Online Matrix Factorization via Broyden Updates

Ömer Deniz Akyıldız

Bogazici University, Istanbul, Turkey.

Introduction

Formally, matrix factorization is the problem of factorizing a data matrix $Y \in \mathbb{R}^{m \times n}$ into [1],

$$Y \approx CX \tag{1}$$

where $C \in \mathbb{R}^{m \times r}$ and $X \in \mathbb{R}^{r \times n}$. Here r is the approximation rank which is typically selected by hand. These methods can be interpreted as dictionary learning where columns of C defines the elements of the dictionary, and columns of X can be thought as associated coefficients.

Online matrix factorization problem consists of updating C and associated columns of X by only using **a subset of columns of** Y which is the problem we are interested in this work.

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{Y} \approx \underbrace{\begin{bmatrix} \times & \times \\ \times & \times \\ \times & \times \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{X}$$

Notation: $[n] = \{1, \dots, n\}$. We denote a random index at time t with $k_t \in [n]$.

Construction of the Objective Function

We would like to update dictionary matrix C and a column of the X matrix x_{k_t} after observing a single column y_{k_t} of the dataset Y. For this purpose, we make the following crucial observations:

- We need to ensure $y_{k_t} \approx C_t x_{k_t}$ at time t for $k_t \in [n]$,
- We need to penalize C_t estimates in such a way that it should be "common to all observations", rather than being overfitted to each observation.

Approach: Suppose we are given y_{k_t} for $k_t \in [n]$ and C_{t-1}, then we solve the following optimization problem for each t,

$$(x_{k_t}^*, C_t^*) = \underset{x_{k_t}, C_t}{\operatorname{argmin}} \left\| y_{k_t} - C_t x_{k_t} \right\|_2^2 + \lambda \left\| C_t - C_{t-1} \right\|_F^2 \tag{2}$$

where $\lambda \in \mathbb{R}$ is a parameter which simply chooses how much emphasis should be put on specific terms in the cost function. This is the same cost function used in quasi-Newton methods to estimate the Hessian matrix [2].

Derivation of Updates

Update for x_{k_t}: Solving for x_{k_t} becomes a least squares problem, the solution is the following pseudoinverse operation,

$$x_{k_t} = (C_t^\top C_t)^{-1} C_t^\top y_{k_t}, \tag{3}$$

Update for C_t: The update is,

$$C_t = (\lambda C_{t-1} + y_{k_t} x_{k_t}^\top)(\lambda I + x_{k_t} x_{k_t}^\top)^{-1}, \tag{4}$$

and by using Sherman-Morrison formula for the term $(\lambda I + x_{k_t} x_{k_t}^\top)^{-1}$, Eq. (4) can be written more explicitly as,

$$C_t = C_{t-1} + \frac{(y_{k_t} - C_{t-1} x_{k_t}) x_{k_t}^\top}{\lambda + x_{k_t}^\top x_{k_t}}, \tag{5}$$

Algorithm 1. Online Matrix Factorization via Broyden Updates (OMF-B)

- Initialise C_0 randomly and set $t = 1$.
- **for** $t = 1 : N$
  - Pick $k_t \in [n]$ at random.
  - Read $y_{k_t} \in \mathbb{R}^m$
  - **for** Iter $= 1 : 2$

$$x_{k_t} = (C_t^\top C_t)^{-1} C_t^\top y_{k_t}$$
$$C_t = C_{t-1} + \frac{(y_{k_t} - C_{t-1} x_{k_t}) x_{k_t}^\top}{\lambda + x_{k_t}^\top x_{k_t}}$$

  - end **for**
- $t \leftarrow t + 1$

One can increase the number of inner iterations.

Some Modifications

Mini-Batch Setting

We denote a mini-batch dataset at time t with y_{v_t}. Hence $y_{v_t} \in \mathbb{R}^{m \times |v_t|}$ where $|v_t|$ is the cardinality of the index set v_t. Update for x_{v_t} reads as,

$$x_{v_t} = (C_t^\top C_t)^{-1} C_t^\top y_{v_t}, \tag{6}$$

and update rule for C_t can be given as,

$$C_t = (\lambda C_{t-1} + y_{v_t} x_{v_t}^\top)(\lambda I + x_{v_t} x_{v_t}^\top)^{-1}, \tag{7}$$

which is no longer same as the Broyden's rule for mini-batch observations.

