Online Passive-Aggressive Algorithms for Non-Negative Matrix Factorization and Completion

Mathieu Blondel **Yotaro Kubo** **Naonori Ueda**
NTT Communication Science Laboratories, Kyoto, Japan

Abstract

Stochastic Gradient Descent (SGD) is a popular online algorithm for large-scale matrix factorization. However, SGD can often be difficult to use for practitioners, because its performance is very sensitive to the choice of the learning rate parameter. In this paper, we present non-negative passive-aggressive (NN-PA), a family of online algorithms for non-negative matrix factorization (NMF). Our algorithms are scalable, easy to implement and do not require the tedious tuning of a learning rate parameter. We demonstrate the effectiveness of our algorithms on three large-scale matrix completion problems and analyze them in the regret bound model.

1 Introduction

Matrix completion is the fundamental task of reconstructing a partially observed matrix. The archetypal application of matrix completion is collaborative filtering, in which the matrix contains ratings given by users to some items. Naturally, since users tend to rate a limited number of items (e.g., movies), only a very small subset of the matrix is typically observed. Completing the missing matrix entries amounts to recovering unknown ratings and can thus be used to recommend new items to users. Matrix completion is also an important pre-processing step in machine learning workflows, since many algorithms or implementations cannot directly handle missing values. Popularized by recommendation systems, matrix factorization based approaches to matrix completion have been among the most successful, as witnessed by the Netflix challenge [Koren et al., 2009]. Although Stochastic Gradient Descent (SGD) is often associated with slow convergence, its low computational complexity, combined with other trade-offs in statistical learning theory [Bottou and Bousquet, 2008], still makes it a candidate of choice for solving large-scale matrix factorization problems [Takács et al., 2009]. However, SGD can often be frustrating to use for practitioners, because its performance is very sensitive to the choice of the learning rate parameter.

In this paper, we propose non-negative passive-aggressive (NN-PA), a family of online algorithms for non-negative matrix factorization (NMF). Our algorithms are scalable, easy to implement and do not require the tedious tuning of a learning rate parameter. Imposing non-negativity in the matrix factorization can reduce prediction error in the case of non-negative matrices compared to conventional methods such as Singular Value Decomposition (SVD) [Zhang et al., 2006] and leads to interpretable and sparse decompositions [Lee and Seung, 1999].

Let R be a non-negative matrix of size $n \times d$. We denote its entries by $r_{u,i}$. For instance, in a recommendation system, $r_{u,i}$ may correspond to the rating given by user u to item i. Let P and Q be two non-negative matrices of size $n \times m$ and $m \times d$, respectively, where $m \leq \min(n, d)$ denotes a user-defined rank. We denote the rows of P by $\boldsymbol{p}_u \in \mathbf{R}_+^m$ and the columns of Q by $\boldsymbol{q}_i \in \mathbf{R}_+^m$. In this paper, we cast non-negative matrix factorization as an online learning problem. On each round t, a matrix entry r_{u_t,i_t} is revealed. The goal of NN-PA is to update $\boldsymbol{p}_{u_t} \in \mathbf{R}_+^m$ and $\boldsymbol{q}_{i_t} \in \mathbf{R}_+^m$ such that $r_{u_t,i_t} \approx \boldsymbol{p}_{u_t} \cdot \boldsymbol{q}_{i_t}$. Let $\boldsymbol{p}_u^t$ be $\boldsymbol{p}_u$ on round t. NN-PA then finds $\boldsymbol{p}_{u_t}^{t+1}$ by solving the following optimization problem:

$$\boldsymbol{p}_{u_t}^{t+1} = \operatorname*{argmin}_{\boldsymbol{p} \in \mathbf{R}_+^m} \frac{1}{2}\|\boldsymbol{p} - \boldsymbol{p}_{u_t}^t\|^2 \text{ s.t. } |\boldsymbol{p} \cdot \boldsymbol{q}_{i_t}^t - r_{u_t,i_t}| = 0. \quad (1)$$

Intuitively, the above optimization problem captures the fact that we want to find the smallest possible change in $\boldsymbol{p}_{u_t}$ so as to predict r_{u_t,i_t}. Similarly, let $\boldsymbol{q}_i^t$

Appearing in Proceedings of the 17^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2014, Reykjavik, Iceland. JMLR: W&CP volume 33.

be $\boldsymbol{q}_i$ on round t. NN-PA then finds $\boldsymbol{q}_{i_t}^{t+1}$ by solving the same optimization problem as above except that the roles of $\boldsymbol{p}_{u_t}^t$ and $\boldsymbol{q}_{i_t}^t$ are swapped:

$$\boldsymbol{q}_{i_t}^{t+1} = \operatorname*{argmin}_{\boldsymbol{q}\in\mathbf{R}_+^m} \frac{1}{2}\|\boldsymbol{q}-\boldsymbol{q}_{i_t}^t\|^2 \text{ s.t. } |\boldsymbol{p}_{u_t}^t\cdot\boldsymbol{q} - r_{u_t,i_t}| = 0. \tag{2}$$

The updates in Eq. (1) and Eq. (2) are called *passive-aggressive* [Crammer et al., 2006] because they may result in rounds that update P or Q "aggressively", or in "passive" rounds that leave P and Q unchanged.

In practice, NN-PA can also be used in a batch setting. Let the set of observed entries in R be Ω. In our experiments, we update P by Eq. (1) for all $(u,i)\in\Omega$ then Q by Eq. (2) for all $(u,i)\in\Omega$ and repeat this process several times. Alternating minimization is a popular approach in the NMF literature (c.f. [Lin, 2007] and references therein) because it essentially allows to reduce learning the NMF to a sequence of non-negative regression problems.

A complete pass over all observed entries $(u,i)\in\Omega$ of R requires NN-PA to solve the optimization problem in Eq. (1) or Eq. (2) $|\Omega|$ times. In the following, we will present algorithms that can solve Eq. (1) or Eq. (2) in $O(m)$ time. Therefore, a complete pass over all observed entries in R takes $O(m|\Omega|)$ time. This is the same complexity as SGD but NN-PA does not require the tedious tuning of a learning-rate parameter. In real-world applications, where the matrices P and Q must be retrained or updated frequently, this is a major advantage of NN-PA in practice.

