Nonnegative Matrix Factorization via (alternating) Projected Gradient Descent

Andersen Ang

Mathématique et recherche opérationnelle, UMONS, Belgium

Homepage: angms.science

First draft: August 2, 2017
Last update: November 27, 2020

The approximate NMF problem

- Given $\mathbf{M} \in \mathbb{R}_+^{m \times n}$, find $\mathbf{W} \in \mathbb{R}_+^{m \times r}$, $\mathbf{H} \in \mathbb{R}_+^{r \times n}$ by solving

$$\begin{aligned} [\mathbf{W}^*, \mathbf{H}^*] &= \underset{\mathbf{W},\mathbf{H}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{M} - \mathbf{W}\mathbf{H}\|_F^2 \\ \text{subject to} & \quad \mathbf{W} \geq 0, \mathbf{H} \geq 0. \end{aligned}$$

- a non-convex optimization problem in two variables $\mathbf{W}$ and $\mathbf{H}$
- a constrained optimization problem: $\mathbf{W}$, $\mathbf{H}$ have to be nonnegative
- a NP-Hard problem, see Vavasis2008: On the complexity of nonnegative matrix factorization
- There are many approaches to solve NMF:
  - Alternating projected gradient descent (this document)
  - Exact Block coordinate descent
  - Extrapolated inexact block coordinate descent

Projected gradient descent algorithm — PGD

- PGD is a way to solve **constrained** optimization problem

$$\min_{x \in \mathcal{Q}} f(x)$$

where $\mathcal{Q}$ is the constraint set.

- Starting from a initial feasible point $x_0 \in \mathcal{Q}$, PGD iterates

$$x_{k+1} = P_{\mathcal{Q}}\Big(x_k - t_k \nabla f(x_k)\Big)$$

where $P_{\mathcal{Q}}(\,\cdot\,)$ is the projection operator, which itself is also an optimization problem:

$$P_{\mathcal{Q}}(x_0) = \arg\min_{x \in \mathcal{Q}} \frac{1}{2}\|x - x_0\|_2^2,$$

i.e. given a point x_0, $P_{\mathcal{Q}}$ finds a point $x \in \mathcal{Q}$ that is closest to x_0.

- Note that projected gradient $=$ proximal gradient on indicator function

The PGD algorithm

Algorithm 1: PGD

Result: A solution x that approximately solves $\min_{x \in \mathcal{Q}} f(x)$

Initialization Pick $x_0 \in \mathcal{Q}$

while *stopping condition is not met* **do**

$$x_{k+1} = P_{\mathcal{Q}}\big(x_k - t_k \nabla f(x_k)\big)$$

end

- If f is L-smooth (∇f is L-Lipschitz), we can pick $t_k = \dfrac{1}{L}$.
- For nonnegative cone, $[\cdot]_+ = \max(\cdot, 0)$ is the projection.

The PGD algorithm on NMF

Algorithm 2: The PGD algorithm on NMF

Result: A solution x that approximately solves $\min_{x \in \mathcal{Q}} f(x)$

Initialization Pick inital matrices $W^0 \geq 0$ and $H^0 \geq 0$

while *stopping condition is not met* **do**

$$W^{k+1} = \left[W^k - t_W^k \nabla f(W^k; H^k) \right]_+;$$

$$H^{k+1} = \left[H^k - t_H^k \nabla f(H^k; W^{k+1}) \right]_+;$$

end

- This algorithm is an inexact block coordinate descent with PGD update.
- That is, we update each block one-by-one, where the sub-problem, e.g.

$$\operatorname*{argmin}_{\mathbf{W} \geq 0} \frac{1}{2} \|\mathbf{M} - \mathbf{W}\mathbf{H}^k\|_F^2$$

  is not solved exactly: the projected gradient update on $\mathbf{W}$ does not solve the above minimization problem.

Lipschitz constant step size

- For $f(W,H) = \frac{1}{2}\|X - WH\|_F^2$, the gradients are

$$\nabla_W f(W,H) = (WH - X)H^T, \quad \nabla_H f(W,H) = W^T(WH - X)$$

Key inequality: for any matrices A and B we have $\|AB\|_F \leq \|A\|_F \|B\|_2$.

