Subsampled online matrix factorization with convergence guarantees

Arthur Mensch, Julien Mairal, Gaël Varoquaux, Bertrand Thirion

Subsampled online matrix factorization with convergence guarantees

Arthur Mensch Inria Parietal Saclay, France
Julien Mairal Inria Thoth Grenoble, France
Gaël Varoquaux Inria Parietal Saclay, France
Bertrand Thirion Inria Parietal Saclay, France
firstname.lastname@inria.fr

Abstract

We present a matrix factorization algorithm that scales to input matrices that are large in both dimensions (i.e., that contains more than 1TB of data). The algorithm streams the matrix columns while subsampling them, resulting in low complexity per iteration and reasonable memory footprint. In contrast to previous online matrix factorization methods, our approach relies on low-dimensional statistics from past iterates to control the extra variance introduced by subsampling. We present a convergence analysis that guarantees us to reach a stationary point of the problem. Large speed-ups can be obtained compared to previous online algorithms that do not perform subsampling, thanks to the feature redundancy that often exists in high-dimensional settings.

Setup. The goal of matrix factorization is to decompose a matrix $\mathbf{X} \in \mathbb{R}^{p \times n}$ – typically n signals of dimension p – as a product of two smaller matrices:

$$\mathbf{X} \approx \mathbf{D}\mathbf{A} \quad \text{with} \quad \mathbf{D} \in \mathbb{R}^{p \times k},\ \mathbf{A} \in \mathbb{R}^{k \times n}, \tag{1}$$

with potential sparsity or structure requirements on $\mathbf{D}$ and $\mathbf{A}$. We consider a sample stream $(\mathbf{x}_t)_{t \geq 0}$ that cycles into the columns $\{\mathbf{x}^{(i)}\}_i$ of $\mathbf{X}$. Matrix factorization can be formulated as a non-convex optimization problem, where the factor $\mathbf{D}$ (the *dictionary*) minimizes the following empirical risk:

$$\mathbf{D} = \operatorname*{argmin}_{\mathbf{D} \in \mathcal{C}} \ \bar{f} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f^{(i)}(\mathbf{D}), \quad \text{where} \quad f^{(i)}(\mathbf{D}) = \min_{\boldsymbol{\alpha} \in \mathbb{R}^k} \frac{1}{2} \left\| \mathbf{x}^{(i)} - \mathbf{D}\boldsymbol{\alpha} \right\|_2^2 + \lambda\, \Omega(\boldsymbol{\alpha}). \tag{2}$$

$\mathcal{C}$ is a column-wise separable convex set of $\mathbb{R}^{p \times k}$, and $\Omega : \mathbb{R}^p \to \mathbb{R}$ is a penalty over the code. The problem of *dictionary learning* [8, 1] sets $\mathcal{C} = \mathcal{B}_2^k$ and $\Omega = \| \cdot \|_1$. Due to the sparsifying effect of ℓ_1 penalty [11], the dictionary forms a basis in which the data admit a *sparse* representation. Setting $\mathcal{C} = \mathcal{B}_1^k$ and $\Omega = \| \cdot \|_2^2$ yields a data-adapted *sparse basis*, akin to sparse PCA[12]. The algorithm presented here accommodates *elastic-net* penalties $\Omega(\boldsymbol{\alpha}) \triangleq (1-\nu)\|\boldsymbol{\alpha}\|_2^2 + \nu\|\boldsymbol{\alpha}\|_1$, and elastic-net ball-constraints $\mathcal{C} \triangleq \{\mathbf{D} \in \mathbb{R}^{p \times k} / \ \|\mathbf{d}^{(j)}\| \triangleq \mu\|\mathbf{d}^{(j)}\|_1 + (1-\mu)\|\mathbf{d}^{(j)}\|_2^2 \leq 1\}$.

Problem. For many applications of matrix factorization, datasets are growing in both sample number and sample dimension. The online matrix factorization algorithm [6] can handle large numbers of samples but was designed to work in relatively small dimension. Recent work [7] has adapted this algorithm to handle very high dimensional dataset. Though it demonstrates good empirical performance, the proposed algorithm yields sequences of iterates with non vanishing variance and is not asymptotically convergent.

