Anti-Sparse Hashing for Collaborative Metric Learning

Anonymous Author(s)

ABSTRACT

For real-life applications, scalability is one of crucial issues in recommender systems. Recently, hashing methods were proposed to circumvent the linear complexity of top-k item recommendation in Euclidean searches and enjoy the sublinear cost of Hamming searches. Currently, most of the existing literature on hashing applied to Collaborative Filtering (CF) focuses on Matrix Factorization (MF) hashing algorithms. The loss of MF is *pointwise* based: it aims at explicitly reconstructing all the user-item interaction matrix and ranks the items based on this estimation. However pointwise based recommender systems are known to be outperformed by *pairwise* based systems, which directly optimize on the users' preferences. Recently, the Collaborative Metric Learning (CML) algorithm, a large-margin pairwise CF system, showed promising results in comparison with both pointwise and pairwise existing CF systems. To the best of our knowledge, this paper is the first attempt to hash the CML algorithm. The hashing procedure consists into a relaxed CML algorithm regularized with a negative l_1 term in the loss, that we shall later refer to as anti-sparse, and constrained in an l_2 ball, then followed by a simple hash with the $sign(.)$ function. The goal of this anti-sparse procedure is both to push away the points from the axis and to remove the role of the norm in the ranking. In addition to be scalable and simple to implement, experiments on three real-world datasets showed that the anti-sparse CML significantly outperformed state-of-the-art CF hashing algorithms.

KEYWORDS

Collaborative Filtering, Metric Learning, Hashing, Regularization

ACM Reference Format:
Anonymous Author(s). 2020. Anti-Sparse Hashing for Collaborative Metric Learning. In *WWW '20, April 20 - 24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3306307.3328180

1 INTRODUCTION

With the expansion of online services such as video and music streaming or social networks, making personalized and accurate recommendations for customers has become an important industrial problematic. The ever growing mass of new users and items made the scalalibity a decisive feature of recommender systems.

Latent factors CF systems are a popular approach for large-scale applications [3, 7, 8, 22]: based on the users' previous interactions, they co-embed users and items in the same Euclidean space $\mathbb{R}^d$ such that the user-item similarities in $\mathbb{R}^d$ reflect the tastes of the users; the recommendation of k items to a user is then done by performing a k-nearest neighbors search. The first CF algorithms that were developed are *pointwise* based algorithms [7, 8] which aim at fully reconstructing the user-item interaction matrix and rank the items afterwards; the most known of them being the MF [7] algorithm. The discrepancy between the optimized objective and the true purpose of ranking items lead *pairwise* based recommender systems to emerge. Pairwise recommender systems directly optimize embeddings in a way that the preferences of the users are conserved in the latent space. Thanks to the removal of the intermediate step, pairwise systems were shown to outperform their pointwise counterparts [3, 15].

We may notice that the complexity of a recommendation for a CF model scales as $O(d \times |\mathcal{I}|)$ where $|\mathcal{I}|$ is the number of items and d is the embedding dimension. Even though this scaling is linear, it is not suited for very large scale and quick recommendations. For instance, the catalog of Amazon contains over 200 millions items; for an embedding dimension of 128, making a single recommendation would involve about 20 billions operations. Fortunately, exact nearest neighbors computation is much less expensive in the case where the search space is binary. Using multiple *inverse-lookup* tables, nearest neighbors search with sublinear time complexity with respect to number of items were implemented [13]. Other approaches were developed to tackle the prohibitive cost of Euclidean nearest neighbors search the most known being Vector Quantization (VQ) [4, 21] which combines a coarse clustering level and a fine grain search within the cluster. However, these methods are out of the scope of this study and we won't further investigate them.

A wide literature has been developed around *hashing* algorithms which map real-valued data into a binary space $\mathbb{B}^d$ and try to preserve the similarities from the Euclidean space into the Hamming space [1, 6, 12, 14]. However, enforcing both the preservation of the similarities and the binary mapping lead to a NP-hard discrete optimization problem [2] for which no generic algorithm exists. As a consequence, most of hashing techniques adopt a two-stages approach with a continuous relaxation of the problem followed by a simple hash rule, like applying a $sign(.)$ function for instance. Such methods are known to be prone to *quantization errors* because of the relaxation process and can destroy the initial geometry of the real-valued data. To limit this phenomenon, recent literature advocated to directly incorporate the hashing into the learning process rather than do a *post-train* quantization [9, 11, 18]; these approaches were shown to improve the performances as the hashing is tailored to a certain learning problem and can partially be reverberated on the loss.