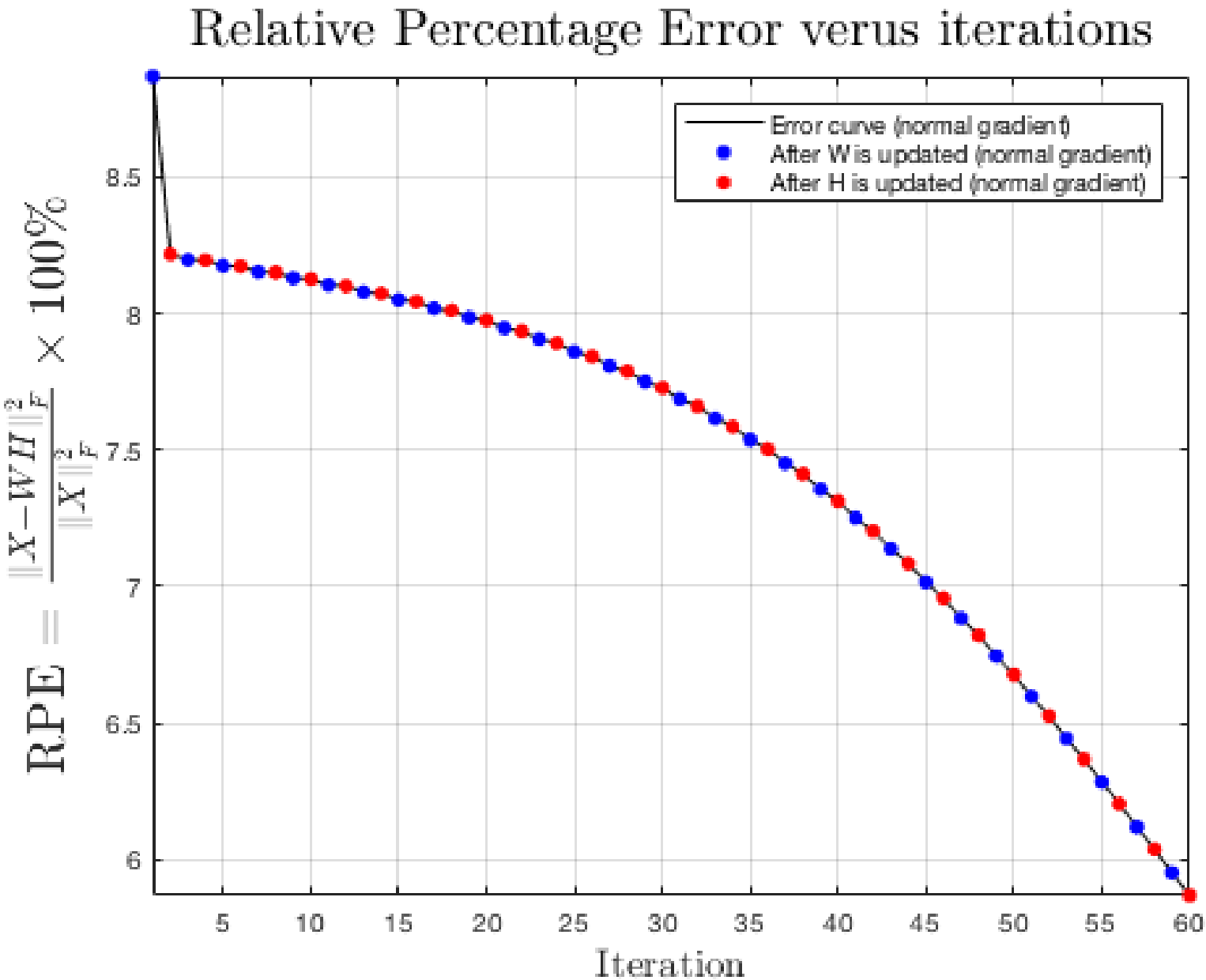
$$\begin{aligned} \|\nabla_W f(W_1) - \nabla_W f(W_2)\|_F &= \|(W_1H - X)H^T - (W_2H - X)H^T\|_F \\ &= \|(W_1 - W_2)HH^T\|_F \\ &\leq \|HH^T\|_2 \cdot \|W_1 - W_2\|_F \end{aligned}$$