Contribution. We address this issue and correct some aspects of the algorithm in [7] to establish convergence and correctness. We thus introduce a new method to efficient solve (2) for numerous high-dimensional data — large p, large n — with theoretical guarantees.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

As in [7], we perform *subsampling* at each iteration of the online matrix factorization algorithm. The sample stream is downsampled in a *stochastic* manner —we observe different subsets of successive colunms. We thus each step of the online algorithm in a space of reduced dimension $q < p$. Unlike [7], we control the variance introduced by the subsampling to obtain convergence. For this, we rely on low-dimensional statistics kept from the past, as many recent stochastic algorithms [10, 3, 2].

1 Algorithm

Original online algorithm. The problem (2) can be solved online, following [6]. At each iteration t, a sample $\mathbf{x}_t$ is drawn from one of the columns $\{\mathbf{x}^{(i)}\}_{1\leq i\leq n}$ of $\mathbf{X}$. Its code $\boldsymbol{\alpha}_t$ is computed from the previous dictionary $\mathbf{D}_{t-1}$: $\boldsymbol{\alpha}_t \triangleq \operatorname{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^p} \frac{1}{2}\left\|\mathbf{x}_t - \mathbf{D}_{t-1}\boldsymbol{\alpha}\right\|_2^2 + \lambda\,\Omega(\boldsymbol{\alpha})$. Then, $\mathbf{D}_t$ is updated as

$$\mathbf{D}_t \in \operatorname*{argmin}_{\mathbf{D}\in\mathcal{C}} \bar{g}_t(\mathbf{D}) \triangleq \Big(\frac{1}{t}\sum_{s=1}^{t}\frac{1}{2}\left\|\mathbf{x}_s - \mathbf{D}\boldsymbol{\alpha}_s\right\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}_s)\Big), \tag{3}$$

In other words, $\mathbf{D}_t$ is chosen to be the best dictionary that relates past codes $(\boldsymbol{\alpha}_s)_{s\leq t}$ to past samples $(\mathbf{x}_t)_{s\leq t}$. Codes are not recomputed from the current iterate $\mathbf{D}$, which would be necessary to compute the true past loss fonction $\bar{f}_t(\mathbf{D})$, of which $\bar{g}_t$ is a strongly-convex upper-bound:

$$\bar{f}_t(\mathbf{D}) \triangleq \frac{1}{t}\sum_{s=1}^{t}\min_{\boldsymbol{\alpha}\in\mathbb{R}^p}\frac{1}{2}\left\|\mathbf{x}_s - \mathbf{D}\boldsymbol{\alpha}\right\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}) \leq \bar{g}_t(\mathbf{D}) \tag{4}$$

It can be shown [see 5, for theoretical grounding] that minimizing $(\bar{g}_t)_t$ yields a sequence of iterates that is asymptotically a critical point of $\bar{f}$ defined in (2). $\bar{g}_t$ can be minimized efficiently by projected block coordinate descent, which makes it useful in practice. Indeed, minimizing $\bar{g}_t$ is equivalent to minimizing the quadratic function

$$\mathbf{D} \to \frac{1}{2}\mathrm{Tr}\left(\mathbf{D}^\top\mathbf{D}\bar{\mathbf{C}}_t\right) - \mathrm{Tr}\left(\mathbf{D}^\top\bar{\mathbf{B}}_t\right), \text{ where } \quad \bar{\mathbf{B}}_t = \frac{1}{t}\sum_{s=1}^{t}\mathbf{x}_s\boldsymbol{\alpha}_s^\top, \quad \bar{\mathbf{C}}_t = \frac{1}{t}\sum_{s=1}^{t}\boldsymbol{\alpha}_s\boldsymbol{\alpha}_s^\top. \tag{5}$$

