Online Kernel Matrix Factorization

Andrés Esteban Páez-Torres(✉) and Fabio A. González

MindLab Research Group, Universidad Nacional de Colombia, Bogotá, Colombia
{aepaezt,fagonzalezo}@unal.edu.co

Abstract. Matrix factorization (MF) has shown to be a competitive machine learning strategy for many problems such as dimensionality reduction, latent topic modeling, clustering, dictionary learning and manifold learning, among others. In general, MF is a linear modeling method, so different strategies, most of them based on kernel methods, have been proposed to extend it to non-linear modeling. However, as with many other kernel methods, memory requirements and computing time limit the application of kernel-based MF methods in large-scale problems. In this paper, we present a new kernel MF (KMF). This method uses a budget, a set of representative points of size $p \ll n$, where n is the size of the training data set, to tackle the memory problem, and uses stochastic gradient descent to tackle the computation time and memory problems. The experimental results show a performance, in particular tasks, comparable to other kernel matrix factorization and clustering methods, and a competitive computing time in large-scale problems.

Keywords: Feature space factorization · Kernel matrix factorization · Large-scale learning

1 Introduction

Matrix factorization (MF) is a popular method in machine learning. The goal of matrix factorization is to find two (or more) matrices, which multiplied better approximate an original input matrix. There different approaches to perform MF, which include principal component analysis (PCA), non-negative matrix factorization (NMF) [9], singular value decomposition (SVD), and independent component analysis (ICA) [7]. All of them have proved their good performance in different machine learning problems such as dimensionality reduction, manifold learning, dictionary learning and clustering. However, the majority of MF methods are linear methods, which is an important limitation when dealing with data exhibiting non-linear dependencies. Kernel methods are an important class of machine learning methods, which address the problem of non-linear pattern modeling by first mapping the data to a higher dimensional feature space induced by a kernel, and then finding linear patterns that correspond to non-linear patterns in the original space. Methods like the support vector machines are widely used and show a very high performance, compared to other supervised methods. Many linear methods can take advantage of kernels by means of the kernel trick.

A. Pardo and J. Kittler (Eds.): CIARP 2015, LNCS 9423, pp. 651–658, 2015.
DOI: 10.1007/978-3-319-25751-8_78

Kernel matrix factorization methods such as kernel matrix factorization (KMF) [6,19], kernel PCA (KPCA) [14], kernel SVD (KSVD) [16], have demonstrated their capability extracting non-linear patterns that is translated in better performance when compared to their linear counterparts. The drawback of such kernel methods is their high computational cost, the time and space required to compute kernel matrices is quadratic in terms of the number of examples. This leads to the impossibility of directly using these methods when the number of samples is large. The purpose of this paper is to present *online kernel matrix factorization* (OKMF), a KMF algorithm that is able to handle the matrix factorization in a kernel induced feature space with large-scale data sets, under a reasonable amount of time and storage resources. OKMF addresses the memory problem imposing a budget restriction, this is, restricting the number of samples needed to represent the feature space base. With respect to the computation time, OKMF uses a stochastic gradient descent (SGD) strategy for optimizing its cost function [2]. SGD has proven to be a fast alternative to solve optimization problems when the amount of samples is large. The method was evaluated in a clustering task of 5 data sets that range from medium to large-scale. We compared OKMF with other kernel and large-scale clustering methods. The paper is organized as follows, the next section present the related work to KMF. In section 3 we present OKMF derivation and algorithm. In section 4, we present the experiments, results and their analysis. Finally, in section 5 we present the conclusions of the work.