Similarly

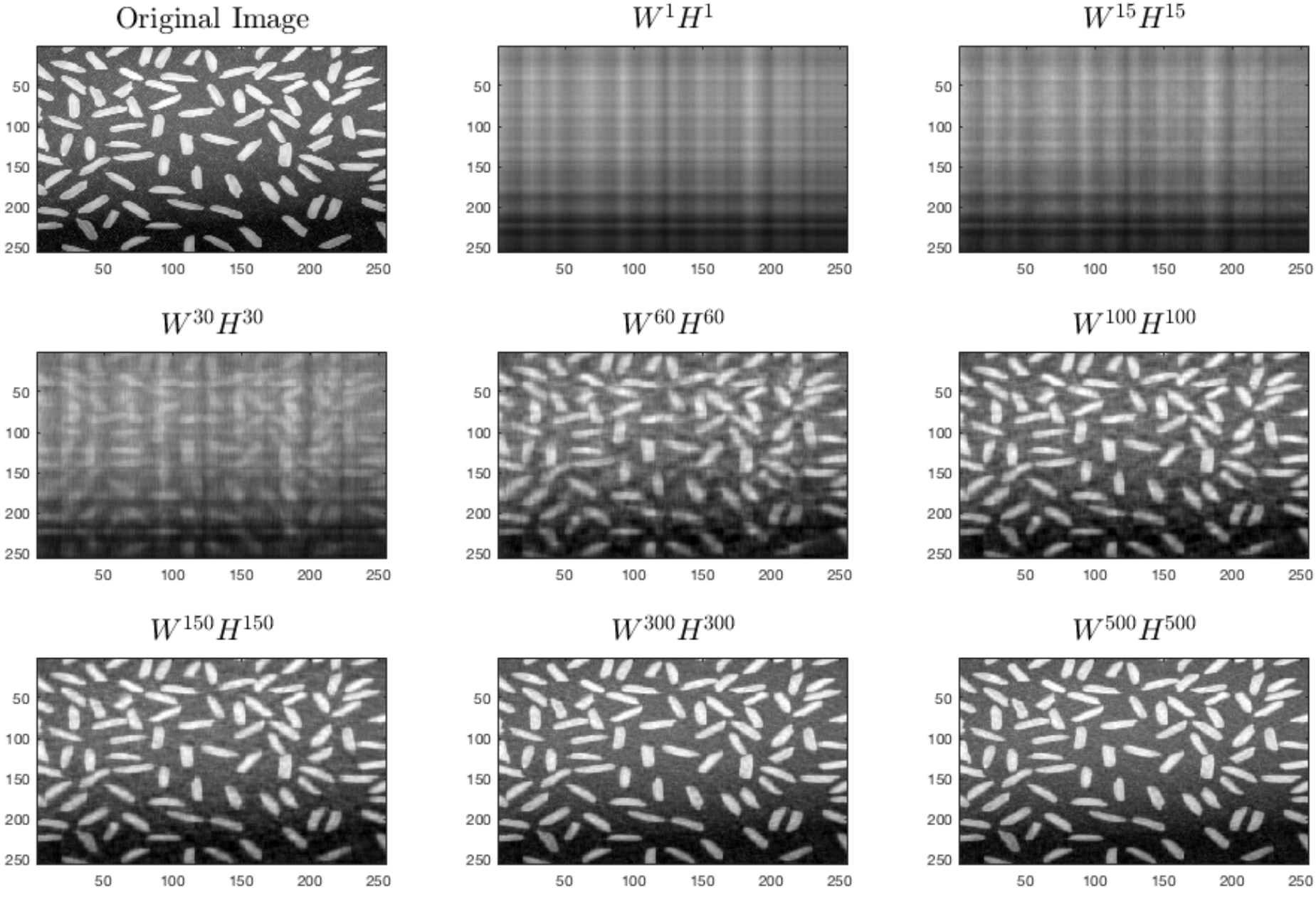
$$\|\nabla_H f(H_1) - \nabla_H f(H_2)\|_F \leq \|W^TW\|_2 \cdot \|H_1 - H_2\|_F$$

We can put $t_W^k = \frac{1}{\|HH^T\|_2}$ and $t_H^k = \frac{1}{\|W^TW\|_2}$ in the previous algorithm.

An example : rice image $(m, n, r) = (256, 256, 86)$



Reconstruction of the rice image at different iteration



PGD on NMF with optimal stepsize

Stepsize t can be selected that achieve maximum decrease in objective function.

$$t_W^* = \underset{t}{\operatorname{argmin}}\ f\Big(W - tg(W), H\Big), \quad t_H^* = \underset{t}{\operatorname{argmin}}\ f\Big(W, H - tg(H)\Big)$$

