Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Multi-Objective Self-Paced Learning

Hao Li[1] **, Maoguo Gong**[1,*]**, Deyu Meng**[2,*]**, Qiguang Miao**[3]
[1]Key Laboratory of Intelligent Perception and Image Understanding, Xidian University, Xi'an, China
[2]School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, China
[3]School of Computer Science and Technology, Xidian University, Xi'an, China
omegalihao@gmail.com, gong@ieee.org, dymeng@mail.xjtu.edu.cn, qgmiao@mail.xidian.edu.cn
[*]Corresponding author

Abstract

Current self-paced learning (SPL) regimes adopt the greedy strategy to obtain the solution with a gradually increasing pace parameter while where to optimally terminate this increasing process is difficult to determine. Besides, most SPL implementations are very sensitive to initialization and short of a theoretical result to clarify where SPL converges to with pace parameter increasing. In this paper, we propose a novel multi-objective self-paced learning (MOSPL) method to address these issues. Specifically, we decompose the objective functions as two terms, including the loss and the self-paced regularizer, respectively, and treat the problem as the compromise between these two objectives. This naturally reformulates the SPL problem as a standard multi-objective issue. A multi-objective evolutionary algorithm is used to optimize the two objectives simultaneously to facilitate the rational selection of a proper pace parameter. The proposed technique is capable of ameliorating a set of solutions with respect to a range of pace parameters through finely compromising these solutions inbetween, and making them perform robustly even under bad initialization. A good solution can then be naturally achieved from these solutions by making use of some off-the-shelf tools in multi-objective optimization. Experimental results on matrix factorization and action recognition demonstrate the superiority of the proposed method against the existing issues in current SPL research.

Introduction

In the field of machine learning and artificial intelligence, *curriculum learning* (CL) (Bengio et al. 2009) and *self-paced learning* (SPL) (Kumar, Packer, and Koller 2010) have been attracting increasing attention to deal with the difficulty of training in the presence of non-convex training criteria and complex data with heavy noises and outliers. Both the learning paradigms are inspired by the learning process of humans/animals, which learns with easier concepts at first and then gradually involves more complex ones into training. It has been proved that this learning paradigm is beneficial in avoiding bad local minima and in achieving a better generalization result (Khan, Mutlu, and Zhu 2011; Basu and Christensen 2013; Tang, Yang, and Gao 2012).

A curriculum defines a set of training samples organized in ascending order of learning difficulty. In CL, the curriculum is assumed to be given beforehand, and remains unchanged thereafter. However, the curriculum design is determined independently of the subsequent learning. Then SPL was proposed by Kumar *et al.* to dynamically generate the curriculum according to what the learner has already learned. In SPL, a regularization term is introduced into the learning objective to jointly learn the curriculum and model parameters, which is called self-paced (SP) regularizer.

In SPL, a binary variable is used in hard weighting scheme (Kumar, Packer, and Koller 2010) to indicate whether the sample is easy or not. Then Jiang *et al.* proposed a soft weighting method (Jiang et al. 2014a), which assigns real-valued weights to reflect the importance of samples and presents three efficient SPL regularizers for different kinds of data sets with different characteristics. SPL has been successfully applied to various applications, such as segmentation (Kumar et al. 2011), domain adaption (Tang et al. 2012), dictionary leaning (Tang, Yang, and Gao 2012), long-term tracking (Supančič and Ramanan 2013), reranking (Jiang et al. 2014a), action and event detection (Jiang et al. 2014b; 2015), and matrix factorization (Zhao et al. 2015).

Compared to traditional machine learning methods, SPL exploits a weight variable to measure the easiness of samples and introduces a gradually increasing pace parameter to control the pace at which the model learns new samples. However, there are some weaknesses in these SPL methods. First, the SPL implementation is generally very sensitive to initialization, which has been indicated by previous investigations (Jiang et al. 2014b; 2015). Second, it is difficult to determine when to stop the iteration in real implementation of a SPL regime. That is, it is hard to select a proper pace where the self-paced learning process should be terminated. Such selection issue is critical since when the pace parameter is large, SPL is prone to get a bad solution in the presence of noisy samples. Third, only a single solution with respect to (wrt) a certain pace parameter can be obtained after SPL calculation. This, however, loses insights for the entire solution spectrum wrt different pace parameters to a certain extent. We always expect to achieve the entire solution path wrt the pace to observe the whole self-paced evolution process and recover useful intrinsic patterns from it.

In this study, we clarify that the SPL model can be equivalently understood as a multi-objective model. It is interesting that by employing multi-objective evolutionary algorithms

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

(MOEAs) (Deb 2001; Coello, Van Veldhuizen, and Lamont 2002) for solving the model, all of the aforementioned issues for current SPL research can be naturally explained and resolved. Recently, researchers have proved that MOEAs can overcome the difficulty that limits the greedy algorithm in some NP-hard problems (Yu, Yao, and Zhou 2013; Qian, Yu, and Zhou 2013). Specifically, MOEAs aim at finding a set of Pareto optimal solutions and then obtain a knee solution to represent the best trade-off of multiple objectives (Branke et al. 2004; Li et al. 2014). And thus it is natural to take the learning objective term and the SP regularizer term as two objectives of multi-objective optimization and embed the traditional SPL problem into this long-term-investigated area to facilitating more known tools to be utilized for better exploring the SPL insight.