2 Solving the optimization problem

In this section, we describe how to solve the optimization problem at the core of NN-PA, Eq. (1) and Eq. (2). To simplify our notation, we define

$\boldsymbol{w}$	=	$\boldsymbol{p}$	or	$\boldsymbol{q}$	(variable)
$\boldsymbol{w}_{t+1}$	=	$\boldsymbol{p}_{u_t}^{t+1}$	or	$\boldsymbol{q}_{i_t}^{t+1}$	(solution)
$\boldsymbol{w}_t$	=	$\boldsymbol{p}_{u_t}^t$	or	$\boldsymbol{q}_{i_t}^t$	(current iterate)
$\boldsymbol{x}_t$	=	$\boldsymbol{q}_{i_t}^t$	or	$\boldsymbol{p}_{u_t}^t$	(input)
y_t	=	r_{u_t,i_t}			(target)

and rewrite the optimization problem as follows

$$\boldsymbol{w}_{t+1} = \operatorname*{argmin}_{\boldsymbol{w}\in\mathbf{R}_+^m} \frac{1}{2}\|\boldsymbol{w}-\boldsymbol{w}_t\|^2 \text{ s.t. } |\boldsymbol{w}\cdot\boldsymbol{x}_t - y_t| \le \epsilon, \tag{3}$$

where for more flexibility we introduced a precision parameter ϵ. We assume $\boldsymbol{w}_t = [w_{t,1},\ldots,w_{t,m}]\in\mathbf{R}_+^m$, $\boldsymbol{x}_t = [x_{t,1},\ldots,x_{t,m}]\in\mathbf{R}_+^m$ and $y_t\in\mathbf{R}_+$. Let us introduce the ϵ-insensitive loss function:

$$\ell^\epsilon(\boldsymbol{w};(\boldsymbol{x},y)) = \max(|\boldsymbol{w}\cdot\boldsymbol{x}-y|-\epsilon,0). \tag{4}$$

It is easy to see that satisfying the constraint $|\boldsymbol{w}\cdot\boldsymbol{x}_t - y_t| \le \epsilon$ in Eq. (3) is equivalent to satisfying the constraint $\ell^\epsilon(\boldsymbol{w};(\boldsymbol{x}_t,y_t)) = 0$. When $\epsilon = 0$, Eq. (4) reduces to the absolute loss $|\boldsymbol{w}\cdot\boldsymbol{x}-y|$. Without the non-negativity constraint, Eq. (3) requires solving the same optimization problem as passive-aggressive regression and enjoys a closed-form solution [Crammer et al., 2006, Section 5]. However, with the non-negativity constraint, Eq. (3) does not enjoy a general closed-form solution anymore. Since Eq. (3) is a quadratic program (QP), one may use any numerical QP solver to solve it. However, as we emphasized previously, a complete pass over the observed entries of R requires solving Eq. (3) $|\Omega|$ times. As an example, on the Yahoo-Music dataset, $|\Omega|$ is equal to $252,800,275$. Therefore, it is crucial to solve the optimization problem efficiently. In the following, we derive an $O(m)$ time procedure.

For convenience, let us introduce the shorthands $\hat{y}_t = \boldsymbol{w}_t\cdot\boldsymbol{x}_t$ and $\ell_t^\epsilon = \max(|\hat{y}_t - y_t| - \epsilon, 0)$. If $|\hat{y}_t - y_t| \le \epsilon$, that is, if $\boldsymbol{w}_t$ readily predicts y_t with loss $\ell_t^\epsilon = 0$, then, assuming that $\boldsymbol{w}_t$ is a feasible solution, we can clearly choose $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$, i.e., $\boldsymbol{w}_t$ is not updated (hence, the name *passive*). When $\ell_t^\epsilon > 0$, the following lemma is a key tool for solving Eq. (3).

Lemma 1 *Assume $\ell_t^\epsilon > 0$. If $\hat{y}_t < y_t$, then solving Eq. (3) is equivalent to solving*

$$\boldsymbol{w}_{t+1} = \operatorname*{argmin}_{\boldsymbol{w}\in\mathbf{R}^m} \frac{1}{2}\|\boldsymbol{w}-\boldsymbol{w}_t\|^2 \text{ s.t. } \boldsymbol{w}\cdot\boldsymbol{x}_t = y_t - \epsilon. \tag{5}$$

If $\hat{y}_t > y_t$, then solving Eq. (3) is equivalent to solving

$$\boldsymbol{w}_{t+1} = \operatorname*{argmin}_{\boldsymbol{w}\in\mathbf{R}_+^m} \frac{1}{2}\|\boldsymbol{w}-\boldsymbol{w}_t\|^2 \text{ s.t. } \boldsymbol{w}\cdot\boldsymbol{x}_t = y_t + \epsilon. \tag{6}$$

Notice that there is a non-negativity constraint in Eq. (6) but *not* in Eq. (5). For Eq. (5), there is therefore a closed-form solution, which is the same as that of the original PA:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \kappa^*\boldsymbol{x}_t \text{ where } \kappa^* = \frac{\ell_t^\epsilon}{\|\boldsymbol{x}_t\|^2}. \tag{7}$$

Unfortunately, due to the additional non-negativity constraint, such a closed-form solution does not exist for Eq. (6). Its Lagrangian is:

$$\mathcal{L}(\boldsymbol{w},\theta,\boldsymbol{\mu}) = \frac{1}{2}\|\boldsymbol{w}-\boldsymbol{w}_t\|^2 + \theta(\boldsymbol{w}\cdot\boldsymbol{x}_t - y_t - \epsilon) - \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{w},$$

where $\theta\in\mathbf{R}_+$ and $\boldsymbol{\mu}\in\mathbf{R}_+^m$ are Lagrange multipliers. The optimization problem in Eq. (6) has a convex objective function and feasible affine constraints. These are sufficient conditions for Slater's condition to

hold. Therefore, satisfying the Karush-Kuhn-Tucker (KKT) conditions is a necessary and sufficient condition for optimality. Let $\boldsymbol{w}^*$ and $(\theta^*, \boldsymbol{\mu}^*)$ be the primal and dual optimal points, where $\boldsymbol{w}^* = [w_1^*, \dots, w_m^*]$ and $\boldsymbol{\mu}^* = [\mu_1^*, \dots, \mu_m^*]$. Differentiating the Lagrangian with respect to $\boldsymbol{w}$ and solving for zero gives:

$$\boldsymbol{w}^* = \boldsymbol{w}_t - \theta^* \boldsymbol{x}_t + \boldsymbol{\mu}^*.$$