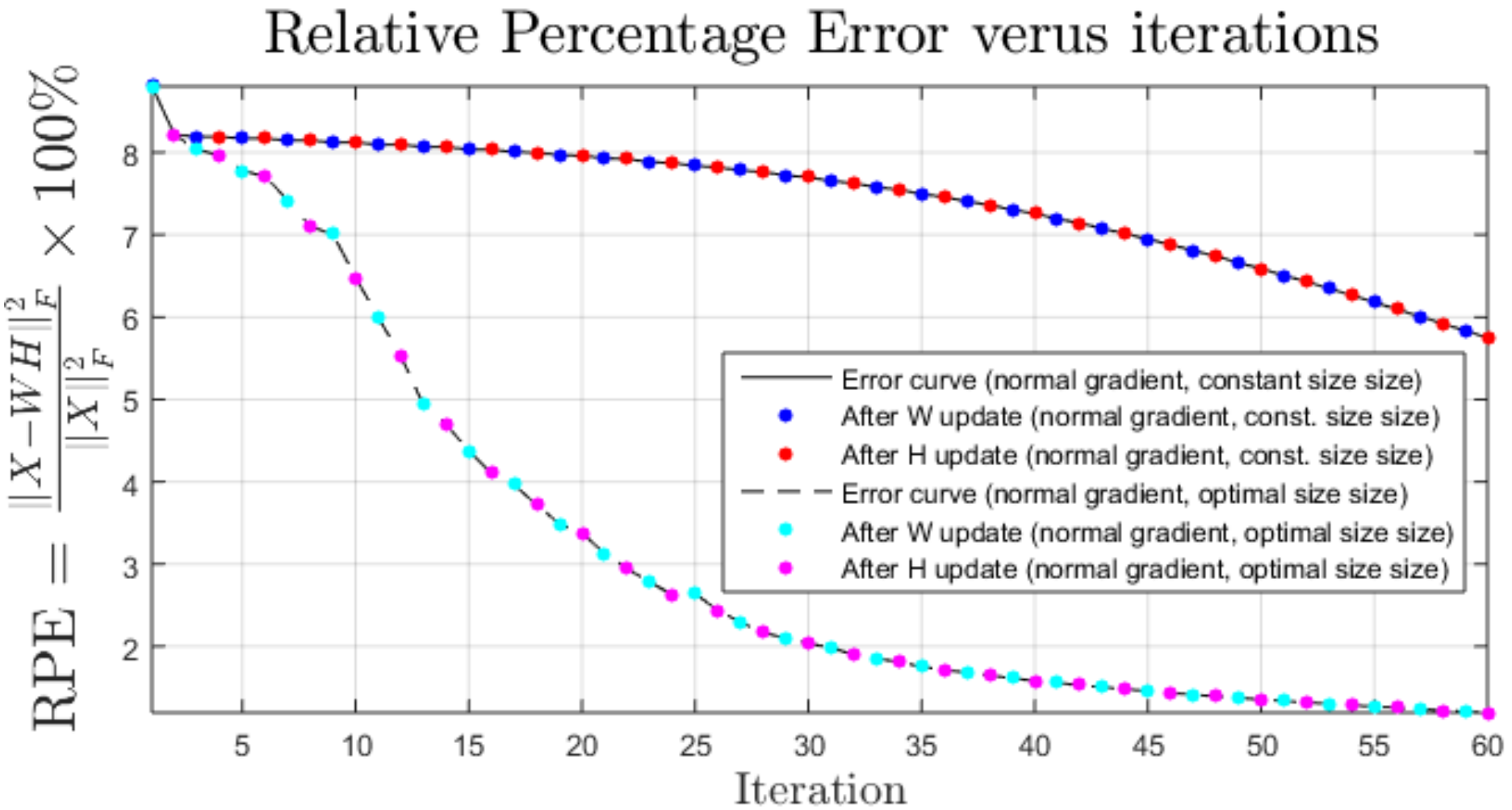
where $g(W) = \nabla_W f(W,H)$ and $g(H) = \nabla_H f(W,H)$. Then,

