ADMM SLIM: Sparse Recommendations for Many Users

Harald Steck
Netflix
Los Gatos, California
hsteck@netflix.com

Maria Dimakopoulou
Netflix
Los Gatos, California
mdimakopoulou@netflix.com

Nickolai Riabov
Netflix
Los Gatos, California
nriabov@netflix.com

Tony Jebara
Spotify & Columbia University
New York, New York
jebara@cs.columbia.edu

ABSTRACT

The Sparse Linear Method (Slim) [15] is a well-established approach for top-N recommendations. This article proposes several improvements that are enabled by the Alternating Directions Method of Multipliers (ADMM) [3, 8, 9], a well-known optimization method with many application areas. First, we show that optimizing the original Slim-objective by ADMM results in an approach where the training time is independent of the number of users in the training data, and hence trivially scales to large numbers of users. Second, the flexibility of ADMM allows us to switch on and off the various constraints and regularization terms in the original Slim-objective, in order to empirically assess their contributions to ranking accuracy on given data. Third, we also propose two extensions to the original Slim training-objective in order to improve recommendation accuracy further without increasing the computational cost. In our experiments on three well-known data-sets, we first compare to the original Slim-implementation and find that not only ADMM reduces training time considerably, but also achieves an improvement in recommendation accuracy due to better optimization. We then compare to various state-of-the-art approaches and observe up to 25% improvement in recommendation accuracy in our experiments. Finally, we evaluate the importance of sparsity and the non-negativity constraint in the original Slim-objective with subsampling experiments that simulate scenarios of cold-starting and large catalog sizes compared to relatively small user base, which often occur in practice.

CCS CONCEPTS

• **Information systems → Collaborative filtering**; **Personalization**; **Recommender systems**.

KEYWORDS

Recommender Systems, Personalization, Sparse Linear Model

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
WSDM '20, February 3–7, 2020, Houston, TX, USA
© 2020 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-6822-3/20/02.
https://doi.org/10.1145/3336191.3371774

ACM Reference Format:
Harald Steck, Maria Dimakopoulou, Nickolai Riabov, and Tony Jebara. 2020. ADMM SLIM: Sparse Recommendations for Many Users. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20), February 3–7, 2020, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3336191.3371774

1 INTRODUCTION

The Sparse Linear Method (Slim) [15] was found to yield competitive top-N recommendation-accuracy in numerous experiments in the literature. At the same time, it is known that training the Slim model can be computationally expensive. This motivated us to develop a computationally efficient approach that is geared toward the scenario where the number of users in the training data can be very large, while the number of items is moderate (of the order of 10,000 to 100,000). This is a realistic scenario in many real world applications, like for instance on premium video-streaming web-sites. Furthermore, this paper sheds light on the contributions of each individual term in Slim's training-objective, which is comprised of several constraints and regularization terms. We also propose two additions to the original Slim-objective as to improve ranking accuracy. To this end, we have employed the Alternating Directions Method of Multipliers (ADMM) [3, 8, 9], a well-known optimization method with numerous applications areas. While it has been applied to related areas (e.g., see [4, 6]), it has not yet been used for optimizing the Slim-objective for improved recommendations. We derive the iterative update equations for this model in Section 3, and outline convergence criteria of the approach in Section 6. The flexibility of ADMM is pivotal, as it enables us to modify the original Slim-objective by switching on and off as well as modifying various terms, resulting in several model-variants as outlined in Section 5. In our experiments on three well-known publicly available data-sets in Section 7, we empirically compare the different model-variants with each other, and identify the modifications that lead to further improvements. We also make a comparison to various state-of-the-art models in Section 7, including probabilistic deep learning approaches, and remarkably find that they are outperformed by our ADMM-based variants by up to 25% in ranking accuracy on the largest of the three data-sets used in our experiments. The various contributions and insights are summarized in Section 8.

2 MODEL DEFINITION

We assume that implicit feedback data are given in the form of a sparse (typically binary) matrix $X \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$, regarding the sets of users $\mathcal{U}$ and items $\mathcal{I}$; and $|\cdot|$ denotes the size of a set. A positive value (typically 1) in X indicates that the user interacted with an item, while a value of 0 indicates that no interaction has been observed.

The item-item weight-matrix $B \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$ is comprised of the model-parameters to be learned. The predicted score $S_{u,j}$ for an item $j \in I$ given a user $u \in \mathcal{U}$ is defined in our model by the dot product

$$S_{u,j} = X_{u,\cdot} \cdot B_{\cdot,j} \tag{1}$$

where $X_{u,\cdot}$ refers to row u, and $B_{\cdot,j}$ to column j. The original optimization problem of Slim was defined in [15] as follows:

$$\begin{aligned} \min_{B} \quad & \frac{1}{2} \cdot \|X - XB\|_F^2 + \frac{\lambda_2}{2} \cdot \|B\|_F^2 + \lambda_1 \cdot \|B\|_1 \\ \text{s.t.} \quad & \operatorname{diag}(B) = 0 \\ & B_{i,j} \geq 0 \quad \forall i, j \in \mathcal{I} \end{aligned} \tag{2}$$

where the matrix of model parameters B is learned by minimizing the square loss between the data X and the predicted scores $S = XB$, using the elastic net regularization [24] with hyper-parameters λ_2 and λ_1 ($\|\cdot\|_F$ denotes the Frobenius norm and $\|B\|_1 = \sum_{i,j} |B_{i,j}|$ the L1-norm of matrix B). The minimization problem is subject to two constraints in the Slim-approach: (1) the diagonal of B has to be zero in order to avoid the trivial solution $B = I$ (where I is the identity matrix), and (2) all the weights $B_{i,j}$ have to be non-negative.

3 OPTIMIZATION VIA ADMM

As to solve the optimization problem in Eq. 2, the Slim-paper [15] as well as follow-up papers [13, 18], took advantage of the fact that this problem regarding the matrix B decouples into independent optimization problems, one for each of the $n = |\mathcal{I}|$ columns (i.e., vectors) $B_{\cdot,j}$, due to the identity

$$\|X - XB\|_F^2 = \sum_{j \in \mathcal{I}} ||X_{\cdot,j} - XB_{\cdot,j}||_2^2$$

Computationally efficient algorithms exist for solving a vector-valued elastic-net problem, e.g., the so-called LARS-EN algorithm [7, 24]. However, $n = |\mathcal{I}|$ such problems need to be solved, which can be computationally expensive.

For this reason, we take a different approach in this paper, optimizing the entire matrix B at once. This results not only in computational speed-ups in our experiments (see Section 7), but we also observe improved optimization compared to the original Slim-implementation [15], as reflected by the increased recommendation accuracy in Table 2. Moreover, this approach allows one to readily switch on and off the various constraints and regularization terms in the original optimization problem (see Eq. 2) – the resulting insights based on the empirical results are discussed in Section 7.

