0 1 3 3 2 1 0;
                   0 0 0.3 0 0 0.2 0;
                   7 0 0 0.1 0.1 0.1 0;
                   7 6 5 4 3 2 1]
    #rand(0:0.01:1,10,10);
```

```
W, H = NMF.randinit(dataset_conv_eg, r);
```



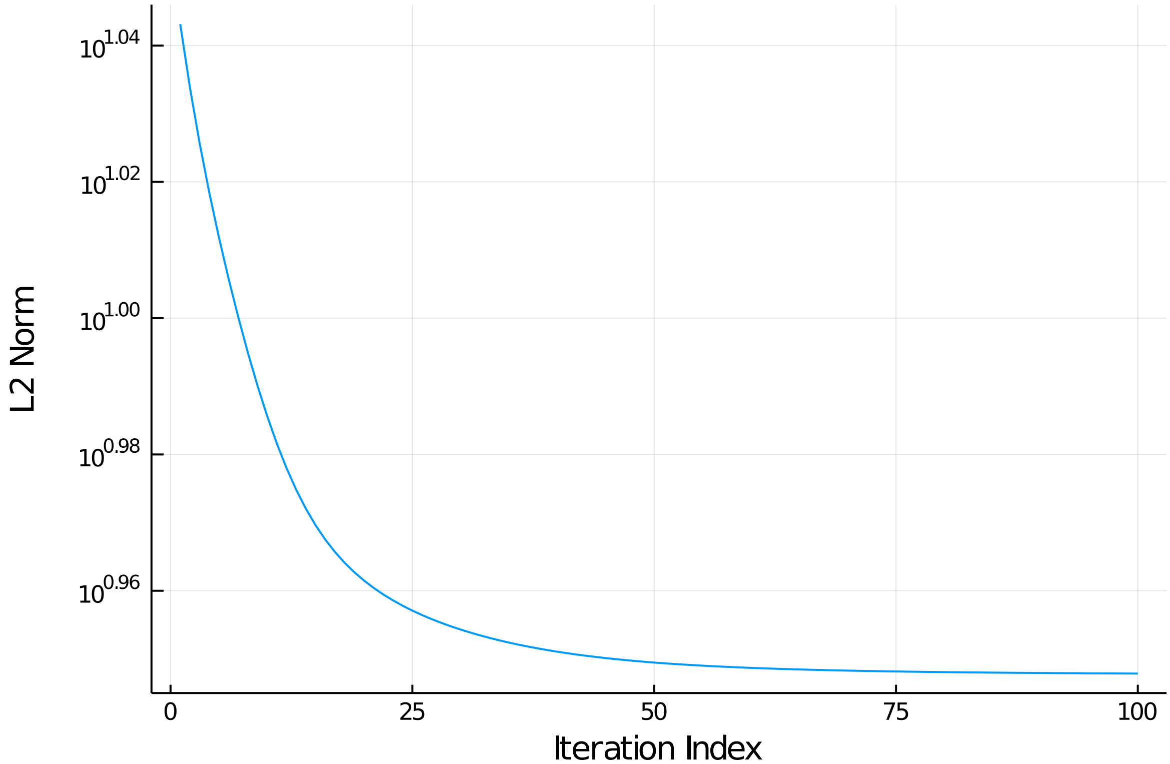

```
W_eg, H_eg, HW_error, L2error = solve_nmf(dataset_conv_eg,W,H,"mse");
```

Convergence Results