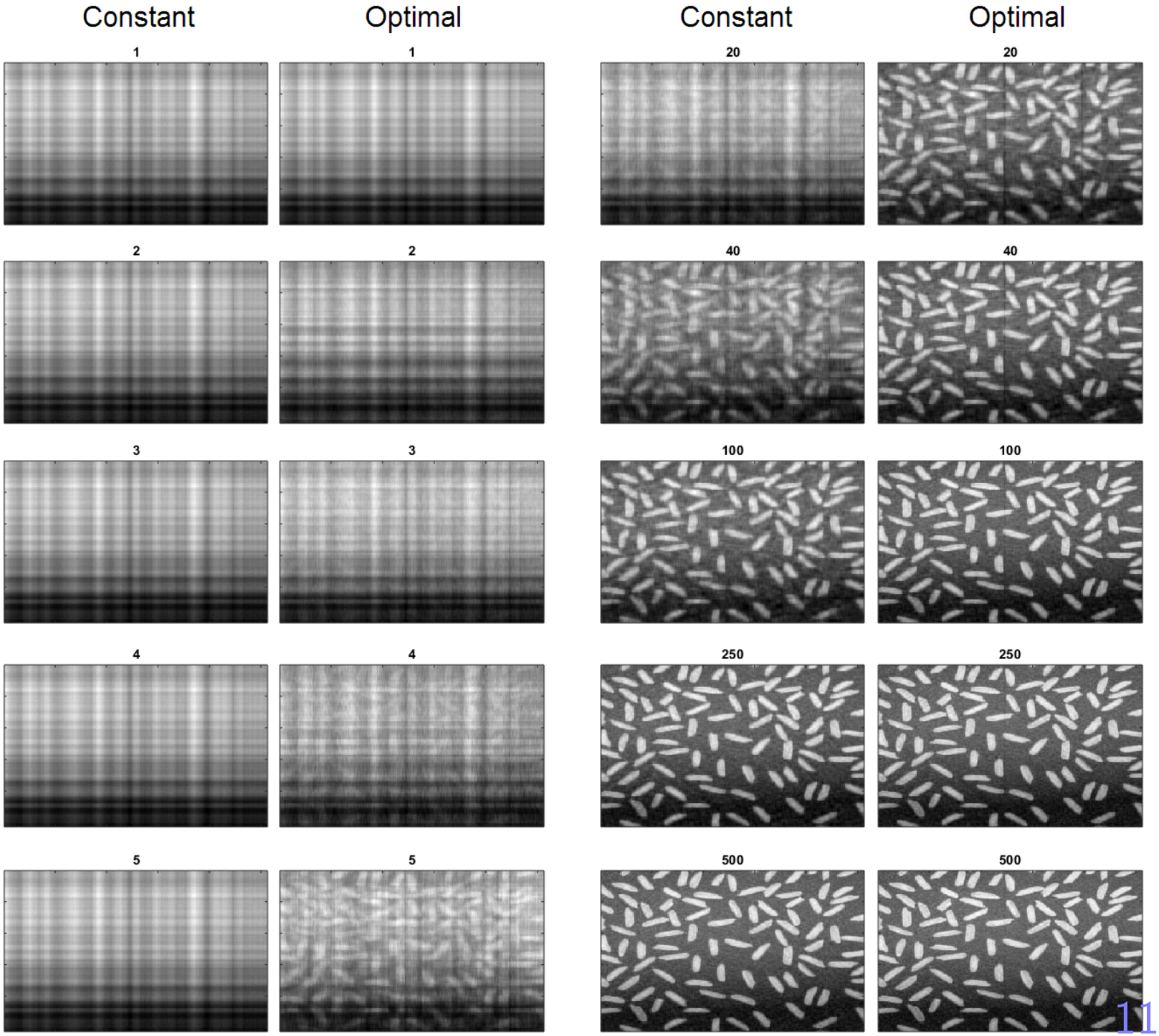
$$\begin{aligned} \frac{\partial f}{\partial t_W} &= \operatorname{Tr}\left(X - WH + tg(W)H\right)^T g(W)H \\ t_W^* &= \frac{\operatorname{Tr}\Big((WH - X)^T g(W)H\Big)}{\operatorname{Tr}\Big(H^T g(W)^T g(W)H\Big)} \end{aligned}$$

$$\begin{aligned} \frac{\partial f}{\partial t_H} &= \operatorname{Tr}\left(X - WH + tWg(H)\right)^T Wg(H) \\ t_H^* &= \frac{\operatorname{Tr}\Big((WH - X)^T Wg(H)\Big)}{\operatorname{Tr}\Big(g(H)^T W^T Wg(H)\Big)} \end{aligned}$$

However, using optimal size, the computational cost is much higher than constant stepsize.

Rice image example




Constant
Optimal
Constant
Optimal
1
1
20
20
2
2
40
40
3
3
100
100
4
4
250
250
5
5
500
500

The Accelerated PG algorithm

- We can apply Nesterov's acceleration on PGD, which adds momentum on the sequence.

- To apply momentum on NMF, the easiest way is to apply it on the exact block coordinate descent scheme:
  1. Pick inital matrices $W^0 \in \mathbb{R}_+^{m\times r}$ and $H^0 \in \mathbb{R}_+^{r\times n}$ and parameters.
  2. On W, loop until stopping condition is met
     2.1 Update and project
     2.2 Extrapolate
  3. On H, loop until stopping condition is met
     3.1 Update and project
     3.2 Extrapolate
  4. Repeat 2 and 3 until converge.

- If we apply momentum on the inexact block coordinate descent scheme, the situation becomes much much more complicated as the theory changes drastically. In this case the algorithm belong to the Bolte-Sabach-Teboulle's PALM, or Xu-Yin's APG, or Hien-Gillis-Patrinos's IBP.