Along such research methodology, we establish a novel *multi-objective self-paced learning* (MOSPL) paradigm to alleviate the deficiencies existed in SPL research. The learning objective and the SP regularizer are optimized by MOEAs simultaneously. Then the entire solution path wrt the pace can approximately be obtained by finding the Pareto optimal front. Each solution in the solution path is obtained by compensating its intra-population neighboring and pre-population solutions. MOSPL is not sensitive to initialization because each solution has the chance to compensate its neighboring solutions and be automatically self-rectified. Furthermore, MOSPL can achieve the entire solution path wrt the pace to gain more insights into the SPL problem. It is easy to determine the final pace, which is naturally achieved at the knee position of this solution path wrt the pace variable by employing some off-the-shelf tools in multi-objective optimization. The aforementioned issues existed in current SPL regime can thus be entirely alleviated.

The rest of the paper is organized as follows. We first briefly introduce the related background knowledge on SPL and multi-objective optimization. Then we propose the model and algorithm of MOSPL. Finally, the experimental results and conclusions are given.

Related Background

Self-paced Learning

Formally, we denote the training dataset as $\mathcal{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathbb{R}^m$ denotes the ith observed sample, and y_i represents its label. Let $L(y_i, g(\mathbf{x}_i, \mathbf{w}))$ denote the loss function which calculates the cost between the ground truth label y_i and the estimated label $g(\mathbf{x}_i, \mathbf{w})$. Here $\mathbf{w}$ represents the model parameter inside the decision function g. In SPL, variable $\mathbf{v}$ is introduced into the learning objective to indicate whether the ith sample is easy or not. The target of SPL is to jointly learn the model parameter $\mathbf{w}$ and the latent weight variable $\mathbf{v} = [v_1, \cdots, v_n]$ by minimizing:

$$\min_{\mathbf{w},\mathbf{v}} \mathbb{E}(\mathbf{w}, \mathbf{v}; \lambda) = \sum_{i=1}^{n} v_i L(y_i, g(\mathbf{x}_i, \mathbf{w})) + f(\mathbf{v}; \lambda), \quad (1)$$

where $\mathbf{v} \in [0, 1]^n$ and $f(\mathbf{v}; \lambda)$ is the SPL regularizer. λ is a gradually increasing pace parameter for controlling the learning pace. The parameter λ plays an important role in the process of learning new samples. When λ is small, only "easy" samples with small losses will be considered into training. As λ grows, more samples with larger losses will be gradually appended to training a more "mature" model.

The original SPL (Kumar, Packer, and Koller 2010) is to minimize the weighted training loss together with the negative l_1-norm regularizer $-\|\mathbf{v}\|_1 = -\sum_{i=1}^{n} v_i$, where v_i is a binary variable. Afterwards, (Jiang et al. 2014a; Zhao et al. 2015) proposed an axiomatic understanding the SP regularizer and further extended more efficient formulations of SP regularizer as follows.

Linear soft weighting regularizer: This scheme is to linearly discriminate samples wrt their losses, which can be realized by the following function:

$$f(\mathbf{v}; \lambda) = \lambda(\frac{1}{2}\|\mathbf{v}\|_2^2 - \sum_{i=1}^{n} v_i), \quad (2)$$

where $\lambda > 0$.

Logarithmic soft weighting regularizer: This approach is to penalize the loss logarithmically, which can be achieved by the following function:

$$f(\mathbf{v}; \lambda) = \sum_{i=1}^{n} (\zeta v_i - \frac{\zeta^{v_i}}{\log \zeta}), \quad (3)$$

where $\zeta = 1 - \lambda$ and $0 < \lambda < 1$.

Mixture weighting regularizer: Mixture scheme is a hybrid of the "soft" and the "hard" scheme, which can be stated by the following function:

$$f(\mathbf{v}; \lambda) = -\zeta \sum_{i=1}^{n} \log(v_i + \frac{1}{\lambda_1}\zeta), \quad (4)$$

where $\zeta = \frac{\lambda_1 \lambda_2}{\lambda_1 - \lambda_2}$ and $\lambda_1 > \lambda_2 > 0$.

The closed-form solutions of the SPL model under the above SP regularizers can be easily deduced. For example, the solution under the linear soft weighting regularizer can be written as:

$$v_i^* = \begin{cases} 1 - \frac{L_i}{\lambda} & L_i < \lambda \\ 0 & L_i \geq \lambda. \end{cases} \quad (5)$$

In this paper, we prefer to utilize the linear SP regularizer due to its simplicity and efficiency in calculation.