The KKT complementary slackness conditions require that $w_j^* \mu_j^* = 0 \; \forall j$. It follows that if $w_j^* > 0$, then $\mu_j^* = 0$ and thus $w_j^* = w_{t,j} - \theta^* x_{t,j}$. Otherwise, the non-negativity constraint implies that $w_j^* = 0$. Wrapping up the two cases, we obtain the following update

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}^* = \max(\boldsymbol{w}_t - \theta^* \boldsymbol{x}_t, 0), \tag{8}$$

where the max operator is taken element-wise. Intuitively, the update in Eq. (8) is parameterized by a single scalar value $\theta^* \in \mathbf{R}_+$. Additionally, we see that it leads to *sparse* solutions. The following lemma shows that finding θ^* can be cast as a root finding problem.

Lemma 2 *Assume $\hat{y}_t > y_t$. Then, the optimal solution of Eq. (6) is $\boldsymbol{w}_{t+1} = \max(\boldsymbol{w}_t - \theta^* \boldsymbol{x}_t, 0)$, where θ^* is a root of:*

$$f_t(\theta) = \max(\boldsymbol{w}_t - \theta \boldsymbol{x}_t, 0) \cdot \boldsymbol{x}_t - y_t - \epsilon. \tag{9}$$

Moreover, if $y_t + \epsilon > 0$, then θ^* is unique.

Proof. The proof technique is similar to Liu and Ye's (2009), who proved a similar result for the problem of Euclidean projection on an ℓ_1 ball. First, following Eq. (8), we must clearly have $\theta^* > 0$, since otherwise $\boldsymbol{w}_t$ is not updated. It follows from the KKT complementary slackness condition $\theta^*(\boldsymbol{w}^* \cdot \boldsymbol{x}_t - y_t - \epsilon) = 0$ that $\boldsymbol{w}^* \cdot \boldsymbol{x}_t - y_t - \epsilon = 0$. Injecting Eq. (8) in $\boldsymbol{w}^* \cdot \boldsymbol{x}_t - y_t - \epsilon = 0$ gives Eq. (9). Next, we need two scalars θ_l and θ_u such that $f_t(\theta_l) > 0$ and $f_t(\theta_u) \leq 0$, respectively. We can choose $\theta_l = 0$, since we clearly have $f_t(0) = \boldsymbol{w}_t \cdot \boldsymbol{x}_t - y_t - \epsilon = \ell_t^\epsilon > 0$. For θ_u, it is easy to verify that $\theta_u = \max_{j : x_{t,j} > 0} \frac{w_{t,j}}{x_{t,j}}$ satisfies the condition. We also know that f_t is continuous and strictly decreasing in $[\theta_l, \theta_u]$. Following the Intermediate Value Theorem, f_t has a root in this interval. If $y_t + \epsilon > 0$, then $f_t(\theta_u) < 0$ and the root is unique. □

In the special case when $y_t + \epsilon = 0$ (which can only happen if $y_t = \epsilon = 0$), solving the above root finding problem is trivial, since $f_t(\theta) = 0 \; \forall \theta \geq \theta_u$. Therefore, $\theta^* = \theta_u$ is a solution. When $y_t + \epsilon > 0$, we choose to solve the root finding problem by bisection, which has worst-case linear-time complexity and can handle the non-differentiable function f_t. Bisection works by repeatedly halving an interval and selecting the subinterval in which the root must lie. We summarize the procedure in Algorithm 2. Note that we used $\frac{|f_t(\theta)|}{y_t + \epsilon} \leq \tau$ as stopping criterion, where the denominator is intended to normalize the error $|f_t(\theta)|$.

We summarize the entire NN-PA procedure for solving Eq. (3) in Algorithm 1. We used the fact that Eq. (7) and Eq. (8) can be combined into Eq. (11) if we set κ^* and θ^* as follows

$$\begin{cases} \kappa^* = \dfrac{\ell_t^\epsilon}{\|\boldsymbol{x}_t\|^2} & \theta^* = 0 \quad \text{if } \hat{y}_t < y_t \\ \kappa^* = 0 & \theta^* = f_t^{-1}(0) \quad \text{if } \hat{y}_t > y_t, \end{cases} \tag{10}$$

where we used $f_t^{-1}(0)$ to denote any root of f_t.

3 Regularized and approximate updates

Unfortunately, NN-PA updates $\boldsymbol{w}_t$ as much as needed to satisfy the constraint imposed by the current training instance. In real-life problems, this behavior might be undesirable, especially in the presence of noise or outliers. To prevent overfitting, soft formulations of support vector machines introduce slack variables in the optimization problem. Similarly, in NN-PA, we can introduce a *non-negative* slack variable ξ into the optimization problem:

$$\begin{aligned} \boldsymbol{w}_{t+1}, \; \xi^* = \underset{\boldsymbol{w} \in \mathbf{R}_+^m, \; \xi \in \mathbf{R}_+}{\operatorname{argmin}} & \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_t\|^2 + C\xi \\ \text{s.t. } & |\boldsymbol{w} \cdot \boldsymbol{x}_t - y_t| \leq \epsilon + \xi, \end{aligned} \tag{12}$$

where $C > 0$ is a parameter which controls the trade-off between being *conservative* (do not change the model too much) and *corrective* (satisfy the constraint). Following the naming convention of Crammer et al. [2006], we call this variant NN-PA-I. Again, similarly to Lemma 1, we can consider the $\hat{y}_t < y_t$ and $\hat{y}_t > y_t$ cases separately. When $\hat{y}_t < y_t$, the non-negativity constraint can be ignored and the closed-form solution is the same as that of the original PA-I [Crammer et al., 2006, Appendix A]: we can set κ^* to $\min(C, \frac{\ell_t^\epsilon}{\|\boldsymbol{x}_t\|^2})$, i.e., the step size is clipped to C. When $\hat{y}_t > y_t$, we can use a similar step size clipping. However, this requires computing the step size by bisection beforehand. The following lemma provides a much more efficient approach.