Among the researches mentioned above, specific hashing frameworks were developed for CF problems [10, 20, 24]. Be it with relaxed schemes or direct discrete optimization, most of the existing literature focuses on hashing pointwise based recommendation algorithms such as MF [7]. Recently the well-known Bayesian Personalized Ranking (BPR) algorithm [15], a pairwise based recommender system, was discretized using a direct optimization procedure [22]. However BPR was shown to be consistently outperformed by CML

[3, 19] which relax the AUC objective with a hinge loss instead of a logistic loss and regularize with an l_2 ball constraint instead of an l_2 penalty in the case of BPR [22] .

In this paper, we provide a regularized version of the CML algorithm that is well suited for a later hard hash with the *sign*(.) function. Pursuing the idea of the Preference Preserving Hashing (PPH) [23] algorithm, we focus on the problem of *ranking inconsistency* induced by the hash rather than on the quantization errors themselves. We identify in this study two major sources of ranking inconsistency. The first one is the heterogeneity of the embeddings' norm that cannot translate into the hamming space. The second one is the closeness to the canonical axis which can tear apart data's neighborhoods. Therefore, we softly limit the ranking inconsistency by adding a $-l_1$ anti-sparse term in the CML loss that we combine with the l_2 ball constraint already present in the original optimization problem. The model is then trained using a joint projected ADAM [5] gradient descent.
Not only the proposed hashing scheme is easy to plug-in but it can also leverage the high computational efficiency of gradient-based methods. Moreover, we show on three real-world datasets that anti-sparse CML consistently outperforms numerous state-of-the-art CF hashing algorithms. The source code is available available on a GitHub repository [1] to reproduce our results.
This paper is organized as follows: in Section 2, we present the CML algorithm and the literature related to hashing for CF. Then in Section 3 we present our anti-sparse CML hashing algorithm. In Section 4 we give an experimental protocol to compare anti-sparse CML with several baselines and we analyze our results in Section 5. Finally, we give our conclusion and future research direction in Section 6.

2 BACKGROUND AND RELATED WORK

In this section we start by presenting the CML algorithm and give qualitative arguments in favor of its superiority over pointwise based CF systems for the task of recommendation. Then we present the different CF hashing techniques that were mostly applied to pointwise based systems so far. Among them, we chose to retain the approach of the PPH algorithm as it appears to be the most relevant for hashing the CML problem. In Section 3, we introduce an extension of this idea for CML anti-sparse hashing.

2.1 Collaborative Metric Learning

The CML is a large-margin pairwise CF system that aims at pulling closer matching user-item pairs and push away, in the squared l_2 sense, non-matching pairs. If we denote by $\mathcal{T}$ the ensemble of triplets (i, j, k) such that the user i positively (resp negatively) interacts with item j (resp k) and if we denote by u_i, v_j and v_k their respective embedding vectors, the problem is mathematically formulated as:

$$\min_{||u||_2 \leq 1, ||v||_2 \leq 1} \sum_{(i,j,k)\in\mathcal{T}} [r + ||u_i - v_j||_2^2 - ||u_i - v_k||_2^2]_+ \tag{1}$$

[1]https://github.com/WorldWideWebConf2020/Anti-sparse-CML

where r is a margin hyperparameter, $[x]_+ = \max(0, x)$ and u (resp. v) is the user (resp. items) matrix of embeddings. Figure 1 [2] shows visually how the CML behaves during the minimization process.

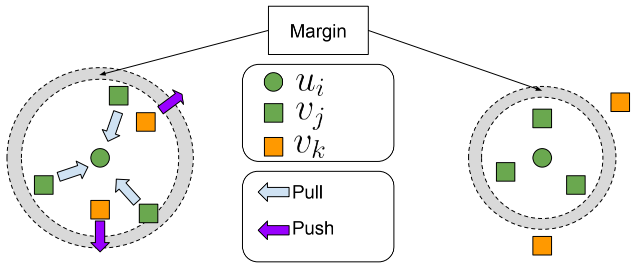

Figure 1: Collaborative Metric Learning. The gradients of the objective function pull closer the positive pairs and push the negative items away up to a certain margin