Multi-objective optimization

A multi-objective optimization problem (MOP) with m decision variables and l objectives can be described as

$$\begin{aligned} \min \quad & F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_l(\mathbf{x}))^T \\ \textit{s.t.} \quad & \mathbf{x} = [x_1, x_2, \cdots, x_m] \in \Omega, \end{aligned} \quad (6)$$

where $\mathbf{x}$ is the *decision vector*, Ω is the *decision space*, $F : \Omega \to R^l$ consists of l real-valued objective functions and R^l is called the *objective space*. The *attainable objective set* is defined as the set $\{F(x) | x \in \Omega\}$. In most instances, the objectives in an MOP are contradictory to each other, which means no point in feasible space can minimize all the

objectives simultaneously. Hence, multi-objective optimization (Deb 2001; Coello, Van Veldhuizen, and Lamont 2002) are designed to find the best trade-off relationship among them simultaneously.

Considering a minimization problem for each objective, it is said that a decision vector $\mathbf{x}_u \in \Omega$ dominates another vector $\mathbf{x}_v \in \Omega$ if and only if

$$\begin{aligned} &\forall i = 1, 2, \cdots, n \ f_i(\mathbf{x}_u) \leq f_i(\mathbf{x}_v) \\ &\wedge \exists j = 1, 2, \cdots, n \ f_i(\mathbf{x}_u) < f_i(\mathbf{x}_v). \end{aligned} \tag{7}$$

And a point x^* in Ω is called a Pareto optimal solution to Eq. (6) in case that there is no such point x in Ω that makes $F(x)$ dominate $F(x^*)$. Then $F(x^*)$ is termed as Pareto optimal vector. The objectives in a Pareto optimal vector have such relationship: a decrease in one objective causes an increase in the others. All the Pareto optimal points constitute a set called Pareto optimal set (Miettinen 1999), and their corresponding Pareto optimal objective vectors are called the Pareto optimal front (PF) (Miettinen 1999).

For multi-objective optimization, it has been recognized that evolutionary algorithms (EAs) are well suited because EAs can deal with a set of possible solutions simultaneously (Fonseca and Fleming 1995; Deb 2001). Various EAs to deal with MOPs have been proposed (Deb et al. 2002; Coello, Pulido, and Lechuga 2004; Zhang and Li 2007) and these EAs are termed as multi-objective evolutionary algorithms (MOEAs). MOEAs seek to obtain a set of Pareto optimal solutions for approximating the true PF in a single run.

Multi-objective Self-paced Learning

MOSPL Model

Through taking the original learning objective (imposed with weights) term and the SP regularizer term as two objectives, the SPL problem (1) can be naturally reformulated as a standard bi-objective optimization problem. A certain SPL model under the pace parameter can then be considered as a certain scalar aggregation between these two terms through the compromising parameter. The weaknesses of such scalarized approach to handle competitive objective in machine learning have been discussed (Matsuyama 1996; Jin and Sendhoff 2008). Specifically, such simplification makes the model hard to determine the compromising (pace) parameter. Under such understanding, the original SPL implementation can just be seen as a greedy strategy through increasing the pace and gradually evaluating the model performance to find an appropriate pace to terminate the iteration.

In this paper, a novel *multi-objective self-paced learning* (MOSPL) model is proposed to address the drawbacks existed in current SPL implementations. Specifically, we can first reformulate the SPL problem in Eq. (1) as the following MOP:

$$\min_{\mathbf{w},\mathbf{v}} (f_1, f_2)^T = \begin{cases} f_1 = \frac{1}{2}\|\mathbf{v}\|_2^2 - \sum_{i=1}^n v_i, \\ f_2 = \sum_{i=1}^n \mathbf{v}_i L(y_i, g(\mathbf{x}_i, \mathbf{w})). \end{cases} \tag{8}$$

Then, we can revisit SPL in a entirely new viewpoint of MOP: In SPL, the pace parameter λ represents the "age" of the learning paradigm, which weighs the two objectives.

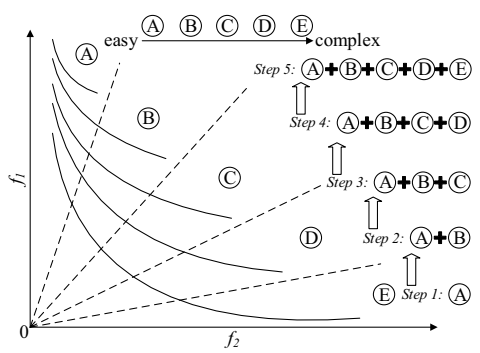


Figure 1: The schematic diagram of MOSPL. MOSPL starts with training in region A, which is the easiest one. Then region B is involved into training. MOSPL stops when all regions have been trained. The evolutive process of the solution path gradually iterating from easy to complex regions is also demonstrated.