```
solve_nmf (generic function with 3 methods)
```

Convergence

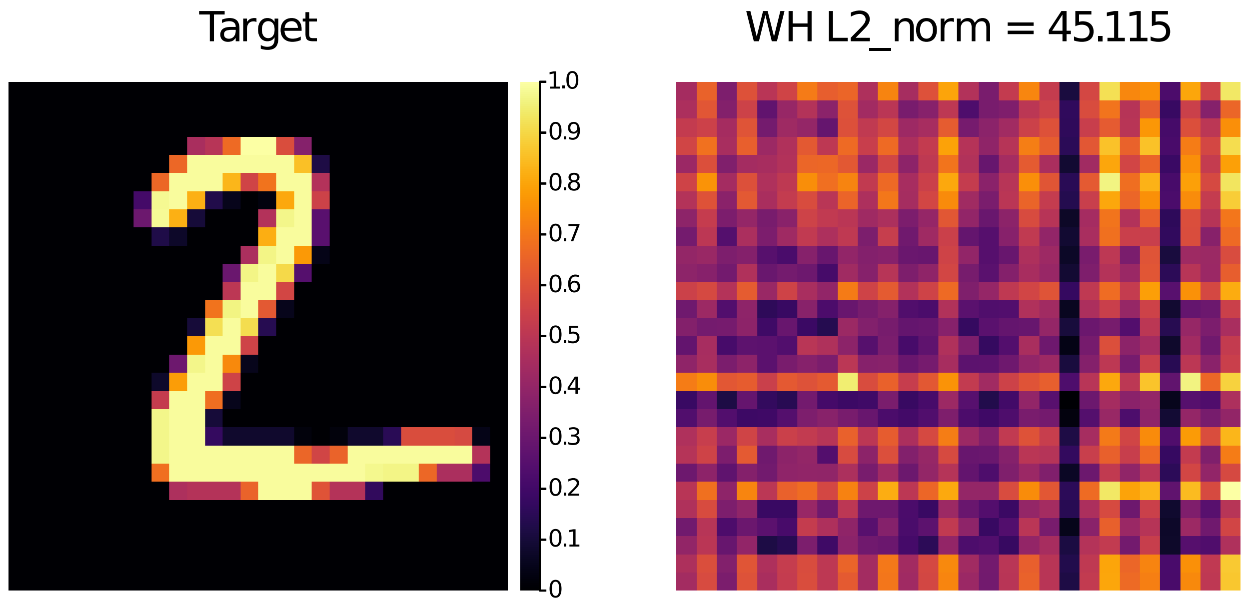


Convergence

example

