Supervised Tensor Learning

Dacheng Tao[1], Xuelong Li[1], Weiming Hu[2], Stephen Maybank[1], and Xindong Wu[3]

1. School of Computer Science and Information Systems, Birkbeck College, University of London, UK
2. National Lab of Pattern Recognition, Institute of Automation, Chinese Academy, P. R. China
3. Department of Computer Science, University of Vermont, USA.
{dacheng, xuelong, sjmaybank}@dcs.bbk.ac.uk, wmhu@nlpr.ia.ac.cn, xwu@cs.uvm.edu

Abstract

This paper aims to take general tensors as inputs for supervised learning. A supervised tensor learning (STL) framework is established for convex optimization based learning techniques such as support vector machines (SVM) and minimax probability machines (MPM). Within the STL framework, many conventional learning machines can be generalized to take nth–order tensors as inputs. We also study the applications of tensors to learning machine design and feature extraction by linear discriminant analysis (LDA). Our method for tensor based feature extraction is named the tenor rank–one discriminant analysis (TR1DA). These generalized algorithms have several advantages: 1) reduce the *curse of dimension* problem in machine learning and data mining; 2) avoid the failure to converge; and 3) achieve better separation between the different categories of samples. As an example, we generalize MPM to its STL version, which is named the tensor MPM (TMPM). TMPM learns a series of tensor projections iteratively. It is then evaluated against the original MPM. Our experiments on a binary classification problem show that TMPM significantly outperforms the original MPM.

1. Introduction

Supervised learning [1–3] is an important topic in machine learning and data mining and their applications. Generally, only one–dimensional vectors are accepted as inputs by existing supervised learning machines, such as support vector machines (SVM) [2], and minimax probability machines (MPM) [3]. However, in the real world many data items such as images are represented by 2nd–order or high–order tensors rather than the one dimensional vectors. We develop a supervised tensor learning (STL) framework in order to apply convex optimization based supervised learning techniques to tensor data.

The motivation for the STL framework comes from the following observations.

- Data structures in the real world: in many computer vision applications objects are represented as 2nd–order, 3rd–order, or higher–order tensors. For example, grey level face images [5] are usually represented as 2nd–order tensors; an attention region [6], which is used in natural image understanding [4], is represented as a 3rd–order tensor; another 3rd–order tensor example is the bi–level down sampled silhouette images used to represent human gait [7]; and moreover, video sequences [8] are 4th–order tensors.

Data structures for conventional learning methods are restricted to one–dimensional vectors, i.e. the 1st–order tensors as inputs. Learning machines such as MPM and SVM obtain top–level performances in machine learning and data mining; however, they have to convert many data items naturally represented by high –order tensors to one–dimensional vectors in order to comply with their input requirements. As a result of this conversion, much useful information in the original data is destroyed, leading to an increased number of classification errors. In addition, the conversion to vector input usually leads to the so called *curse of dimension* problem since the dimension of the feature space becomes much larger than the number of training samples. If the data are represented in their natural, high–order tensors, then this *curse of dimension* problem is usually reduced.

The main contributions of this paper are as follows.

- Tensor–plane (a set of projection orientations): by an iterative scheme, the STL–based classifier's optimal tensor–plane can be obtained. Our method aims to approach an optimization criterion iteratively; and this criterion can be any convex functions, such as, the margin maximization (SVM) and the probability of correct classification of future data maximization (MPM).
- Kernelization: a kernel space representation of general tensor inputs is developed.
- Tensor MPM (TMPM): as an example, MPM is generalized to its STL version, named the tensor MPM (TMPM). TMPM is then compared with MPM on a sample image classification problem. In addition, TMPM is kernelized.

Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05)
1550-4786/05 $20.00 © 2005 IEEE

- Stability analysis: in a series of experiments we observe that TMPM converges quickly within a very few iterations (around 5). Furthermore, unlike many other learning models, it is stable under different choices of initial values [11], i.e. it does not become trapped in a *local minimum*.
- Feature extraction: we study tensor–based feature extraction based on linear discriminant analysis (LDA). In this effort [18], we inherit the merits from both the defined differential scatter based discriminant criterion (DSDC) and the rank–one (or rank–*n*) tensor decomposition; as a result DSDC can extract features from tensors. The new established feature extraction algorithm is named the tenor rank–one discriminant analysis (TR1DA).

2. Supervised tensor learning (STL)

This section introduces some basic concepts in convex optimization based supervised learning; establishes a STL framework for general tensors with an iterative procedure; and also deduces a kernel extension of the STL framework for high–order tensor pattern classification using convex optimization.

2.1. Convex optimization based learning

The intrinsic connections between machine learning methods and convex optimization techniques have been well studied. Examples include quadratic programming [1] in SVM and second order cone programming [3] in MPM. More recently, some convex optimization tools, such as semi–definite programming [12], have become popular because they are simple and yet powerful. Below, x_i ($1 \le i \le n_x$) and y_i ($1 \le i \le n_y$) are tensors belonging to the positive class and the negative class respectively.