However, in the MOSPL model, we minimize the objectives simultaneously by using evolutionary algorithms to obtain a set of solutions. Therefore the "age" of MOSPL can be analyzed in objective space, which can be divided into several regions ranked in ascending order of learning difficulty. As shown in Figure 1, MOSPL starts with training in the easy region, and then gradually takes more complex regions into consideration. Each solution can compensate its neighboring solutions and has the chance to be self-rectified inbetween. Finally, MOSPL stops until all regions have been trained and obtains a solution sequence corresponding to the solution path wrt pace of the original SPL model, which facilitates us to use some off-the-shelf tools in MOP to select a proper knee point along the path as the final model parameters.

MOSPL Algorithm

Instead of extracting the solution path of SPL wrt λ, the task can be transformed to an easier one by searching the solution path wrt the sample number involved into training, i.e., the number of samples with non-zero importance weights v_i (Jiang et al. 2014a). Therefore we can predefine a sample number sequence, we also call it pace sequence for convenience, $\widehat{\mathbf{N}} = \{\widehat{N}_1, \widehat{N}_2, \cdots, \widehat{N}_p\}$ ($\widehat{N}_i < \widehat{N}_j$ for $i < j$) representing the number of selected samples in the process of SPL. Each $\widehat{N}_i$ denotes how many samples will be selected in the ith SPL stage, and $\widehat{N}_p = n$ means finally all samples are chosen into training. Then we use a weight sequence, which corresponds to a population in MOEAs implementation, $P = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_p\}$ to represent the solution path, where $\mathbf{v}_j$ is the weight variable. Obviously, the population P with the pace sequence $\widehat{\mathbf{N}}$ is the expected solution path at the end of MOSPL. Specifically, for each element $\mathbf{v}_j$ along this solution path P, the number of its nonzero entries is $\widehat{N}_j$. The aim of our MOSPL algorithm is then to incrementally rectify this solution path by employing certain off-the-shelf MOEA tricks.

The key idea of the proposed MOSPL algorithm is to

start training a population in easy regions, and then gradually evolve the population to more complex regions. We now introduce the implementation details of our MOSPL algorithm. $\mathbf{N}^{(i)}$ is used to represent how many samples will be selected in the population of the ith iteration of MOSPL. At the beginning evolution step, we enforce each number in the pace sequence $\mathbf{N}^{(1)}$ with comparably small values, i.e., we let all $N_j^{(1)} = \widehat{N}_1$ $(j = 1, 2, \cdots, p)$ in $\mathbf{N}^{(1)}$ in the first iteration. With the evolution process continuing, we gradually increase the backwards paces in $\mathbf{N}^{(i)}$, progressively complete the solution path especially wrt larger paces and evolve more complex knowledge into the learning population. Therefore in the next evolution steps, the pace sequence is defined as

$$N_j^{(i)} = \begin{cases} \widehat{N}_j, & j \leq i \\ \widehat{N}_i, & \text{others} \end{cases} \text{ for any } j = 1, 2, ..., n. \quad (9)$$

By increasing the pace sequence in this way, we have constructed a gradually evolutionary population from easy to complex. In the last iteration, we obtain the solution path with the expected pace sequence $\widehat{\mathbf{N}}$. MOSPL gradually involves more complex regions into training and stops until all regions have been trained. Obviously, we need p iterations to traverse the entire regions in objective space.

For each iteration, it is a fundamental optimization problem, which can be solved by certain off-the-shelf MOEAs. The optimal solutions in ith iteration can be obtained by optimizing the following MOP:

$$\begin{aligned} &\min(f_1, f_2)^T \\ &\text{s.t.} \|\mathbf{v}_j\|_0 = N_j^{(i)}, j = 1, 2, \cdots, p. \end{aligned} \quad (10)$$

where $\mathbf{v}_j$ is the jth individual of P and $\|\mathbf{v}_j\|_0$ counts the number of nonzero entries in $\mathbf{v}_j$. To solve this MOP, MOEA/D (Zhang and Li 2007) is used as the fundamental optimization tool to simultaneously optimize the objectives (Eq. (8)) with the predefined pace sequence. The proposed MOSPL is shown in **Algorithm 1**. In the initialization, P is randomly initialized with a few number of samples, which is in the easy region of objective space. During iterations, most individuals of P involve more samples into training to consider more complex regions of objective space. For each iteration, the learning objective and the SPL regularizer are optimized by MOEA/D simultaneously. Each individual in P is initiated by the output of the corresponding individual of the population obtained in the last iteration, and optimized by using information from its several neighboring individuals. MOEA/D uses a differential evolution (DE) operator for producing an offspring, which generates the offspring from several parent solutions. Then the offspring is used to update the neighboring solutions.

