Nonnegative Matrix Factorization with Regularization

Andersen Ang

ECS, Uni. Southampton, UK
andersen.ang@soton.ac.uk

Homepage angms.science

Version: February 7, 2024

First draft: June 6, 2017
Mathématique et de Recherche opérationnelle
Faculté polytechnique de Mons, UMONS
Belgium

Content

Setup: the regularized numerical NMF Problem

- Input: $\boldsymbol{X} \in \mathbb{R}_+^{m \times n}$
  - m rows
  - n columns
  - all entries are nonnegative $X_{ij} \geq 0$

- Task: find two nonnegative matrices $\boldsymbol{W} \in \mathbb{R}_+^{m \times r}$ and $\boldsymbol{H} \in \mathbb{R}_+^{r \times n}$ such that

$$(\mathcal{P}) \ : \ \left(\boldsymbol{W}^*, \boldsymbol{H}^*\right) \ = \ \underset{\boldsymbol{W} \geq \boldsymbol{0}, \boldsymbol{H} \geq \boldsymbol{0}}{\operatorname{argmin}} \ \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2 + \lambda g(\boldsymbol{W}, \boldsymbol{H}).$$

  where we consider $(\boldsymbol{W}^*, \boldsymbol{H}^*)$ is a solution to Problem $(\mathcal{P})$ if the objective value is small enough

  - $f(\boldsymbol{W}, \boldsymbol{H}) = \frac{1}{2}\|\boldsymbol{W}\boldsymbol{H} - \boldsymbol{X}\|_F^2$ is the primary objective function
    - it is called data-fitting
    - it is measured in Frobenius norm (i.e. Euclidean distance)
  - $g(\boldsymbol{W}, \boldsymbol{H})$ is the regularization, here we assume g is differentiable
  - $\lambda \geq 0$ is the weight
  - $r \in \mathbb{N}$ is the factorization rank

Inexact Block-Coordinate Descent based on gradient descent update

$$(\mathcal{P}) \ : \ (\boldsymbol{W}^*, \boldsymbol{H}^*) \ = \ \underset{\boldsymbol{W}\geq \boldsymbol{0}, \boldsymbol{H}\geq \boldsymbol{0}}{\operatorname{argmin}} \underbrace{\frac{1}{2}\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2}_{=:f} + \lambda g(\boldsymbol{W}, \boldsymbol{H}).$$

- We can solve $\mathcal{P}$ by inexact Block-Coordinate Descent (BCD):
$$\boldsymbol{W}_1 = \text{update}(\boldsymbol{W}, \boldsymbol{H}_0), \ \boldsymbol{H}_1 = \text{update}(\boldsymbol{W}_1, \boldsymbol{H}), \ \boldsymbol{W}_2 = \text{update}(\boldsymbol{W}, \boldsymbol{H}_1), \ \boldsymbol{H}_2 = \text{update}(\boldsymbol{W}_2, \boldsymbol{H}), \ \cdots$$
- For the "update", there are many options, one option is projected gradient descent
$$\begin{aligned} \boldsymbol{W}_{k+1} &= \Big[\boldsymbol{W}_k - \alpha \nabla_{\boldsymbol{W}} (f + \lambda g)(\boldsymbol{W}_k, \boldsymbol{H}_k)\Big]_+ \\ \boldsymbol{H}_{k+1} &= \Big[\boldsymbol{H}_k - \beta \nabla_{\boldsymbol{H}} (f + \lambda g)(\boldsymbol{W}_{k+1}, \boldsymbol{H}_k)\Big]_+ \end{aligned}$$
  - α, β denote stepsize
  - $\nabla_{\boldsymbol{W}}$ denotes the matrix-wise gradient with respect to $\boldsymbol{W}$. For compactness we will just write ∇
  - $[x]_+$ denotes the projection of x onto the nonnegative orthant. This operation makes sure the variable is feasible (satisfying the $\geq \boldsymbol{0}$ constraint)
- The terms $\nabla f(\boldsymbol{W}, \boldsymbol{H})$ are already derived (see here for the expansion of f) and thus
$$\begin{aligned} \boldsymbol{W}_{k+1} &= \Big[\boldsymbol{W}_k - \alpha\Big(\boldsymbol{W}_k \boldsymbol{H}_k \boldsymbol{H}_k^\top - \boldsymbol{X}\boldsymbol{H}_k^\top + \lambda \nabla g(\boldsymbol{W}_k, \boldsymbol{H}_k)\Big)\Big]_+ \\ \boldsymbol{H}_{k+1} &= \Big[\boldsymbol{H}_k - \beta\Big(\boldsymbol{W}_k^\top \boldsymbol{W}_k \boldsymbol{H}_k - \boldsymbol{W}_k^\top \boldsymbol{X} + \lambda \nabla g(\boldsymbol{W}_{k+1}, \boldsymbol{H}_k)\Big)\Big]_+ \end{aligned}$$