Its gradient $\nabla\bar{g}_t : \mathbf{D} \to \mathbf{D}\bar{\mathbf{C}}_t - \bar{\mathbf{B}}_t$ can be tracked online by updating $\bar{\mathbf{C}}_t$ and $\bar{\mathbf{B}}_t$ at each iteration:

$$\bar{\mathbf{C}}_t = (1-\frac{1}{t})\bar{\mathbf{C}}_{t-1} + \frac{1}{t}\boldsymbol{\alpha}_t\boldsymbol{\alpha}_t^\top \qquad \bar{\mathbf{B}}_t = (1-\frac{1}{t})\bar{\mathbf{B}}_{t-1} + \frac{1}{t}\mathbf{x}_t\boldsymbol{\alpha}_t^\top. \tag{6}$$

Those two statistics are thus sufficient to yield the sequence $(\mathbf{D}_t)_t$. The weight $\frac{1}{t}$ used above can be replaced by a more general w_t. In addition, the online algorithm has a minibatch extension.

Subsampled algorithm. We adapt the algorithm from [6] to handle large sample dimension p. The complexity of this algorithm linearly depends on the dimension p in three aspects:

- $\mathbf{x}_t \in \mathbb{R}^p$ is used to compute the code $\boldsymbol{\alpha}_t$,
- it is used to update the surrogate parameters $\bar{\mathbf{C}}_t \in \mathbb{R}^{p\times k}$,
- $\mathbf{D}_t \in \mathbb{R}^{p\times k}$ is fully updated at each iteration.

Our new *subsampling online matrix factorization* algorithm (SOMF) reduces the dimensionality of each of these steps, so that the single-iteration complexity in p depends on $q = \frac{p}{r}$ rather than p. $r > 1$ is a *reduction factor* that is close to the computational speed-up per iteration in the large dimensional regime $p \gg k$. Formally, we randomly draw, at iteration t, a mask $\mathbf{M}_t$ that "selects" a random subset of $\mathbf{x}_t$. $\mathbf{M}_t$ is a $\mathbb{R}^{p\times p}$ random diagonal matrix, such that each coefficient is a Bernouilli variable with parameter $\frac{1}{r}$, normalized to be 1 in expectation. With this definition at hand, $\mathbf{M}_t\mathbf{x}_t$ constitutes a non-biased, low-dimensional estimator of $\mathbf{x}_t$: $\mathbb{E}[\|\mathbf{M}_t\mathbf{x}_t\|_0] = \frac{p}{r} = q$, and $\mathbb{E}[\mathbf{M}_t\mathbf{x}_t] = \mathbf{x}_t$, with $\|\cdot\|_0$ counting the number of non-zero coefficients. Thus, r is the average proportion of observed features at each iteration. We further define the pair of orthogonal projectors $\mathbf{P}_t \in \mathbb{R}^{q\times p}$ and $\mathbf{P}_t^\perp \in \mathbb{R}^{(p-q)\times p}$ that projects $\mathbb{R}^p$ onto $\mathrm{Im}(\mathbf{M}_t)$ and $\mathrm{Ker}(\mathbf{M}_t)$, which we will use for the dictionary update step.
In brief, SOMF, defined in Alg. 1, follows the outer loop of online matrix factorization, with the following major modifications at iteration t:

- it uses $\mathbf{M}_t\mathbf{x}_t$ and low-size statistics instead of $\mathbf{x}_t$ to estimate the code $\boldsymbol{\alpha}_t$ and the surrogate g_t,
- it updates a subset of the dictionary $\mathbf{P}_t\mathbf{D}_{t-1}$ to reduce the surrogate value $\bar{g}_t(\mathbf{D})$. Relevant parameters of $\bar{g}_t$ are computed using $\mathbf{P}_t\mathbf{x}_t$ and $\boldsymbol{\alpha}_t$ only.

We describe in detail the new code computation and dictionary update steps. We then state convergence guarantees for SOMF. Those are non trivial to obtain as SOMF is not an exact (stochastic) majorization-minimization algorithm.