By utilizing such an algorithm, the population in each evolution step is capable of not only making use of the helpful population knowledge of the last evolution step, but also compensating all solutions inbetween. A good solution path wrt various pace number $\widehat{\mathbf{N}}$ is thus expected to be recursively ameliorated during this evolution process through its full intra- and inter-population interactive manner. Then an angle-based method proposed in (Branke et al. 2004) is used to obtain the knee solution from the solution path, where further improvement in one objective causes a rapid degradation in other objectives. The knee solution represents the best trade-off between the learning objective and the SPL regularizer.

Algorithm 1 Algorithm of Multi-objective Self-paced Learning.

Input: The pace sequence: $\widehat{\mathbf{N}} = \{\widehat{N}_1, \widehat{N}_2, \cdots, \widehat{N}_p\}$, the training dataset $\mathcal{D}$.
Output: The knee solution.

```
Initialize the population P = {v_1, ..., v_p} constrained with ||v_j||_0 = N_j^(1), j = 1, 2, ..., p.
for i=1 to p do
    if i > 1 then The new region is involved into training. Therefore individuals of P are initiated by the model parameters from the last population and updated with the pace sequence N^(i).
    end if
    //Update P by minimizing Eq. (10) using MOEA/D.
    for j=1 to p do
        Reproduce new offspring v̂_j and update the neighboring solutions. For more information, please refer to (Zhang and Li 2007).
    end for
end for
Obtain the solution path wrt the pace N̂ and an angle-based method (Branke et al. 2004) is used to obtain the knee solution.
return the knee solution.
```

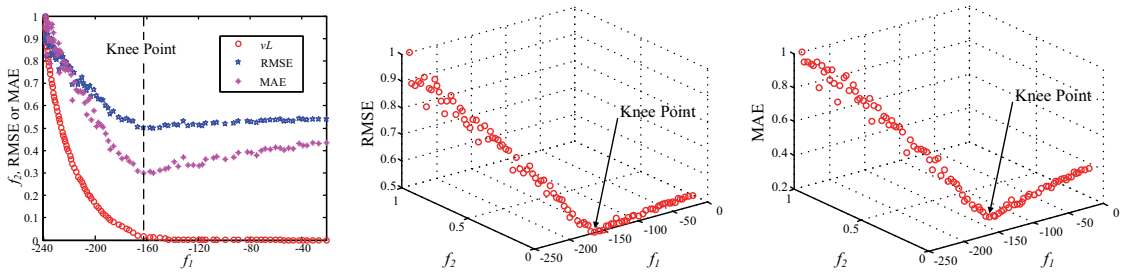
The proposed method addresses the issues existing in current SPL implementations. MOSPL is robust to initialization because each solution can use information from its neighboring solutions and be automatically self-rectified. Moreover, MOSPL guides the solutions to converge to the Pareto optimal front and approaches the entire solution path wrt the pace to gain more insights into the SPL problem. A good solution can be obtained by employing an off-the-shelf tool in multi-objective optimization, which is naturally achieved at the knee position of this solution path wrt the pace.

Experiments

In order to validate the advantages of the proposed MOSPL, matrix factorization and action recognition are considered in our experiments.

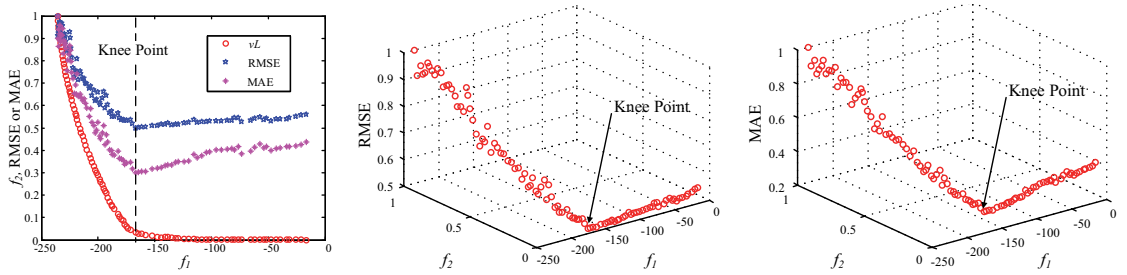
Matrix Factorization

Matrix factorization aims to factorize an $m \times n$ data matrix $\mathbf{Y}$, whose entries are denoted as y_{ij}s, into two smaller factors $\mathbf{U} \in R^{m \times r}$ and $\mathbf{V} \in R^{n \times r}$, where $r \ll \min(m, n)$, such that $\mathbf{U}\mathbf{V}^T$ is possibly close to $\mathbf{Y}$ (Chatzis 2014; Zhao et al. 2015). Matrix factorization has many applications in various disciplines (Tomasi and Kanade 1992; Mnih and Salakhutdinov 2007). Here we test the proposed MOSPL scheme on synthetic matrix factorization problems.



(a) (b) (c)

Figure 2: Relationships between f_1, f_2, RMSE and MAE with the LS loss. (a) shows the variation of f_2, RSME and MAE with change in f_1. (B) shows 3-D plot of RSME, f_1 and f_2. (c) shows 3-D plot of MAE, f_1 and f_2.