The reason why CML manages to outperform pointwise CF algorithms can be understood by analyzing the link of the objective function with the AUC criteria. If we denote by $\hat{x}_{ij}$ the estimated similarity between item i and user j and by $\delta(. > 0)$ the Heaviside function, we have up to weighting factors:

$$AUC \propto \sum_{(i,j,k)\in\mathcal{T}} \delta(\hat{x}_{ij} - \hat{x}_{ik} > 0) \tag{2}$$

which accounts for the number of accurately ranked triplets. One of the goals of recommender systems, rather than predicting $\hat{x}_{ij}$ with a high precision, as it is the case for pointwise algorithms, is to maximize the *AUC* quantity, *i.e.* is to be able to discriminate between items for each user. However, since the Heaviside function has a zero derivative almost everywhere, optimizing *AUC* cannot be done via gradient-based methods (actually, it was shown to be a NP-hard problem [22]) and the problem needs to be relaxed. On the one hand, if we replace the Heaviside by the r-margin hinge function $l(t) = max(0, r - t)$ and if we chose $\hat{x}_{ij} \propto -\|u_i - v_j\|^2$, we end-up with the CML objective. On the other hand, if the similarity measure is given by the dot product and if the Heaviside loss is relaxed to the logistic loss $l(t) = \log(1 + \exp(-t))$, we obtain the BPR objective.

As mentioned in the introduction, CML was shown to outperform BPR on numerous recommendation tasks [3, 19]. The most common explanation is that, thanks to the triangle inequality, the Euclidean distance can propagate the users' tastes while the dot product does not have such guarantees. However, we cannot base ourselves on this argument when it comes to choosing to hash CML instead of BPR. Indeed, hashed embeddings all share the same norm, therefore the dot product and the Euclidean distance coincide on the hashed space (up to a factor 2). The main reason why we chose to hash CML rather than BPR finds its origin in the regularization schemes adopted by these methods. The complete optimization problem that BPR solves is:

$$\min_{u,v} \sum_{(i,j,k)\in\mathcal{T}} -\log(1 + \exp(u_i v_k^\top - u_i v_j^\top)) + \lambda(\|u\|_F^2 + \|v\|_F^2) \tag{3}$$

[2]this figure was inspired from the paper of [3]

where λ is a regularization hyperparameter and $\|.\|_F$ is the Frobenius norm. While the optimization of CML is explicitly constrained on the l_2 ball, the BPR aglorithm optimizes on the associated Lagragian primal problem with equal Lagrangian multipliers and uses an l_2 penalty. As we will see in Section 3, the l_2 ball constraint is readily suited for our hashing procedure while the l_2 penalty, that tends to minimize the norms of the embeddings, might, on the contrary, harm it.

2.2 Hashing for Collaborative Filtering

In this subsection, we review the literature around hashing algorithms designed for CF problems. The common starting point of these algorithms is the problem:

$$\min_{u,v} \mathcal{L}(u,v) \;\text{ s.t. }\; u \in \{-1,1\}^{m\times d}, v \in \{-1,1\}^{n\times d} \tag{4}$$

where m (resp n) is the number of users (resp items) and d is the embedding dimension. In most of the existing literature [10, 20, 24], $\mathcal{L}$ is the loss associated with a MF problem:

$$\mathcal{L}(u,v) = \sum_{(i,j)\in\mathcal{S}} (r_{ij} - u_i v_j^\top)^2 \tag{5}$$

where $\mathcal{S}$ can either be the set of interacting user-item pairs (i,j) or the whole set of user-item pairs and r_{ij} is the rating of item j by the user i. The problem (4) being discrete NP-hard [2], with no straightforward resolution scheme, the first algorithms that were developed continuously relaxed the problem with an appropriate regularization. For instance, following the work of [6] who noticed that an isotropic projection of the data could reduce the quantization error after applying the $sign(.)$ function, [24] implemented an MF algorithm with a penalty for imbalanced embeddings of the form $\lambda\|\sum_i u_i\|^2$, which is the first order condition of isotropy. On their side, the authors of [10] developed an iterative local search algorithm in the spirit of [1]: a variable discrete representation of users and items is relaxed within the neighborhood of a proper fixed continuous representation to solve the MF problem. Then, an orthogonal transformation is applied to fit the continuous representation on the discrete one and those steps are repeated until convergence. This algorithm is known as Collaborative Hashing (CH).

Despite the efforts made, the methods above were criticized for their quantization erros [16, 20, 22], harming the overall recommendation process. A radical answer was first given in a generic hashing algorithm for supervised learning [16]. They noticed that in a discrete least squares regression problem, fixing all the bits but the k-th gave exact discrete closed form solutions. Therefore, they proposed a *discrete coordinate descent* algorithm, where they sequentially loop over the k-th bit to optimize until no flips are observed. This scheme was applied to MF [20] and to BPR [22], which was actually the first pairwise based CF algorithm to be hashed. By definition, this procedure does not suffer any quantization error but it comes at a certain price and it has two major drawbacks:

(1) the cyclic loops over the bits cannot be parallelized and the overall training cost scales quadratically with the dimension.
(2) the inherent suboptimality of coordinate-wise descents is largely amplified by the discrete nature of the problem and lead to relatively poor results as we will see in Section 5.