A. Support Vector Machines (SVM)

SVM [2] is an effective bi–category classification algorithm with sound theoretical foundations and a good generalization ability. It aims to maximize the margin between the positive and negative samples as:

$$\begin{cases} \min\left(w^T w\right) \\ s.t. \begin{array}{l} w^T x_i + b \ge +1, 1 \le i \le n_x \\ w^T y_j + b \le -1, 1 \le j \le n_y . \end{array} \end{cases} \tag{1}$$

where w is the optimal projection orientation for bi-category classification.

B. Minimax probability machines (MPM)

MPM maximizes the probability of correct classification for future data, or alternatively, minimizes the maximum of the Mahalanobis distances of the positive and negative samples. It is a second order cone programming:

$$\begin{cases} \max\limits_{\alpha, w \ne 0, b} \left(\alpha\right) \\ s.t. \begin{array}{l} w^T \overline{x} - b \ge \kappa\left(\alpha\right)\sqrt{w^T \Sigma_x w} \\ b - w^T \overline{y} \ge \kappa\left(\alpha\right)\sqrt{w^T \Sigma_y w} \end{array} \end{cases} \tag{2}$$

where $\kappa\left(\alpha\right) = \sqrt{\frac{\alpha}{1-\alpha}}$, $\overline{x} = \frac{1}{n_x}\sum_{i=1}^{n_x} x_i$, $\overline{y} = \frac{1}{n_y}\sum_{j=1}^{n_y} y_j$, Σ_x and Σ_y are the covariance matrices of x and y , and w is the optimal projection orientation for classification. MPM has a solid theoretical foundation based on the powerful Marshall and Olkin's theorem [15]. MPM can outperform SVM consistently [3].

2.2. Supervised tensor learning framework

Motivated by the successes of convex optimization based learning algorithms and the effective image representation by general tensors [13] [14], we propose a tensor framework for this type of learning algorithm. Generally, convex optimization based learning can be written as:

$$\begin{cases} \max\limits_{w \ne 0} f\left(w, b, \theta\right) \\ s.t. \begin{array}{l} C_i^x\left(w^T x_i, b, \theta\right) \ge \eta_i, 1 \le i \le n_x \\ C_j^y\left(w^T y_j, b, \theta\right) \ge \varsigma_j, 1 \le j \le n_y \end{array} \end{cases} \tag{3}$$

where C_i^x and C_j^y are linear or quadratic constrained functions. It is not difficult to show that (1) and (2) can be unified into (3). The optimal hyper–plane for the classification can thus be written as:

$$g\left(z\right) = \operatorname{sign}\left(w^T z - b\right) . \tag{4}$$

Here, if $g\left(z\right) = +1$, then z belongs to the positive class; otherwise, it belongs to the negative class.

Based on (3) and (4), we can use the tensor idea to re–represent convex optimization learning with n^{th}–order tensors as inputs by:

$$\begin{cases} \max\limits_{w \ne 0} f\left(w_1, w_2, \ldots, w_n, b, \theta\right) \\ s.t. \begin{array}{l} C_i^x\left(x_i \times_1 w_1 \times_2 w_2 \times \ldots \times_n w_n, b, \theta\right) \ge \eta_i, 1 \le i \le n_x \\ C_j^y\left(y_j \times_1 w_1 \times_2 w_2 \times \ldots \times_n w_n, b, \theta\right) \ge \varsigma_j, 1 \le j \le n_y \end{array} \end{cases} \tag{5}$$

and the optimal classification function is:

$$g\left(z\right) = \operatorname{sign}\left(z \times_1 w_1 \times_2 w_2 \times \ldots \times_n w_n - b\right) \tag{6}$$

where $\times_k w_k$ is the mode–*k* product in tensor analysis [14]. It is defined as: $B = A \times_k w_k$. Here, we use a set of projection orientations $w_k |_{k=1}^{n}$ (*optimal tensor plane*) for classification.

Table 1. Alternating procedure (AP) for STL.

Input:	The positive n^{th}–order tensor points x_i, $1 \le i \le n_x$ and the negative n^{th}–order tensor points y_i, $1 \le i \le n_y$.
Output	A supervised tensor classifier based on the optimal tensor plane and the following bias: $g(z) = \operatorname{sign}\left(z \times_1 w_1 \times_2 w_2 \times \ldots \times_n w_n - b\right)$, where $\times_i$ denotes the mode–*i* product [14].
Step 1.	Initialization: Set m as the n^{th}–order tensor defined by $m = \frac{1}{2n_x}\sum_{i=1}^{n_x} x_i + \frac{1}{2n_x}\sum_{j=1}^{n_y} y_j$.
Step 2.	Estimate the initial values for w_k ($1 \le k \le n$), according to $\min_{w_k, 1 \le k \le n} \left\| m - \lambda \times_1 w_1^0 \times_2 w_2^0 \times \ldots \times_n w_n^0 \right\|_F$, where $\lambda = m \times_1 w_1^0 \times_2 w_2^0 \times \ldots \times_n w_n^0$.
Step 3.	For $p = 1$, 2, …, *N* or (until converged), do: For $l = 1,\ldots,n$, do: $\begin{cases} \max_{w_l^p \neq 0} f\left(w_l^p, b_l, \theta\right) \\ C_i^x\left(\left(w_l^p\right)^T\left(x_i \times_{-l} w_{-l}^{p-1}\right), b_l, \theta\right) \ge \eta_i \\ s.t. C_j^y\left(\left(w_l^p\right)^T\left(y_j \times_{-l} w_{-l}^{p-1}\right), b_l, \theta\right) \ge \varsigma_j \\ 1 \le i \le n_x, 1 \le j \le n_y \end{cases}$ where C_i^x and C_j^y are constrained functions. Here, $x_i \times_{-l} w_{-l}^{p-1}$ is defined by: $x_i \times_1 w_1^{p-1} \times \ldots \times_{l-1} w_{l-1}^{p-1} \times_{l+1} w_{l+1}^{p-1} \ldots \times_n w_n^{p-1}$.
Step 4.	Find the optimal bias b for all the projected positive and negative training samples by: $\begin{cases} \max f(b, \theta) \\ C_i^x\left(x_i \times_1 w_1^N \times_2 w_2^N \times \ldots \times_n w_n^N, b, \theta\right) \ge \eta_i \\ s.t. C_j^y\left(y_j \times_1 w_1^N \times_2 w_2^N \times \ldots \times_n w_n^N, b, \theta\right) \ge \varsigma_j \\ 1 \le i \le n_x, 1 \le j \le n_y \end{cases}$