The key idea of this paper is to use the method of Lagrangian multipliers (see also [21]), as well as one of its generalizations, Alternating Directions Method of Multipliers (ADMM). ADMM was developed in the 1970s in [8, 9], and is summarized in the recent review article [3]. The original optimization problem in Eq. 2, where B is involved in several functions and constraints, can be recast in an equivalent optimization problem where the matrix B is now replaced by two matrices $B, C \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$ that are constrained to be equal:

$$\begin{aligned} \min_{B,C} \quad & f(B) + g(C) \\ \text{s.t.} \quad & B = C \end{aligned} \tag{3}$$

In the formulation of Eq. 3,

$$f(B) = \frac{1}{2}\|X - XB\|_F^2 + \frac{\lambda_2}{2} \cdot \|B\|_F^2, \quad \operatorname{dom} f = \{B | \operatorname{diag}(B) = 0\} \tag{4}$$

is the objective of a ridge regression problem with a restricted domain (dom) that absorbs the zero-diagonal constraint, and

$$g(C) = \lambda_1 \cdot \|C\|_1, \quad \operatorname{dom} g = \{C | C_{i,j} \geq 0\} \tag{5}$$

is the L1 penalty term with a restricted domain (dom) regarding the non-negativity constraint.

In ADMM, the equality constraint $B = C$ is absorbed into the objective function by using the so-called augmented Lagrangian:

$$L_\rho(B, C, \Gamma) = f(B) + g(C) + \langle \Gamma, B - C \rangle_F + \frac{\rho}{2} \cdot \|B - C\|_F^2 \tag{6}$$

where $\Gamma \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$ denotes the matrix of Lagrangian multipliers associated with the constraint $B - C = 0$, and $\rho > 0$ is the penalty parameter that applies to the squared difference between B and C, while $\langle \Gamma, B - C \rangle_F$ is the Frobenius inner product of matrices Γ and $B - C$. Matrices B and C are called the *primal variables*, and matrix Γ the *dual variable* in ADMM.

ADMM proceeds with iterative update equations that solve the constrained optimization-problem in Eq. 3. ADMM comes with several convergence guarantees regarding these iterative updates (e.g., see [3]), which allow for coordinate descent, i.e., one variable can be updated at a time, as opposed to a joint update of all variables at once. At each iterative step $k + 1$, one may hence update first matrix B, then matrix C, and finally matrix Γ:[1]

$$B^{(k+1)} = \operatorname*{argmin}_{B} L_\rho\left(B, C^{(k)}, \Gamma^{(k)}\right) \tag{7}$$

$$C^{(k+1)} = \operatorname*{argmin}_{C} L_\rho\left(B^{(k+1)}, C, \Gamma^{(k)}\right) \tag{8}$$

$$\Gamma^{(k+1)} = \Gamma^k + \rho \cdot \left(B^{(k+1)} - C^{(k+1)}\right) \tag{9}$$

3.1 Update of B

We observe from Eq. 6 that the update of matrix B is not subject to the L1-norm regularization nor the non-negativity constraint. Hence, Eq. 7 can be re-written as:

$$\begin{aligned} B^{(k+1)} = \operatorname*{argmin}_{B} \Bigg(& \frac{1}{2}\|X - XB\|_F^2 + \frac{\lambda_2}{2} \cdot \|B\|_F^2 + \gamma^\top \operatorname{diag}(B) \\ & + \langle \Gamma^{(k)}, B - C^{(k)} \rangle_F + \frac{\rho}{2} \cdot \|B - C^{(k)}\|_F^2 \Bigg) \end{aligned} \tag{10}$$

In Eq. 10, we have introduced the vector of Lagrangian multipliers $\gamma \in \mathbb{R}^{|\mathcal{I}|}$ in order to absorb the zero-diagonal domain-constraint, $\operatorname{diag}(B) = 0$, of Eq. 4 in the term $\gamma^\top \operatorname{diag}(B)$. The update of matrix B can be solved in closed form by setting the derivative of Eq. 10 to zero and rearranging terms (see also [21]). Given the values of

[1]Note that we use the unscaled notation of ADMM throughout this paper, while the scaled version $U = \Gamma/\rho$ is often used in [3].

$C^{(k)}$ and $\Gamma^{(k)}$ obtained at the previous iteration k of ADMM, the solution for $B^{(k+1)}$ at iteration $k+1$ is given by

$$\begin{aligned} B^{(k+1)} &= \left(X^\top X + (\lambda_2+\rho)I\right)^{-1}\left(X^\top X - \Gamma^{(k)} + \rho C^{(k)} - \mathrm{diagMat}(\gamma)\right) \\ &= \tilde{B}^{(k+1)} - P \cdot \mathrm{diagMat}(\gamma) \end{aligned} \tag{11}$$

where diagMat(γ) denotes the diagonal matrix with the vector of Lagrangian multipliers γ on its diagonal, and we defined

$$P \quad := \quad \left(X^\top X + (\lambda_2+\rho)\cdot I\right)^{-1} \tag{12}$$

$$\tilde{B}^{(k+1)} \quad := \quad P \cdot X^\top X + P \cdot \left(\rho \cdot C^{(k)} - \Gamma^{(k)}\right) \tag{13}$$

Note that both P and $P \cdot X^\top X$ can be pre-computed. Also note that both look like the solution of a ridge-regression problem (with the difference that the diagonal is additionally loaded with ρ prior to inversion). Finally, the vector γ of Lagrangian multipliers is determined by the constraint of the zero diagonal, so that it follows from Eq. 11:

$$\mathrm{diag}\left(B^{(k+1)}\right) = \mathrm{diag}\left(\tilde{B}^{(k+1)}\right) - \mathrm{diag}(P) \odot \gamma = 0$$

where $\odot$ denotes the elementwise product of the vectors. This yields the closed-form solution

$$\gamma = \mathrm{diag}\left(\tilde{B}^{(k+1)}\right) \oslash \mathrm{diag}(P) \tag{14}$$

where $\oslash$ denotes the elementwise division of the two vectors on the diagonal of the matrices $\tilde{B}^{(k+1)}$ and P.

3.2 Update of C

We observe from Eq. 6 that the update of matrix C is only subject to the L1-norm regularization and the non-negativity constraint. Hence, it follows from Eq. 8 that

$$C^{(k+1)} = \underset{C}{\mathrm{argmin}}\left(\lambda_1\|C\|_1 + \langle\Gamma^{(k)}, B^{(k+1)} - C\rangle_F + \frac{\rho}{2}\|B^{(k+1)} - C\|_F^2\right) \tag{15}$$

As outlined in [3], the solution for $C^{(k+1)}$ at iteration $k+1$ is given by

$$C^{(k+1)} = \left(s_{\lambda_1/\rho}\left(B^{(k+1)} + \frac{1}{\rho}\cdot\Gamma^{(k)}\right)\right)_+ \tag{16}$$

where $s_\kappa(a) = (a-\kappa)_+ - (-a-\kappa)_+$ is the soft-thresholding operator (e.g., see [3]) applied elementwise to matrix $B^{(k+1)} + \frac{1}{\rho}\cdot\Gamma^{(k)}$, while the function $(\cdot)_+$ preserves all positive values and sets all negative values to zero.