2 Related Work

This section shows a short review of the current development of kernel matrix factorization. Basically KMF methods extend the linear matrix factorization methods with kernels in order to achieve a factorization able tot extract non-linear patterns. As with SVM, the kernel trick allows to extend linear methods to work in a high-dimensional space, called feature space, without calculating an explicit mapping to that space. One of the first methods to extend MF with kernels is the work of Zhang et al. [19] that considers a factorization of the form $\phi(X)^T\phi(X) = \phi(X)^T W_\phi H$. This method uses a modification of the multiplicative rules proposed by Lee et al. [9]. In the field of recommender systems the work of Rendle et al. [13] propose to use a regularized kernel matrix factorization in order to create a recommender system. Jun et al. [8] proposed two methods, P-NMF and KP-NMF, which are projective variations of NMF [9] and KMF [19]. Another of the seminal works in the field of KMF is the method proposed by Ding et al. [6], this method uses a different factorization of the form $\Phi(X) = \Phi(X)WG^T$. An et al. [1] proposes a method for multiple kernel factorization. The work of Pan et al. [12] proposes a method with a self-constructed kernel that preserves the non-negativity restriction in feature space. The article presented by Li et al. [10] uses a similar approach to Ding et al., but constructs a new kernel that represent, in a non-supervised approach, the manifold of the feature space. The work of Li and Ngom [11] provides some modifications to

classical KMF [6,19], in order to simplify the calculations. Xia et al. [18] propose a robust kernel matrix factorization using the same factorization as in [6], but instead the cost function uses a different norm. All the mentioned works use the complete kernel matrix to compute the KMF, and therefore, they are inapplicable to large scale problems. This motivates the development of of a kernel matrix factorization method capable of dealing with large scale problems. Selecting a sample of the data, is one of the most common approaches for reducing time and memory requirements for machine learning methods. This approach has been used to improve the scalability of kernel methods through a strategy called *learning in a budget* [4]. In this strategy, the budget is a representative set of the training set with only $p \ll n$, samples, where n is total number of samples in the training data set. The kernel is then calculated between the training set and the budget, reducing the size from $O(n^2)$ to $O(np)$. Following this approach, Wang et al. proposed ECKF a KMF method [17]. This method uses a subset of the data in order to compute a KMF. Another related method is presented the work of Chen and Cai on spectral clustering[3], which uses a subset of data samples called landmarks to calculate a sparse representation of data based on kernels.

3 Online Kernel Matrix Factorization

In the following discussion we assume a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which induces a mapping $\Phi : \mathcal{X} \to \mathcal{F}$ from the problem space, $\mathcal{X}$, to a feature space, $\mathcal{F}$. For simplicity's sake, we assume $\mathcal{X} = \mathbb{R}^m$and $\mathcal{F} = \mathbb{R}^n$. Also, a set of l samples in the problem space is noted as X corresponding to a matrix in $\mathbb{R}^{m \times l}$, and a subset, called a budget, is noted as $B \in \mathbb{R}^{m \times p}$.

To understand OKMF, let's consider a factorization of the feature space into a product of a linear combination of feature space vectors and a low-dimensional latent space representation.

$$\Phi(X) = \Phi(B)WH \tag{1}$$

where $\Phi(X) \in \mathbb{R}^{n \times l}$ is the mapping of all data into a feature space , $\Phi(B) \in \mathbb{R}^{n \times p}$ is the mapping of the budget B into the feature space and B satisfies the budget restriction, i.e, $|B| \ll |X|$. $W \in \mathbb{R}^{p \times r}$ is a weight matrix . Finally, $H \in \mathbb{R}^{r \times l}$ is the latent space representation for every element of $\Phi(X)$.
The previous factorization lead us to the following optimization problem:

$$\min_{W,h_i} J_i(W, h_i) = \min_{W,h_i} \frac{1}{2}\|\Phi(x_i) - \Phi(B)Wh_i\|^2 + \frac{\lambda}{2}\|W\|_F^2 + \frac{\alpha}{2}\|h_i\|^2 \tag{2}$$

To find the optimization rules for SGD, the partial derivatives of the cost in equation 2 with respect W and h_i.

$$\frac{\partial J_i(W, h_i)}{\partial h_i} = W^T\Phi(B)^T\Phi(x_i) - W^T\Phi(B)^T\Phi(B)Wh_i + \alpha h_i \tag{3}$$

$$\frac{\partial J_i(W,h_i)}{\partial W} = \Phi(B)^T\Phi(x_i)h_i^T - \Phi(B)^T\Phi(B)Wh_ih_i^T + \lambda W \tag{4}$$