According to the definition of this novel tensor oriented supervised learning framework, we can directly represent objects in their original format for machine learning and data mining applications in computer vision etc. However, so far, no closed–form solutions to (5) are known. In the next section, an alternating approach is developed to solve (5).

2.3. Alternating approach

In our tensor based supervised learning framework, we define and study an *optimal tensor plane* ($w_i |_{i=1}^{n}$, a set of projection orientations) to approach the objective function with some constrained functions, which are related to the positive and negative samples. A close–form solution does not exist, so we develop an alternating procedure (AP) for the framework, listed in Table 1. In this table, Steps 2 and 3 finds the *optimal tensor plane* $w_i |_{i=1}^{n}$ and Step 4 finds the optimal bias *b*.

2.4. Kernelization

As for many other machine learning and data mining algorithms, the STL framework can be kernelized directly. The kernelized model is:

$$\begin{cases} \max_{w \neq 0} f\left(\varphi(w_1), \ldots, \varphi(w_n), b, \theta\right) \\ C_i^x\left(\varphi(x_i) \times_1 \varphi(w_1) \times \ldots \times_n \varphi(w_n), b, \theta\right) \ge \eta_i, 1 \le i \le n_x \\ s.t. \\ C_j^y\left(\varphi(y_j) \times_1 \varphi(w_1) \times \ldots \times_n \varphi(w_n), b, \theta\right) \ge \varsigma_j, 1 \le j \le n_y \end{cases} \quad (7)$$

and the optimal classification function is:

$$g(z) = \operatorname{sign}\left(\varphi(z) \times_1 \varphi(w_1) \times \ldots \times_n \varphi(w_n) - b\right) \quad (8)$$

where $\times_k$ is the mode–*k* product in tensor analysis [14] and $\varphi(\cdot)$ is a nonlinear mapping function.

The key issue in kernelization is how to calculate $\varphi(x_i) \times_1 \varphi(w_1) \times \ldots \times_n \varphi(w_n)$. The answer is to perform tensor rank–one decomposition, i.e. x can be decomposed into $x_i = \lambda_i \times_1 w_1^i \times \ldots \times_n w_n^i = \lambda_i \prod_{j=1}^{n} \times_j w_j^i$. In the kernel space the decomposition can be written as: $\varphi(x_i) = \lambda_i \times_1 \varphi(w_1^i) \times \ldots \times_n \varphi(w_n^i) = \lambda_i \prod_{j=1}^{n} \times_j \varphi(w_j^i)$.

With this representation, the kernel trick can be utilized directly:

$$\begin{aligned} &\varphi(x_i) \times_1 \varphi(w_1) \times \ldots \times_n \varphi(w_n) = \lambda_i \prod_{k=1}^{n} \times_k \varphi(w_k^i) \prod_{l=1}^{n} \times_l \varphi(w_l) \\ &= \lambda_i \prod_k \prod_l \delta_k^l \varphi(w_k^i) \cdot \varphi(w_l) = \lambda_i \prod_k \varphi(w_k^i) \cdot \varphi(w_k) \\ &= \lambda_i \prod_k K^k(w_k^i, w_k) \end{aligned} \quad (9)$$

where $K^k(\cdot)$ is a typical kernel function, such as the Gaussian function and the polynomial function. By this kind of representation, we can have n kernel functions for the tensor learning framework in the kernel space with the n^{th}–order tensors as inputs.

The alternating procedure can also be kernelized because the l^{th} kernel tensor projection orientation is $\varphi(w_l) = \sum_{i=1}^{n_x+n_y} \alpha_l^i \varphi(w_l^i)$, which is similar to vector based kernel algorithms, such as kernel discriminant analysis. Here, α_l^i is the linear combination coefficient for the l^{th} direction of the i^{th} training sample. Based on these observations, it is straightforward to outline the kernel AP. In this paper, we do not focus on the kernel method, because the kernel parameter, defined in kernel functions, tuning is still a problem and the number of the kernel parameters is much more than that in the 1st–order form.