Algorithm 1 Subsampled online matrix factorization (SOMF)

Input: Initial iterate $\mathbf{D}_0$, weight sequences $(w_t)_{t>0}$, $(\gamma_c)_{c>0}$, sample set $\{\mathbf{x}^{(i)}\}_{i>0}$, # iterations T.
for t from 1 to T **do**
Draw $\mathbf{x}_t = \mathbf{x}^{(i)}$ at random and $\mathbf{M}_t$ (see text)
Update the regression parameters for sample i: $c^{(i)} \leftarrow c^{(i)} + 1$, $\gamma \leftarrow \gamma_{c^{(i)}}$

$$(\boldsymbol{\beta}_t^{(i)}, \mathbf{G}_t^{(i)}) \leftarrow (1-\gamma)(\boldsymbol{\beta}_{t-1}^{(i)}, \mathbf{G}_{t-1}^{(i)}) + \gamma(\mathbf{D}_{t-1}^\top \mathbf{M}_t \mathbf{x}^{(i)}, \mathbf{D}_{t-1}^\top \mathbf{M}_t \mathbf{D}_{t-1}),\ (\boldsymbol{\beta}_t, \mathbf{G}_t) \leftarrow (\bar{\boldsymbol{\beta}}_t^{(i)}, \bar{\mathbf{G}}_t^{(i)})$$

Compute the approximate code for $\mathbf{x}_t$: $\boldsymbol{\alpha}_t \leftarrow \operatorname{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^k} \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{G}_t \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \boldsymbol{\beta}_t + \lambda\,\Omega(\boldsymbol{\alpha})$.
Update the parameters of the aggregated surrogate $\bar{g}_t$:

$$\bar{\mathbf{C}}_t \leftarrow (1-w_t)\bar{\mathbf{C}}_t + w_t \boldsymbol{\alpha}_t \boldsymbol{\alpha}_t^\top. \qquad \mathbf{P}_t\bar{\mathbf{B}}_t \leftarrow (1-w_t)\mathbf{P}_t\bar{\mathbf{B}}_t + w_t \mathbf{P}_t \mathbf{x}_t \boldsymbol{\alpha}_t^\top. \tag{7}$$

Compute simultaneously (using (using [7, Alg 2]) for 1st expression):

$$\mathbf{P}_t\mathbf{D}_t \leftarrow \operatorname*{argmin}_{\mathbf{D}^r\in\mathcal{C}^r} \frac{1}{2}\mathrm{Tr}\left(\mathbf{D}^{r\top}(\mathbf{D}^r\bar{\mathbf{C}}_t - \bar{\mathbf{B}}_t)\right),\ \mathbf{P}_t^\perp\bar{\mathbf{B}}_t \leftarrow (1-w_t)\mathbf{P}_t^\perp\bar{\mathbf{B}}_{t-1} + w_t\mathbf{P}_t^\perp\mathbf{x}_t\boldsymbol{\alpha}_t^\top. \tag{8}$$

Output: Final iterate $\mathbf{D}_T$.

Code computation. In the online algorithm, $\boldsymbol{\alpha}_t$ is obtained solving the linear regression problem

$$\boldsymbol{\alpha}_t = \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^k} \frac{1}{2}\boldsymbol{\alpha}^\top \mathbf{G}_t^\star \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \boldsymbol{\beta}_t^\star + \lambda\Omega(\boldsymbol{\alpha}), \quad \text{where} \quad \mathbf{G}_t^\star = \mathbf{D}_{t-1}^\top\mathbf{D}_{t-1} \text{ and } \boldsymbol{\beta}_t^\star = \mathbf{D}_{t-1}^\top\mathbf{x}_t \tag{9}$$