Orthogonal NMF

- NMF with orthogonality constraints

$$\underset{\boldsymbol{W}\geq 0,\boldsymbol{H}\geq 0}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{W}\boldsymbol{H}\|_F^2 \text{ s.t. } \boldsymbol{W}^\top\boldsymbol{W}=\boldsymbol{I} \text{ and } \boldsymbol{H}^\top\boldsymbol{H}=\boldsymbol{I}.$$

- To enforce orthogonality, we can consider adding the penalty of orthogonality into the optimization formulation

$$(\mathcal{P}_{\text{ONMF}}) \ : \ \underset{\boldsymbol{W}\geq 0,\boldsymbol{H}\geq 0}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{W}\boldsymbol{H}\|_F^2+\frac{\lambda}{4}\|\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}\|_F^2+\frac{\mu}{4}\|\boldsymbol{H}^\top\boldsymbol{H}-\boldsymbol{I}\|_F^2.$$

  - Here $g(\boldsymbol{W},\boldsymbol{H})$ can be split into two independent functions $\|\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}\|_F^2$ and $\|\boldsymbol{H}^\top\boldsymbol{H}-\boldsymbol{I}\|_F^2$.
  - The meaning is that, we are minimizing the whole function, hence we want the whole expression to have small value. Then if $\|\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}\|_F^2$ is small, we have $\boldsymbol{W}^\top\boldsymbol{W}\approx\boldsymbol{I}$, which is what we want.

- Now we need to know $\nabla_{\boldsymbol{W}}\|\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}\|_F^2$ and $\nabla_{\boldsymbol{H}}\|\boldsymbol{H}^\top\boldsymbol{H}-\boldsymbol{I}\|_F^2$.

- You will soon see why we divided-by-4

Evaluating gradient of $\|\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}\|_F^2$

$$
\begin{aligned}
\|\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}\|_F^2 &= \mathrm{Tr}\Big((\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I})^\top(\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I})\Big)\\
&= \mathrm{Tr}\Big((\boldsymbol{W}^\top\boldsymbol{W})^\top(\boldsymbol{W}^\top\boldsymbol{W})-2\boldsymbol{W}^\top\boldsymbol{W}+\boldsymbol{I}\Big)\\
\nabla\|\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}\|_F^2 &= \nabla\mathrm{Tr}\Big((\boldsymbol{W}^\top\boldsymbol{W})^\top(\boldsymbol{W}^\top\boldsymbol{W})\Big)-2\nabla\mathrm{Tr}\Big(\boldsymbol{W}^\top\boldsymbol{W}\Big)\\
\nabla\mathrm{Tr}\Big(\boldsymbol{W}^\top\boldsymbol{W}\Big) &= 2\boldsymbol{W}\\
\nabla\mathrm{Tr}\Big((\boldsymbol{W}^\top\boldsymbol{W})^\top(\boldsymbol{W}^\top\boldsymbol{W})\Big) &= 4\boldsymbol{W}(\boldsymbol{W}^\top\boldsymbol{W})\\
\nabla\|\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}\|_F^2 &= 4\boldsymbol{W}(\boldsymbol{W}^\top\boldsymbol{W})-2(2\boldsymbol{W})\\
&= 4\boldsymbol{W}(\boldsymbol{W}^\top\boldsymbol{W})-4\boldsymbol{W}\boldsymbol{I}_r\\
&= 4\boldsymbol{W}(\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}_r)
\end{aligned}
$$

How to find $\nabla\mathrm{Tr}\Big((\boldsymbol{W}^\top\boldsymbol{W})^\top(\boldsymbol{W}^\top\boldsymbol{W})\Big)$?