In the previous equations we can replace $\Phi(B)^T\Phi(B)$ by the matrix $k(B,B) \in \mathbb{R}^{p\times p}$ defined as $k(B,B) = \{k(b_i,b_j\}_{i,j}$, where $b_i \in \mathbb{R}^m$ corresponds to the i-th column of B. In the same way we can replace $\Phi(B)^T\Phi(x_i)$ by $k(B,x_i)$ defined in a similar way. This means that we can avoid computing the explicit mapping of data into feature space and use the more common kernel-trick approach. The update rule for h_t is a closed formula resulting from equating the derivative in equation 3 to zero.

$$h_t = (W_{t-1}^T k(B,B)W_{t-1} - \alpha I)^{-1}W_{t-1}^T k(B,x_t) \tag{5}$$

The update rule for W_t is the standard SGD rule

$$W_t = W_{t-1} - \gamma(k(B,x_t){h_t}^T - k(B,B)W_{t-1}h_t{h_t}^T + \lambda W_{t-1}) \tag{6}$$

With the rules found in equation 5 and equation 6 we can design an algorithm to compute the feature space factorization. It takes as arguments the data matrix, budget matrix, the learning rate γ, regularization parameters λ and α. The output correspond to the matrix W.

Algorithm 1. Online kernel matrix factorization

procedure OKMF($X, budget, W, \gamma, \lambda, \alpha$)
  $KB \leftarrow k(budget, budget)$
  for all $x_i \in X$ **do**
    $kxi \leftarrow k(budget, x_i)$
    $h_i \leftarrow (W^T KBW - \alpha I)^{-1}W^T kxi$
    $W \leftarrow W - \gamma(kxi{h_i}^T - KBWh_i{h_i}^T + \lambda W)$
  end for
  return W
end procedure

The use of a budget poses a new problem which is the selection of this budget. To tackle this problem, two approaches were used. The first was randomly picking p instances of the data set. The second one is computing a k-Means clustering with $k = p$, the resulting k cluster centers can be viewed as a prototype set of the data.

4 Experiments

One of the applications of matrix factorization is clustering, hence we choose it as the task to evaluate OKMF. The columns of the factors matrix, $\Phi(B)W$, can be viewed as a set of cluster centers in feature space. In turn, the latent space representation matrix H contains the information of the membership of each element to each cluster.

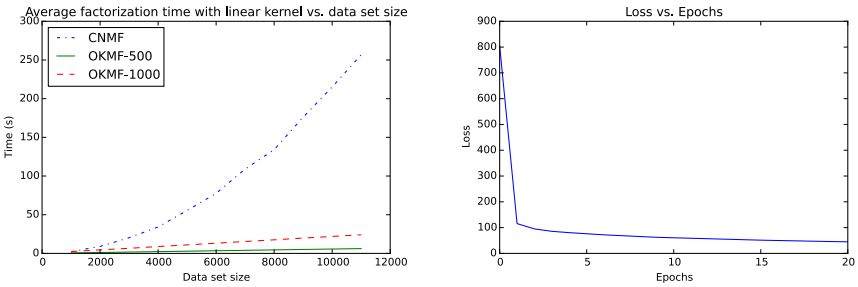

Fig. 1. Average factorization time of 30 runs vs. data set size (left) and OKMF average loss vs. epochs(right)

Table 1. Datasets Characteristics

Name	# of Samples	# of Classes	# of Attributes
Abalone	4,177	3	8
WineQ	4,898	3	11
Synthetic	5,000	2	2
Seismic	98,528	3	50
Covtype	581,012	7	54

Two kinds of data sets were used, medium-scale and large-scale, the characteristics of the data sets is presented in Table 1. The performance measure selected is the clustering accuracy, which is the fraction of correctly clustered points, showed in equation 7.

$$AC = \frac{\sum_{i=1}^{N} \delta(cf_i, map(c_i))}{N} \tag{7}$$

Where cf_i is the found label and c_i the ground truth, N the number of data points. $\delta(\cdot)$ is 1 when the found label cf_i matches c_i. $map(\cdot)$ is the best match of the found clusters and the ground truth computed using the Hungarian Algorithm. The compared algorithms are kernel k-Means [5], kernel convex non-negative matrix factorization [6] and online k-Means [15]. The kernels used in the experiments are linear and Gaussian (RBF). A parameter exploration was performed to find parameters of OKMF and the σ parameter of the RBF kernel. 30 tests were conducted and average accuracy and average time are reported. Finally the budget size was fixed to 500. The randomly picked budget is labeled as OKMF-R in the results, the strategy using k-Means cluster centers is labeled as OKMF-K. All algorithms were implemented in Python using Anaconda's MKL Extension on a computer with Intel Core i5 with four cores at 3.30GHz and 8GB of memory. Given kernel k-Means and CNMF methods require to have the whole kernel matrix in main memory, it is not feasible to use them to cluster the Seismic and Covtype data sets.