2.5. LDA–based feature extraction: TR1DA

TR1DA extends the STL framework for feature extraction based on linear discriminant analysis (LDA). Because TR1DA includes many variables, we first define the variables. $\mathbf{X}_{i,j}^k$ is the i^{th} object in the j^{th} class in the k^{th} training iteration. For k=1, we have $\mathbf{X}_{i,j}^1 = \mathbf{X}_{i,j}$. Moreover, $\mathbf{X}_{i,j}^k$ is an M^{th}–order tensor. $\mathbf{M}_j^k = \left(\sum_{i=1}^{n_j} \mathbf{X}_{i,j}^k\right) \big/ n_j$ is the j^{th} class mean tensor in the k^{th} training iteration and $\mathbf{M}^k = \left(\sum_{j=1}^{c} \mathbf{M}_j^k\right) \big/ c$ is the total mean tensor of all objects in the k^{th} training iteration. u_i^j is the j^{th} direction base vector for decomposition in the i^{th} training iteration. With these definitions, TR1DA is defined by the following equations:

$$\mathbf{X}_{i,j}^k = \mathbf{X}_{i,j}^{k-1} - \lambda_{i,j}^{k-1} \prod_{d=1}^{M} {}_{\times_d}\, u_{k-1}^d ; \lambda_{i,j}^{k-1} = \mathbf{X}_{i,j}^{k-1} \prod_{d=1}^{M} {}_{\times_d}\, u_{k-1}^d \qquad (10)$$

$$\arg\max_{u_k^l|_{l=1}^M} \left(\begin{array}{l} \sum_{i=1}^{c} \left(\begin{array}{l} n_i \left(\left(\mathbf{M}_i^k - \mathbf{M}^k\right) \prod_{l=1}^{M} {}_{\times_l} \left(u_k^l\right)^T \right) \\ \left(\left(\mathbf{M}_i^k - \mathbf{M}^k\right) \prod_{l=1}^{M} {}_{\times_l} \left(u_k^l\right)^T \right)^T \end{array} \right) \\ -\zeta \sum_{i=1}^{c} \sum_{j=1}^{n_i} \left(\begin{array}{l} \left(\left(\mathbf{X}_{j,i}^k - \mathbf{M}_i^k\right) \prod_{l=1}^{M} {}_{\times_l} \left(u_k^l\right)^T \right) \\ \left(\left(\mathbf{X}_{j,i}^k - \mathbf{M}_i^k\right) \prod_{l=1}^{M} {}_{\times_l} \left(u_k^l\right)^T \right)^T \end{array} \right) \end{array} \right) \qquad (11)$$

where $\mathbf{X}_{i,j}^1 = \mathbf{X}_{i,j}$.

From the definition of the problem, which is given by (10) and (11), we know that TR1DA can be calculated by a greedy approach, because of the lack of the closed form solution for the problem. The greedy approach is illustrated in Figure 1. The calculation of $\mathbf{X}_{i,j}^r$ is based on the given $\mathbf{X}_{i,j}^{r-1}$ and $u_{r-1}^d|_{d=1}^M$. With the given $\mathbf{X}_{i,j}^{r-1}$ and $u_{r-1}^d|_{d=1}^M$, we calculate $\lambda_{i,j}^{r-1}$ via $\lambda_{i,j}^{r-1} = \mathbf{X}_{i,j}^{r-1} \prod_{d=1}^{M} {}_{\times_d}\, u_{r-1}^d$. The projection orientations $u_{r-1}^d|_{d=1}^M$ are obtained by the alternating least square (ALS) method. In ALS, we obtain the optimal base vector u_{r-1}^d given $u_{r-1}^i|_{1\le i\le M}^{i\ne d}$. We can conduct the procedure iteratively to obtain $u_{r-1}^d|_{d=1}^M$. The flowchart of the algorithm is given in Figure 1 and the detailed procedure for TR1DA is given in [18] due to length constraints. More extensions, such as a graph embedding extension and a kernel extension for TR1DA, are also given in [18].

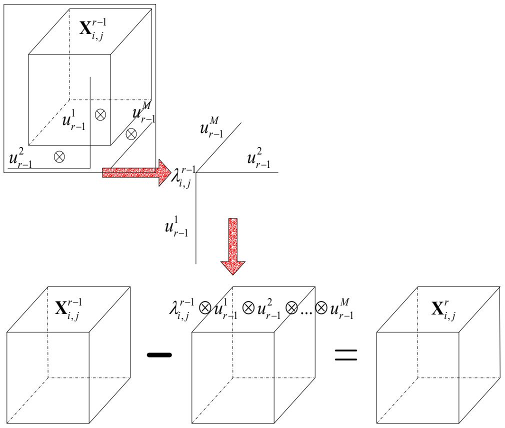

Figure 1. Greedy approach in TR1DA.

With TR1DA, we can obtain $u_{r-1}^d|_{d=1}^M$ iteratively. The coordinate value $\lambda_{i,j}^{k-1}$ can represent the original tensor $\mathbf{X}$. For recognition, the prototype $\mathbf{X}^p$ for each individual class in the database and the test tensor $\mathbf{X}^t$ to be classified are projected onto the bases to get the prototype weight vector $\lambda_p^k|_{k=1}^R$ and test weight vector $\lambda_t^k|_{k=1}^R$. The test tensor class is found by minimizing the distance $\varepsilon = \left\| \lambda_t^k|_{k=1}^R - \lambda_p^k|_{k=1}^R \right\|$.