(a) (b) (c)

Figure 3: Relationships between f_1, f_2, RMSE and MAE with the LAD loss. (a) shows the variation of f_2, RSME and MAE with change in f_1. (B) shows 3-D plot of RSME, f_1 and f_2. (c) shows 3-D plot of MAE, f_1 and f_2.

The data were generated as follows: two matrices $\mathbf{U}$ and $\mathbf{V}$, both of which are of size 40×20, were first randomly generated with each entry drawn from the Gaussian distribution $\mathcal{N}(0,1)$, leading to a ground truth rank-4 matrix $\mathbf{Y}_0 = \mathbf{U}\mathbf{V}^T$. Then 40% of the randomly selected entries were designed as missing data, 20% of the other randomly selected entries were added to uniform noise on [-20,20], and the rest entries were added to Gaussian noise drawn from $\mathcal{N}(0, 0.01)$. Two commonly used loss functions are considered in this paper: the LS loss $L(y_{ij}, [\mathbf{U}\mathbf{V}^T]_{ij}) = (y_{ij} - [\mathbf{U}\mathbf{V}^T]_{ij})^2$ and the LAD loss $L(y_{ij}, [\mathbf{U}\mathbf{V}^T]_{ij}) = |y_{ij} - [\mathbf{U}\mathbf{V}^T]_{ij}|$. The loss values are calculated by modifying the solver proposed by (Zheng et al. 2012) to solve the trace-norm regularized matrix factorization with both the LS and LAD loss. Two criteria were adopted for performance assessment. (1) *root mean square error* (RMSE): $\frac{1}{\sqrt{mn}}\|\mathbf{Y}_0 - \hat{\mathbf{U}}\hat{\mathbf{V}}^T\|_F$, and (2) *mean absolute error* (MAE): $\frac{1}{mn}\|\mathbf{Y}_0 - \hat{\mathbf{U}}\hat{\mathbf{V}}^T\|_1$, where $\hat{\mathbf{U}}$, $\hat{\mathbf{V}}$ denote the outputs from a utilized matrix factorization method.

Figure 2 and Figure 3 show the relationship among the SPL regularizer (f_1), the learning objective (f_2), RMSE and MAE. The values of f_2 are normalized into [0, 1]. The values of RMSE and MAE are normalized into [0.5, 1] and [0.3, 1], respectively. The left graphs show 2-D plots of the variance of f_2, RMSE and MAE with change in f_1. The middle graphs depict 3-D views of RSME, f_1 and f_2. The right graphs provide 3-D views of MAE, f_1 and f_2. Figure 2(a) and Figure 3(a) show the Pareto fronts obtained by the proposed method. The knee point does exist in the Pareto front with the LS and LAD loss, which represents the best compromise between the learning objective and the SPL regularizer. If more entries are added into the process of matrix factorization, the loss values will increase rapidly in the presence of noise entries. Therefore the values of learning objective have fast growth beyond the knee point. As shown in Figure 2(a) and Figure 3(a), the solution with the smallest RMSE and MAE is next to the knee point. To better understand of the convergence of the proposed MOSPL, tendency curves of RMSE and MAE with respect to iterations are shown in Figure 4. In the first several iterations, SPL gradually takes more entries into consideration. When the iteration continues, SPL cannot stop at the expected "position" with small RMSE and MAE, which is prone to get a bad solution. The figure shows that the results obtained by MOSPL is more stable than those obtained by SPL.

Then we compare the results provided by the proposed technique with those obtained by three state-of-the-art MF methods: RegL1ALM (Zheng et al. 2012), CWM (Meng et al. 2013), and MoG (Meng and De la Torre 2013). Four datasets with different noises are used here. The performance of each competing method was evaluated in terms of RMSE and MAE, as the average over the 50 realizations, and reported in Table 1. Because SPL is prone to get a bad solution with the gradually increasing pace parameter, both the optimal and stable results are presented in the table. As can be seen from Table 1, the proposed MOSPL performs well in the presence of outliers and missing data.

Table 1: Performance comparison of MF methods in terms of RMSE and MAE on synthetic data. The results are averaged over 50 runs. The best result in each experiment is highlighted in bold.