Exact BCD algorithm with momentum for NMF

1. (On W) Initalize W^0, ϵ, set $V^0 = W^0$, $\lambda^0 = 0$, then loop until stopping condition is met :
   - 1.1 $\lambda^k = \frac{1}{2}(1 + \sqrt{1 + 4\lambda_{k-1}^2})$, $\gamma^k = \frac{1 - \lambda^{k-1}}{\lambda^k}$
   - 1.2 Update: $V^k = \max\left([W^k - t_W^k \nabla_W f(W^k; H)]_{ij}, \epsilon\right)$
   - 1.3 Extrapolation : $W^k = (1 - \gamma^k)V^k + \gamma V^{k-1}$
2. (On H) Initalize H^0, ϵ, set $G^0 = H^0, \lambda^0 = 0$, then loop until stopping condition is met :
   - 2.1 $\lambda^k = \frac{1}{2}(1 + \sqrt{1 + 4\lambda_{k-1}^2})$, $\gamma^k = \frac{1 - \lambda^{k-1}}{\lambda^k}$
   - 2.2 Update: $G^k = \max\left([H^k - t_H^k \nabla_H f(H^k; W)]_{ij}, \epsilon\right)$
   - 2.3 Extrapolation : $H^k = (1 - \gamma^k)G^k + \gamma G^{k-1}$

The acceleration is achieved by extrapolating current W, H by the coupling variables V, G with the extrapolation weights $\{\gamma^k\}$.

MATLAB code (click me)

The wrong way to apply momentum on NMF

Note in each Update step the other variable is held fix (without the superscript k). That is, the matrix H when updating W^k for $k = 1, 2, ...$ is all the same. The following shows a wrong way to apply APGD on NMF :

1. Initalize W^0, H^0 and ϵ, set $V^0 = W^0, G^0 = H^0$, and $\lambda^0 = 0$
2. Loop until stopping condition is met
   - 2.1 Update: $V^k = \max\left([W^k - t_W^k \nabla_W f(W^k; H^k)]_{ij}, \epsilon\right)$
   - 2.2 Extrapolation : $W^k = (1 - \gamma^k)V^k + \gamma V^{k-1}$
   - 2.3 Update: $G^k = \max\left([H^k - t_H^k \nabla_H f(H^k; W^{k+1})]_{ij}, \epsilon\right)$
   - 2.4 Extrapolation : $H^k = (1 - \gamma^k)G^k + \gamma G^{k-1}$

Why it is wrong : consider step 2, at $k = 1$

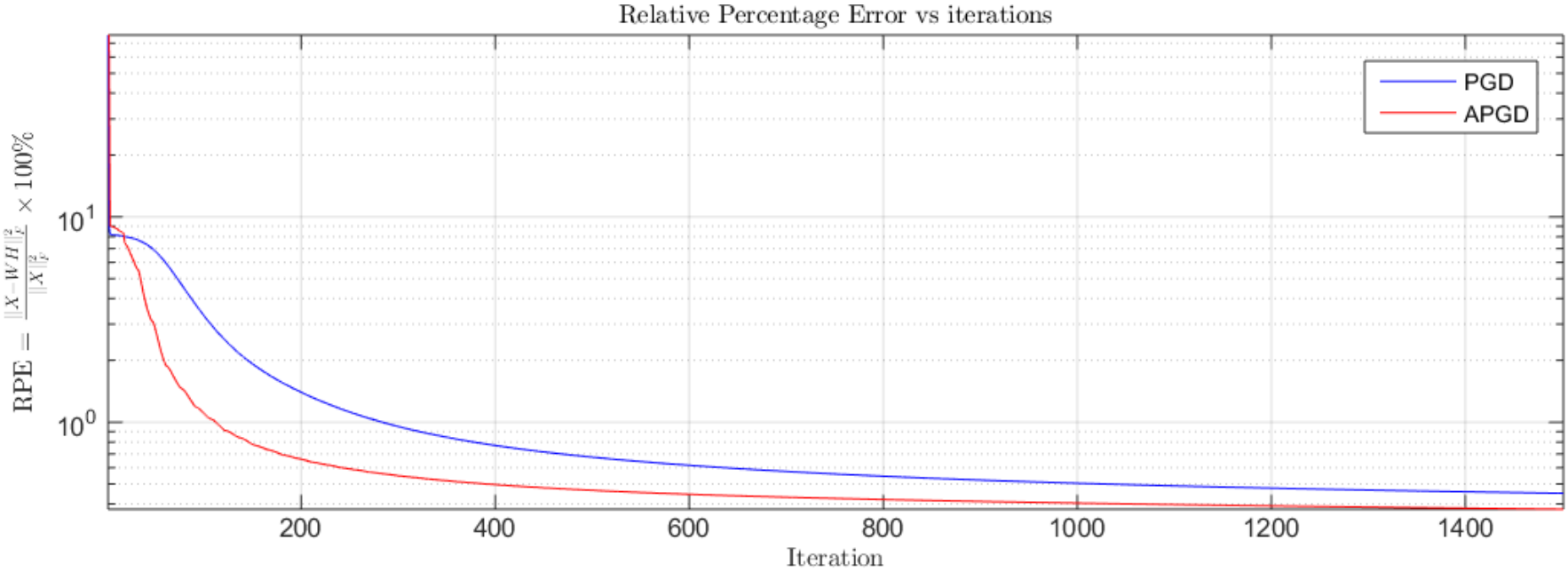
$$k = 1 \qquad V^1 = \max\left([W^1 - t_W^1 \nabla_W f(W^1; H^1)]_{ij}, \epsilon\right).$$