Unlike existing tensor extensions of discriminant analysis, TR1DA can converge (all $u_r^d|_{1\le d\le M}^{1\le r\le R}$ do not change any more during training stage). We can check the convergence through $\left\| \left(u_k^l\right)_t \cdot \left(u_k^l\right)_{t+1} \right| - 1 \right| \le \varepsilon$ where ε is a small number. If $\left| \left(u_k^l\right)_t \cdot \left(u_k^l\right)_{t+1} \right| = 1$, the calculated projection orientation in the t^{th} iteration is equivalent to the $(t+1)^{\text{th}}$ iteration.

3. Tensor minimax probability machines (TMPM): an instance of STL framework

We can easily generalize existing STL based convex optimization based machine learning and data mining algorithms. In this section, we generalize MPM to tensor MPM (TMPM) as an example. MPM is recently built and reported to achieve a top–level performance [3]. Moreover, MPM has a very strong probability theory foundation based on the powerful Marshall and Olkin's theorem [15]. TMPM aims at maximizing the probability of the correct classification for future data points according to:

$$\begin{cases} \max\limits_{\alpha, b, w_k \neq 0, 1 \le k \le n} (\alpha) \qquad \text{s.t.} \\ \overline{(x \times_{-k} w_{-k})} \times_k w_k - b \ge \kappa(\alpha) \sqrt{w_k^T \Sigma_{(x \times_{-k} w_{-k})} w_k} \\ b - \overline{(y \times_{-k} w_{-k})} \times_k w_k \ge \kappa(\alpha) \sqrt{w_k^T \Sigma_{(y \times_{-k} w_{-k})} w_k} \end{cases} \tag{12}$$

where $\kappa(\alpha) = \sqrt{\frac{\alpha}{1-\alpha}}$, $\overline{(x \times_{-k} w_{-k})} = \frac{1}{n_x} \sum_{i=1}^{n_x} x_i \times_{-k} w_{-k}$, $\overline{(y \times_{-k} w_{-k})} = \frac{1}{n_y} \sum_{j=1}^{n_y} y_j \times_{-k} w_{-k}$, and $\Sigma_{x \times_{-k} w_{-k}}$ and $\Sigma_{y \times_{-k} w_{-k}}$ are the covariance matrices of $x \times_{-k} w_{-k}$ and $y \times_{-k} w_{-k}$ respectively.

Table 2. Alternating procedure (AP) for tensor minimax probability machines (TMPM).

Input:	Same as in Table 1.
Output	A tensor minimax probability machine using the optimal tensor plane and the bias: $g(z) = \operatorname{sign}(z \times_1 w_1 \times_2 w_2 \times \ldots \times_n w_n - b)$.
Step1.	Same as Step 1 in Table 1.
Step2.	Same as Step 2 in Table 1.
Step 3.	For $p = 1, 2, \ldots, N$ or (until converged), do: For $l = 1, \ldots, n$, do: $\begin{cases} \max\limits_{\alpha, b, w_k^p \neq 0, 1 \le k \le n} (\alpha) \qquad \text{s.t.} \\ \overline{(x \times_{-k} w_{-k}^{p-1})} \times_k w_k^p - b_k^p \ge \kappa(\alpha) \sqrt{w_k^{pT} \Sigma_{(x \times_{-k} w_{-k}^{p-1})} w_k^p} \\ b_k^p - \overline{(y \times_{-k} w_{-k}^{p-1})} \times_k w_k^p \ge \kappa(\alpha) \sqrt{w_k^{pT} \Sigma_{(y \times_{-k} w_{-k}^{p-1})} w_k^p} \end{cases}$
Step 4.	Find the optimal bias b for all the projected positive/negative training samples according to: $\begin{cases} \max\limits_{b} (\alpha) \qquad \text{s.t.} \\ \overline{x_i \times_1 w_1^N \times_2 w_2^N \times \ldots \times_n w_n^N} - b \ge \kappa(\alpha) \\ b - \overline{y_j \times_1 w_1^N \times_2 w_2^N \times \ldots \times_n w_n^N} \ge \kappa(\alpha) \end{cases}$

The optimization problem for learning in (12) is a sequential second–order cone programming in convex optimization. Based on the procedure developed in Section 2.3, to solve (12) is straightforward. In Table 2, we have only listed Step 3 and Step 4, which require more detail, while Step 1 and Step 2 are the same as in the procedure shown in Table 1.

We now turn to analyze the computational complexity of this generalized TMPM. If the second order cone programming is solved by the primal–dual interior–point method, the computational complexity of TMPM is $O\left(N \sum_{i=1}^{L} (n^i)^3\right)$, when the input tensors belong to $R^{n^1 \times \ldots n^L}$ and N iterations are required for convergence.