- Chain rule: see details here
- By Matrix Cookbook formula (123)

$$\frac{\partial\mathrm{Tr}\Big\{\boldsymbol{B}^\top\boldsymbol{X}^\top\boldsymbol{C}\boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{C}\boldsymbol{X}\boldsymbol{B}\Big\}}{\partial\boldsymbol{X}}=\boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{C}\boldsymbol{X}\boldsymbol{B}\boldsymbol{B}^\top+\boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{B}\boldsymbol{B}^\top\boldsymbol{X}^\top\boldsymbol{C}^\top\boldsymbol{X}+\boldsymbol{C}\boldsymbol{X}\boldsymbol{B}\boldsymbol{B}^\top\boldsymbol{X}^\top\boldsymbol{C}\boldsymbol{X}+\boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{B}\boldsymbol{B}^\top$$

Note that here $\boldsymbol{W}$ can be any matrix, so the same conclusion holds for $\nabla\|\boldsymbol{H}^\top\boldsymbol{H}-\boldsymbol{I}\|_F^2=4\boldsymbol{H}(\boldsymbol{H}^\top\boldsymbol{H}-\boldsymbol{I}_n)$.

Now we have...

$$\nabla\|\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}_r\|_F^2=4\boldsymbol{W}(\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}),\qquad \nabla\|\boldsymbol{H}^\top\boldsymbol{H}-\boldsymbol{I}_n\|_F^2=4\boldsymbol{H}(\boldsymbol{H}^\top\boldsymbol{H}-\boldsymbol{I}).$$

- Then

$$\frac{\lambda}{4}\nabla\|\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}_r\|_F^2=\lambda\boldsymbol{W}\big(\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}_r\big),\qquad \frac{\mu}{4}\nabla\|\boldsymbol{H}^\top\boldsymbol{H}-\boldsymbol{I}_n\|_F^2=\mu\boldsymbol{H}\big(\boldsymbol{H}^\top\boldsymbol{H}-\boldsymbol{I}_n\big)$$

- Thus

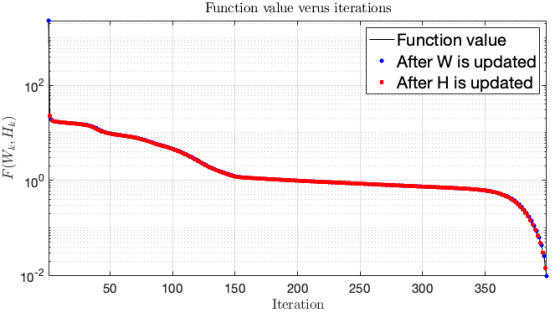
$$\boldsymbol{W}_{k+1} = \Big[\boldsymbol{W}_k-\alpha\Big(\boldsymbol{W}_k\boldsymbol{H}_k\boldsymbol{H}_k^\top-\boldsymbol{X}\boldsymbol{H}_k^\top+\lambda\boldsymbol{W}_k\big(\boldsymbol{W}_k^\top\boldsymbol{W}_k-\boldsymbol{I}_r\big)\Big)\Big]_+$$

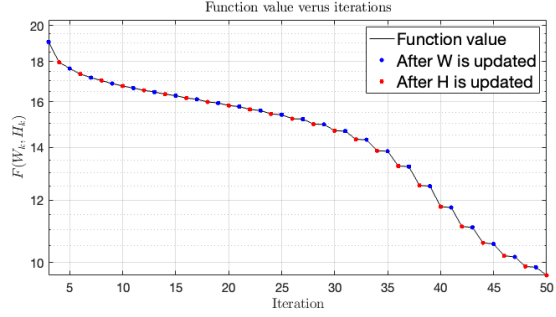
$$\boldsymbol{H}_{k+1} = \Big[\boldsymbol{H}_k-\beta\Big(\boldsymbol{W}_k^\top\boldsymbol{W}_k\boldsymbol{H}_k-\boldsymbol{W}_k^\top\boldsymbol{X}+\mu\boldsymbol{H}_k\big(\boldsymbol{H}_k^\top\boldsymbol{H}_k-\boldsymbol{I}_n\big)\Big)\Big]_+$$