3.3 Update of Γ

Finally, the update of the dual variable $\Gamma^{(k+1)}$ at iteration $k+1$ is given by Eq. 9 above. Note that the penalty parameter ρ in the augmented Lagrangian in Eq. 6 plays the role of the step-size in Eq. 9, where the matrix of Lagrangian multipliers Γ is updated. Hence, the value chosen for the hyper-parameter ρ has a crucial effect on the speed of convergence of ADMM in practice. This is discussed in more detail in Section 6.

ALGORITHM 1: ADMM Training in Python 2.7

```
Input: User-item interaction-matrix X
       Regularization parameters lambda1, lambda2 > 0
       ADMM penalty parameter rho > 0
       Number of ADMM iterations K
Output: Item-item weight-matrix C
# pre-compute
XtX = X.T.dot(X)
diag_indices = numpy.diag_indices(XtX.shape[0])
XtX[diag_indices] += lambda2 + rho
P = numpy.linalg.inv(XtX)
XtX[diag_indices] -= lambda2 + rho
B_aux = P.dot(XtX)
# initialize
Gamma = numpy.zeros(XtX.shape, dtype=float)
C = numpy.zeros(XtX.shape, dtype=float)
# iterate until convergence
for _ in range(K):
    B_tilde = B_aux + P.dot(rho * C - Gamma)
    gamma = numpy.diag(B_tilde) / numpy.diag(P)
    B = B_tilde - P * gamma
    C = softthreshold(B + Gamma/rho, lambda1/rho)
    C = numpy.maximum(C, 0.)
    Gamma += rho * (B - C)
return C
```

4 ALGORITHM

The Python 2.7 code of ADMM for collaborative filtering is presented in Algorithm 1, where the function 'softthreshold' may be implemented as outlined in Section 3.2. Note that the result is given by matrix C (and not B), as it obeys all the constraints.

5 MODEL VARIANTS

In this section we discuss several variants of Slim: two simplified and two improved ones.

5.1 Removing the Non-Negativity Constraint

When the non-negativity constraint regarding the model parameters B is dropped from Slim, the model can learn both similarities *and* dissimilarities between pairs of items. In this case, the optimization problem becomes

$$\min_B \ \frac{1}{2}\|X - XB\|_F^2 + \frac{\lambda_2}{2}\|B\|_F^2 + \lambda_1\|B\|_1 \quad \text{s.t. } \mathrm{diag}(B) = 0 \tag{17}$$

Even in this simpler formulation, the presence of the non-smooth L1 penalty justifies the usage of ADMM. The ADMM formulation is identical to the one outlined above, except for the function $g(C)$ in Eq. 5 that now has an unrestricted domain. The update of $C^{(k+1)}$ at step $k+1$ is now given by (instead of Eq. 16):

$$C^{(k+1)} = s_{\lambda_1/\rho}\left(B^{(k+1)} + \frac{1}{\rho}\cdot\Gamma^{(k)}\right) \tag{18}$$

5.2 Removing the L1 Penalty

When the L1 penalty is dropped, but the non-negativity constraint is kept, ADMM can be used as outlined above, but with a modified update of $C^{(k+1)}$ at step $k+1$ (instead of Eq. 16):

$$C^{(k+1)} = \left(B^{(k+1)}\right)_+ \tag{19}$$

so that only the positive values in $B^{(k+1)}$ are retained in $C^{(k+1)}$.

5.3 Removing the Non-Negativity Constraint and the L1 Penalty

A dense weight matrix B is obtained when both the non-negativity constraint and the L1-norm regularization are dropped from the original Slim objective in Eq. 2. In this case, the resulting objective is a ridge-regression problem with the constraint that the diagonal of B is zero:

$$\min_B \quad \frac{1}{2}\,||X - XB||_F^2 + \frac{\lambda_2}{2}\,||B||_F^2 \quad \text{s.t.} \quad \text{diag}(B) = 0 \tag{20}$$

This can be solved in closed form, using the method of Lagrangian multipliers, analogous to the derivation of the update of B in Section 3.1. Here, it is much simpler, however: again, we use the term $\gamma^\top \cdot \text{diag}(B)$ in the Lagrangian. Compared to Section 3.1, we now have the simplified case where $C^k = \Gamma^k = 0$ and $\rho = 0$. Hence, Eqs. 12 and 13 immediately simplify and we obtain

$$P = \left(X^\top X + \lambda_2 \cdot I\right)^{-1}$$

as well as $\tilde{B}^{(k+1)} = P \cdot X^\top X$. It follows from Eq. 11 that the closed-form solution $B^{(\text{dense})}$ with a zero diagonal is [2]

$$B^{(\text{dense})} = P \cdot X^\top X - P \cdot \text{diagMat}(\gamma) = I - P \cdot \text{diagMat}(1 \oslash \text{diag}(P)) \tag{21}$$

Computing the dense solution $B^{(\text{dense})}$ is hence considerably simplified, as no iterations are needed, unlike ADMM.

5.4 Adding Item-Bias Terms

The weight matrix B is the only model parameter in the original Slim-approach [15]. Besides these pairwise parameters in B, one may also use so-called (unary) item-bias terms b_i for each $i \in \mathcal{I}$ (also called the intercepts of the items in regression). Instead of learning them as part of the model training, it is easier to estimate their values in a pre-processing step, where $b_i = \mu_i = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}} X_{u,i}$ is the mean of column $i \in \mathcal{I}$ of the user-item matrix X. The matrix B is then learned (as outlined above) by using the centered matrix $X - \vec{1}\vec{b}^\top$ in place of the original matrix X, where $\vec{1}$ is a vector of ones in the outer product with the vector of item-biases $\vec{b} = (b_1, ..., b_{|\mathcal{I}|})^\top$. In other words, the mean is subtracted from each column, resulting in a matrix with zero mean in each column. Having determined B and $\vec{b}$ from the training data, the resulting prediction rule for the score regarding user u and item j now reads (instead of Eq. 1) $S_{u,j} = (X_{u,\cdot} - \vec{b}^\top) \cdot B_{\cdot,j} + b_j$.