Noise	Criteria	RegL1ALM	CWM	MoG	SPL(LS)		SPL(LAD)		MOSPL (LS)	MOSPL (LAD)
					Optimal	Stable	Optimal	Stable		
[-20,20]	RMSE	8.9788	2.7574	5.5509	1.7837	8.1222	1.6219	8.8914	**1.6170**	1.7227
(0,0.01)	MAE	4.5626	1.5974	2.7340	1.0068	4.2949	0.9031	4.3996	**0.8837**	0.9449
[-20,20]	RMSE	9.1648	2.8366	5.3706	1.8944	9.1874	1.8839	8.9963	1.7995	**1.7603**
(0,0.1)	MAE	4.6833	1.6663	2.6400	1.1169	4.7714	1.1108	4.3023	1.0377	**1.0045**
[-40,40]	RMSE	18.6236	4.9154	11.4176	1.5654	17.7414	1.6596	18.2794	1.5961	**1.5909**
(0,0.01)	MAE	8.4375	2.1360	5.5225	0.7503	7.8657	0.7876	8.4553	**0.7668**	0.7738
[-40,40]	RMSE	18.1732	4.9530	11.5925	1.7116	18.0802	1.7531	18.5029	1.6364	**1.5144**
(0,0.1)	MAE	8.5308	2.2056	5.7110	0.8537	8.0022	0.8363	8.5951	0.8109	**0.7374**

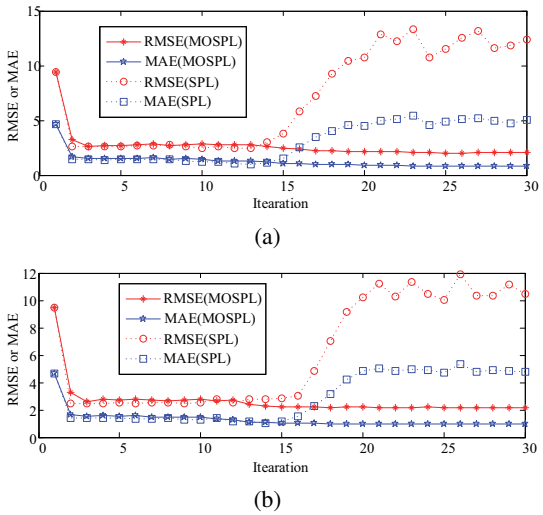


Figure 4: Tendency curves of RMSE and MAE with respect to iterations.

Table 2: Performance comparison of all competing methods in terms of MAP on Hollywood2.

ID	BatchTrain	SPL		MOSPL
		Optimal	Stable	
H01	18.775	33.719	19.583	39.646
H02	95.790	95.790	95.790	95.790
H03	71.750	71.750	71.750	71.750
H04	81.960	81.960	81.960	81.960
H05	62.787	62.786	62.786	62.786
H06	42.988	42.982	42.982	42.982
H07	16.716	34.378	11.441	37.509
H08	63.340	60.567	60.539	60.874
H09	85.751	79.454	79.276	80.279
H10	53.595	81.530	81.323	82.055
H11	35.860	38.870	38.870	38.870
H12	65.657	80.861	80.841	81.445
MAP	58.164	63.720	60.595	**64.662**

Action Recognition

The goal of action recognition is to recognize human actions in videos. Hollywood2 was collected from 69 different Hollywood movies (Marszalek, Laptev, and Schmid 2009). It contains 1707 videos belonging to 12 actions, splitting into a training set (823 videos) and a test set (884 videos). The performance was evaluated on the Hollywood2 dataset by the Mean Average Precision (MAP). The improved dense trajectory features are extracted and represented by fisher vector (Perronnin and Dance 2007). Then the spatial and temporal extension is applied to the improved dense trajectory (Lan, Li, and Hauptmann 2014). The linear kernel matrix is precalculated to accelerate the experiments. For more information, please refer to (Jiang et al. 2014b).

Table 2 lists the MAP comparison obtained by BatchTrain, SPL and MOSPL. BatchTrain represents a standard train approach in which a model is trained simultaneously using all samples. Here we use kernel SVM as the standard train approach (Jiang et al. 2014a; 2014b; 2015). As shown in Table 2, MOSPL has a better performance than SPL. MOSPL is more robust to the initialization than SPL and gets relatively satisfactory results.

Conclusion

In this study, we have proposed a novel multi-objective self-paced learning (MOSPL) method for SPL calculation. This method provides a new viewpoint to see SPL, and evidently alleviates the deficiencies existed in current SPL regime. Specifically, the MOSPL method not only largely enhances the robustness of SPL calculation to different initializations, but also outputs an entire solution path for the SPL solution wrt different pace parameters, facilitating a overall understanding to the entire SPL solution spectrum. Based on this information, some off-the-shelf multi-objective optimization techniques can be readily employed to set a rational knee point for a proper tuning of the terminating pace, which simplifies this fairly hard issue in the previous SPL research. In the future, we will explore more SPL regularizers to be suitable for various problems and pay interest in improving the multi-objective evolutionary algorithms to reduce the time complexity.

Acknowledgments

This work was supported in part by the National Nature Science Foundation of China under Grants 61273317, 61422209, 61373114, 11131006, in part by the National Program for Support of Top-Notch Young Professionals of China, in part by the Specialized Research Fund for the Doctoral Program of Higher Education under Grant 20130203110011, and in part by the Fundamental Research Funds for the Central Universities under Grant K5051202053.

References

Basu, S., and Christensen, J. 2013. Teaching classification boundaries to humans. In *AAAI*, 109–115.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *ICML*, 41–48.