- Remarks
  - Do not set α,β too large
  - Imagine in the extreme case that $\alpha=\infty$. You can see that in case the gradient bracket is positive, then everything inside the $[\cdot]_+$ is a strictly negative number, and then $[\cdot]_+$ will set everything to zero
  - How to tune α,β: there are many ways, for simplicity we consider bisection
  - Is there a theoretical bound for the value of α,β: it seems no. There is no a global constant Lipschitz constant for $\nabla\|\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I}_r\|_F^2$

Quick-and-dirty code

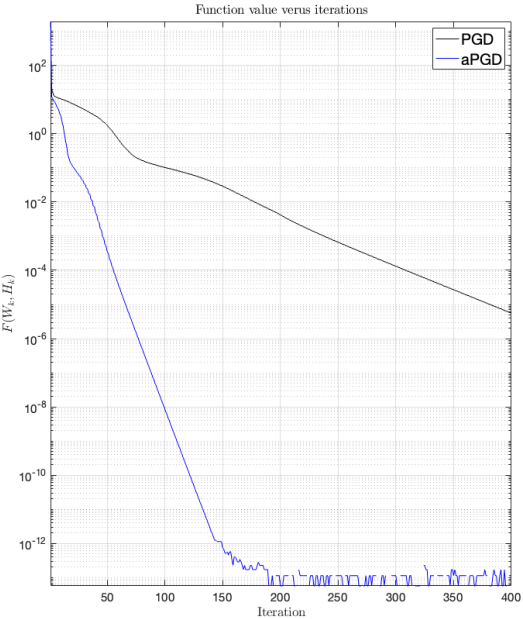
```
for k = 1 : itermax
    %% Update of W
    alpha = 1e-3; % initial stepsize

    F_pre = computeF(X,W,H,Ir,In,lambda,mu);
      HHt = H*H';
      XHt = X*H';

      % Block coordinate descent on W
      for jj = 1 : 25
              WtW = W'*W;
            gradW = W*HHt - XHt + lambda*W*(WtW - Ir);

            Wtemp = max(0,  W - alpha*gradW);

            % bisection line search on stepsize
            F_cur = computeF(X,Wtemp,H,Ir,In,lambda,mu);
              if F_cur > F_pre
                alpha = alpha/2;
                Wtemp = max(0,  W - alpha*gradW);
                F_cur = computeF(X,Wtemp,H,Ir,In,lambda,mu);
              end
                W = Wtemp;
          F(1,k) = computeF(X,W,H,Ir,In,lambda,mu);
    end

    %% Update of H
    alpha = 1e-3; % initial stepsize

    F_pre = computeF(X,W,H,Ir,In,lambda,mu);
      WtW = W'*W;
      WtX = W'*X;

      % Block coordinate descent on H
      for jj = 1 : 25
              HtH = H'*H;
            gradH = WtW * H - WtX +mu*H*(HtH-In);
            Htemp = max(H - alpha*gradH, 0);

            % bisection line search on stepsize
            F_cur = computeF(X,W,Htemp,Ir,In,lambda,mu);
              if F_cur > F_pre
                alpha = alpha/2;
                Htemp = max(H - alpha*gradH, 0);
                F_cur = computeF(X,W,Htemp,Ir,In,lambda,mu);
              end
                H = Htemp;
          F(2,k) = computeF(X,W,H,Ir,In,lambda,mu);
    end
end
```





PGD with Nesterov's acceleration

- Why PGD: we can easily add Nesterov's acceleration in the PGD update within the BCD framework.
- On using Nesterov's acceleration solving NNLS
- What about multiplicative update?
  Slow, not worth mentioning



Last page - summary

Regularized NMF

Inexact Block-Coordinate Descent

Orthogonal NMF

$\nabla \|\boldsymbol{W}^\top \boldsymbol{W} - \boldsymbol{I}_r\|_F^2 = 4\boldsymbol{W}(\boldsymbol{W}^\top \boldsymbol{W} - \boldsymbol{I})$

Quick-and-dirty code

End of document