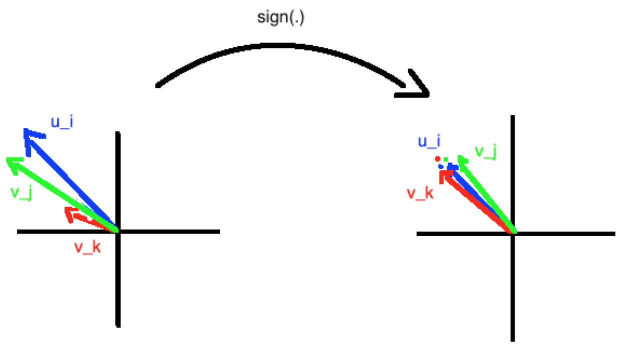

Figure 2: Ranking inconsistency due to the heterogeneity of the norms: user was i was closer to item j than item k in the real-valued space but the points all merge in the hashed space.

The authors of PPH argued that quantization errors should not be the main focus for recommendation tasks. What we should be concerned with is the ranking inconsistency induced by the hash *i.e.* when the embeddings are such that $u_i v_j^\top > u_i v_k^\top$ but $h(u_i)h(v_j)^\top > h(u_i)h(v_k)^\top$ with $h(.) = sign(.)$. Indeed, hashing with zero quantization can achieve a full rank consistency as the quantization error upperbounds pairwise dot products. However, the discrete problem associated with null quantization error is hard to solve as seen in the paragraph above. The authors noticed the ranking induced by the continuous dot products was highly dependent on the norms of the embeddings and that such a property could not translate into the hashed space as all the hashed embeddings share the same norm. Therefore, they decided to remove the norm dependency of the ranking by softly imposing a constant norm for all the embeddings, with the addition of a term $\lambda(\|u\|^2 - \alpha)^2$ in the MF loss. This is known as the Constant Features Norm (CFN) constraint.

Our contribution is an improvement over the work of PPH. First, we chose to work with CML instead of MF as it is better suited to recommendation. Second we add an anti-sparse term $-l_1$ in the loss of CML instead of the constant feature penalty proposed by the authors. Combined with the anti-sparse effect, the l_2 ball constraint implicitly pushes all the points to have norm 1. As we will observe in Section 5, the anti-sparse effect further improves the ranking consistency of the hash as it also pushes away the points from the axis. Finally, we provide an explicit optimization procedure that scales well and, as a gradient-based method, can be efficiently computed.

3 ANTI-SPARSE CML

In this section we give the motivations for anti-sparse hashing then we give a mathematical formulation for the anti-sparse CML and finally, we propose an optimization procedure for the associated problem.

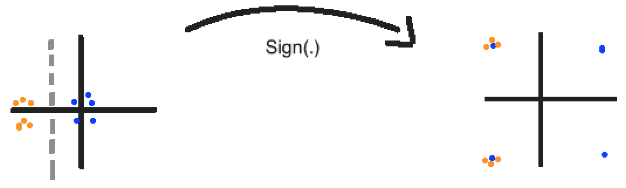

Figure 3: Sparsity bias, clusters on the axis are torn apart by the sign function

3.1 Motivations

As pointed out by [23] the key property that hashed recommender systems must satisfy is the ranking consistency. In the case of CML, since Hamming distance $d_{\mathcal{H}}(x, y) \propto -xy^\top$, that would be:

$$||u_i - v_j||^2 < ||u_i - v_k||^2 \Rightarrow h(u_i)h(v_j)^\top > h(u_i)h(v_k)^\top \quad (6)$$

The semantic structure induced by the Euclidean distance partially relies on the norm of the embeddings themselves. For instance, points with a high norm compared to the rest of the embeddings will be structurally far away from many points. As Figure 2 shows, the hashed embeddings cannot carry semantics relations encoded in the norms in the real-valued space. Indeed, all the hashed embeddings share the same norm $||h(x)||_2 = \sqrt{d}$.