Handling Missing Data

Define a mask $M \in \{0,1\}^{m \times n}$. We denote the data matrix with missing entries with $M \odot Y$ where $\odot$ denotes the Hadamard product. Suppose we have an observation y_{k_t} at time t and some entries of the observation are missing. We denote the mask vector for this observation as m_{k_t} which is k_t'th column of M. We need another mask $M_{C_t} \in \{0,1\}^{m \times r}$.

$$M_{C_t} = \underbrace{[m_{k_t}, \dots, m_{k_t}]}_{r \text{ times}}.$$

The update rule for x_{k_t} becomes the following pseudoinverse operation (see paper for derivation),

$$x_{k_t} = ((M_{C_t} \odot C_t)^\top (M_{C_t} \odot C_t))^{-1} (M_{C_t} \odot C_t)^\top (m_{k_t} \odot y_{k_t}),$$

and the update rule for C_t (for fixed x_{k_t}) can trivially be given as,

$$C_t = C_{t-1} + \frac{(m_{k_t} \odot (y_{k_t} - C_{t-1} x_{k_t})) x_{k_t}^\top}{\lambda + x_{k_t}^\top x_{k_t}}.$$

We denote the results on dataset with missing entries in Experiments.

Experimental Results

- Comparison with stochastic gradient matrix factorisation (SGMF) (left column) and nonnegative matrix factorisation (NMF) (right column).

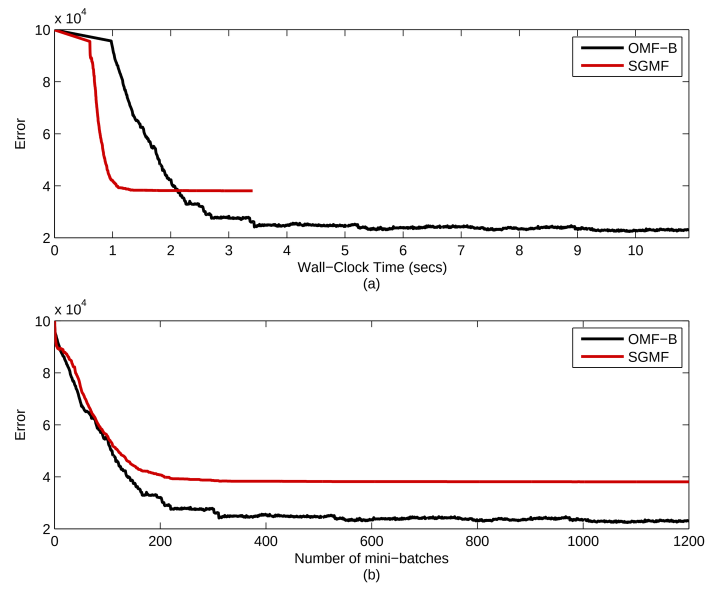

Figure 1: Comparison with SGMF. (a) SGMF processes the dataset in a much less wall-clock time, but we achieve a lower error in the same wall-clock time. (b) Our algorithm uses samples in a more efficient manner.

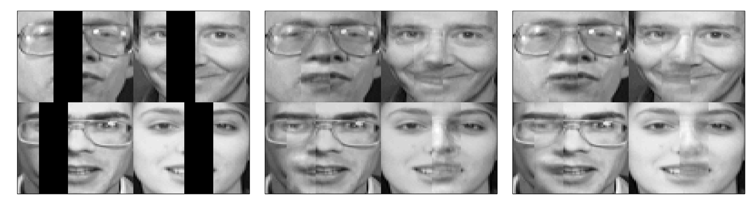

Figure 2: A demonstration on Olivetti faces dataset consists of 400 faces of size 64×64 with %25 missing data. We vectorized each face and construct a data matrix of size 4096×400. Some example faces with missing data are on the left. Comparison of results of OMF-B (middle) with 30 online passes over dataset and NMF with 1000 batch iterations (right). Signal-to-noise ratios (SNR) are: OMF-B: 11.57, NMF: 12.13 where initial SNR is 0.75.

References

1. D. D. Lee and H. S. Seung, Learning the parts of objects by nonnegative matrix factorization, *Nature*, vol. 401, no. 6755, pp. 788791, Oct. 1999.
2. Philipp Hennig, and Martin Kiefel. "Quasi-Newton methods: A new direction." *The Journal of Machine Learning Research*, 14.1 (2013): 843-865.

Acknowledgements: Thanks to Philipp Hennig for stimulating discussions. This work is supported by TUBITAK under the grant 113M492 (PAVERA).

This poster was presented at Machine Learning Summer School (MLSS) 2015, Tubingen, Germany.