For large p, computing $\mathbf{G}_t^\star$ and $\boldsymbol{\beta}_t^\star$ dominates the complexity of the code computation step. To reduce this complexity, we introduce *estimators* for $\mathbf{G}_t$ and $\boldsymbol{\beta}_t$, computable at a cost proportional to q, whose use does not break convergence. Recall that the sample $\mathbf{x}_t$ is drawn from a finite set of samples $\{\mathbf{x}^{(i)}\}_i$. We estimate $\mathbf{G}_t^\star$ and $\boldsymbol{\beta}_t^\star$ from $\mathbf{M}_t\mathbf{x}_t$ and data from previous iterations $s < t$ for which $\mathbf{x}^{(i)}$ was drawn. Namely, we keep in memory $2n$ estimators, written $(\mathbf{G}_t^{(i)}, \boldsymbol{\beta}_t^{(i)})_{1\le i\le n}$, observe the sample $i = i_t$ at iteration t and use it to update the i-th estimators $\bar{\mathbf{G}}_t^{(i)}$, $\bar{\boldsymbol{\beta}}_t^{(i)}$ following

$$\boldsymbol{\beta}_t^{(i)} = (1-\gamma)\mathbf{G}_{t-1}^{(i)} + \gamma\mathbf{D}_{t-1}^\top\mathbf{M}_t\mathbf{x}^{(i)}, \qquad \mathbf{G}_t^{(i)} = (1-\gamma)\mathbf{G}_{t-1}^{(i)} + \gamma\mathbf{D}_{t-1}^\top\mathbf{M}_t\mathbf{D}_t^{(i)}, \tag{10}$$

where γ is a weight factor determined by the number of time sample i has been previously observed at time t. Precisely, given $(\gamma_c)_c$ a decreasing sequence of weights, $\gamma \triangleq \gamma_{c_t^{(i)}}$, where $c_t^{(i)} = |\{s \le t, \mathbf{x}_s = \mathbf{x}^{(i)}\}|$ All others estimators $\{\mathbf{G}_t^{(j)}, \boldsymbol{\beta}_t^{(j)}\}_{j\neq i}$ are left unchanged from iteration $t-1$. The set $\{\mathbf{G}_t^{(i)}, \boldsymbol{\beta}_t^{(i)}\}_{1\le i\le n}$ is used to define the *averaged* estimators at iteration t, related to sample i:

$$\mathbf{G}_t \triangleq \mathbf{G}_t^{(i)} = \sum_{s\le t, \mathbf{x}_s=\mathbf{x}^{(i)}} \gamma_{s,t}^{(i)}\mathbf{D}_{s-1}^\top\mathbf{M}_s\mathbf{D}_{s-1}, \quad \boldsymbol{\beta}_t \triangleq \boldsymbol{\beta}_t^{(i)} = \sum_{s\le t, \mathbf{x}_s=\mathbf{x}^{(i)}} \gamma_{s,t}^{(i)}\mathbf{D}_{s-1}^\top\mathbf{M}_s\mathbf{x}^{(i)}, \tag{11}$$

where $\gamma_{s,t}^{(i)} = \gamma_{c_t^{(i)}}\prod_{s<t, \mathbf{x}_s=\mathbf{x}^{(i)}}(1-\gamma_{c_s^{(i)}})$. Replacing $(\mathbf{G}_t^\star, \boldsymbol{\beta}_t^\star)$ by $(\mathbf{G}_t, \boldsymbol{\beta}_t)$ in (9), $\boldsymbol{\alpha}_t$ minimizes the masked loss averaged over the previous iterations where sample i appeared:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^k} \sum_{s\le t\mathbf{x}_s=\mathbf{x}^{(i)}} \frac{\gamma_{s,t}^{(i)}}{2}\|\mathbf{M}_s(\mathbf{x}^{(i)} - \mathbf{D}_{s-1}^\top\boldsymbol{\alpha})\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}). \tag{12}$$

The sequences $(\mathbf{G}_t)_t$ and $(\boldsymbol{\beta}_t)_t$ are *consistent* estimations of $(\mathbf{G}_t^\star)_t$ and $(\boldsymbol{\beta}_t^\star)_t$ — consistency arises from the fact that a single sample $\mathbf{x}^{(i)}$ is observed with different masks along iterations. This was not the case in the algorithm proposed in [7], which use estimators that does involve averaging from past data. Solving (12) is made closer and closer to solving (9), in a manner that ensures the correctness of the algorithm. Yet, computing the estimators (11) is r times as costly as computing $\mathbf{G}_t^\star$ and $\boldsymbol{\beta}_t^\star$ from (9) and permits to speed up the code computation steop close to r times. The weight sequences $(w_t)_t$ and $(\gamma_c)_c$ are selected appropriately to ensure convergence. For instance, we can set $w_t = \frac{1}{t^v}, \gamma_c = \frac{1}{c^{2.5-2v}}$, with $v \in (\frac{3}{4}, 1)$.

