Multiplicative Updates for Learning with Stochastic Matrices

Zhanxing Zhu, Zhirong Yang*, and Erkki Oja

Department of Information and Computer Science, Aalto University**
P.O. Box 15400, 00076, Aalto, Finland
zhanxing.zhu@gmail.com, {zhirong.yang,erkki.oja}@aalto.fi

Abstract. Stochastic matrices are arrays whose elements are discrete probabilities. They are widely used in techniques such as Markov Chains, probabilistic latent semantic analysis, etc. In such learning problems, the learned matrices, being stochastic matrices, are non-negative and all or part of the elements sum up to one. Conventional multiplicative updates which have been widely used for nonnegative learning cannot accommodate the stochasticity constraint. Simply normalizing the nonnegative matrix in learning at each step may have an adverse effect on the convergence of the optimization algorithm. Here we discuss and compare two alternative ways in developing multiplicative update rules for stochastic matrices. One reparameterizes the matrices before applying the multiplicative update principle, and the other employs relaxation with Lagrangian multipliers such that the updates jointly optimize the objective and steer the estimate towards the constraint manifold. We compare the new methods against the conventional normalization approach on two applications, parameter estimation of Hidden Markov Chain Model and Information-Theoretic Clustering. Empirical studies on both synthetic and real-world datasets demonstrate that the algorithms using the new methods perform more stably and efficiently than the conventional ones.

Keywords: nonnegative learning, stochastic matrix, multiplicative update.

1 Introduction

Nonnegativity has shown to be a useful constraint in many machine learning problems, such as Nonnegative Matrix Factorization (NMF) (e.g. [1–3]) and clustering (e.g. [4, 5]). Multiplicative updates are widely used in nonnegative learning problems because they are easy to implement, while naturally maintaining the nonnegativity of the elements to be learned after each update. Consider a matrix problem: suppose we are minimizing an objective function $\mathcal{J}(W)$ over a nonnegative matrix W, that is, $W_{ik} \geq 0$ for all (i, k). Multiplicative update is easily

* Corresponding author.

** This work is supported by the Academy of Finland (grant numbers 251170 and 140398).

J.-K. Kämäräinen and M. Koskela (Eds.): SCIA 2013, LNCS 7944, pp. 143–152, 2013.

derived from the gradient $\nabla = \partial \mathcal{J} / \partial W$. The conventional multiplicative update rule is given by

$$W_{ik} \leftarrow W_{ik} \frac{\nabla_{ik}^{-}}{\nabla_{ik}^{+}}, \tag{1}$$

where ∇^+ and ∇^- are the positive and (unsigned) negative parts of ∇, respectively, i.e. $\nabla = \nabla^+ - \nabla^-$.

When dealing with probabilistic problems such as Markov Chains, the matrix elements are discrete probabilities. Then, in addition to the nonnegativity constraint, some or all entries of such matrices must sum up to one. Such matrices are called *stochastic matrices*. Given a nonnegative matrix W, there are three typical stochasticities in nonnegative learning:

- (*left stochastic*) or column-sum-to-one: $\sum_i W_{ik} = 1$, for all k,
- (*right stochastic*) or row-sum-to-one: $\sum_k W_{ik} = 1$, for all i,
- (*vectorized stochastic*) or matrix-sum-to-one: $\sum_{ik} W_{ik} = 1$.

The multiplicative update rule in Eq. (1) cannot handle as such the stochasticity constraints. A normalization step is usually needed to enforce the unitary sum (e.g. [6–8]). However, this simple remedy may not work well with the multiplicative updates and may cause unstable or slow convergence. There is little literature on adapting the multiplicative update rules for stochastic matrices.

We focus here on how to develop multiplicative update rules for stochastic matrices in nonnegative learning in a general way. For clarity we only solve the right stochastic case in this paper, while the development procedure and discussion can easily be extended to the other two cases.

We present two approaches in Section 2. The first applies the principle of multiplicative update rules in a reparameterized space, while the second performs multiplicative updates with the relaxed constraints by using Lagrangian multipliers. While both approaches are quite generic and widely used, they have not been applied in detail on stochastic matrix learning problems. In Section 3, these methods demonstrate their advantages in terms of stability and convergence speed over the conventional normalization approach in two applications, estimating Hidden Markov Chain Models and clustering, on both synthetic and real-world data. Section 4 concludes the paper.