Lemma 3 *Assume $\hat{y}_t > y_t$. Then, the optimal solution of Eq. (12) is $\boldsymbol{w}_{t+1} = \max(\boldsymbol{w}_t - \theta^* \boldsymbol{x}_t, 0)$, where $\theta^* = C$ if $f_t(C) \geq 0$ and $\theta^* = f_t^{-1}(0)$ otherwise.*

Lemma 3 is remarkable in that we can completely avoid finding $f_t^{-1}(0)$ if $f_t(C) \geq 0$. It is easy to see that the smaller C (the more regularized), the more

Algorithm 1 NN-PA algorithms

Input: $\boldsymbol{w}_t$, $\boldsymbol{x}_t$, y_t, C, ϵ
Compute prediction $\hat{y}_t = \boldsymbol{w}_t \cdot \boldsymbol{x}_t$
Suffer loss $\ell_t^\epsilon = \max(|\hat{y}_t - y_t| - \epsilon, 0)$
if $\ell_t^\epsilon = 0$ **then**
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$
else
  Compute κ^* and θ^* by
    Eq. (10) for NN-PA
    Eq. (13) for NN-PA-I
    Eq. (14) for NN-PA-II

$$\boldsymbol{w}_{t+1} = \max\Big(\boldsymbol{w}_t + (\kappa^* - \theta^*)\boldsymbol{x}_t, 0\Big) \tag{11}$$

end if
Output: $\boldsymbol{w}_{t+1}$

Algorithm 2 Root finding by bisection

Input: $\boldsymbol{w}_t$, $\boldsymbol{x}_t$, y_t, ϵ, τ
$l \leftarrow \theta_l = 0$
$u \leftarrow \theta_u = \max_{j:x_{t,j}>0} \frac{w_{t,j}}{x_{t,j}}$
$s \leftarrow \infty$
while $|s|/(y_t + \epsilon) > \tau$ **do**
  Update midpoint $\theta \leftarrow (l+u)/2$
  $s \leftarrow f_t(\theta)$ (NN-PA-I) or $\hat{f}_t(\theta)$ (NN-PA-II)
  if $s < 0$ **then**
    Update upper bound $u \leftarrow \theta$
  else
    Update lower bound $l \leftarrow \theta$
  end if
end while
Output: θ

likely this condition will hold. Therefore, the smaller C, the more likely we will not need to find $f_t^{-1}(0)$ at all. To summarize, NN-PA-I sets κ^* and θ^* in Eq. (11) as follows

$$\begin{cases} \kappa^* = \alpha_t & \theta^* = 0 & \text{if } \hat{y}_t < y_t \\ \kappa^* = 0 & \theta^* = C & \text{if } \hat{y}_t > y_t \text{ and } f_t(C) \geq 0 \\ \kappa^* = 0 & \theta^* = f_t^{-1}(0) & \text{if } \hat{y}_t > y_t \text{ and } f_t(C) < 0, \end{cases} \tag{13}$$

where $\alpha_t = \min(C, \frac{\ell_t^\epsilon}{\|\boldsymbol{x}_t\|^2})$.

We also introduce a NN-PA-II variant, which replaces $C\xi$ in Eq. (12) by $C\xi^2$, i.e., we penalize constraint violations quadratically instead of linearly. NN-PA-II (derivation omitted) sets κ^* and θ^* in Eq. (11) as follows

$$\begin{cases} \kappa^* = \beta_t & \theta^* = 0 & \text{if } \hat{y}_t < y_t \\ \kappa^* = 0 & \theta^* = \hat{f}_t^{-1}(0) & \text{if } \hat{y}_t > y_t, \end{cases} \tag{14}$$

where $\beta_t = \frac{\ell_t^\epsilon}{\|\boldsymbol{x}_t\|^2 + \frac{1}{2C}}$ and $\hat{f}_t^{-1}(0)$ denotes any root of the "regularized" function $\hat{f}_t(\theta) = f_t(\theta) - \frac{\theta}{2C}$.

NN-PA-I is fundamentally more efficient than NN-PA-II, because, thanks to Lemma 3, we can completely avoid solving the root finding problem whenever $f_t(C) \geq 0$. However, it can still be computationally expensive when $f_t(C) < 0$. To alleviate the need to solve a root finding problem, both in NN-PA-I and NN-PA-II, we propose to replace $f_t^{-1}(0)$ in Eq. (13) and $\hat{f}_t^{-1}(0)$ in Eq. (14) by α_t and β_t, respectively. The resulting updates are exact when $\hat{y}_t < y_t$ but may only be approximate when $\hat{y}_t > y_t$. Indeed, in the latter case, the update ensures that $\ell^\epsilon(\boldsymbol{w}_{t+1}; (\boldsymbol{x}_t, y_t)) \leq \ell^\epsilon(\boldsymbol{w}_t; (\boldsymbol{x}_t, y_t))$ but not that the optimization problem is solved optimally. Despite being approximate, we will show in the next section that such updates are sufficient to derive an $O(\sqrt{T})$ regret bound, where T is the total number of rounds.

4 Regret analysis

We study the performance of NN-PA-I in the regret bound model, in which the cumulative loss of our algorithm is compared to the cumulative loss of any competing hypothesis. The competing hypothesis can be chosen in hindsight, i.e., after observing the entire sequence of input-target pairs. The following theorem holds for *both* the regular and approximate versions of NN-PA-I.