Dictionary update. In the original online algorithm, the whole dictionnary $\mathbf{D}_{t-1}$ is updated at iteration t. To reduce the time complexity of this step, we add a "freezing" constraint to the minimization of the quadratic function (5). Every row r of $\mathbf{D}$ that corresponds to an *unseen* row r at iteration r (such that $\mathbf{M}_t[r, r] = 0$) remains unchanged. $\mathbf{D}_t$ is thus obtained solving

$$\mathbf{D}_t \in \underset{\substack{\mathbf{D}\in\mathcal{C} \\ \mathbf{P}_t^{\perp}\mathbf{D}=\mathbf{P}_t^{\perp}\mathbf{D}_{t-1}}}{\operatorname{argmin}} \frac{1}{2}\operatorname{Tr}\left(\mathbf{D}^\top\mathbf{D}\bar{\mathbf{C}}_t\right) - \operatorname{Tr}\left(\mathbf{D}^\top\bar{\mathbf{B}}_t\right), \text{ with } \mathbf{P}_t \text{ orth. projector on } \operatorname{Im}(\mathbf{M}_t) \quad (13)$$

With elastic-net ball constraints, solving (13) reduces to performing the partial dictionary update (8) (Alg. 1), with $\mathcal{C}^r = \{\mathbf{D}^r \in \mathbb{R}^{k\times q}, \|(\mathbf{d}^r)^{(j)}\| \leq 1 - \|\mathbf{d}_{t-1}^{(j)}\| + \|\mathbf{P}_t\mathbf{d}_{t-1}^{(j)}\|\}$. We perform this update using a single pass of projected block coordinate descent with blocks in the reduced space $\mathbb{R}^q$. The dictionary update step is thus performed r times faster than the original algorithm.

Surrogate computation The gradient we use to solve (8) requires to know only $\bar{\mathbf{C}}_t$ and $\mathbf{P}_t\bar{\mathbf{B}}_t$. We thus parallelize the partial update of the dictionary and the update of $\mathbf{P}_t^{\perp}\bar{\mathbf{B}}_t$, using a second thread. The update of $\mathbf{P}_t\bar{\mathbf{B}}_t$ is performed in the main thread at a cost proportional to q. As the parallel computation is dominated by dictionary update, this is enough to effectively reduce the computation time of $\bar{\mathbf{B}}_t$ computation.

Convergence guarantees All in all, the three steps whose complexity depends on p in the original algorithm now depends on q, which speeds-up a single iteration by a factor close to r. Yet SOMF retain convergence guarantees. We assume that there exists ν such that for all $t > 0$, $\mathbf{D}_t^\top\mathbf{D}_t \succ \nu\mathbf{I}$ (met in practice or by adding a small ridge regularization to (2)), and make a technical data-independent hypothesis on $(w_t)_t$ and $(\gamma_c)_c$ decay. Then SOMF iterates converge in the same sense as in the original algorithm [6]: $\mathbf{D}_t \to \mathbf{D}_\infty \in \mathbb{R}^{p\times k}$ and $\nabla\bar{f}(\mathbf{D}_\infty, \mathbf{D} - \mathbf{D}_\infty) \geq 0$ for any $\mathbf{D} \in \mathcal{C}$, where $\bar{f}$ is the empirical risk defined in (2).