5.5 Item-Specific Regularization

In the original Slim-approach [15], only two (scalar) regularization parameters λ_1, λ_2 are used, i.e., either one takes the same value across all items. In this section, we replace them by item-specific regularization parameters: given that the item-popularities typically follow a power-law distribution in good approximation (e.g., see [19]), we parameterize the item-specific regularization parameters in terms of these item-popularities. If X is a binary user-item interaction matrix, then the popularity of item i is proportional to the mean μ_i in column i of X. We now define the L1 and L2 penalties as follows:

$$\lambda_1 \cdot ||\text{diagMat}(\vec{\mu})^{\alpha_1/2} \cdot B||_1 = \lambda_1 \cdot \sum_{i,j\in\mathcal{I}} \mu_i^{\alpha_1/2} \cdot |B_{i,j}|$$
$$\lambda_2 \cdot ||\text{diagMat}(\vec{\mu})^{\alpha_2/2} \cdot B||_F^2 = \lambda_2 \cdot \sum_{i,j\in\mathcal{I}} \mu_i^{\alpha_2} \cdot B_{i,j}^2 \tag{22}$$

where we introduced the exponents $\alpha_1, \alpha_2 > 0$ as to control the power-law in the regularization terms. The optimal values of these exponents are determined via grid-search, alongside the values of $\lambda_1, \lambda_2 > 0$.

When *both* L1 and L2 regularization are applied, then the learned parameters $B_{i,j}$ are shrunk towards zero by both penalties. In [24], it was motivated that un-doing the shrinkage due to the L2-norm regularization improves prediction accuracy: $B^{(\text{rescaled})} = \text{diagMat}(1 + \frac{\vec{\lambda}^{(2)}}{|\mathcal{U}|}) \cdot B$, where $\vec{\lambda}^{(2)} = \lambda_2 \cdot (\mu_1^{\alpha_2}, ..., \mu_{\mathcal{I}}^{\alpha_2})^\top$ denotes the vector of L2-norm regularization parameters (see above).[3] Given that we are eventually interested in only the *ranking* of the items, this correction factor only has an effect if $\vec{\lambda}^{(2)}$ has different values for different items (as motivated in this section). If all its values are identical, like in Slim [15], this correction factor has no effect on the ranking and hence may be ignored.

6 CONVERGENCE

In this section, we adapt the stopping criteria presented in [3] for the purpose of the optimization problem presented in Section 3: the necessary and sufficient optimality conditions for ADMM is primal and dual feasibility.

The *primal* and *dual residuals* at iteration $k+1$ of ADMM are [3]

$$R_{\text{primal}}^{(k+1)} = B^{(k+1)} - C^{(k+1)}$$
$$R_{\text{dual}}^{(k+1)} = -\rho \cdot \left(C^{(k+1)} - C^{(k)}\right)$$

Hence, a reasonable termination criterion for the ADMM iterations is that the primal and dual residuals are small,

$$||R_{\text{primal}}^{(k+1)}||_F \le \epsilon_{\text{primal}} \quad \text{and} \quad ||R_{\text{dual}}^{(k+1)}||_F \le \epsilon_{\text{dual}} \tag{23}$$

where $\epsilon_{\text{primal}}, \epsilon_{\text{dual}} > 0$ are the tolerances for primal and dual feasibility, respectively. Following [3], one can choose an absolute tolerance, ϵ_{abs}, which depends on the scale of the values of matrix B, as well as a relative tolerance, ϵ_{rel}, which is typically chosen to be 10^{-3} or 10^{-4}, see [3]. Then the primal and dual feasibility tolerances are given by the formulae

$$\epsilon_{\text{primal}} = |\mathcal{I}| \cdot \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \cdot \max\left\{||B^{(k+1)}||_F, ||C^{(k+1)}||_F\right\}$$
$$\epsilon_{\text{dual}} = |\mathcal{I}| \cdot \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \cdot ||\Gamma^{(k+1)}||_F$$

As discussed in [3] and verified in our experiments in Section 7, ADMM reaches an accuracy level that is reasonable for most large-scale practical applications within a few tens of iterations. For

[2]Substituting $X^\top X = P^{-1} - \lambda_2 \cdot I$ into $P \cdot X^\top X = P \cdot (P^{-1} - \lambda_2 \cdot I) = I - P \cdot \lambda_2 \cdot I$, and then from the zero-diagonal constraint it follows $\gamma + \lambda_2 = 1 \oslash \text{diag}(P)$.

[3]Note that dividing by $|\mathcal{U}|$ in the correction factor is necessary here, unlike in [24], as we did not divide the square loss by $|\mathcal{U}|$.

this reason, we ran 50 iterations of ADMM in our experiments in Section 7. However, if one wishes for higher accuracy solutions, the aforementioned stopping criteria can be used.

In practice, the choice of penalty parameter ρ is also crucial for convergence speed. In Section 7, we swept a set of values for ρ as part of the grid-search. Convergence may be sped up in practical applications by adapting the value of ρ along the ADMM iterations, which also makes it less dependent on the initial choice of ρ. As outlined in [3], ρ may be increased when the primal residuals are much larger than the dual residuals (i.e., if $\|R_{\text{primal}}^{(k+1)}\|_F > t \cdot \|R_{\text{dual}}^{(k+1)}\|_F$) as follows: $\rho^{(k+1)} = \tau \cdot \rho^{(k)}$. Typical choices are $\tau = 2$ and $t = 10$ [3]. Conversely, the value of ρ may be decreased when the dual residuals are much larger than the primal residuals (i.e., if $\|R_{\text{dual}}^{(k+1)}\|_F > t \cdot \|R_{\text{primal}}^{(k+1)}\|_F$) according to $\rho^{(k+1)} = \rho^{(k)}/\tau$, and keeping its value unchanged when the primal and dual residuals are within a factor of t.

7 EXPERIMENTS

In the experimental set-up, we follow the setting in [14], as the authors provided publicly available code[4] for reproducibility of the results. We first summarize the three data-sets, the evaluation protocol as well as the baseline approaches used in [14] (for details, see their paper). Then, regarding the model-variants proposed in this paper, we introduce the naming used throughout this section, including the tables. We discuss the results regarding the full-size training data in Section 7.1, and regarding the sub-sampled training data in Section 7.2.

Three data-sets were used in the experiments in [14], and were pre-processed and filtered for items and users with a certain activity level, resulting in the following data-set sizes, see [14] and their publicly available code for details:

- MovieLens 20 Million (***ML-20M***) data [10]: 136,677 users and 20,108 movies with about 10 million interactions,
- Netflix Prize (***Netflix***) data [1]: 463,435 users and 17,769 movies with about 57 million interactions,
- Million Song Data (***MSD***) [2]: 571,355 users and 41,140 songs with about 34 million interactions.

We also follow the evaluation protocol used in [14], which is based on *strong generalization*, i.e., we create train-user, validation-user and test-user sets, $\mathcal{U}^{\text{train}}$, $\mathcal{U}^{\text{valid}}$, $\mathcal{U}^{\text{test}}$, which are disjoint (while the set $\mathcal{I}$ of items is the same in test, validation and training). This is in contrast to *weak generalization*, where the training and test sets are disjoint in terms of user-item interaction-pairs, but not in terms of users. The test data $X^{\text{test}} \in \mathbb{R}^{|\mathcal{U}^{\text{test}}| \times |\mathcal{I}|}$ are further split into two disjoint sets of user-item interactions: one serves as the input to the recommender system, based on which the other one has to be predicted, see [14] for details.