Regarding kernelization, the key issue in a kernel tensor minimax probability machine (KTMPM) is the adaptation of (12) to the kernel trick. Here, we present our results for the positive class only. For the negative class, the deduction process is similar. We define

$$\theta_{-k}^{i} = \lambda_i \prod_{\substack{l=1 \\ l \neq k}}^{n} \sum_{q=1}^{n_x + n_y} \gamma_q^l K(w_l^i, w_l^q) \tag{13}$$

then we have:

$$\begin{aligned} & \overline{\varphi(x) \times_{-k} \varphi(w_{-k})} \times_k \varphi(w_k) \\ & = \frac{1}{n_x} \left\{ \sum_{i=1}^{n_x} \sum_{j=1}^{n_x + n_y} \gamma_j^k \theta_{-k}^i K(w_k^i, w_k^j) \right\} \end{aligned} \tag{14}$$

and

$$\begin{aligned} & \frac{1}{n_x} (\varphi(w_k))^T \left\{ \sum_{i=1}^{n_x} \sum_{j=1}^{n_x} \begin{pmatrix} \varphi(x_i) \times_{-k} \varphi(w_{-k}) \\ -\overline{\varphi(x) \times_{-k} \varphi(w_{-k})} \end{pmatrix} \begin{pmatrix} \varphi(x_j) \times_{-k} \varphi(w_{-k}) \\ -\overline{\varphi(x) \times_{-k} \varphi(w_{-k})} \end{pmatrix}^T \right\} (\varphi(w_k)) \\ & = \frac{1}{n_x} \left\{ \sum_{i=1}^{n_x} \sum_{j=1}^{n_x} \begin{pmatrix} \left(\sum_{p=1}^{n_x + n_y} \gamma_p^k \theta_{-k}^i K(w_k^p, w_k^i) \right) \\ -\frac{1}{n_x} \left(\sum_{p=1}^{n_x + n_y} \sum_{q=1}^{n_x} \gamma_p^k \theta_{-k}^q K(w_k^p, w_k^q) \right) \end{pmatrix} \begin{pmatrix} \left(\sum_{p=1}^{n_x + n_y} \gamma_p^k \theta_{-k}^j K(w_k^p, w_k^j) \right) \\ -\frac{1}{n_x} \left(\sum_{p=1}^{n_x + n_y} \sum_{q=1}^{n_x} \gamma_p^k \theta_{-k}^q K(w_k^p, w_k^q) \right) \end{pmatrix} \right\}. \end{aligned} \tag{15}$$

where γ_q^l is linear combination coefficients, like α_l^i defined in Section 2.4. With (13), (14), and (15), the KTMPM can be implemented according to AP, which is similar to the procedure listed in Table 2.

Figure 2 illustrates TMPM for learning with n^{th}–order tensors as inputs. All the data in the figure come from the experiment section below. First, we have a set of samples to train TMPM. We then get three 1st–order tensors as the projection orientations for classification. To represent the outer product of the three 1st–order tensors conveniently, we only do the outer product for the first and second projection 1st–order tensors, the first and third projection 1st–order tensors, and the second and third projection 1st–order tensors, respectively. The results of each production are shown as a 2–dimensional tensor plane. With these tensor planes, we can project the training or testing data onto a real axis easily through $\lambda_i = x_i \times_1 w_1 \times_2 w_2 \times \ldots \times_n w_n$. After projections, we can calculate the optimal bias according to the criterion of the 0th–order TMPM.

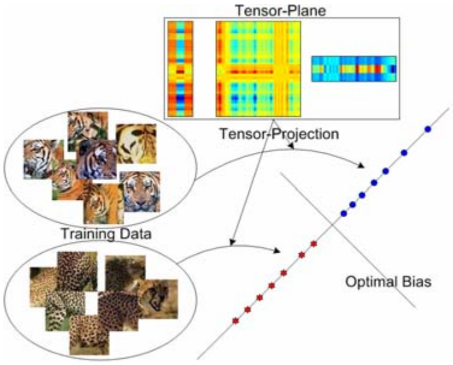

Figure 2. The proposed learning framework.

4. Experimental results

With the established STL framework, TMPM is developed as an example of a generalized learning machine. In this section, its performance is examined with a binary image classification problem. The experimental results show that the STL version outperforms the original vector based version in terms of the ability for generalization and the stability for different initial parameters $w_i|_{i=1}^{n}$ as referred to in Abstract.

4.1. Sample binary classification problem

To categorize images into groups based on their semantic contents is a very important and challenging issue. The fundamental task is binary classification. A hierarchical structure can be built using a series of binary classifiers. As a result, this semantic image classification [19] can make the growing image repositories easy to search and browse [17]; moreover, the semantic image classification is of great help for many other applications.

In this STL based classification environment, two groups of images are separated from each other by a trained TMPM. Inputs (representing features) of TMPM are the regions of interest (ROIs) within the pictures, which are extracted by the attention model [6] and represented as 3rd–order tensors.

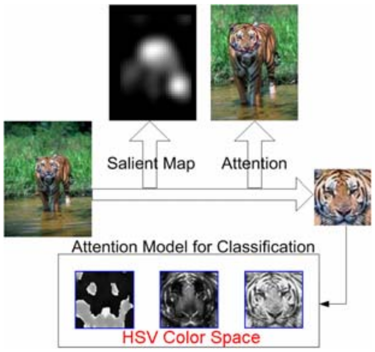

Figure 3. Attention model for classification.