Theorem 1 *Let $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_T, y_T)$ be a sequence of input-target pairs, where $\boldsymbol{x}_t \in \mathbf{R}_+^m$ and $y_t \in \mathbf{R}_+$. Let $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_T$ be a sequence of vectors obtained by the regular or approximate NN-PA-I update rules. Then $\forall \boldsymbol{w} \in \mathbf{R}_+^m$*

$$\sum_{t=1}^T \ell^\epsilon(\boldsymbol{w}_t; (\boldsymbol{x}_t, y_t)) - \sum_{t=1}^T \ell^\epsilon(\boldsymbol{w}; (\boldsymbol{x}_t, y_t)) \leq \frac{\|\boldsymbol{w}\|^2}{2C} + \frac{TC\rho}{2}, \tag{15}$$

where $\rho = \max_t \|\boldsymbol{x}_t\|^2$.

To prove this result (c.f. supplementary material), we adopt a primal-dual view of online learning [Shalev-Shwartz and Singer, 2007]. In this view, we define an optimization problem and we cast online learning as the task of incrementally increasing the dual objective function. We prove that the NN-PA-I update rules guarantee that the dual increases. Note that we can minimize the right-hand side of (15) with respect to C. By choosing $C = \frac{\|\boldsymbol{w}\|}{\sqrt{T\rho}}$, the right-hand side becomes

$\|\boldsymbol{w}\|\sqrt{T\rho}$, i.e., we obtain an $O(\sqrt{T})$ regret bound. In other words, the difference between the cumulative loss suffered by NN-PA-I and the cumulative loss of any fixed non-negative weight vector is bounded by a term that is sub-linear in the number of iterations T.

As an immediate corollary of Theorem 1, we analyze the performance of NN-PA-I for doing k passes over the observed entries of matrix R.

Corollary 1 *Let $\mathcal{S}$ be a sequence of rounds of length $T = k|\Omega|$ such that on each round t we choose an entry $(u_t, i_t) \in \Omega$, and all entries of R are chosen k times in total. If we keep Q fixed on all rounds and use NN-PA-I to update $\boldsymbol{p}_{u_t}^t$ to $\boldsymbol{p}_{u_t}^{t+1}$, then $\forall P \in \mathbf{R}_+^{n \times m}$*

$$\sum_{t \in \mathcal{S}} \ell^\epsilon(\boldsymbol{p}_{u_t}^t; (\boldsymbol{q}_{i_t}, r_{u_t,i_t})) - \sum_{t \in \mathcal{S}} \ell^\epsilon(\boldsymbol{p}_{u_t}; (\boldsymbol{q}_{i_t}, r_{u_t,i_t})) \leq \frac{\|P\|_F^2}{2C} + \frac{TC\rho_q}{2},$$

where $\|.\|_F$ denotes the Frobenius norm and $\rho_q = \max_i \|\boldsymbol{q}_i\|^2$. If we keep P fixed on all rounds and use NN-PA-I to update $\boldsymbol{q}_{i_t}^t$ to $\boldsymbol{q}_{i_t}^{t+1}$, then the same bound holds but with the roles of P and Q swapped.

In future work, we plan to further study NN-PA's theoretical guarantees in a batch setting.

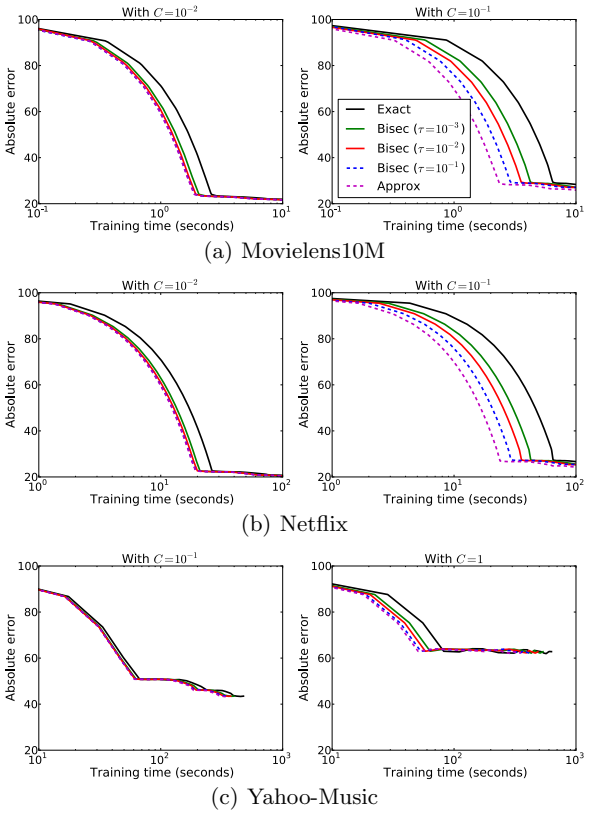
5 Empirical results

We conduct experiments on the following three popular large-scale recommendation system datasets.

Table 1: Datasets used in our experiments.

Dataset	Users	Items	Ratings
Movielens10M	69,878	10,677	10,000,054
Netflix	480,189	17,770	100,480,507
Yahoo-Music	1,000,990	624,961	252,800,275

Due to space limitations, we focus on the NN-PA-I variant. To determine prediction error, we use stratified selection (w.r.t. the number of ratings per user) in order to split each dataset's ratings into 4/5 training and 1/5 testing. The task is to predict the test ratings. Throughout our experiments we set $\epsilon = 0$, which is equivalent to choosing the *absolute loss* as our evaluation measure. For easier comparison, we normalize results by the absolute norm of the rating matrix R. Throughout this section, we report the average results over 5 runs. For each run, we initialize P to $\mathbf{0}$ and Q randomly with rank $m = 30$ (all algorithms use the same initialization in a given run). The number of ratings indicated in Table 1 corresponds to the number of times the NN-PA-I optimization problem must be solved in order to make one pass over the rating matrix R.