In [14], results for the following models were reported, which we now use as baselines:

- Sparse Linear Method (**Slim**) [15]: Besides the original model, a computationally faster approximation that drops the constraints on the weights [13] was also considered. However, the results of this approximation were not found to be on par with the other models in the experiments in [14].
- Weighted Matrix Factorization (**wmf**) [12, 16]: A linear model with a latent representation of users and items.
- Collaborative Denoising Autoencoder (**cdae**) [23]: A non-linear model with one hidden layer.
- Denoising Autoencoder (**Mult-dae**) and Variational Autoencoder (**Mult-vae** $^{\text{PR}}$) [14]: Both are trained using the multinomial likelihood, which was found to outperform the Gaussian and logistic likelihoods. Best results were obtained in [14] for the Mult-vae $^{\text{PR}}$ and Mult-dae models that were rather shallow 'deep models', namely with a 200-dimensional latent representation, as well as a 600-dimensional hidden layer in both the encoder and decoder. Both models are non-linear, and Mult-vae $^{\text{PR}}$ is also probabilistic.

[4]The code regarding *ML-20M* in [14] is publicly available at `https://github.com/dawenl/vae_cf`. Upon request, the authors kindly provided the code for the other two data-sets.

Table 1: NDCG@100 for various values of λ_1 and λ_2 on *Netflix* data for 'ADMM L1' ($\rho = 10,000$). Results within the standard error of 0.001 of best NDCG@100 are highlighted.

λ_1 \ λ_2	0	1	5	50	1,000	5,000
0	–	0.381	0.383	0.388	**0.393**	0.391
0.1	0.390	0.390	0.390	0.391	**0.393**	0.391
0.5	0.390	0.390	0.390	0.391	**0.393**	0.391
1	0.391	0.391	0.391	0.391	**0.394**	0.391
2	0.392	0.392	0.392	0.392	**0.394**	0.391
3	0.392	0.391	0.392	0.392	**0.394**	0.391
4	0.392	**0.393**	**0.393**	**0.393**	**0.394**	0.391
5	**0.393**	**0.393**	**0.393**	**0.393**	**0.394**	0.391
10	**0.393**	**0.393**	**0.393**	**0.393**	**0.393**	0.391
20	0.392	0.392	0.392	0.392	0.392	0.388
50	0.386	0.386	0.386	0.386	0.385	0.381

We do not use Neural Collaborative Filtering [11] or Bayesian Personalized Ranking (BPR) [17] as baselines, as their accuracies were found to be so low on these data-sets that no results were reported in [14], see also [5]. We compare these baselines to the following variants of the ADMM-approach outlined in Section 3:

- **ADMM $\geq$ 0 & L1**: ADMM with non-negative weights, L1-norm and L2-norm regularization – the original objective function in the Slim-paper [15], but optimized via ADMM (instead of the original optimizer).
- **ADMM** $\geq$ 0: ADMM with non-negative weights and only L2-norm regularization, i.e., without the L1 penalty. Sparsity is only induced due to the non-negativity constraint.
- **ADMM L1**: ADMM with unrestricted weights, L1-norm and L2-norm regularization – this yields a sparse solution, where the weights are allowed to be positive, zero or negative.

In our experiments, we ran ADMM for 50 iterations. We also ran ADMM to full convergence based on the stopping criteria outlined in Section 6. The ranking metrics were almost identical as with stopping after 50 iterations in our experiments, which aligns with the remarks in [3] and in Section 6 that ADMM reaches sufficient accuracy for most practical applications after a few tens of iterations. The values of the ADMM penalty term ρ and the L1/L2-norm

Table 2: Ranking accuracy (with standard errors ≈ 0.002, 0.001, and 0.001 on *ML-20M*, *Netflix*, and *MSD* data).

	(a) *ML-20M*			(b) *Netflix*			(c) *MSD*		
model-variants	Recall @20	Recall @50	NDCG @100	Recall @20	Recall @50	NDCG @100	Recall @20	Recall @50	NDCG @100
ADMM ≥ 0	0.376	0.502	0.407	0.352	0.435	0.384	0.330	0.425	0.388
ADMM L1	0.391	0.521	0.420	0.362	0.445	0.394	**0.334**	0.428	0.390
ADMM ≥ 0 & L1	0.376	0.502	0.407	0.352	0.435	0.384	0.330	0.425	0.388
Dense	0.391	0.521	0.420	0.362	0.445	0.393	**0.333**	0.428	0.389
Dense ≥ 0	0.373	0.499	0.402	0.345	0.429	0.377	0.324	0.418	0.379
Sparse Approx.	0.391	0.521	0.420	0.361	0.445	0.393	**0.333**	0.427	0.389
Sparse Approx. ≥ 0	0.373	0.499	0.402	0.344	0.428	0.376	0.324	0.417	0.377
Centered	0.391	0.521	0.421	0.363	0.446	0.394	**0.333**	0.428	0.389
Centered & item-L2	0.391	0.522	0.422	**0.365**	**0.448**	**0.397**	**0.334**	**0.429**	**0.391**
Centered & item-L2 & L1	0.391	0.522	0.422	**0.365**	**0.449**	**0.398**	**0.334**	**0.430**	**0.392**
	Results reproduced from [14]:								
SLIM	0.370	0.495	0.401	0.347	0.428	0.379	— did not finish in [14] —		
WMF	0.360	0.498	0.386	0.316	0.404	0.351	0.211	0.312	0.257
CDAE	0.391	0.523	0.418	0.343	0.428	0.376	0.188	0.283	0.237
MULT-VAE $^{\text{PR}}$	**0.395**	**0.537**	**0.426**	0.351	0.444	0.386	0.266	0.364	0.316
MULT-DAE	0.387	0.524	0.419	0.344	0.438	0.380	0.266	0.363	0.313

regularization-parameters λ_1 and λ_2 were chosen w.r.t. NDCG@100 via grid-search. An interesting observation was that NDCG@100 changed very slowly with these hyper-parameters, as illustrated in Table 1 for the model 'ADMM L1' (with unrestricted weights). For this reason, it was essential to search over several orders of magnitude, while a rather coarse granularity of our grid-search was sufficient for obtaining close-to-optimal results. When training the model with non-negative weights (ADMM ≥ 0 & L1), however, the relevant range in grid-search was considerably reduced, as was also used in [15].