Taking these considerations into account, the authors of PPH [23] imposed a CFN constraint in the embeddings. With such a constraint, the dot product, the Euclidean distance and the cosine distance $\cos(u, v) = \frac{uv^\top}{||u||_2||v||_2}$ coincide (up to an affine transformation) and the ranking consistency problem becomes:

$$\cos(u_i, v_j) > \cos(u_i, v_k) \Rightarrow h(u_i)h(v_j)^\top > h(u_i)h(v_k)^\top \quad (7)$$

While the authors argue that the correlation factor between $\cos(u_i, v_j)$ and $h(u_i)h(v_j)^\top$ is sufficiently positive for uniformly distributed points on a sphere, we chose to reshape the individual distributions of the embeddings in order to further increase the correlation between these two quantities and have a better ranking consistency. As Figure 3 shows, the closeness of the embeddings to the canonical axis can induce a high inconsistency as clusters are torn apart by the *sign*(.) function. Therefore, points need to be pushed away from the axis. In other words we want embeddings without sparse, or close to sparse, components.

3.2 Mathematical formulation

As mentioned in the last paragraph the ranking consistency condition implies that the embeddings should both compel with the CFN condition and the anti-sparse condition.

The anti-sparse condition can be enforced with the addition of a $-l_1$ term in the loss. As the Figure 4 shows, the gradients of this term (represented by the arrows) induce a force that push away from the axis. We can notice, that this repelling force also tends to increase the l_2 norm. Therefore, the CFN condition can be imposed by adding an l_2 unit ball constraint to the anti-sparse regularization: during the optimization process, as the l_1 norm grows, the l_2 norm eventually hits the border of the l_2 unit ball.

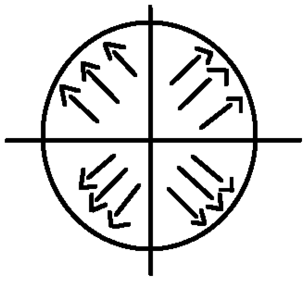

Figure 4: The arrows represent the gradients of the anti-sparse regularization $-l_1$. We expect the embeddings to hit the unit l_2 sphere during the optimization process and stick on it.

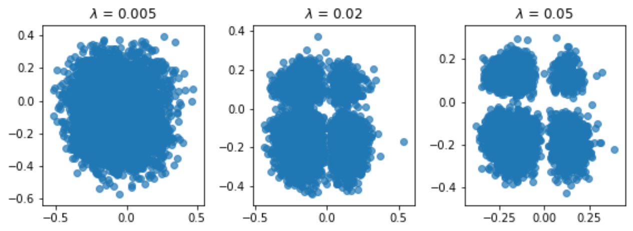

Figure 5: User embeddings with anti-sparse λ regularization projected on two randomly chosen axis. Clusters appear as we increase the value of λ.

When we incorporate these two regularizations in the CML framework, we obtain the following minimization problem:

$$\min_{||u||_2 \leq 1, ||v||_2 \leq 1} \sum_{(i,j,k) \in \mathcal{T}} [r + ||u_i - v_j||_2^2 - ||u_i - v_k||_2^2]_+ - \lambda(||u||_1 + ||v||_1) \quad (8)$$

where λ is an hyperparameter which controls for the intensity of the anti-sparse regularization. Figure 5 shows the evolution of embeddings trained with anti-sparse CML for an increasing value of λ; as it was expected the intensity of anti-sparsity increases with λ and we can see clusters appearing.

3.3 Practical resolution

We solve the problem (8) with a joint ADAM gradient descent on users and items. We note that the loss is computationally intractable for real-life data as the number of triplets $|\mathcal{T}|$ grows with $O(|\mathcal{I}| \times |\mathcal{U}|)$. We circumvent this barrier in the same fashion as in CML. At each step of the descent, we sample a batch of user-item positively interacting pairs and one negative item for each user in the batch. The embeddings are hashed with the *sign*(.) function after a fixed number of epochs. Therefore, the anti-sparse CML only scales linearly with the number of observed interactions and it enjoys the high computational efficiency of gradient-based methods.

Table 1: Characteristics of the training datasets

Dataset	#users	#items	#interactions	density
MovieLens-1M	6032	3183	696752	3,63 %
Amazon-CDs	4846	25094	219512	0,18 %
Yelp	44666	52915	1179331	0,05 %

4 EXPERIMENTS

We detail in this section our experimental procedure, the datasets that we use and the baselines we compare the anti-sparse CML to.

4.1 Experimental procedure

For each dataset, we have a set of training interactions and a test set. For each of its hyperparmeters, a model is trained on the train set and evaluated on the test set; the final score associated with a model is its best score on the test set.