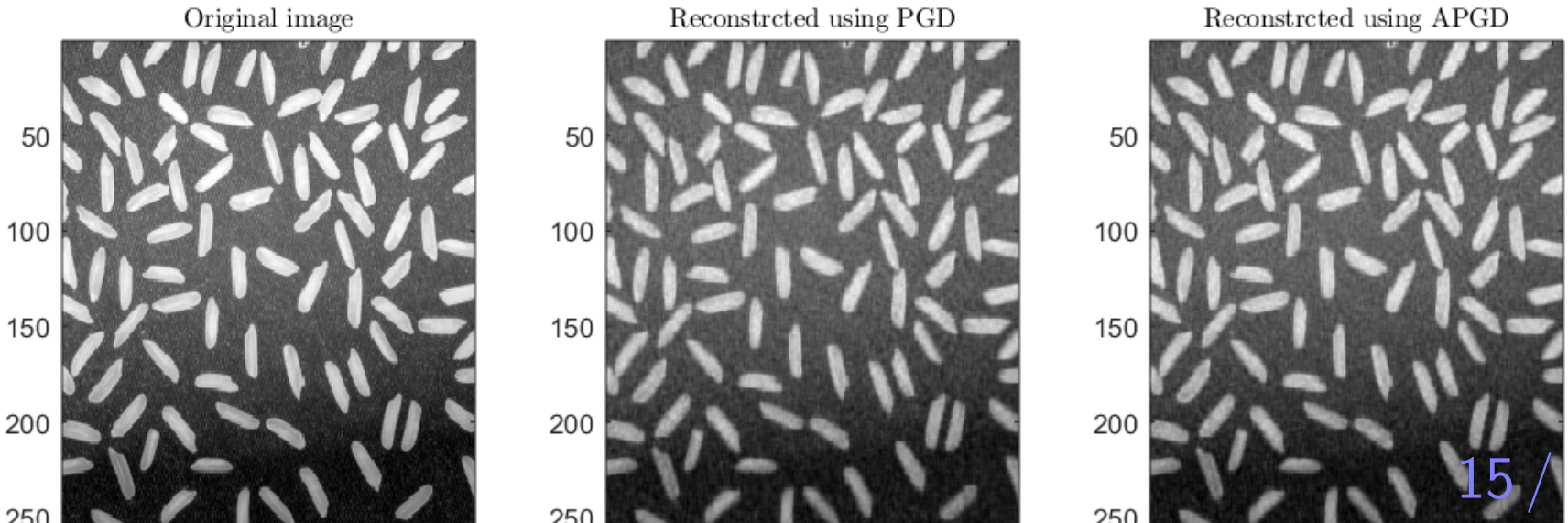
$$k = 2 \qquad V^2 = \max\left([W^2 - t_W^2 \nabla_W f(W^2; H^2)]_{ij}, \epsilon\right)$$

Unless variable H is already optimal (H converges and so $H^1 = H^2$) otherwise the update of H at $k = 1$ makes $H^2 \neq H^1$ and so the optimization problem on minimizing W at $k = 1$ is different from that at $k = 2$.
Therefore, for applying extrapolation directly on the block variable, we have to use the framework of PALM, APG or IBP.