The attention model [6] is capable of reproducing human–level performances for a number of pop–out tasks [16]. A target “pops–out” from its surroundings when it has it’s a unique orientation, color, intensity, or size. Pop–out targets are always easily noticed by an observer. Therefore, utilizing the attention model to describe an image’s semantic information is reasonable.

As shown in Figure 3, to represent an attention region from an image consists of several steps: 1) extract the salient map as introduced by Itti *et al.* in [6]; 2) find the most attentive region, whose center has the largest value in the salient map; 3) extract the attention region by a square (called ROI) in size of 64×64; and 4) finally, represent this ROI in the hue, saturation, and value (HSV) perceptual color space. Consequently, we have a 3rd–order tensor for the image representation.

Note that although we only select a small region from the image, the size of the extracted 3rd–order tensor is already as large as $64 \times 64 \times 3$; if we vectorize it, the dimensionality of the vector will be 12288. From the next subsection, we will be aware that the numbers of elements in the training and test sets are only of hundreds, much smaller than 12288 .Therefore, the *small samples size* (SSS) problem is always met when a 3rd–order tensor is converted to a vector for input to a conventional learning machine. In contrast, our tensor oriented supervised learning scheme can reduce the SSS problem and at the same time represent the ROIs much more naturally.

4.2. Training/test data sets

The training set and the test set for the following experiments are built upon the Corel photo gallery [17], from which 100 images are selected for each of the two sets. These 200 images are processed to extract the 3^{rd}–tensor attention features for TMPM.

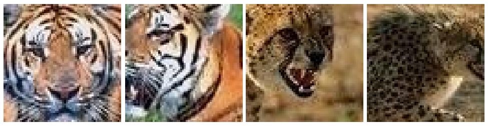

Figure 4. Some successful attention ROIs.

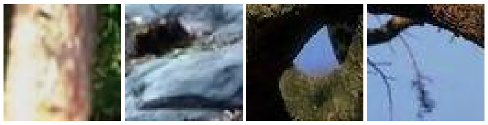

Figure 5. Some unsuccessful attention ROIs.

We choose the "Tiger" category and the "Leopard" category for binary classification experiments since it is a very difficult task for a machine to distinguish between them. The "Tiger" and "Leopard" classification is carried out in the next subsection. We choose the top *N* images as a training set according to the image IDs, while all the images are used to form the corresponding test set.

Table 3. TMPM vs. MPM.

	Training Error Rate		Testing Error Rate	
STS	TMPM	MPM	TMPM	MPM
5	0.0000	0.4000	0.4600	0.5050
10	0.0000	0.5000	0.4250	0.4900
15	0.0667	0.4667	0.3250	0.4150
20	0.0500	0.5000	0.2350	0.4800
25	0.0600	0.4800	0.2400	0.4650
30	0.1167	0.5000	0.2550	0.4600

We introduced 3^{rd}–order tensor attention ROIs, which can mostly be found correctly from the images. Some successful results, respectively extracted from the "Tiger" category and the "Leopard" category, are shown in Figure 4. By this means, the underlying data structures are well kept for the next step: classification. However, we should note that the attention model sometimes cannot depict the semantic information in an image. This is because the attention model always locates the region that is different from its surroundings and thus might be "cheated" when some complex or bright background exists. Some unsuccessful ROIs in the "tiger" (top) and "leopard" (bottom) categories are shown in Figure 5. It should be emphasized that in order to keep the following comparative experiments fair and automatic, these wrongly extracted ROIs were not excluded from each training set.

4.3. Binary classification performance

We carried out the binary classification ("Tiger" and "Leopard") experiments on the above training/ test sets. The proposed tensor based TMPM algorithm is compared with the original MPM. The experimental results are shown in Table 3. Error rates for both training and testing are reported, as the size of the training set (STS) increases from 5 to 30.

From the training error rates in Table 3, it can be seen that the traditional method (MPM) cannot learn a satisfactory model for classification when the size of the sample set is small. However, the learning algorithm TMPM under the proposed STL framework has a good characteristic on the volume control according to the computational learning theory and its real performances.

Also from Table 3, based on the testing error rates of the comparative experiments, the proposed TMPM algorithm more effectively represents the intrinsic discriminant information (in forms of 3^{rd}–order ROIs). TMPM learns a better classification model for unseen data classification than MPM and thus has a better performance on the testing set. It is observed that the TMPM error rate is a decreasing function of the size of the training set. This is consistent with statistical learning theory.

5. Experimental–based convergence and stability analysis

In this section we study two important issues in machine learning and data mining, namely, the convergence property and the insensitiveness to the initial values.

We carry out a series of experiments based on the same database employed in Section 4. Experimental results prove that TMPM converges well. In addition, it is insensitive to the initial values and thus has a good stability.

Figure 6 shows tensor projected position values $\lambda_i = x_i \times_1 w_1 \times_2 w_2 \times \ldots \times_n w_n$ of the original general tensors with an increasing number of learning iterations using 10 training samples for each class. We find that the projected values converge to stable values. As shown in Figure 6, five to six iterations are usually enough to achieve convergence.