Table 2. Average clustering accuracy of 30 runs. Results for CNMF and KK-means are not reported for the larger datasets, since it was not possible to evaluate them on the whole dataset.

Method	Abalone	WineQ	Synthetic	Seismic	Covtype
CNMF-Linear	0.5253	0.3966	0.6369	n.a.	n.a.
CNMF-RBF	0.5269	0.4488	0.9800	n.a.	n.a.
K K-means-Linear	0.5280	0.3962	0.6377	n.a.	n.a.
K K-means-RBF	0.5268	0.4487	1.000	n.a.	n.a.
Online k-means	0.5286	0.3918	0.6494	0.4007	0.4276
OKMF-K-Linear	0.3658	0.4261	0.5041	0.3923	0.4405
OKMF-K-RBF	0.5188	0.4447	0.6230	0.5000	0.4875
OKMF-R-Linear	0.3658	0.4276	0.5033	0.3918	0.4300
OKMF-R-RBF	0.5331	0.4410	0.6925	0.5000	0.4876

4.1 Analysis

Table 2 show the results of the different methods and the different data sets. Our method performs very well in the task of clustering. Also the capability of using kernels enhance the performance of the accuracy without incurring in large memory usage or time to compute the factorization. Also the accuracy is not greatly affected by the budget selection scheme, given there is not a large difference between the performance of OKMF-R and OKMF-K.

Table 3 presents the clustering times, online k-means is the fastest algorithm, it outperforms by much the other algorithms. However, OKMF is faster than the CNMF factorization algorithm. The figure 1 shows a comparison between CNMF and OKMF with 500 and 1000 budget size, whilst CNMF average time behaves quadratic, OKMF average time behaves linear, also shows that OKMF converges within 2 to 5 epochs. Figure 2 shows the impact of budget size on clustering accuracy, this support the idea of having a small budget and achieve good performances.

Table 3. Average clustering time of 30 runs. Results for CNMF and KK-means are not reported for the larger datasets, since it was not possible to evaluate them on the whole dataset.

Method	Abalone	WineQ	Synthetic	Seismic	Covtype
CNMF-Linear	35.91	52.26	49.92	n.a.	n.a.
CNMF-RBF	37.15	50.88	50.95	n.a.	n.a.
K K-means-Linear	1.10	1.55	1.54	n.a.	n.a.
K K-means-RBF	2.35	2.84	3.04	n.a.	n.a.
Online k-means	0.06	0.06	0.05	0.41	1.90
OKMF-K-Linear	3.06	3.22	3.62	45.45	422.21
OKMF-K-RBF	4.54	4.84	5.18	82.65	648.59
OKMF-R-Linear	1.23	1.33	1.20	34.18	396.94
OKMF-R-RBF	2.72	3.14	3.76	70.23	635.67