PGD(inexact BCD with gradient update) vs APG(exact BCD with momentum)

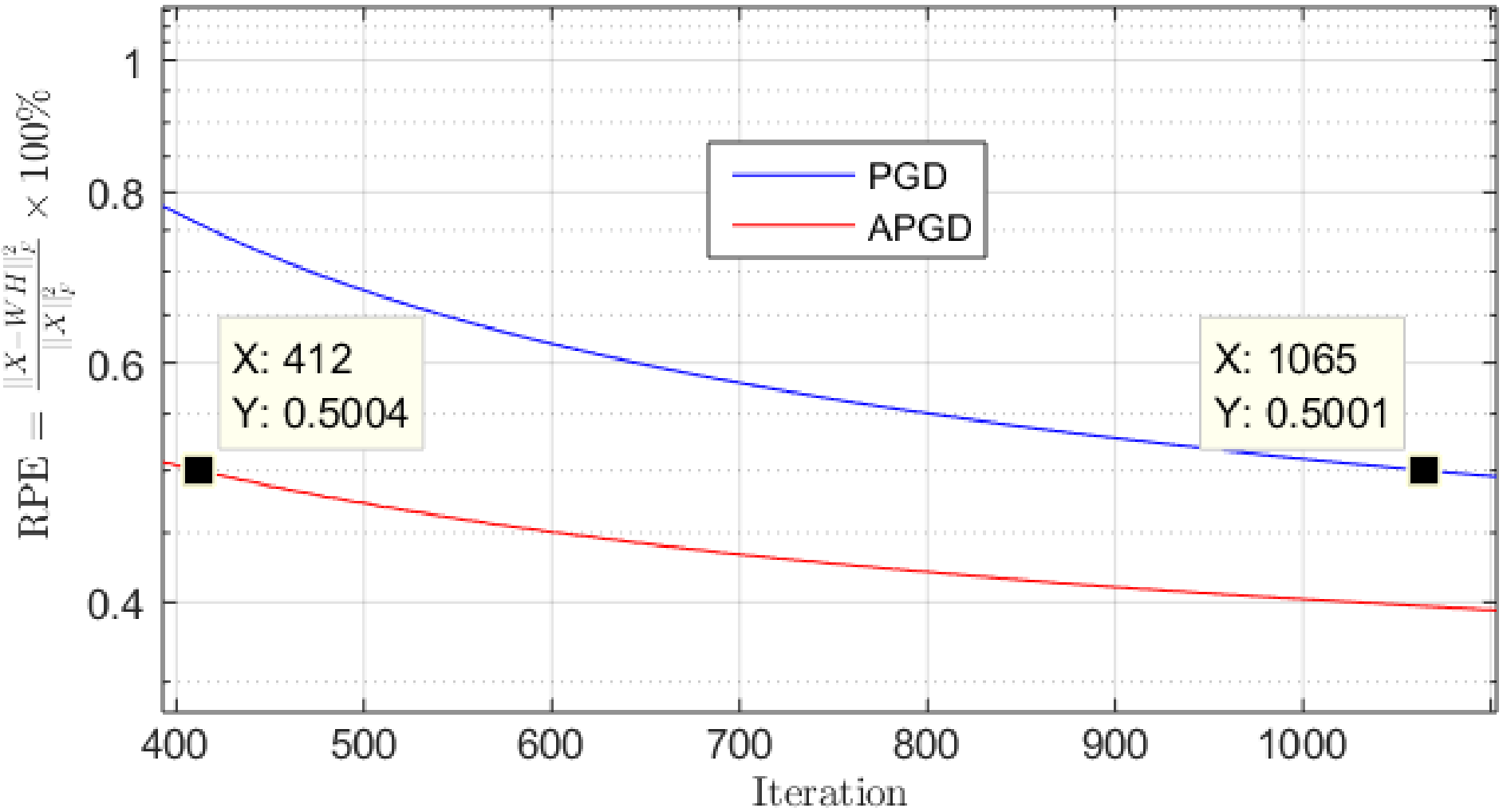
Example : MATLAB rice image $(m, n, r) = (256, 256, 64)$





PGD vs APGD

Here APGD takes about 50% of the computation to achieve the same amount of fitting error of PGD.



Further discussion

Recall that accelerated GD for unconstrained convex problem with single variable is not monotone, it has ripples in the trace of the objective function value.

In this case adaptive restart can be used. That is, if the extrapolated solution produces objective function value higher than that of the previous iterate, the extrapolated solution is thrown away, the momentum and extrapolation weight are all reset to initial state.

That is : if at step k, $f(W^k) > f(W^{k-1})$:

- $W^k = W^{k-1}$
- $V^k = W^k$
- $\lambda^k = 0$

Last page - summary

- Alternating projected gradient descent for NMF
  MATLAB code (click me)

- Alternating projected gradient descent = Inexact block coordinate descent

- Exact block coordinate descent with internal Nesterov's acceleration

End of document