Since we are in the context of recommendation, we evaluate the algorithms based on their average Normalized Discounted Cumulative Gain for k items (NDCG@k). For each user u, the algorithm recommends items $(i_1, \cdots, i_k)$ that u has not seen in the training set and we compute:

$$NDCG@k = \frac{\sum_{l=1}^{k} \frac{rel(i_l)}{\log(l+1)^l}}{\sum_{l=1}^{\min(n,k)} \frac{1}{\log(l+1)^l}} \tag{9}$$

where $rel(i_l)$ is 1 if item i_l is relevant to user u, 0 if not and n is the total number of unseen relevant items for user u. The numerator computes the number of relevant items in the selection with a logarithmic discount factor which accounts for the short-term preference. The denominator normalizes the score in $[0, 1]$. In our experiments, we set k to 10 as it is a commonly used value in the literature [10, 19, 20, 23, 24].

As we expect the representational capacity of the embeddings to increase with the dimension, we tested the algorithms with a growing embedding dimension $d = [8, 16, 32, 64, 128, 256]$.

4.2 Datasets

We used three real-world datasets for our experiments. Since most of the literature involves pointwise based algorithms, we worked with *explicit-feedback* datasets. Table 1 shows the characteristics of the three training datasets, ordered by density as it is one of the key factor to determine the difficulty of the recommendation task [17]. All datasets are triplets of user-id, item-id and the associated rating ranging from one to five. The Movie-Lens 1Million (ML-1M) [3] dataset comprises one million ratings of movies, the Amazon-CDs [4] dataset are reviews of CDs and Vyniles left by the customers on the Amazon website and Yelp [5] gather online reviews of different businesses.

We used 70% of the interactions for the train set and the rest for the test set. An interaction was considered as positive when the associated rating was 4 or above. Each dataset was filtered such that each user has at least five positive train interactions.

[3]https://grouplens.org/datasets/movielens/1m/
[4]http://jmcauley.ucsd.edu/data/amazon/
[5]https://www.kaggle.com/yelp-dataset/yelp-dataset

4.3 Baselines

We compared the performance of the proposed anti-sparse CML with the following baselines (for a more detailed presentation, please refer to Section 2):

CML: our real valued baseline. We expect it to perform better than all methods as it is both pairwise based and real-valued.

Sign CML: the naive method that simply applies the $sign(.)$ after training a CML model. Because of its simplicity, it should be the minimum performance to expect from pairwise hashing CF algorithms. However, we will show in Section 4 that it coincides with PPH applied to CML.

Discrete Personalized Ranking (DPR): a hashing method for BPR therefore pairwsise-based. Uses a discrete coordinate descent scheme. We used the implementation provided by the authors. Though the training scales linearly with the number of observed interactions, the code was not efficient and we could not test it on the Yelp dataset. We also had to stop at dimension 128.

Discrete Collaborative Filtering (DCF): also uses a discrete coordinate descent but applied to MF. The scaling is also linear in the number of interactions. We used the implementation provided by the authors.

CH (Collaborative Hashing): an MF based hashing algorithm that looks for iterative and local improvements of the binary codes to fit the entire interaction matrix. We implemented the algorithm ourselves. Because of the scaling in $O(|\mathcal{U}|\times|\mathcal{I}|)$, we did not manage to test the algorithm on the Yelp dataset.

4.4 Parameters setting

The parameters of the hashing CF baselines were tuned in the range proposed by their respective authors. For the CML algorithm, we used a batch size of 512, a learning rate of 7.5e-4, 100 epochs and a margin that we kept at 0.5. For anti-sparse CML, we tuned the parameter λ within [1e-4, 1e-1] with a logarithmic spacing; the other parameters we used were the same as in CML.

5 RESULTS

Figure 6 plots the evolution of the NDCG@10 score for our proposed model and all the baselines as the latent dimension increases. The real-valued CML (in blue) upper-bounds all hashing baselines as it was expected. Our algorithm (in orange) uniformly outperforms the other hashing CF methods and in particular the DPR (in violet), which is also pairwise-based. Besides, it also highly limits the relative performance drop when compared to CML. As Table 2 shows, more than 80 % of the real-valued performance can be attained in high dimension. Indeed, the problem that CML tries to solve is rather coarse: for each user, it tries to bring closer its positive items and repel the negative ones. The structure it tries to impose does not need the very fine grain of the continuous Euclidean distance. For instance as Figure 6 shows, the anti-sparse CML in dimension 128 yields comparable performances as the real-valued algorithm in dimension 32 yet the hash codes contain much less information than the continuous embeddings even if stored in single-precision format. However, continuity is a very handy way to make differentiable the problem of embedding a structure and to look for local improvements. Combining a real-valued training