Branke, J.; Deb, K.; Dierolf, H.; and Osswald, M. 2004. Finding knees in multi-objective optimization. In *PPSN*, 722–731.

Chatzis, S. P. 2014. Dynamic bayesian probabilistic matrix factorization. In *AAAI*, 1731–1737.

Coello, C. A. C.; Pulido, G. T.; and Lechuga, M. S. 2004. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation* 8(3):256–279.

Coello, C. A. C.; Van Veldhuizen, D. A.; and Lamont, G. B. 2002. *Evolutionary algorithms for solving multi-objective problems*. Norwell, MA: Kluwer.

Deb, K.; Pratap, A.; Agarwal, S.; and Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2):182–197.

Deb, K. 2001. *Multi-objective optimization using evolutionary algorithms*. New York: Wiley.

Fonseca, C. M., and Fleming, P. J. 1995. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation* 3(1):1–16.

Jiang, L.; Meng, D.; Mitamura, T.; and Hauptmann, A. G. 2014a. Easy samples first: Self-paced reranking for zero-example multimedia search. In *ACM MM*, 547–556.

Jiang, L.; Meng, D.; Yu, S.-I.; Lan, Z.; Shan, S.; and Hauptmann, A. 2014b. Self-paced learning with diversity. In *NIPS*, 2078–2086.

Jiang, L.; Meng, D.; Zhao, Q.; Shan, S.; and Hauptmann, A. G. 2015. Self-paced curriculum learning. In *AAAI*, 2694–2700.

Jin, Y., and Sendhoff, B. 2008. Pareto-based multiobjective machine learning: An overview and case studies. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 38(3):397–415.

Khan, F.; Mutlu, B.; and Zhu, X. 2011. How do humans teach: On curriculum learning and teaching dimension. In *NIPS*, 1449–1457.

Kumar, M. P.; Turki, H.; Preston, D.; and Koller, D. 2011. Learning specific-class segmentation from diverse data. In *ICCV*, 1800–1807.

Kumar, M. P.; Packer, B.; and Koller, D. 2010. Self-paced learning for latent variable models. In *NIPS*, 1189–1197.

Lan, Z.; Li, X.; and Hauptmann, A. G. 2014. Temporal extension of scale pyramid and spatial pyramid matching for action recognition. *arXiv preprint arXiv:1408.7071*.

Li, L.; Yao, X.; Stolkin, R.; Gong, M.; and He, S. 2014. An evolutionary multiobjective approach to sparse reconstruction. *IEEE Transactions on Evolutionary Computation* 18(6):827–845.

Marszalek, M.; Laptev, I.; and Schmid, C. 2009. Actions in context. In *CVPR*, 2929–2936.

Matsuyama, Y. 1996. Harmonic competition: a self-organizing multiple criteria optimization. *IEEE Transactions on Neural Networks* 7(3):652–668.

Meng, D., and De la Torre, F. 2013. Robust matrix factorization with unknown noise. In *ICCV*, 1337–1344.

Meng, D.; Xu, Z.; Zhang, L.; and Zhao, J. 2013. A cyclic weighted median method for $l1$ low-rank matrix factorization with missing entries. In *AAAI*, 704–710.

Miettinen, K. 1999. *Nonlinear multiobjective optimization*. Norwell, MA: Kluwer.

Mnih, A., and Salakhutdinov, R. 2007. Probabilistic matrix factorization. In *NIPS*, 1257–1264.

Perronnin, F., and Dance, C. 2007. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 1–8.

Qian, C.; Yu, Y.; and Zhou, Z.-H. 2013. An analysis on recombination in multi-objective evolutionary optimization. *Artificial Intelligence* 204:99–119.

Supančič, J. S., and Ramanan, D. 2013. Self-paced learning for long-term tracking. In *CVPR*, 2379–2386.

Tang, K.; Ramanathan, V.; Fei-Fei, L.; and Koller, D. 2012. Shifting weights: Adapting object detectors from image to video. In *NIPS*, 638–646.

Tang, Y.; Yang, Y.-B.; and Gao, Y. 2012. Self-paced dictionary learning for image classification. In *ACM MM*, 833–836.

Tomasi, C., and Kanade, T. 1992. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision* 9(2):137–154.

Yu, Y.; Yao, X.; and Zhou, Z.-H. 2013. On the approximation ability of evolutionary optimization with application to minimum set cover. In *IJCAI*, 3190–3194.

Zhang, Q., and Li, H. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation* 11(6):712–731.

Zhao, Q.; Meng, D.; Jiang, L.; Xie, Q.; Xu, Z.; and Hauptmann, A. G. 2015. Self-paced learning for matrix factorization. In *AAAI*, 3196–3202.

Zheng, Y.; Liu, G.; Sugimoto, S.; Yan, S.; and Okutomi, M. 2012. Practical low-rank matrix approximation under robust l_1-norm. In *CVPR*, 1410–1417.