2 Multiplicative Updates for Stochastic Matrices

2.1 Normalization

The simplest approach for maintaining the constraints is re-normalization, which has conventionally been used for updating stochastic matrices (e.g. [6, 7]). After each multiplicative update by Eq. (1), this method normalizes the matrix: $W_{ik} \leftarrow W_{ik} / \sum_b W_{ib}$. However, as shown in Section 3.2, the normalization method often causes instability or leads the objective function to fall into some poor local minima.

2.2 Reparameterization

Alternatively, we reparameterize the right stochastic matrix W into a non-stochastic matrix U via $W_{sb} = U_{sb}/\sum_a U_{sa}$. We can then optimize over U without the stochasticity constraints, instead of W. For notation brevity, in this paper, we denote ∇^+ and ∇^- as positive and (unsigned) negative gradient parts of the objective function $\mathcal{J}(W)$ for W. For the gradients of other matrices, we use subscripts to differentiate those of W, for example, ∇^+_U and ∇^-_U.

With the reparameterization, we have $\frac{\partial W_{sb}}{\partial U_{st}} = \frac{\delta_{bt}}{\sum_a U_{sa}} - \frac{U_{sb}}{(\sum_a U_{sa})^2}$, where δ_{bt} is the Kronecker delta function. Applying the chain rule, the derivative of $\mathcal{J}$ with respect to U is

$$\frac{\partial \mathcal{J}}{\partial U_{st}} = \underbrace{\frac{\nabla^+_{st}}{\sum_a U_{sa}} + \frac{(\nabla^- U^T)_{ss}}{(\sum_a U_{sa})^2}}_{\nabla^+_U} - \underbrace{\Big(\frac{\nabla^-_{st}}{\sum_a U_{sa}} + \frac{(\nabla^+ U^T)_{ss}}{(\sum_a U_{sa})^2}\Big)}_{\nabla^-_U}. \tag{2}$$

This suggests multiplicative update rules for U and W:

$$U_{st} \leftarrow U_{st}\frac{(\nabla^-_U)_{st}}{(\nabla^+_U)_{st}} = U_{st}\frac{\nabla^-_{st} + \sum_k \nabla^+_{sk} W_{sk}}{\nabla^+_{st} + \sum_k \nabla^-_{sk} W_{sk}}, \tag{3}$$

followed by the row normalization of W, i.e. $W_{sb} = U_{sb}/\sum_a U_{sa}$.

2.3 Relaxation

The Lagrangian technique, as a well-known method to tackle constraints in optimization, can be employed here to relax the stochasticity constraints. Given the set of equality constraints $\sum_k W_{ik} - 1 = 0, i = 1, \ldots, m$ for a right stochastic matrix, the relaxed objective function can be formulated by introducing Lagrangian multipliers $\{\lambda_i\}_{i=1}^m$: $\widetilde{\mathcal{J}}(W, \{\lambda_i\}_{i=1}^m) = \mathcal{J}(W) - \sum_i \lambda_i(\sum_k W_{ik} - 1)$. Its gradient w.r.t. W is $\frac{\partial \widetilde{\mathcal{J}}}{\partial W_{ik}} = \nabla^+_{ik} - \nabla^-_{ik} - \lambda_i$. Thus, we can obtain the preliminary update rule: $W'_{ik} = W_{ik}\frac{\nabla^-_{ik} + \lambda_i}{\nabla^+_{ik}}$. Inserting the preliminary update rule into the constraint $\sum_b W'_{ib} = 1$, we have $\sum_b W_{ib}\frac{\nabla^-_{ib}}{\nabla^+_{ib}} + \lambda_i \sum_b \frac{W_{ib}}{\nabla^+_{ib}} = 1$. Solving the equation gives $\lambda_i = \frac{1 - \sum_b W_{ib}\nabla^-_{ib}/\nabla^+_{ib}}{\sum_b W_{ib}/\nabla^+_{ib}}$. Putting them back to the preliminary rule, we have $W'_{ik} = W_{ik}\frac{\nabla^-_{ik} A_{ik} + 1 - B_{ik}}{\nabla^+_{ik} A_{ik}}$, where $A_{ik} = \sum_b \frac{W_{ib}}{\nabla^+_{ib}}$ and $B_{ik} = \sum_b W_{ib}\frac{\nabla^-_{ib}}{\nabla^+_{ib}}$. There is a negative term $-B_{ik}$ in the numerator, which may cause negative entries in the updated W. To overcome this, we apply the "moving term" trick [9–12] to resettle B_{ik} to the denominator, giving the final update rule