```
eg2 = reshape(dataset[:, example], 28, 28);
```

```
W2, H2 = NMF.randinit(eg2, r);
```

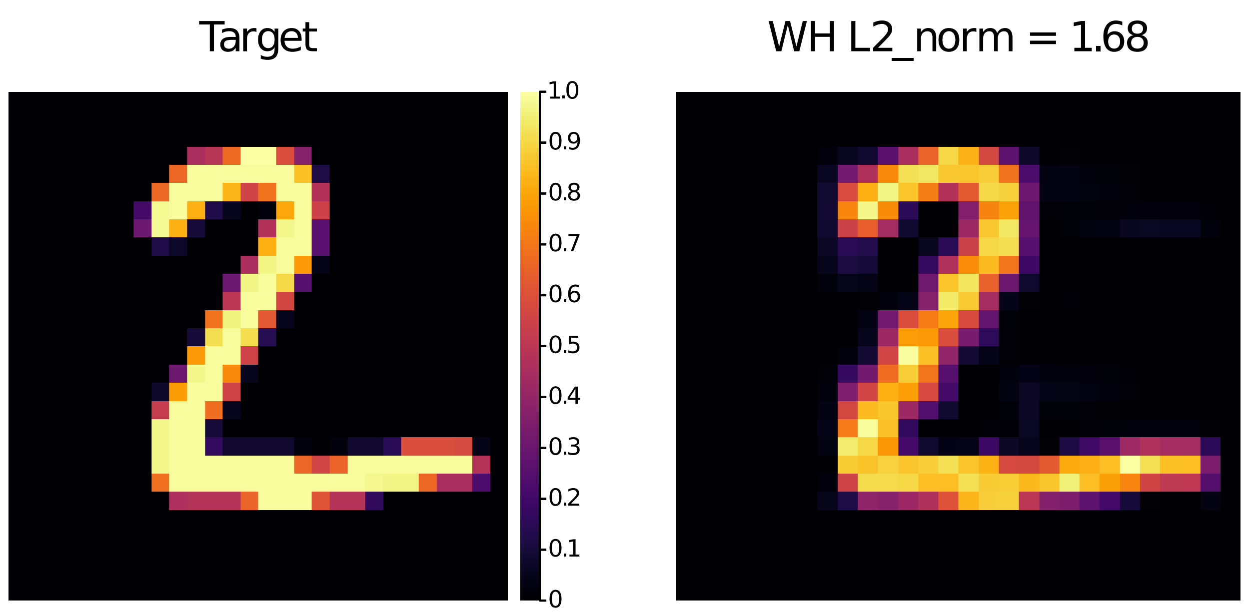



```
TwoColumn(
    heatmap(rotl90(eg2),size=(325,325),title="Target",axis=false),
    heatmap(rotl90(W2*H2),size=(325,325),axis=false,legend=false,
        title="WH L2_norm = $(round(norm(eg2-(W2*H2),2),digits=3))"
    )
)
```

Convergence

features (r) = 1 50

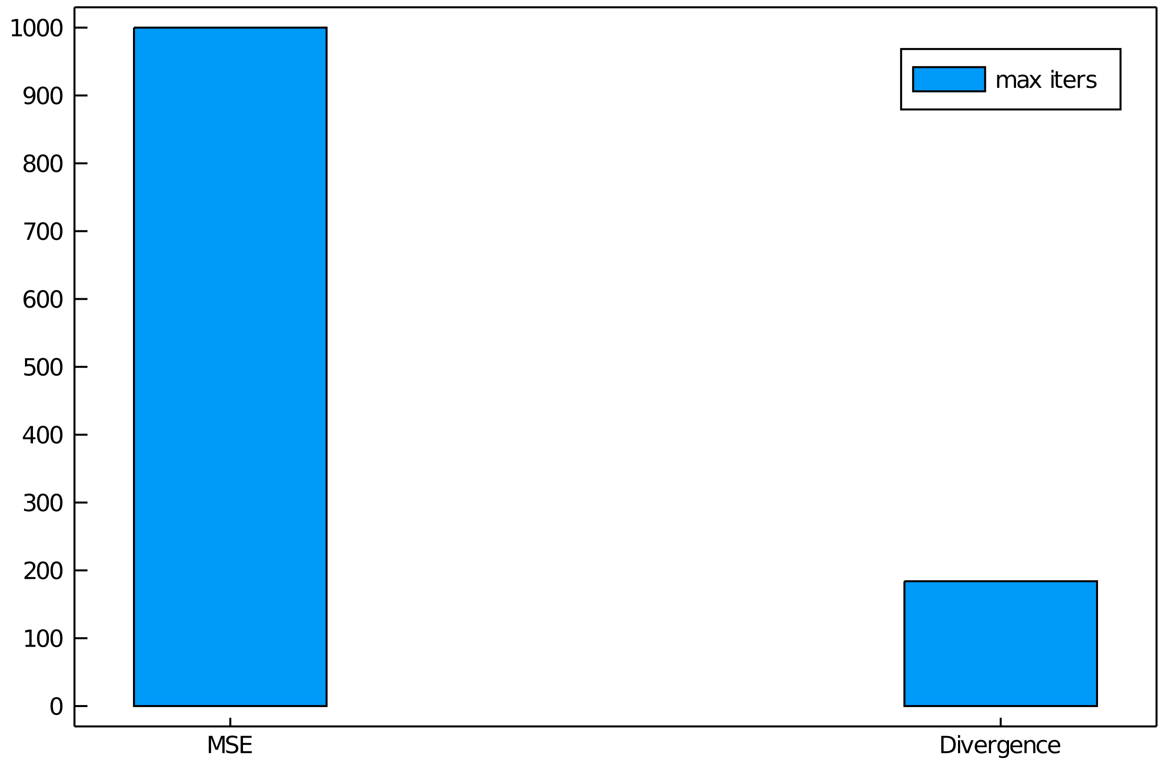
```
result_conv_eg = NMF.solve!(NMF.MultUpdate{Float64}(obj=:mse,maxiter=100), eg2, W2,
H2);
```



Convergence

```
N = 5;
```

```
compare (generic function with 1 method)
```



Summary and Current work

- widely used tool for the analysis of high-dimensional data
- Other algorithms: other divergences (chi-square statistic), Pearson-Neyman distances, etc
- ALS Projected Gradient Methods, Coordinate Descent Methods
- searching for global minima of the factors and factor initialization.
- how to factorize million-by-billion matrices, (Distributed Nonnegative Matrix Factorization) Scalable Nonnegative Matrix Factorization (ScalableNMF) etc.

References

[1]:

> Lee, DD & Seung, HS (1999). Learning the parts of objects by non-negative matrix factorization. Nature.

[2]:

> D. D. Lee and H. S. Seung. Algorithms for non-negative matrixfactorization. InNeural Information Processing Systems, pages 556–562, 2000.

[3]:

> C. Boutsidis and E. Gallopoulos. SVD based initialization: a headstart for nonnegative matrix factorization.Pattern Recognition,41(4):1350–1362, 2008.

[4]:

> S.P. Boyd and L. Vandenberghe.Convex optimization. CambridgeUniversity Press, 2004

[5]:

> Pluto.jl, Julia, https://juliahub.com/docs/Pluto/OJqMt/0.7.4/

[6]:

> LeCun, Y. & Cortes, C. (2010). MNIST handwritten digit database