In addition, we evaluated the dense weight-matrix, which we obtained via the closed-form solution of the simplified training objective (see Section 5.3). As a computationally cheap alternative to ADMM, we also tried the simple heuristic of thresholding the dense solution as to obtain non-negative or sparse weight-matrices:

- **Dense**: the closed-form solution $B^{(\text{dense})}$ in Eq. 21, where both the L1 penalty and the non-negativity constraint are dropped from the training objective (see Eq. 20).
- **Dense** $\geq$ 0: based on the dense solution, the suboptimal non-negative solution $(B^{(\text{dense})})_+$, where the function $(\cdot)_+$ sets all negative elements to zero.
- **Sparse Approx.**: based on the dense solution, the suboptimal sparse solution obtained by setting all entries of $B^{(\text{dense})}$ to zero where $|B^{(\text{dense})}_{u,i}| \leq \theta$; as to allow for a fair comparison, θ is chosen so that the resulting matrix has the same level of sparsity as the 'ADMM L1' solution.
- **Sparse Approx.** ≥ 0 : based on the dense solution, the suboptimal sparse solution obtained by setting all entries of $B^{(\text{dense})}$ to zero where $B^{(\text{dense})}_{u,i} \leq \theta$; θ is chosen so that the resulting matrix has the same level of sparsity as the 'ADMM ≥ 0 & L1' solution.

When learning the dense solution, $B^{(\text{dense})}$, we found the optimal L2-norm regularization parameter λ_2 to be about 500 on *ML-20M*, 1,000 on *Netflix*, and 200 on *MSD* data. Note that these values are much larger than the typical values used for SLIM, which often are of the order of 1 [15].

Besides these simplifications of the SLIM approach, we also evaluated the model-variants outlined in Sections 5.4 and 5.5:

- **Centered**: the dense weight-matrix $B^{(\text{dense})}$ in Section 5.3 is trained on centered data (see Section 5.4), i.e., additional item-bias terms are used.
- **Centered & item-L2**: the dense weight-matrix $B^{(\text{dense})}$ in Section 5.3 is trained on centered data (see Section 5.4) using item-specific L2-penalties (see Section 5.5).
- **Centered & item-L2 & L1**: the weight-matrix is trained on centered data (see Section 5.4) using item-specific L2 and also L1 penalties (see Section 5.5).

Note that the first two variants can be obtained via the closed-form solution in Eq. 21, while the third variant is obtained via ADMM.

7.1 Discussion: Full-size Data-Sets

Table 2 summarizes the evaluation of the various approaches with respect to the ranking metrics Recall@20, Recall@50 as well as NDCG@100 on the three data-sets *ML-20M*, *Netflix* and *MSD*, following the experimental set-up in [14]. In the following, we discuss several interesting insights.

ADMM vs. best competing baseline in [14]. In Table 2, we observe that the best model-variant trained via ADMM (Centered & item-L2 & L1) is slightly worse (-1%) on *ML-20M*, slightly better (+3%) on *Netflix* and considerably better (+25%) on *MSD* data in terms of NDCG@100 compared to the best competing model, which is MULT-VAE $^{\text{PR}}$.

ADMM vs. SLIM. Even though 'ADMM $\geq$ 0 & L1' optimizes exactly the same training objective as the original SLIM-code [15] does (see

Eq. 2, i.e., including the non-negativity constraint and the L1-norm regularization), Table 2 shows that it obtains significantly better ranking accuracy. This suggests that ADMM is able to get closer to the optimum of the SLIM-objective than the optimizer in the original code does. Moreover, it is also computationally more efficient: the 50 iterations of ADMM took about 40 minutes, 30 minutes and 5 hours on the *ML20M*, *Netflix* and *MSD* data-sets, respectively, running on an AWS instance with 64 GB RAM and 16 vCPUs (same for the other ADMM approaches). This is in contrast to the original SLIM-code [15], where [14] reported that parallelized grid search for SLIM took approximately two weeks on the *Netflix* data and that the MSD data 'was too large for SLIM to finish in a reasonable amount of time' in their experiments [14].

Non-negativity Constraint on Weights. The constraint regarding non-negative weights in SLIM [15] tends to hurt performance on the full-size data in Table 2: the variant 'ADMM $\geq$ 0 & L1', which optimizes the original SLIM-objective, obtains significantly lower accuracy than the variant 'ADMM L1', where the non-negativity constraint is dropped. The same finding also holds when comparing the variants 'ADMM $\geq$ 0' and 'Dense', which also differ only in the non-negativity constraint; also 'Dense' vs. 'Dense $\geq$ 0' as well as 'Sparse Approx.' vs 'Sparse Approx. $\geq$ 0' show the same behavior. Intuitively, this means that increased recommendation accuracy can be achieved by removing the non-negativity constraint and allowing the model-parameters to learn both similarities and dissimilarities among the items. Note, however, that we found the non-negativity constraint to possibly improve recommendation accuracy when training on extremely small data-sets (see Section 7.2).

Sparsity via L1-Penalty. Although slightly better in some cases, remarkably we found that the sparse model ('ADMM L1') did not perform significantly better than the dense solution ('Dense') on the full-size data-sets in Table 2. Note that the dense solution can be obtained via the closed-form solution in Eq. 21, which took less than 2, 2 and 20 minutes on the *ML20M*, *Netflix* and *MSD* data-sets, respectively. This is about 15 times faster than running 50 iterations of ADMM (see above). Note that the training-time may be reduced even further by exploiting model-sparsity as to reduce the number of parameters that have to be estimated, as outlined in [22]. Once learned, a sparse solution may have several advantages such as smaller memory footprint and decreased recommendation time [15]. Even then, obtaining an approximate sparse solution by thresholding the dense solution ('Sparse Approx.') also performs about equally well as obtaining the optimal sparse solution via ADMM ('ADMM L1') in Table 2. On the other hand, obtaining an approximate non-negative solution by setting the negative weights of the dense solution to zero ('Dense $\geq$ 0') performs worse than obtaining the optimal non-negative solution via 'ADMM $\geq$ 0'. Note, however, that these findings were different when we trained on extremely small data-sets (see Section 7.2).

Centered Data and Item-specific Regularization. Table 2 shows that adding item-bias terms to the model did not significantly improve recommendation accuracy ('Centered' vs. 'Dense'). In the grid-search regarding the optimal $\alpha_1, \alpha_2 \in \{0, 1/4, 1/2, 3/4, 1\}$, we found that exponent $\alpha_1 = 0$, i.e., a constant L1 penalty across items, was optimal, while the optimal L2 penalty grew with the item's popularity ($\alpha_2 = 1/4$ on *ML-20M*, and $\alpha_2 = 3/4$ on *Netflix* and *MSD*). Table 2 shows that item-specific L2 penalties ('Centered & item-L2' vs. 'Centered') may result in small but significant improvements in recommendation accuracy. Interestingly, these improvements were larger than the ones resulting from L1-norm regularization in our experiments. As an aside, note that further variants are outlined in [20].