$$W_{ik} \leftarrow W_{ik}\frac{\nabla^-_{ik} A_{ik} + 1}{\nabla^+_{ik} A_{ik} + B_{ik}}. \tag{4}$$

We call the above update rule *iterative Lagrangian solution* [9] for the nonnegative learning problem with stochasticity constraints.

The final update rule reveals that there are two forces, weighted by A_{ik}, in the update procedure of W. One keeps the original multiplicative update rule $W_{ik} \leftarrow W_{ik}\nabla^-_{ik}/\nabla^+_{ik}$, and the other steers the matrix to approach the manifold specified by the stochasticity constraints. Incorporating the two forces into one update rule can ultimately optimize the matrix W and fulfill the constraints approximately as well.

Moreover, if there exists a certain auxiliary upper-bounding function (see e.g. [1, 9]) for $\mathcal{J}(W)$ and minimization of the auxiliary function guarantees that $\mathcal{J}(W)$ monotonically decreases under multiplicative updates, we can easily design an augmented auxiliary function for $\widetilde{\mathcal{J}}(W, \{\lambda_i\})$ by the principle in [9] to keep the relaxed objective function descending under the update rule (4). This can theoretically ensure the objective function convergence of multiplicative update rules for stochastic matrices.

3 Applications

3.1 Parameter Estimation in HMM

A *Hidden Markov Model* (HMM) chain is widely used for dealing with temporal or spatial structures in a random process. In the basic chain model, $s(i)$ denotes the hidden state at time i, taking values from $\mathcal{S} = \{1, 2, \ldots, r\}$, and its corresponding observed output is denoted by $x(i)$. The state joint probability is $S(s_1, s_2) = P(s(i) = s_1, s(i+1) = s_2)$. The factorization of the joint probability of two consecutive observations $\widehat{X}(x_1, x_2) = P(x(i+1) = x_2, x(i) = x_1)$, can be expressed as $\widehat{X}(x_1, x_2) = \sum_{s_1,s_2} P(x_1|s_1)S(s_1, s_2)P(x_2|s_2)$. For the discrete and finite random variable alphabet $|\mathcal{X}| = m$, $\widehat{X}$ forms an $m \times m$ vectorized stochastic matrix. Then, the above factorization can be expressed in matrix form: $\widehat{X} = PSP^T$, where $\widehat{X} \in \mathbb{R}_+^{m\times m}$, $P \in \mathbb{R}_+^{m\times r}$ and $S \in \mathbb{R}_+^{r\times r}$.

The matrix $\widehat{X}$ is generally unknown in practice. Based on the observations, $x(1), \ldots, x(N)$, $\widehat{X}$ is usually approximated by the joint normalized histogram

$$X(x_1, x_2) = \frac{1}{N-1}\sum_{i=1}^{n-1} \delta(x(i) = x_i)\delta(x(i+1) = x_2), \tag{5}$$

where δ is Dirac delta function. With the Euclidean distance metric, the NMF optimization problem can be formulated as $\min_{P\geq 0, S\geq 0} \mathcal{J}(P, S) = \frac{1}{2}\|X - PSP^T\|^2$, subject to $\sum_i P_{is} = 1, \sum_{st} S_{st} = 1$ for $s = 1, \ldots, r$.

Lakshminarayanan and Raich recently proposed an iterative algorithm called NNMF-HMM for the above optimization problem [13]. Their method applies normalization after each update of P and S, which is in turn based on matrix pseudo-inversion and truncation of negative entries.

