Computer Lab: Nonnegative Matrix Factorization
SD-TSIA 211

Olivier Fercoq, Hamid Jalalzai, Alex Lambert, Rémi Leluc

13 December 2019

1 Submission and grading information

The computer lab should be done by groups of 2. You will write a report with your implementation of the function before Thursday, 19 December 2018.

Please send it via the Gnomio web site: `http://sdtsia211.gnomio.com`. Your username is your last name, except if two of you have the same last name. You can ask for your username by clicking on "Forgotten your username or password?" The default password is `OptimizationIsGreat`. Each student submits his copy of the report in "Site home > Nonnegative Matrix Factorization". It can be either a python notebook or a pdf with python supplement.

Each of you will then be assigned two reports to grade through the web interface.

This computer lab will count for 10% of the final SD-TSIA 211 grade:
- 1 point for being present on December 13th
- 1 point for submitting a report
- 1 point for commenting 2 reports

2 Database

Question 2.1
Download and extract the database of faces, collected by AT&T Laboratories Cambridge, available in `orl_faces.zip` in the course's website. After extracting the images, you may use `nmf_utils.py` or `TP_NMF.ipynb` to load the database into Python.

How many images are there in the database? How many pixels are there in each image?

3 Presentation of the model

We have n images with p pixels. We are going to vectorize the images, that is consider them as a mere list of p pixel values. Then, we stack them all in a matrix M of size $n \times p$.

The goal of Nonnegative Matrix Factorization (NMF) is to (approximately) factorize the matrix M, which has only nonnegative entries, into the product of two matrices W and H

so that $M \approx WH$. W is a $n \times k$ nonnegative matrix and H is a $k \times p$ nonnegative matrix, where $k \leq \min(n, p)$ is a parameter of the model called the dimension of the latent space. Each line $H_{j,:}$ of H can be interpreted as the image of a piece of face. The coefficient $W_{i,j}$ tells us which proportion of image $H_{j,:}$ is present in the face $M_{i,:}$.

The model can be trained by solving the optimization problem

$$(\hat{W}, \hat{H}) \in \arg\min_{W \geq 0, H \geq 0} \frac{1}{2np} \sum_{i=1}^{n} \sum_{l=1}^{p} \left(M_{i,l} - \sum_{j=1}^{k} W_{i,j} H_{j,l} \right)^2 \tag{1}$$

$$(\hat{W}, \hat{H}) \in \arg\min_{W \geq 0, H \geq 0} \frac{1}{2np} \|M - WH\|_F^2$$

where $\|\cdot\|_F$ is Frobenius's norm.

When $k = 1$ the solution is proportional to the left and right singular vectors of M associated with the singular value with highest magnitude. In general, we need to use an optimization algorithm.

Question 3.1
Show with a 1D example that the objective function is not convex.

Question 3.2
Calculate the gradient of the objective function.

4 Find W when H_0 is fixed

Question 4.1
We initialize the optimization algorithm as follows:

```
W0, S, H0 = scipy.sparse.linalg.svds(M, k)
W0 = numpy.maximum(0, W0 * numpy.sqrt(S))
H0 = numpy.maximum(0,(H0.T * numpy.sqrt(S)).T)
```

What is the advantage of this choice? What would be other possibilities for the initialization?

In a first part, we would like to solve the simpler problem:

$$g(W) = \frac{1}{2np} \|M - WH_0\|_F^2$$

$$W^1 \in \arg\min_{W \geq 0} g(W)$$

Question 4.2
Is the objective function g convex? Calculate its gradient. We will admit that the gradient of g is Lipschitz continuous with constant $L_0 = \|(H^0)^\top H^0\|_F$.

Question 4.3
Write a function to compute $g(W)$ and another to compute $\nabla g(W)$.

You can check your computations using the function `scipy.optimize.check_grad` (as `check_grad` cannot deal with matrix variable, you may need to vectorize your variables).

Question 4.4
Let us define the function

$$\begin{aligned} \iota_{\mathbb{R}_+} : \quad & \mathbb{R} \to \mathbb{R} \cup \{+\infty\} \\ & x \mapsto \begin{cases} 0 & \text{if } x \geq 0 \\ +\infty & \text{if } x < 0 \end{cases} \end{aligned}$$

Show that for all $\gamma > 0$, $\mathrm{prox}_{\gamma \iota_{\mathbb{R}_+}}$ is the projection onto $\mathbb{R}_+$.

Question 4.5
Code a function `projected_gradient_method(val_g, grad_g, W0, gamma, N)` that minimizes a function g subject to nonnegativity constraints by the projected gradient method with a constant step size γ, starting from W_0 and stopping after N iterations.

Question 4.6
Use the function to minimize g with $N = 100$ and $k = 38$.

5 Algorithmic refinement for the problem with H_0 fixed

Question 5.1
Implement a line search to the projected gradient method, in order to free ourselves from the need of a known Lipschitz constant.

Question 5.2
Compare the performance of both algorithms.

6 Resolution of the full problem

Question 6.1
Solve Problem (1) by the projected gradient method with line search for $N = 1000$ iterations. What does the algorithm return?

Question 6.2
When W (resp. H) is fixed, the problem is easier to solve. The method of alternate minimizations uses this fact and consists in the following method:

Show that the value of the objective in decreasing at each iteration. Deduce from this that the value converges.

```
**for** $t \geq 1$ **do**
    $W_t \leftarrow \arg\min_W \ \frac{1}{2np}\|M - WH_{t-1}\|_F^2$
    $H_t \leftarrow \arg\min_H \ \frac{1}{2np}\|M - W_tH\|_F^2$
**end for**
```

Question 6.3
Code the alternate minimizations method.

Question 6.4
Compare projected gradient and alternate minimizations methods. Are the solutions the same? Is the objective value the same? How do the computing times compare?

Question 6.5
What stopping criterion could be used for the algorithms instead of just the number of iterations?