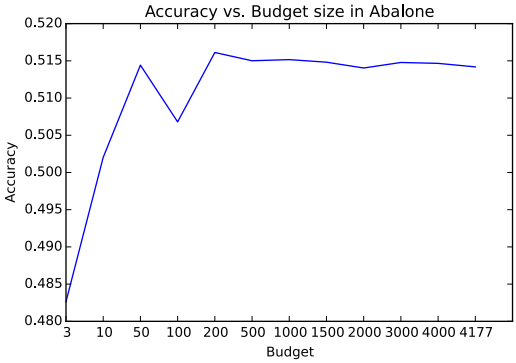

Fig. 2. Average clustering accuracy of 30 runs vs. budget size

5 Conclusions

We presented a novel kernel matrix factorization method that uses a budget restriction to tackle the memory issue and computation time. Using a budget leads to OKMF save memory, given it is not necessary to store the complete kernel matrix of size $n \times n$, but a smaller kernel matrix of the budget of size $p \times p$, which is a significant reduction if $p \ll n$. However, this leads to two problems, the budget size selection and the selection of the budget itself. The first problem can be solved by the exploration of this budget size in order to minimize the reconstruction error. To solve the second problem, we studied two ways, the first is randomly pick p instances of the original data, the second is applying k-means fixing k to the budget size. Also, in the task of clustering our method performs as well as other kernel matrix factorization methods, including CNMF. OKMF has low memory requirements, because it depends directly on the budget size, so having millions of samples, doesn't affect the memory usage. Using SGD as the method to optimize also implies a memory saving when performing the KMF, instead of keeping in memory a kernel matrix of the budget against all the data with a size of $p \times n$, OKMF only needs a kernel vector of the budget against the current instance of size $p \times 1$. Besides OKMF has an advantage in the time required to process large number of instances, because it converges with a small number of epochs and the time factorization takes grows linearly with the number of instances. As future work, other strategies for the budget selection can be studied and the application of OKMF in other tasks different from clustering.

References

1. An, S., Yun, J.M., Choi, S.: Multiple kernel nonnegative matrix factorization. In: 2011 ICASSP Conference, pp. 1976–1979. IEEE (2011)
2. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: COMPSTAT 2010, pp. 177–186. Springer (2010)
3. Chen, X., Cai, D.: Large-scale spectral clustering with landmark-based representation. In: Twenty-Fifth AAAI Conference (2011)
4. Crammer, K., Kandola, J., Singer, Y.: Online classification on a budget. In: Advances in Neural Information Processing Systems, pp. 225–232 (2004)
5. Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means: spectral clustering and normalized cuts. In: Tenth ACM SIGKDD Conference, pp. 551–556. ACM (2004)
6. Ding, C., Li, T., Jordan, M.I.: Convex and semi-nonnegative matrix factorizations. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(1), 45–55 (2010)
7. Hyvärinen, A., Karhunen, J., Oja, E.: Independent component analysis, vol. 46. John Wiley & Sons (2004)
8. Jun, Y., Jintao, M., Xiaoxu, L.: A kernel based non-negative matrix factorization. In: Second IITA-GRS Conference, vol. 1, pp. 376–379. IEEE (2010)
9. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature **401**(6755), 788–791 (1999)
10. Li, P., Chen, C., Bu, J.: Clustering analysis using manifold kernel concept factorization. Neurocomputing **87**, 120–131 (2012)
11. Li, Y., Ngom, A.: A new kernel non-negative matrix factorization and its application in microarray data analysis. In: 2012 CIBCB Symposium, pp. 371–378. IEEE (2012)
12. Pan, B., Lai, J., Chen, W.S.: Nonlinear nonnegative matrix factorization based on mercer kernel construction. Pattern Recognition **44**(10), 2800–2810 (2011)
13. Rendle, S., Schmidt-Thieme, L.: Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In: 2008 ACM RecSys Conference, pp. 251–258. ACM (2008)
14. Schölkopf, B., Smola, A., Müller, K.-R.: Kernel principal component analysis. In: Gerstner, W., Hasler, M., Germond, A., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327, pp. 583–588. Springer, Heidelberg (1997)
15. Sculley, D.: Web-scale k-means clustering. In: WWW 2010 Conference, pp. 1177–1178. ACM (2010)
16. Shawe-Taylor, J., Cristianini, N.: Kernel methods for pattern analysis. Cambridge University Press (2004)
17. Wang, L., Rege, M., Dong, M., Ding, Y.: Low-rank kernel matrix factorization for large-scale evolutionary clustering. IEEE Transactions on Knowledge and Data Engineering **24**(6), 1036–1050 (2012)
18. Xia, Z., Ding, C., Chow, E.: Robust kernel nonnegative matrix factorization. In: 12th ICDMW Conference, pp. 522–529. IEEE (2012)
19. Zhang, D., Zhou, Z.-H., Chen, S.: Non-negative Matrix Factorization on Kernels. In: Yang, Q., Webb, G. (eds.) PRICAI 2006. LNCS (LNAI), vol. 4099, pp. 404–412. Springer, Heidelberg (2006)