Here we solve the HMM problem by multiplicative updates. First, we decompose the gradients of $\mathcal{J}$ to positive and negative parts: $\nabla_P = \nabla^+_P - \nabla^-_P$ and $\nabla_S = \nabla^+_S - \nabla^-_S$ where $\nabla^+_P = PSP^TPS^T + PS^TP^TPS$, $\nabla^-_P = XPS^T + X^TPS$, $\nabla^+_S = P^TPSP^TP$, and $\nabla^-_S = P^TXP$. Multiplicative update rules for P and S

Table 1. Comparison of convergence time (in seconds) and final objective on (top) synthetic dataset and (bottom) English words dataset. Results are in the form mean ± standard deviation.

Methods	NNMF-HMM	norm.-HMM	repa.-HMM	rela.-HMM
Time	3.52 ± 2.10	1.00 ± 0.44	1.84 ± 1.04	1.89 ± 0.56
Objective	$4 \times 10^{-3} \pm 1 \times 10^{-9}$	$7 \times 10^{-6} \pm 9 \times 10^{-9}$	$4 \times 10^{-5} \pm 9 \times 10^{-5}$	$7 \times 10^{-6} \pm 9 \times 10^{-9}$

Methods	NNMF-HMM	norm.-HMM	repa.-HMM	rela.-HMM
Time	0.15 ± 0.09	2.94 ± 1.43	2.53 ± 1.28	2.89 ± 0.62
Objective	$6 \times 10^{-3} \pm 2 \times 10^{-3}$	$8 \times 10^{-4} \pm 0.2 \times 10^{-4}$	$9 \times 10^{-4} \pm 1 \times 10^{-4}$	$8 \times 10^{-4} \pm 0.2 \times 10^{-4}$

are then formed by inserting these quantities to the rules in Section 2. Note that P is left stochastic and S is vectorized stochastic. Some supporting formulae for these two cases are given in the appendix.

We have compared the multiplicative algorithms against the implementation in [13] on both synthetic and real-world data. First, a data set was generated under the same settings as in [13]. That is, we constructed an HMM with $r = 3$ states and a transition matrix $X(s_2|s_1) = \begin{bmatrix} 0 & 0.9 & 0.1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$. Next, $x^{'}$ was sampled based on the conditional distributions $x^{'}|s = 1 \sim \mathcal{N}(11, 2)$, $x^{'}|s = 2 \sim \mathcal{N}(16, 3)$ and $x^{'}|s = 3 \sim \text{Uniform}(16, 26)$. Then x was generated by rounding $x^{'}$ to its nearest integer. We sampled $N = 10^5$ observations and calculated the joint probability normalized histogram matrix X by Eq. (5).

The iterative algorithms stop when $\frac{\|P - P_{\text{old}}\|_F}{\|P\|_F} < \epsilon$ and $\frac{\|S - S_{\text{old}}\|_F}{\|S\|_F} < \epsilon$, where $\|\cdot\|_F$ stands for Frobenius norm and $\epsilon = 10^{-6}$. We run the algorithms 50 times with different random initialization for P and S. All the experiments are performed on a PC with 2.83GHz CPU and 4GB RAM. Results are given in Table 1 (top), from which we can see that the three methods with multiplicative updates are faster and give better objectives than NNMF-HMM.

For one group of random implementations, we compare their convergence speeds, i.e, objective value as a function of time, in Fig. 1 (top). We can observe that NNMF-HMM algorithm gets stuck into a high objective. The other three algorithms converge much faster and have similar convergence speed while satisfying the stochasticity constraints.

In a real-world application, 1510 commonly used English words have been selected for exploring word-building rules in English by HMM modeling. We treat each word as a sequence of letters (observations) and consider two consecutive letters as two successive observations. We calculated the joint normalized histogram matrix $X \in \mathbb{R}_+^{26 \times 26}$ by Eq. (5). We empirically set the number of hidden states $r = 6$ in our experiment. Table 1 (bottom) presents the convergence time and objectives based on 50 independent runs. Fig. 1 (bottom) shows the convergence speed of the four algorithms on of the runs, exhibiting the same advantage of multiplicative updates over NNMF-HMM which was demonstrated for the synthetic data set.