Figure 7 shows TMPM is insensitive to the initial values. Many learning algorithms converge to different local minima with different initial parameter values for

$w_i |_{i=1}^{n}$. This is the so–called *local minimum* problem. However, our new TMPM does not slump into this *local minimum* problem, which is proved by a set of experiments, with different initial parameters, 10 learning iterations, and 20 training samples. Theoretically, this is also true, because each sub–optimization problem to optimize w_i is convex.

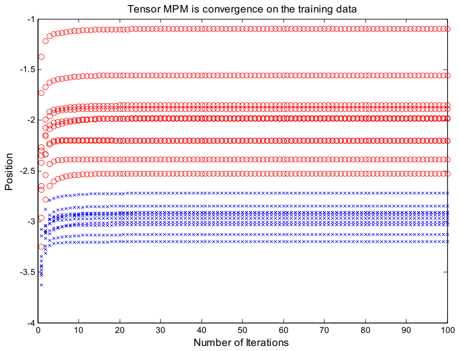

Figure 6. TMPM converges effectively.

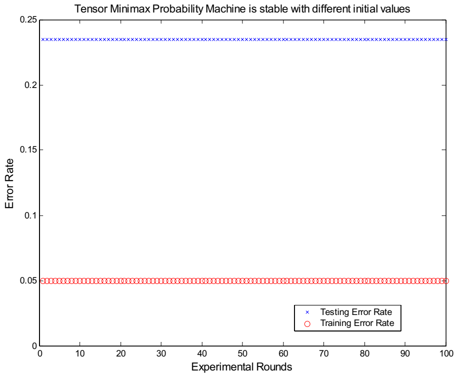

Figure 7. TMPM is stable with different initial values in 10 learning iterations (20 training samples in each class).

6. Conclusion

In this paper, we have developed a supervised tensor learning (STL) framework to generalize convex optimization based schemes so that they accept n$^{\text{th}}$–order tensors as inputs. We have also studied the *tensor rank–one discriminant analysis* (*TR1DA*) method for extracting features from tensors. Under this STL framework, a novel approach called tensor minimax probability machines (TMPM) has been developed to learn a series of projections for classification. TMPM has the following properties: 1) it can reduce the *curse of dimension* problem meaningfully and directly by using the tensor based representation, which reduces the number of parameters in the learning procedure; 2) it makes a better use of the information on the input than vector based discriminant analysis, efficiently; 3) it converges within a few training iterations; and 4) it is insensitive to the initial parameter values. The stability of the proposed scheme has also been analyzed.

References

[1] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge University Press, 2003.

[2] V. Vapnik, The Nature of Statistical Learning Theory. Springer–Verlag, New York, 1995.

[3] G. Lanckriet *et al.*, “A Robust Minimax Approach to Classification,” *JMLR*, 3, 555–582, 2002.

[4] J. Li and J. Wang, “Automatic Linguistic Indexing of Pictures by a Statistical Modeling Approach,” *IEEE Trans. PAMI*, 25(9), 1075–1088, 2003.

[5] M. Turk and A. Pentland, “Face recognition using eigenfaces,” in *Proc. of IEEE CVPR*, 586–591, 1991.

[6] L. Itti *et al.*, “A Model of Saliency–Based Visual Attention for Rapid Scene Analysis,” *IEEE Trans. PAMI*, 20(11), 1254–1259, 1998.

[7] S. Sarkar *et al.*, “The Human ID Gait Challenge Problem: Data Sets, Performance, and Analysis,” *IEEE Trans. PAMI*, 27(2), 162–177, 2005.

[8] S. Chang, “The Holy Grail of Content–Based Media Analysis,” *IEEE Multimedia Magazine*, 9(2), 6–10, 2002.

[9] R. Bellman, Adaptive Control Processes: A Guided Tour. Princeton University Press, 1960.

[10] M. Anthony *et al.*, Computational Learning Theory. Cambridge Univ. Press, 1997.

[11] R. O. Duda *et al.*, Pattern Classification. 2/e, Wiley.

[12] G. Lanckriet *et al.*, “Learning the Kernel Matrix with Semidefinite Programming,” *JMLR*, 5, 27–72, 2004.

[13] A. Shashua and A. Levin, “Linear Image Coding for Regression and Classification Using the Tensor–rank Principle,” in *Proc. of IEEE CVPR*, 2001.

[14] L. Lathauwer, Signal Processing based on Multilinear Algebra. PhD Thesis, Universiteit Leuven, 1997.

[15] A. Marshall, and I. Olkin, “Multivariate Chebyshev Inequalities,” *Annals of Mathematical Statistics*, 31(4), 1001–1014, 1960.

[16] A. Treisman and G. Gelade, “A Feature–Integration Theory of Attention,” *Cognitive Psychology*, 12(1), 97–136, 1980.

[17] J. Wang *et al.*, “SIMPLIcity: Semantics–Sensitive Integrated Matching for Picture Libraries,” *IEEE Trans. PAMI*, 23(9), 947–963, 2001.

[18] D. Tao *et al.*, Tensor Rank One Discriminant Analysis. Tech. Rep., Birkbeck College, University of London, Nov. 2004.

[19] A. Vailaya *et al.*, “On Image Classification: City Images vs. Landscapes,” *Pattern Recognition*, 31(12), 1921–1935, 1998.