7.2 Discussion: Subsampled Data-Sets

The surprising findings on the full-sized training-sets in the previous section were that (1) the non-negativity constraint tended to decrease recommendation accuracy, and (2) the sparsity promoted by the L1-norm regularization did not significantly improve it. This motivated us to examine these two terms in the original SLIM-objective further by training the models on *smaller* data-sets (the test-set remained unchanged). To this end, we randomly down-sampled the training data (without replacement), as to obtain smaller data-sets of various sizes, all the way down to only 1,000 users. For each subsample-size, we randomly generated 5 samples, trained the model on each, and then averaged the obtained ranking metrics. All results in Table 3 are based on the variant 'ADMM L1', which corresponds to the original SLIM-objective, where we dropped the non-negativity constraint as to focus on the effect of L1-norm regularization. For sample-size 1,000, Table 3 also shows the results obtained by optimizing the original SLIM-objective, including the non-negativity constraint, using the variant 'ADMM $\geq$ 0 & L1'. Regarding the full-size training-data, the results of 'ADMM $\geq$ 0 & L1' are shown in Table 2.

Moreover, Table 3 also shows the improvement of the sparse solution due to the L1 penalty, obtained by 'ADMM L1' (or 'ADMM $\geq$ 0 & L1'), over the dense solution as well as over the heuristic of thresholding the dense solution as to match the density of the ADMM-based solution.

Our subsampling experiments may simulate two real-world scenarios. First, subsampling simulates the case of a small number of training observations compared to a very large catalog size, which may occur in applications such as e-commerce, especially for the long-tail items. Second, subsampling may also simulate the case of item cold-starting, in which there are very few user-item interactions for certain or all items.

Comparing the three data-sets in Table 3, it is apparent that the *ML-20M* data has unexpected properties, which are also different from the (expected) behavior of the *Netflix* and *MSD* data: the sparse solution does not improve over the dense solution as the sample size diminishes, and the density as well as the fraction of positive weights stays about constant across the different sample sizes. The *Netflix* and *MSD* data agree with each other, and we base the following discussion on these two data-sets.

L1-norm regularized solution via ADMM vs. dense solution for small sample-sizes. Although the variants 'ADMM L1' and 'Dense' are on par for the full-size training-data in Table 2, the advantage of 'ADMM L1' over 'Dense' increases toward smaller sample-sizes (i.e., $\Delta_{\text{drop}}^{(\text{dense})}$ increases) in Table 3 on the *Netflix* and *MSD* data-sets. In this table, the number of non-zero model parameters tend to decrease with shrinking sample size on the *Netflix* and *MSD* data-sets, which shows that the L1-penalty adapts the

Table 3: Results on sub-sampled training-data for 'ADMM L1' (also 'ADMM $\geq$ 0 & L1' shown for sample size 1,000), averaged over 5 samples: besides the absolute values of the ranking metrics, also the difference to the dense solution ($\Delta^{(\text{dense})}_{\text{drop}}$: 'ADMM L1' - 'Dense') and to the thresholded dense solution that has the same level of sparsity as the ADMM-solution ($\Delta^{(\text{sparse})}_{\text{drop}}$: 'ADMM L1' - 'Sparse Approx.') are shown. We highlighted differences that are larger than the standard errors of about 0.002, 0.001, and 0.001 on the *ML-20M, Netflix,* and *MSD* data, respectively. Also the density of the learned weight matrix, and the fraction of positive weights among the non-zero weights are shown. The last columns show the best hyper-parameters determined via grid-search for each sample-size.

	Recall@20			Recall@50			NDCG@100							
Sub-sample size	ADMM L1	$\Delta^{(\text{Dense})}_{\text{drop}}$	$\Delta^{(\text{Sparse})}_{\text{drop}}$	ADMM L1	$\Delta^{(\text{Dense})}_{\text{drop}}$	$\Delta^{(\text{Sparse})}_{\text{drop}}$	ADMM L1	$\Delta^{(\text{Dense})}_{\text{drop}}$	$\Delta^{(\text{Sparse})}_{\text{drop}}$	Density	Positives	ρ	λ_2	λ_1
(a) *ML-20M*														
full-size	0.391	0.000	0.000	0.521	0.000	0.000	0.420	0.000	0.000	7.48%	54 %	10,000	500	1
10,000	0.366	0.000	0.000	0.490	0.000	0.000	0.396	0.000	0.000	4.75%	65 %	10,000	500	0.5
1,000	0.324	**0.003**	**0.003**	0.437	**0.003**	0.002	0.352	**0.003**	**0.003**	1.03%	74%	10,000	2	0.5
trained with non-negative weights (i.e., model variant 'ADMM $\geq$ 0 & L1'), and $\Delta^{(\text{sparse})}_{\text{drop}}$ w.r.t. model 'Sparse Approx. $\geq$ 0':														
1,000	0.321	0.000	**0.010**	0.433	-0.001	**0.011**	0.350	0.001	**0.013**	0.65%	100%	1,000	100	0.2
(b) *Netflix*														
full-size	0.362	0.000	0.000	0.445	0.000	0.001	0.394	0.000	0.001	7.83%	53 %	10,000	1,000	3
10,000	0.334	**0.004**	**0.006**	0.412	**0.002**	**0.005**	0.366	**0.004**	**0.006**	0.35%	64 %	10,000	50	5
1,000	0.293	**0.009**	**0.011**	0.365	**0.008**	**0.011**	0.324	**0.010**	**0.012**	0.26%	74%	10,000	2	2
trained with non-negative weights (i.e., model variant 'ADMM $\geq$ 0 & L1'), and $\Delta^{(\text{sparse})}_{\text{drop}}$ w.r.t. model 'Sparse Approx. $\geq$ 0':														
1,000	0.295	**0.011**	**0.016**	0.367	**0.010**	**0.017**	0.327	**0.013**	**0.021**	0.80%	100%	1,000	100	0.5
(c) *MSD*														
full-size	0.333	0.000	0.001	0.428	0.000	0.001	0.390	0.000	0.001	5.05%	57 %	1,000	50	1
10,000	0.272	**0.022**	**0.025**	0.354	**0.025**	**0.027**	0.321	**0.025**	**0.026**	0.16%	78 %	1,000	0.5	1
1,000	0.147	**0.013**	**0.016**	0.197	**0.014**	**0.016**	0.182	**0.015**	**0.017**	0.22%	98%	1,000	0.2	0.5
trained with non-negative weights (i.e., model variant 'ADMM $\geq$ 0 & L1'), and $\Delta^{(\text{sparse})}_{\text{drop}}$ w.r.t. model 'Sparse Approx. $\geq$ 0':														
1,000	0.148	**0.015**	**0.021**	0.199	**0.015**	**0.022**	0.184	**0.017**	**0.023**	0.22%	100%	1,000	0	0.5

model-complexity to the sample-size in the expected way, resulting in the observed improvements over the dense model on small sample-sizes. The improvement due to the L1 penalty is most striking on the *MSD* data, on which 'ADMM L1' has about 8% higher NDCG@100 than 'Dense' on a training sample with 1,000 users. In comparison, 'ADMM L1' has about 3% higher NDCG@100 than 'Dense' for a training sample with 1,000 users in the *Netflix* data-set. This is to be expected, as L1 regularization is helpful when the ratio of training observations (i.e., users in this case) to the number of parameters (i.e., items in this case) is small, and the *MSD* data-set has more than twice the items of the *Netflix* data-set.