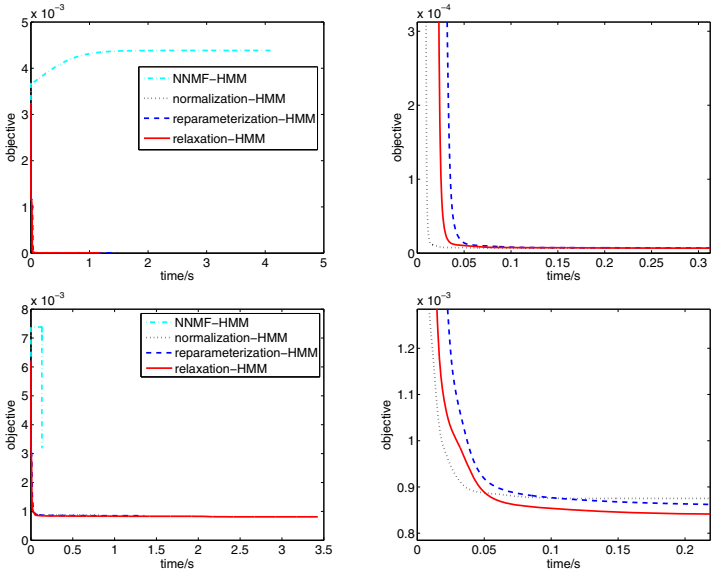

Fig. 1. Objective evolution of the compared algorithms in HMM parameter estimation on (top) synthetic data set and (bottom) English words. The right sub-figures are the zoom-in plot for early epochs.

3.2 Nonparametric Information Theoretic Clustering

Next, we demonstrate another application of the proposed methods to a nonnegative learning problem beyond matrix factorization: *Nonparametric Information Clustering* (NIC) [4]. This recent clustering approach is based on maximizing the mutual information between data points and formulates a clustering criterion analogous to k-means, but with locality information enhanced.

Given a set of data points $\{x_1, x_2, \ldots, x_N\}$ and the number of clusters n_C, denote the cluster of x_i by $C(x_i) = c_i$, and the number of points assigned to the kth cluster by n_k. NIC minimizes the following score as the clustering criterion: $\mathcal{J}_{NIC}(C) = \sum_k \frac{1}{n_k - 1} \sum_{i \neq j | c_i = c_j = k} \log \|x_i - x_j\|^2$. Faivishevsky and Goldberger [4] proposed a greedy algorithm to optimize the score function $\mathcal{J}_{NIC}(C)$. Initialized by a random partition, the greedy algorithm repeatedly picks a data point and determines its cluster assignment that maximizes the score function $\mathcal{J}_{NIC}(C)$. Note that for each data point, the greedy algorithm has to recompute the score function, which is expensive for large data sets.

To overcome this issue, we reformulate the hard clustering problem into a soft version such that differential calculus in place of combinatorial optimization can be used. Moreover, we use batch updates of cluster assignments for all data points. First, we relax the score function $\mathcal{J}^S_{NIC} = \sum_k p(k) \sum_{ij} p(i|k) p(j|k) D_{ij}$ by using probabilities, where $p(k) = \sum_a p(k|a) p(a)$ is the cluster probability, and $p(i|k)$ and $p(j|k)$ are the observation probabilities in cluster k. D abbreviates the dissimilarity matrix for the whole data set, with $D_{ij} = \log \|x_i - x_j\|^2$.

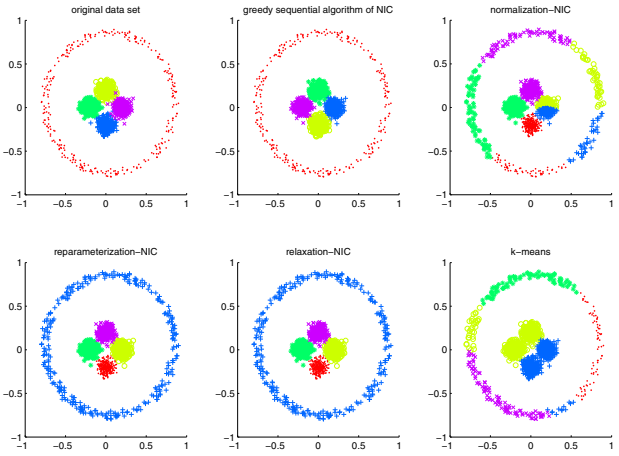

Fig. 2. Clustering analysis for a synthetic dataset (top left) using five algorithms: greedy algorithm (top middle), normalization (top right), reparameterization (bottom left), relaxation (bottom middle) and k-means (bottom right)