(a) Movielens10M

(b) Netflix

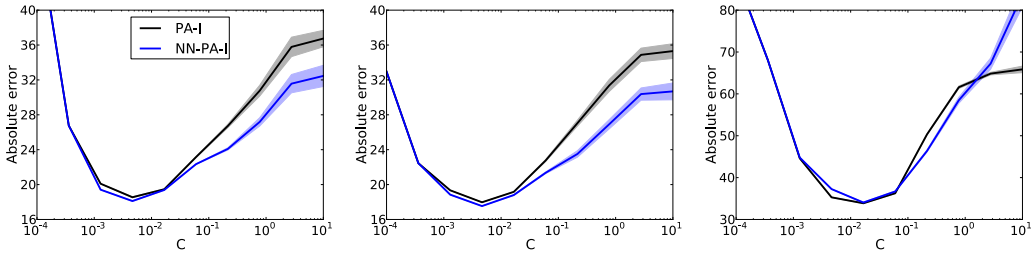
(c) Yahoo-Music

Figure 1: **Convergence of different methods for solving the NN-PA-I optimization problem.** We compare convergence for two different values of C. Absolute error is normalized, training time (seconds) is in log-scale. Results are the average of 5 runs.

5.1 Solving the NN-PA-I problem

As we mentioned previously, the optimization problem associated with NN-PA-I can be broken down into two sub-problems. If $\hat{y}_t < y_t$, there exists a computationally cheap closed-form solution. If $\hat{y}_t > y_t$, we need to solve Eq. (6), for which no closed-form solution is available. In this experiment, we compare three methods for solving Eq. (6):

- Exact: pivot algorithm for finding an exact solution (c.f. Section 6.3),
- Bisec: bisection algorithm for finding a solution with tolerance τ (c.f. Algorithm 2),
- Approx: approximate update (c.f. Section 3).



(a) Movielens10M (b) Netflix (c) Yahoo-Music

Figure 2: **Comparison between PA-I and NN-PA-I.** We compare prediction error for different values of C. Absolute error is normalized. Results are the average of 5 runs (each run uses a different initialization).

Figure 1 compares the speed of convergence of the above methods for two different values of the regularizer parameter C. When C is small, we see that the different methods perform comparably. This is because the condition $f_t(C) \geq 0$ in Lemma 3 is more likely to hold if C is small. We can thus avoid solving an expensive optimization problem most of the time. When C is larger, we see that solving the sub-problem exactly or accurately is slower and does not reduce the final prediction error. In the next sections, we therefore always use the approximate variant of NN-PA-I.

5.2 Comparison with existing methods

Comparison with PA-I. To investigate the validity of using NMF for the task of matrix completion, we compare NN-PA-I to the original PA-I [Crammer et al., 2006] (i.e., without non-negativity constraint). Both algorithms use the same non-negative initializations for P and Q and perform 5 passes over the rating matrix. Figure 2 compares the prediction error achieved by the two algorithms for different values of the regularization parameter C. PA-I and NN-PA-I perform comparably when using the optimal C parameter. However, NN-PA-I is more robust with respect to the choice of C, which suggests that non-negativity constraints act as a form of regularization. For PA-I, we found that initializing P and Q with both positive and negative values performed poorly compared to non-negative initialization. This confirms that good solutions tend to be non-negative.

Comparison with SGD and Coordinate Descent. We also compare NN-PA-I to Stochastic Gradient Descent (SGD) and Coordinate Descent (CD). We choose these methods because they are considered state-of-the-art in the recommendation system literature and can easily incorporate the non-negativity constraint. We use SGD and CD to minimize the NMF objective $\sum_{(u,i)\in\Omega} \ell(\boldsymbol{p}_u \cdot \boldsymbol{q}_i, r_{u,i}) + \frac{\lambda}{2}(\|P\|_F^2 + \|Q\|_F^2)$ with respect to $P \in \mathbf{R}_+^{n\times m}$ and $Q \in \mathbf{R}_+^{m\times d}$, where ℓ is a loss function and $\lambda > 0$ is a regularization parameter. We use the absolute loss for SGD and the squared loss for CD, since CD cannot handle non-differentiable loss functions. Common learning rates for SGD include $\eta_t = \frac{\eta_0}{\sqrt{t}}$ and $\eta_t = \frac{1}{\lambda t}$. We choose the latter, since it does not require any extra hyper-parameter and is known to achieve a $O(\sqrt{T})$ regret bound [Shalev-Shwartz, 2007]. The complexity of CD for doing one pass is the same as that of NN-PA-I and SGD, i.e., $O(m|\Omega|)$ [Yu et al., 2012]. However, in practice, CD typically requires more memory than NN-PA-I and SGD because of the necessity to maintain a residual matrix.

It is difficult to compare the convergence of methods that minimize different objectives, because the choice of the regularization parameter influences the trade-off between fast early convergence and low final prediction error. We therefore compare the prediction error achieved by NN-PA-I, SGD and CD for a given "computational budget": 1, 3, 5 passes over the rating matrix R. We hold 25% of the training set to select the regularization parameter (C for NN-PA-I, λ for SGD and CD) that minimizes absolute error then retrain using the entire training set once the parameter has been selected.

Results are given in Table 2. NN-PA-I achieves the lowest prediction error on all datasets when doing 1 or 3 passes. CD achieves slightly lower prediction error than NN-PA-I on two datasets when doing 5 passes, although at the cost of substantially longer training times. Interestingly, the standard deviation of the prediction error for NN-PA-I was much smaller than SGD, meaning that NN-PA-I is less sensitive than SGD to initialization. This is because NN-PA-I chooses the step size (κ^* and θ^* in Eq. (11)) by solving a data-dependent optimization problem, whereas SGD uses a data-independent learning rate.

In terms of sparsity, we found that NN-PA-I and SGD do not achieve very sparse solutions. For example, on the Yahoo-Music dataset, the percentage of non-

Table 2: **Comparison between NN-PA-I, Stochastic Gradient Descent (SGD) and Coordinate Descent (CD).** We compare absolute error (normalized) and training time (seconds) for a given "computational budget": 1, 3 or 5 passes over the datasets. We tune the regularization parameter (C or λ) using a development set and report the results on the test set.