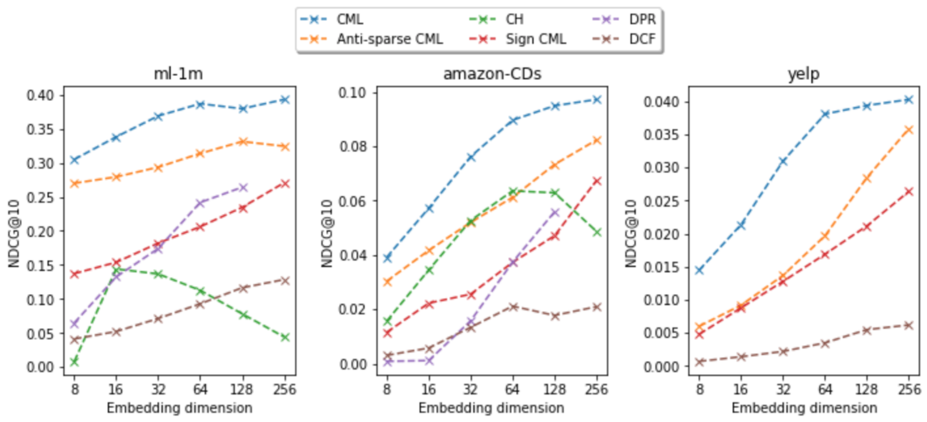

Figure 6: Evolution of the NDCG@10 scores with the embedding dimension. In blue, the real-valued CML and in orange, our algorithm. Our method uniformly outperforms all the hash CF baselines

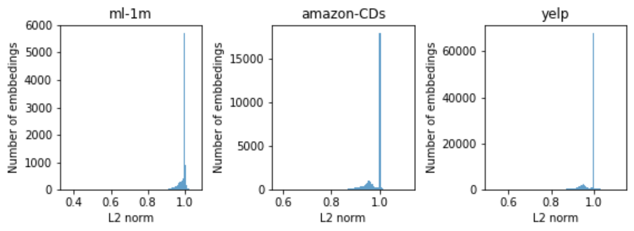

Figure 7: Histogram of norms trained with CML on the three datasets

followed by a quantization is a way to obtain "the best of both worlds".

Surprisingly, the naive Sign CML (in red) holds competitive results, especially on the ML-1M and Yelp datasets. We understand this phenomenon when we check the histogram of the norms of the embeddings on Figure 7. The CFN condition of PPH authors is actually almost respected in the standard CML. The intuition is that the "easy" way to split negative pairs is simply to increase the norms.

ML-1M	Amazon-CDs	Yelp
88 % (r=128)	84 % (r=256)	89 % (r=256)

Table 2: Ratio of best NDCG@10 of anti-sparse CML/real-valued CML. The value in the parenthesis corresponds to the embedding dimension for which anti-sparse CML performed best.

This remark highlights the fact that the anti-sparse condition is a true improvement of the CFN condition. Finally let us analyze the behavior of the pointwise baselines. CH (in green) is the only algorithm for which increasing the dimension quickly degrades the performance. One possible explanation is that CH initializes its real-valued features with PCAs and can incorporate more noise as dimension grows. Nevertheless, it is a quite strong as it manages to quickly outperform the DPR algorithm on the Amazon dataset. Even though DCF is also pointwise based, it would not be completely fair to make a comparison and state that CH is superior. Indeed, while DCF solely regresses on the observed interactions, CH regresses on the full interaction matrix like the original MF algorithm. Yet, unlike its real valued counterpart which scales as $O(|U| + |I|)$, CH does not come with an effective resolution. This could explain the choice of the authors of DCF to modify the CH algorithm. In their experiment, they droped the PCA initialization and the Iterative Quantization scheme. Actually what they implemented instead is the algorithm of [24] with an extra bits-decorrelation constraint to meet the second order condition of isotropy.

6 CONCLUSION AND FUTURE WORKS

In this paper, we developed an anti-sparse regularization framework that we used to hash the CML algorithm. Not only it outperformed all the baselines, but also, unlike numerous hashing methods, it is remarkably simple to plug-in, it enjoys the high computational efficiency of gradient-based methods and finally, its training solely scales linearly with the number of interactions.

We can legitimately wonder whether the anti-sparse regularization is intrinsically a good CF hashing framework or whether the gain of performance over the baselines is uniquely due to the initial superiority of the real-valued CML. The experimental results tend to infirm the latter: the anti-sparse CML both outperformed Sign CML, which turned out to be roughly the PPH algorithm applied to CML and can therefore be considered as a serious baseline, and DPR that almost exclusively distinguish of our algorithm on the optimization method that is used.

In the future, we intend to apply the anti-sparse framework to the MF algorithm and compare it to other pointwise based CF hashing algorithm. The outcome of such an experiment could confirm the intrinsic superiority of the anti-sparse regularization over existing CF hashing methods.