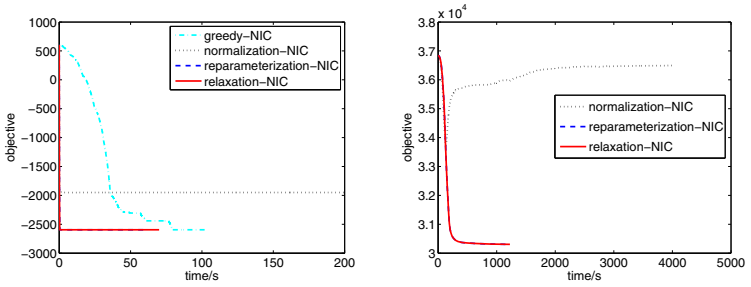

Fig. 3. Objective evolution of a typical trial of compared NIC algorithms: (left) synthetic dataset and (right) pen digit dataset

In clustering, our aim is to optimize conditional probability $p(k|i)$, the soft assignment to kth cluster for data point x_i. Applying the Bayes theorem, we have $p(i|k) = \frac{p(k|i)p(i)}{\sum_a p(k|a)p(a)} = \frac{p(k|i)}{\sum_a p(k|a)}$, where we assume a uniform prior for data points x_i, $p(i) = \frac{1}{N}$.

For notational simplicity, we define the assignment matrix $W_{ik} = p(k|i)$. The NIC model can thus be reformulated as the following nonnegative optimization problem: $\min_{W \geq 0} \mathcal{J}^S_{NIC}(W) = \frac{1}{N} \sum_k \frac{(W^T DW)_{kk}}{\sum_a W_{ak}}$, subject to $\sum_b W_{ib} = 1$ for all i. The gradient to W is $\frac{\partial \mathcal{J}^S_{NIC}}{\partial W_{ik}} = \frac{1}{N} \left(\frac{2(DW)_{ik}}{\sum_a W_{ak}} - \frac{(W^T DW)_{kk}}{(\sum_a W_{ak})^2} \right)$, with which we can construct multiplicative updates for W as described in Section 2. The iterative algorithms terminate when $\frac{\|W - W_{\text{old}}\|_F}{\|W\|_F} < 10^{-5}$.

We first generated a synthetic data set with five clusters: four "pie" clusters and a circular cluster surrounding the pies (Fig. 2, top left sub-figure). Each cluster has 300 data points. A typical run of the compared NIC algorithms is

Table 2. Comparison of convergence time and final objective on synthetic clusters

Methods	greedy	norm.-NIC	repa.-NIC	rela.-NIC	k-means
Time (s)	80.83 ± 17.75	535.11 ± 37.25	7.22 ± 7.06	6.98 ± 6.94	0.03 ± 0.05
Objective	$-2.60 \times 10^3 \pm 0$	$-2.53 \times 10^3 \pm 204$	$-2.60 \times 10^3 \pm 0.01$	$-2.60 \times 10^3 \pm 0.01$	$-1.79 \times 10^3 \pm 490$
Purity	0.99 ± 0	0.97 ± 0.06	0.99 ± 0	0.99 ± 0	0.72 ± 0.14

Table 3. Clustering performance for the five compared methods on datasets i) ecoli, ii) glass, iii) parkinsons, iv) iris, v) wine, vi) sonar, and vii) pima. Each cell shows the mean and standard deviation of *clustering purity (the first line)* and *converged time (the second line, in seconds)*. The abbreviations *norm.*, *repa.*, and *rela.* stand for normalization, reparameterization, and relaxation, respectively. Bold numbers indicate the best in each row.