Dataset	Passes		NN-PA-I	SGD	CD
Movielens10M	1	Error	**23.75 ± 0.05**	31.58 ± 1.91	34.59 ± 0.03
		Time	3.24 ± 0.01	2.68 ± 0.01	3.88 ± 0.01
	3	Error	**20.91 ± 0.04**	25.27 ± 0.02	21.38 ± 0.05
		Time	10.28 ± 0.01	8.09 ± 0.08	12.73 ± 0.01
	5	Error	20.61 ± 0.01	24.54 ± 0.02	**20.47 ± 0.01**
		Time	17.40 ± 0.06	13.44 ± 0.03	22.57 ± 0.01
Netflix	1	Error	**22.32 ± 0.01**	27.29 ± 0.81	34.31 ± 0.01
		Time	34.29 ± 0.10	27.68 ± 0.41	36.58 ± 0.37
	3	Error	**20.01 ± 0.01**	24.28 ± 0.01	21.60 ± 0.01
		Time	109.53 ± 2.97	82.98 ± 0.14	153.46 ± 0.72
	5	Error	19.64 ± 0.01	23.70 ± 0.14	**19.37 ± 0.01**
		Time	181.43 ± 0.22	133.59 ± 0.60	270.28 ± 0.49
Yahoo-Music	1	Error	**50.64 ± 0.33**	52.52 ± 0.68	57.08 ± 0.28
		Time	114.16 ± 0.05	96.89 ± 0.04	170.38 ± 0.06
	3	Error	**38.44 ± 0.16**	44.63 ± 1.24	45.32 ± 0.23
		Time	335.13 ± 0.34	291.59 ± 0.24	468.86 ± 0.69
	5	Error	**36.26 ± 0.09**	41.62 ± 1.15	37.97 ± 0.21
		Time	576.08 ± 0.73	475.86 ± 2.90	787.57 ± 1.68

zero coefficients in matrix Q obtained by NN-PA-I and SGD with best-tuned regularization parameter was 85% whereas it was 45% by CD. The fact that solutions are not very sparse is a well-known issue of online algorithms in general (see, e.g., [Blondel et al., 2013] and references therein).

5.3 Model interpretability

In previous work, Zhang et al. [2006] found that NMF can empirically reduce prediction error compared to conventional methods such as Singular Value Decomposition (SVD). Another advantage of NMF is that it tends to obtain interpretable solutions [Lee and Seung, 1999]. In the case of recommendation system data, the bases contained in matrix Q can be interpreted as topics or genres. Thanks to the non-negativity constraint, the coefficients in a basis correspond to the relative importance of items (e.g., movies) in that basis. It is therefore possible to sort items in a basis in order of decreasing coefficients.

To verify whether NN-PA-I can indeed extract relevant topics, we compute the matrix decomposition of the Movielens10M dataset for 5 different initializations and select the one which achieves the lowest absolute error. In the Movielens10M dataset, each movie was manually assigned to one or more of twenty categories (e.g., comedy, thriller, etc...). Following Zhang et al. [2006], we thus set the rank to $m = 20$ for this experiment. Table 3 indicates the top 5 movies in each topic or genre. Although some topics contain irrelevant movies, we see that related movies tend to cluster into the same topics. For example, Topic 1 contains horror movies and Topic 2 contains comedies. NN-PA can therefore be used to gain insights about the data.

6 Related work

6.1 Non-negative Matrix Factorization

NMF [Lee and Seung, 1999] is an unsupervised matrix factorization method, the goal of which is to minimize the reconstruction error between a non-negative matrix and its low rank decomposition. In this decomposition, the matrix is represented as a strictly additive weighted sum of bases. Popular optimization methods for NMF include multiplicative methods [Lee and Seung, 2001], projected gradient descent [Lin, 2007] and coordinate descent [Hsieh and Dhillon, 2011]. Zhang et al. [2006] previously applied NMF to the task of non-negative matrix completion and showed that NMF can reduce prediction error compared to conventional methods such as Singular Value Decomposition (SVD). Sindhwani et al. [2010] proposed a weighted NMF method, in which they also enforced non-negativity on the decomposition so as to lead to interpretable models.

To handle large-scale or real-time data, several authors have proposed online NMF algorithms [Cao et al., 2007, Mairal et al., 2010, Wang et al., 2011]. However, they all assume that the data matrix is fully observed. On the other hand, several state-of-the-art optimization methods developed in the recommendation system literature, such as SGD (e.g., [Koren et al., 2009]) or coordinate descent (e.g., [Yu et al., 2012]), can easily incorporate a non-negativity constraint. We thus compared NN-PA to these methods in Section 5.

Table 3: **Topic model.** The top 5 ranked movies in 6 out of 20 topics extracted from the Movielens10M dataset by NN-PA-I. Categories in parentheses are the tags movies are annotated with in the Movielens10M dataset.

Topic 1	Topic 2	Topic 3
Scream (Comedy, Horror, Thriller)	Dumb & Dumber (Comedy)	Pocahontas (Animation, Children, Musical, ...)
The Fugitive (Thriller)	Ace Ventura: Pet Detective (Comedy)	Aladdin (Adventure, Animation, Children, ...)
The Blair Witch Project (Horror, Thriller)	Five Corners (Drama)	Merry Christmas Mr. Lawrence (Drama, War)
Deep Cover (Action, Crime, Thriller)	Ace Ventura: When Nature Calls (Comedy)	Toy Story (Adventure, Animation, Children, ...)
The Plague of the Zombies (Horror)	Jump Tomorrow (Comedy, Drama, Romance)	The Sword in the Stone (Animation, Children, Fantasy, ...)

Topic 4	Topic 5	Topic 6
Belle de jour (Drama)	Four Weddings and a Funeral (Comedy, Romance)	Terminator 2: Judgment Day (Action, Sci-Fi)
Jack the Bear (Comedy, Drama)	The Birdcage (Comedy)	Braveheart (Action, Drama, War)
The Cabinet of Dr. Caligari (Crime, Drama, Fantasy, ...)	Shakespeare in Love (Comedy, Drama, Romance)	Aliens (Action, Horror, Sci-Fi)
M*A*S*H (Comedy, Drama, War)	Henri V (Drama, War)	Mortal Kombat (Action, Adventure, Fantasy)
Bed of Roses (Drama, Romance)	Three Men and a Baby (Comedy)	Congo (Action, Adventure, Mystery, ...)