L1-norm regularized solution via ADMM vs. sparse approximation from dense solution in small sample sizes. Although, the variants 'ADMM L1' and 'Sparse Approx.' are on par in Table 2, when using all the available training data for *ML-20M, Netflix* and *MSD*, on smaller sample sizes 'Sparse Approx.' does not perform as well ($\Delta^{(\text{sparse})}_{\text{drop}}$ grows) in Table 3. Specifically, 'ADMM L1' has about 10% higher NDCG@100 than 'Sparse Approx.' for a training sample with 1,000 users on *MSD* data, and 4% higher NDCG@100 than 'Sparse Approx.' when trained on 1,000 users of the *Netflix* data.

Non-negativity Constraint on Weight-Matrix. Table 3 shows that the fraction of positive weights among the non-zero weights increases as the sample size diminishes. Intuitively, this means that, as the sample size decreases, the solution focuses more and more on learning similarities rather than dissimilarities among the items. This agrees with the finding for the sample size of 1,000 users in Table 3, where the non-negativity constraint introduced by [15] tends to help improve the ranking accuracy when the training data are small. Hence, even though the non-negativity constraint introduced in [15] tends to hurt in large samples (see Table 2), it may become helpful in (extremely) small samples.

Optimal Regularization Parameters. As determined by grid search, Table 3 shows that the optimal L2-penalty diminishes considerably with decreasing sample-size, while the optimal L1-penalty stays approximately constant. This agrees with our results regarding the item-specific regularization terms, where we found that the optimal L2 penalty increases with item-popularity, while the optimal L1 penalty is (approximately) constant.

8 SUMMARY AND CONCLUSIONS

In this paper, we demonstrated in experiments on several well-known data-sets that ADMM [3, 8, 9] is a scalable and flexible approach for learning Slim-like models [15]. Besides its computational efficiency, we found empirically that ADMM is able to reach the optimum of Slim's convex-optimization problem more closely than the original implementation does, resulting in statistically significant increases in ranking accuracy. Compared to various competing models in our experiments, including probabilistic deep learning models, we observed up to 25% higher recommendation accuracy, especially on the largest of the three well-known data-sets used. The flexibility of ADMM was essential for switching on and off the various constraints and regularization-terms in the original Slim-objective, as to understand their individual contributions. The experimental results showed that the non-negativity constraint on the learned weight-matrix as well as the sparsity-promoting L1-norm regularization-term are able to improve recommendation accuracy in the regime where the number of users is much smaller than the number of items that are available for recommendation, like in the cold-start scenario or in the case of a large catalog size compared to a relatively small user-base. In the case of having a large number of users, however, our experiments indicate that the non-negativity constraint tends to hurt, while the sparsity-promoting L1-penalty may not significantly improve ranking-accuracy. If both of these terms are dropped from the original Slim-objective, ADMM is not necessary, and the provided closed-form solution can be used, which requires orders of magnitude less training time. Among the two improvements of the training-objective we proposed – which do not increase training time – the item-specific L2-norm regularization turned out to yield accuracy-gains that are statistically significant, except for the smallest publicly available data-set used in our experiments.

ACKNOWLEDGMENTS

We are very grateful to Dawen Liang for providing his code for the experimental setup of all three data-sets, and we also want to thank our Netflix colleagues who engaged in conversation and provided useful feedback.

REFERENCES

[1] J. Bennet and S. Lanning. 2007. The Netflix Prize. In *Workshop at SIGKDD-07, ACM Conference on Knowledge Discovery and Data Mining*.

[2] T. Bertin-Mahieux, D.P.W. Ellis, B. Whitman, and P. Lamere. 2011. The Million Song Dataset. In *International Society for Music Information Retrieval Conference (ISMIR)*.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. 2011. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Found. Trends Mach. Learn.* 3 (2011), 1–122.

[4] Y. Cheng, L. Yin, and Y. Yu. [n.d.]. LorSLIM: Low Rank Sparse Linear Methods for Top-N Recommendations. In *IEEE International Conference on Data Mining (ICDM)*. 90–9.

[5] M.F. Dacrema, P. Cremonesi, and D. Jannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *ACM Conference on Recommender Systems (RecSys)*.

[6] P. Danaher, P. Wang, and D.M. Witten. 2014. The Joint Graphical Lasso for Inverse Covariance Estimation across Multiple Classes. *Journal of the Royal Statistical Society, Series B* 76 (2014), 373–97.

[7] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. 2004. Least Angle Regression. *Annals of Statistics* 32 (2004). Issue 2.

[8] D. Gabay and B. Mercier. 1976. A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximations. *Computers and Mathematics with Applications* 2 (1976), 17–40.

[9] R. Glowinski and A. Marrocco. 1975. Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation–dualite, d'une classe de problems de Dirichlet non lineares. *Revue Francaise d'Automatique, Informatique, et Recherche Operationelle* 9 (1975), 41–76.

[10] F. M. Harper and J. A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5 (2015). Issue 4.

[11] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. 2017. Neural Collaborative Filtering. In *International World Wide Web Conference (WWW)*.

[12] Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *IEEE International Conference on Data Mining (ICDM)*.

[13] M. Levy and K. Jack. 2013. Efficient Top-N Recommendation by Linear Regression. In *RecSys Large Scale Recommender Systems Workshop*.

[14] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *International World Wide Web Conference (WWW)*.

[15] X. Ning and G. Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *IEEE International Conference on Data Mining (ICDM)*. 497–506.

[16] R. Pan, Y. Zhou, B. Cao, N. Liu, R. Lukose, M. Scholz, and Q. Yang. 2008. One-Class Collaborative Filtering. In *IEEE International Conference on Data Mining (ICDM)*.

[17] S. Rendle, Ch. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Conference on Uncertainty in Artificial Intelligence (UAI)*. 452–61.

[18] S. Sedhain, A. K. Menon, S. Sanner, and D. Braziunas. 2016. On the Effectiveness of Linear Models for One-Class Collaborative Filtering. In *AAAI Conference on Artificial Intelligence*.

[19] H. Steck. 2011. Item popularity and recommendation accuracy. In *ACM Conference on Recommender Systems (RecSys)*. 125–32.

[20] H. Steck. 2019. Collaborative Filtering via High-Dimensional Regression. arXiv:1904.13033.

[21] H. Steck. 2019. Embarrassingly Shallow Autoencoders for Sparse Data. In *International World Wide Web Conference (WWW)*.

[22] H. Steck. 2019. Markov Random Fields for Collaborative Filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[23] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester. 2016. Collaborative Denoising Auto-Encoders for top-N Recommender Systems. In *ACM Conference on Web Search and Data Mining (WSDM)*.

[24] H. Zou and T. Hastie. 2005. Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society, Series B* 67 (2005).