Data	greedy	norm.	repa.	rela.	rela.+greedy
i)	**0.80±0.02**	0.51±0.02	0.80±0.03	0.80±0.03	0.80±0.03
	4.05±1.06	3.83±1.33	**1.70±2.12**	2.1±3.22	2.63±1.14
ii)	**0.64±0.05**	0.44±0.01	**0.64±0.05**	**0.64±0.05**	**0.64±0.05**
	1.29±0.28	1.20±0.50	0.48±0.31	**0.47±0.32**	0.89±0.42
iii)	**0.75±0.00**	**0.75±0.00**	**0.75±0.00**	**0.75±0.00**	**0.75±0.00**
	0.42±0.13	0.40±0.03	0.55±0.07	0.49±0.05	**0.34±0.11**
iv)	0.72±0.04	0.47±0.05	0.72±0.07	0.72±0.07	**0.73±0.06**
	0.26±0.09	0.74±0.17	0.20±0.29	**0.16±0.10**	0.18±0.11
v)	0.77±0.16	0.40±0.01	0.65±0.13	0.65±0.13	**0.81±0.18**
	0.58±0.21	0.56±0.30	0.56±0.11	0.50±0.11	**0.42±0.16**
vi)	**0.55±0.00**	0.53±0.01	0.55±0.02	0.55±0.02	**0.55±0.00**
	0.32±0.04	0.41±0.00	0.57±0.02	0.54±0.00	**0.30±0.05**
vii)	**0.67±0.00**	0.65±0.00	0.66±0.01	0.66±0.01	0.67±0.01
	19.65±5.67	5.44±0.38	**4.46±0.17**	4.60±0.16	12.74±6.42

shown in Fig. 3 (left). We can see that the normalization method gets stuck and returns a high objective. The other three methods can find better NIC scores, whereas the greedy algorithm requires much more time to converge. By contrast, reparameterization and relaxation achieve nearly the same efficiency. We have repeated the compared algorithms 50 times with different random initializations. The means and standard deviations of the running time and objective are shown in Table 2, which again shows the efficiency advantage of the algorithms using the proposed methods.

We validate the converged estimates by checking their corresponding clustering results. From Fig. 2, k-means and normalization methods split the large circle into several parts and fail to identify the four pies in the middle. The NIC algorithms can correctly identify the five clusters. The clustering performance is quantified by the *purity* measurement, which is calculated by $\frac{1}{N}\sum_k \max_{1\le l\le q} n_k^l$, with n_k^l the number of samples in kth cluster that belong to true class l. A larger purity usually corresponds to better clustering performance. The last row in Table 2 indicates that the new algorithms are not only fast but are able to produce satisfactory clustering results.

We have also tested the compared algorithms for NIC on several real-world datasets selected from the UCI Machine Learning Repository[1]. The results are shown in Table 3. We can see that in six out of seven datasets, the reparameterization and relaxation methods can achieve similar purity as the greedy algorithm

[1] `http://archive.ics.uci.edu/ml/`

Table 4. Comparison of convergence time, final objective and purity on pen digits dataset

Methods	greedy	norm.-NIC	repa.-NIC	rela.-NIC	k-means
Time (s)	–	4509 ± 328	1276 ± 41	1274 ± 38	0.7 ± 0.4
Objective	–	36445 ± 51	30457 ± 177	30457 ± 177	30812 ± 285
Purity	–	0.37 ± 0.001	0.73 ± 0.04	0.73 ± 0.04	0.68 ± 0.04

but with less learning time. For the *wine* dataset, though the purities obtained by the proposed methods are inferior to those by the greedy algorithm, we can use the relaxation (or reparameterization) method as initialization, followed by the greedy approach. This way we can achieve satisfactory purity but still with reduced learning time. Surprisingly, this hybrid method produces even better clustering purities for the datasets *iris* and *wine*. By contrast, the normalization method is again unstable and its clustering purity is much lower than the others for four out of seven datasets.

The efficiency advantage brought by the proposed method is clearer for larger datasets, e.g. *pima*. Furthermore, we applied the compared methods to an even larger real world data set, "Pen-Based Recognition of Handwritten Digits Data Set" also from the UCI Repository. This data set contains 10992 samples, each of which is 16-dimensional. In clustering the digits, the two proposed methods can achieve purity 0.73 except the normalization approach (purity 0.37), see Table 4. However, the greedy algorithm needs more than one day to converge, while the proposed reparameterization and relaxation multiplicative updates can converge within 20 minutes, as compared with the normalization method in the right panel of Fig. 3.

4 Conclusions

We have introduced two methods, reparameterization and relaxation, for multiplicatively updating stochastic matrices. Both methods share the property of good convergence speed and can fulfill the stochasticity constraints. They outperform the conventional normalization method in terms of stability. We have applied the proposed algorithms to two applications, parameter estimation in HMM and NIC clustering. Experimental results indicate that the proposed two methods are advantageous for developing multiplicative algorithms for nonnegative learning with stochastic matrices.