Proof sketch. We use the aggregated nature of $\bar{g}_t$ and the fact that we can obtain a *geometric* rate of convergence for a single pass of projected block coordinate descent [*e.g* see 9] to control the terms $\mathbf{D}_t - \mathbf{D}_{t-1}$ and $\bar{g}_t(\mathbf{D}_t) - \bar{g}_t(\mathbf{D}_t^*)$, where $\mathbf{D}_t^* = \operatorname{argmin}_{\mathbf{D}\in\mathcal{C}}\bar{g}_t$. We obtain $\bar{g}_t(\mathbf{D}_t) - \bar{g}_t(\mathbf{D}_{t-1}) = \mathcal{O}(w_t)$, a crucial result on estimate stability. Simultaneously, we show that the partial minimization yields the same result as the full minimization for $t \to \infty$, as $\theta_t - \theta_t^\star \to 0$. Using this result, with appropriate selection of $(w_t)_t$ and $(\gamma_t)_t$, the noise induced by the use of estimators in (9) can be bounded in the derivations of [6]. We then write the difference (12) - (9) as the sum of a *lag* term and an empirical mean over $\{\mathbf{M}_s\}_{\mathbf{x}_s=\mathbf{x}_t}$. Both can be bounded with appropriate selection of weights. Formally, $(\bar{g}_t)_t$ are no longer upper-bounds of $(\bar{f}_t)_t$, but become so for $t \to \infty$, at a sufficient rate to guarantee convergence. □

2 Experiments

Hyperspectral images. We benchmark our algorithm by performing dictionary learning on a large hyperspectral image. Dictionary learning is indeed used on patches of hyperspectral images, as in [4]. Extracting 16×16 patches from an 1GB hyperspectral image from the AVIRIS project with 224 channels yields samples of dimension $p = 57,000$. Figure 1 demonstrates that the newly proposed algorithm is faster than the original, non-subsampled algorithm from [6] by a factor close to $r = 4$. This speed-up is helped by the redundancy in the different channels of the hyperspectral patches. In the first epochs, the proposed method also outperforms the recent subsampled algorithm from [7], thanks to the introduction of consistent estimators in the code computation step. All algorithms are implemented in Cython and benched on two cores. They cycle over 100,000 normalized samples with minibatches of size 50. The code for reproduction is available at `github.com/arthurmensch/modl`..

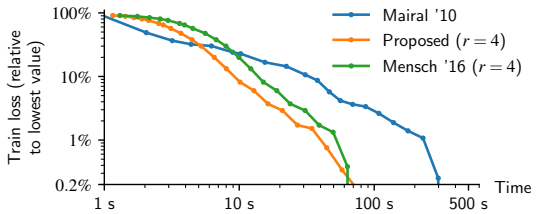

Figure 1 Performing dictionary learning on hyperspectral data (224 channels, 16 × 16 patches) is faster with stochastic subsampling, and even faster with the newly proposed variance control.

Conclusion The new SOMF algorithm can efficiently factorize large and tall matrices. It preserves the speed gains of [7] but has convergence guarantees that are beneficial in practice.

Acknowledgments

The research leading to these results was supported by the ANR (MACARON project, ANR-14-CE23-0003-01 – NiConnect project, ANR-11-BINF-0004NiConnect) and has received funding from the MetaMRI Inria Associate Team.

References

[1] A. Agarwal, A. Anandkumar, P. Jain, P. Netrapalli, and R. Tandon. Learning sparsely used overcomplete dictionaries. In *Conference on Learning Theory*, 2014.

[2] A. Defazio, F. Bach, and S. Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, 2014.

[3] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*. 2013.

[4] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi. Nonlocal transform-domain filter for volumetric data denoising and reconstruction. *IEEE Transactions on Image Processing*, 22(1):119–133, 2013.

[5] J. Mairal. Stochastic majorization-minimization algorithms for large-scale optimization. In *Advances in Neural Information Processing Systems*, 2013.

[6] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research*, 11:19–60, 2010.

[7] A. Mensch, J. Mairal, B. Thirion, and G. Varoquaux. Dictionary learning for massive matrix factorization. In *International Conference on Machine Learning*, 2016.

[8] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

[9] P. Richtárik and M. Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144:1–38, 2014.

[10] M. Schmidt, N. L. Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013.

[11] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[12] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.