REFERENCES

[1] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 12 (2013), 2916–2929. https://doi.org/10.1109/TPAMI.2012.193

[2] Johan Hastad. 1997. Some optimal inapproximability results. In *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*. ACM, 1–10.

[3] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 193–201. https://doi.org/10.1145/3038912.3052639

[4] H. Jegou, M. Douze, and C. Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (Jan 2011), 117–128. https://doi.org/10.1109/TPAMI.2010.57

[5] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. (12 2014). http://arxiv.org/abs/1412.6980

[6] Weihao Kong and Wu jun Li. 2012. Isotropic Hashing. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1646–1654. http://papers.nips.cc/paper/4846-isotropic-hashing.pdf

[7] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS.

[8] Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for Non-negative Matrix Factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems (NIPS'00)*. MIT Press, Cambridge, MA, USA, 535–541. http://dl.acm.org/citation.cfm?id=3008751.3008829

[9] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. 2015. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 07-12-June-2015. IEEE Computer Society, 2475–2483. https://doi.org/10.1109/CVPR.2015.7298862

[10] Xianglong Liu, Junfeng He, Cheng Deng, and Bo Lang. [n.d.]. *Collaborative Hashing*. Technical Report.

[11] Y. Liu, J. Song, K. Zhou, L. Yan, L. Liu, F. Zou, and L. Shao. 2019. Deep Self-Taught Hashing for Image Retrieval. *IEEE Transactions on Cybernetics* 49, 6 (June 2019), 2229–2241. https://doi.org/10.1109/TCYB.2018.2822781

[12] Mohammad Norouzi, David J Fleet, and Ruslan R Salakhutdinov. 2012. Hamming Distance Metric Learning. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1061–1069. http://papers.nips.cc/paper/4808-hamming-distance-metric-learning.pdf

[13] M. Norouzi, A. Punjani, and D. J. Fleet. 2012. Fast search in Hamming space with multi-index hashing. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 3108–3115. https://doi.org/10.1109/CVPR.2012.6248043

[14] Loïc Paulevé, Hervé Jégou, and Laurent Amsaleg. 2010. Locality Sensitive Hashing: A Comparison of Hash Function Types and Querying Mechanisms. *Pattern Recogn. Lett.* 31, 11 (Aug. 2010), 1348–1358. https://doi.org/10.1016/j.patrec.2010.04.004

[15] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian Personalized Ranking from Implicit Feedback. (5 2012). http://arxiv.org/abs/1205.2618

[16] F. Shen, C. Shen, W. Liu, and H. T. Shen. 2015. Supervised Discrete Hashing. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 37–45. https://doi.org/10.1109/CVPR.2015.7298598

[17] Joao Felipe Silva, Natanael Moura Junior, and Luiz Caloba. 2018. Effects of Data Sparsity on Recommender Systems based on Collaborative Filtering. 1–8. https://doi.org/10.1109/IJCNN.2018.8489095

[18] Changchang Sun, Xuemeng Song, Fuli Feng, Wayne Xin Zhao, Hao Zhang, and Liqiang Nie. 2019. Supervised Hierarchical Cross-Modal Hashing. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. ACM, New York, NY, USA, 725–734. https://doi.org/10.1145/3331184.3331229

[19] Tran Dang Quang Vinh, Yi Tay, Shuai Zhang, Gao Cong, and Xiao-Li Li. 2018. Hyperbolic Recommender Systems. *CoRR* abs/1809.01703 (2018). arXiv:1809.01703 http://arxiv.org/abs/1809.01703

[20] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete Collaborative Filtering. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval - SIGIR '16*. ACM Press, New York, New York, USA, 325–334. https://doi.org/10.1145/2911451.2911502

[21] Ting Zhang, Chao Du, and Jingdong Wang. 2014. Composite Quantization for Approximate Nearest Neighbor Search. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14)*. JMLR.org, II–838–II–846. http://dl.acm.org/citation.cfm?id=3044805.3044986

[22] Yan Zhang, Defu Lian, and Guowu Yang. [n.d.]. *Discrete Personalized Ranking for Fast Collaborative Filtering from Implicit Feedback*. Technical Report. www.aaai.org

[23] Zhiwei Zhang, Qifan Wang, Lingyun Ruan, and Luo Si. 2014. Preference preserving hashing for efficient recommendation. Association for Computing Machinery (ACM), 183–192. https://doi.org/10.1145/2600428.2609578

[24] Ke Zhou and Hongyuan Zha. 2012. Learning binary codes for collaborative filtering. Association for Computing Machinery (ACM), 498. https://doi.org/10.1145/2339530.2339611