In future work, we would like to investigate the theoretical proof on the convergence of reparameterization and relaxation methods, which can be handled by constructing proper auxiliary functions. Moreover, the connection between the proposed methods and the Dirichlet process prior deserves further investigation.

A Formulae for Left and Vectorized Stochastic Matrices

In reparameterization method, for left stochastic matrix,

$$\frac{\partial \mathcal{J}}{\partial U_{st}} = \frac{\nabla^+_{st}}{\sum_a U_{at}} + \frac{(U^T \nabla^-)_{tt}}{(\sum_a U_{at})^2} - \Big(\frac{\nabla^-_{st}}{\sum_a U_{at}} + \frac{(U^T \nabla^+)_{tt}}{(\sum_a U_{at})^2} \Big), \tag{6}$$

for vectorized stochastic matrix,

$$\frac{\partial \mathcal{J}}{\partial U_{st}} = \frac{\nabla^+_{st}}{\sum_{ab} U_{ab}} + \frac{\sum_{ab} \nabla^-_{ab} U_{ab}}{(\sum_{ab} U_{ab})^2} - \Big(\frac{\nabla^-_{st}}{\sum_{ab} U_{ab}} + \frac{\sum_{ab} \nabla^+_{ab} U_{ab}}{(\sum_{ab} U_{ab})^2} \Big). \tag{7}$$

In relaxation method, for left stochastic matrix,

$$A_{ik} = \sum_a \frac{W_{ak}}{\nabla^+_{ak}}, \quad B_{ik} = \sum_a W_{ak} \frac{\nabla^-_{ak}}{\nabla^+_{ak}}, \tag{8}$$

and for vectorized stochastic matrix,

$$A_{ik} = \sum_{ab} \frac{W_{ab}}{\nabla^+_{ab}}, \quad B_{ik} = \sum_{ab} W_{ab} \frac{\nabla^-_{ab}}{\nabla^+_{ab}}. \tag{9}$$

References

1. Dhillon, I.S., Sra, S.: Generalized nonnegative matrix approximations with bregman divergences. In: Advances in Neural Information Processing Systems, vol. 18, pp. 283–290 (2006)
2. Choi, S.: Algorithms for orthogonal nonnegative matrix factorization. In: Proceedings of IEEE International Joint Conference on Neural Networks, pp. 1828–1832 (2008)
3. Cichocki, A., Zdunek, R., Phan, A.H., Amari, S.: Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis. John Wiley (2009)
4. Faivishevsky, B., Goldberger, J.: A nonparametric information theoretic clustering algorithm. In: The 27th International Conference on Machine Learning (2010)
5. Jin, R., Ding, C., Kang, F.: A probabilistic approach for optimizing spectral clustering. In: Advances in Neural Information Processing Systems, pp. 571–578 (2005)
6. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401, 788–791 (1999)
7. Ding, C., Li, T., Peng, W.: On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. Computational Statistics and Data Analysis 52(8), 3913–3927 (2008)
8. Mørup, M., Hansen, L.: Archetypal analysis for machine learning. In: IEEE International Workshop on Machine Learning for Signal Processing (MLSP), pp. 172–177. IEEE (2010)
9. Yang, Z., Oja, E.: Linear and nonlinear projective nonnegative matrix factorization. IEEE Transaction on Neural Networks 21(5), 734–749 (2010)
10. Yang, Z., Oja, E.: Unified development of multiplicative algorithms for linear and quadratic nonnegative matrix factorization. IEEE Transactions on Neural Networks 22(12), 1878–1891 (2011)
11. Yang, Z., Oja, E.: Quadratic nonnegative matrix factorization. Pattern Recognition 45(4), 1500–1510 (2012)
12. Yang, Z., Oja, E.: Clustering by low-rank doubly stochastic matrix decomposition. In: International Conference on Machine Learning (ICML) (2012)
13. Lakshminarayanan, B., Raich, R.: Non-negative matrix factorization for parameter estimation in hidden markov models. In: IEEE International Workshop on Machine Learning for Signal Processing (MLSP), pp. 89–94 (2010)