6.2 Passive-aggressive algorithms

Passive-aggressive algorithms [Crammer et al., 2006] are a family of online algorithms for supervised learning. On each round, passive-aggressive algorithms solve a constrained optimization problem which balances between two competing goals: being *conservative*, in order to retain information acquired on preceding rounds, and being *corrective*, in order to make a more accurate prediction when presented with the same instance again. Passive-aggressive algorithms enjoy a certain popularity in the Natural Language Processing community, where they are often used for large-scale batch learning. Using the result of an optimization problem comprising two opposing terms was previously advocated by several authors for deriving online updates [Littlestone, 1989, Herbster, 2001] and can be used to motivate SGD updates as well [Kivinen and Warmuth, 1997]. Consider the following optimization problem:

$$\boldsymbol{w}_{t+1} = \operatorname*{argmin}_{\boldsymbol{w} \in \mathbf{R}^m_+} \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_t\|^2 + \eta_t \ell^\epsilon(\boldsymbol{w}; (\boldsymbol{x}_t, y_t)),$$

where η_t is the learning rate parameter. Since the above optimization problem does not enjoy an analytical solution, (projected) SGD performs, on any round when $\ell^\epsilon_t > 0$, the following approximate update instead

$$\begin{aligned}\boldsymbol{w}_{t+1} &= \max(\boldsymbol{w}_t - \eta_t \boldsymbol{g}_t, 0) \\ &= \max(\boldsymbol{w}_t + \eta_t s_t \boldsymbol{x}_t, 0),\end{aligned}$$

where $\boldsymbol{g}_t$ is a subgradient of the function $\ell^\epsilon(\cdot, (\boldsymbol{x}_t, y_t))$ evaluated at $\boldsymbol{w}_t$ and $s_t = \text{sign}(y_t - \hat{y}_t)$. The similarity between the above update and NN-PA is striking and sheds some light on NN-PA's behavior. Namely, NN-PA can be seen as automatically adjusting the learning rate η_t in a data-dependent fashion, i.e., based on $\boldsymbol{w}_t$, $\boldsymbol{x}_t$, y_t, C and ϵ.

6.3 Relation to projection onto the simplex

The optimization problem in Eq. (6) can be seen as the problem of Euclidean projection on the convex set $\{\boldsymbol{w} \in \mathbf{R}^m_+ \mid \boldsymbol{w} \cdot \boldsymbol{x}_t = y_t + \epsilon\}$. Interestingly, this problem generalizes the well-known problem of projection onto the standard simplex. Indeed, the two problems are the same if we set $\boldsymbol{x}_t$ to a vector of all ones and $y_t + \epsilon$ to 1. Duchi et al. [2008] propose an *exact* pivot algorithm with expected $O(m)$ complexity for solving the projection onto the simplex. For completeness, we derive a similar algorithm for solving Eq. (6) in the supplementary material. Our algorithm includes Duchi et al.'s algorithm as a special case. However, as pointed out in Section 5.1, solving the problem exactly is slower and does not reduce prediction error.

7 Conclusion

We introduced NN-PA and demonstrated its effectiveness on three large-scale matrix completion problems. We also confirmed that NN-PA is able to learn easy-to-interpret matrix decompositions and we provided $O(\sqrt{T})$ regret bounds. A straightforward extension is to apply NN-PA to semi-NMF, e.g., by applying a non-negativity constraint to the basis matrix Q but not the coefficient matrix P. Future work also includes investigating the effectiveness of NN-PA on real-time streaming data.

Acknowledgements

This work was partially supported by the FIRST program. We thank Olivier Grisel for reading an early draft of this paper and suggesting the idea of approximate updates.

References

Mathieu Blondel, Kazuhiro Seki, and Kuniaki Uehara. Block coordinate descent algorithms for large-scale sparse multiclass classification. *Machine Learning*, 93(1):31–52, 2013.

Leon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems 20*, pages 161–168. 2008.

Bin Cao, Dou Shen, Jian-Tao Sun, Xuanhui Wang, Qiang Yang, and Zheng Chen. Detect and track latent factors with online nonnegative matrix factorization. IJCAI'07, pages 2689–2694, 2007.

Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the International Conference on Machine Learning*, pages 272–279, 2008.

Mark Herbster. Learning additive models online with fast evaluating kernels. In *Proceedings of the 14th Annual Conference on Computational Learning Theory*, pages 444–460, 2001.

Cho-Jui Hsieh and Inderjit S. Dhillon. Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1064–1072, 2011.

Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*, pages 556–562, 2001.

Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19 (10):2756–2779, 2007.

N. Littlestone. *Mistake bounds and logarithmic linear-threshold learning algorithms*. PhD thesis, Santa Cruz, CA, USA, 1989.

Jun Liu and Jieping Ye. Efficient euclidean projections in linear time. In *Proceedings of the International Conference on Machine Learning*, pages 657–664, 2009.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

O.L. Mangasarian. A finite newton method for classification. *Optimization Methods and Software*, pages 913–929, 2002.

Shai Shalev-Shwartz. *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, 2007.

Shai Shalev-Shwartz and Yoram Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, pages 1567–1599, 2006.

Shai Shalev-Shwartz and Yoram Singer. A primal-dual perspective of online learning algorithms. *Machine Learning*, 69(2-3):115–142, 2007.

V. Sindhwani, S.S. Bucak, Jianying Hu, and A. Mojsilovic. One-class matrix completion with low-density factorizations. In *ICDM*, pages 1055–1060, 2010.

Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research*, 10:623–656, 2009.

Fei Wang, Ping Li, and Arnd Christian Knig. Efficient document clustering via online nonnegative matrix factorizations. In *SDM'11*, pages 908–919, 2011.

Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit S. Dhillon. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *ICDM*, pages 765–774, 2012.

Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *SDM*, 2006.