

Hierarchical Modeling with Tensor Inputs

Yada Zhu, Jingrui He, Rick Lawrence

IBM T.J. Watson Research Center
1101 Kitchawan Road Route 134
Yorktown Heights, NY 10598

Abstract

In many real applications, the input data are naturally expressed as tensors, such as virtual metrology in semiconductor manufacturing, face recognition and gait recognition in computer vision, etc. In this paper, we propose a general optimization framework for dealing with tensor inputs. Most existing methods for supervised tensor learning use only rank-one weight tensors in the linear model and cannot readily incorporate domain knowledge. In our framework, we obtain the weight tensor in a hierarchical way – we first approximate it by a low-rank tensor, and then estimate the low-rank approximation using the prior knowledge from various sources, e.g., different domain experts. This is motivated by wafer quality prediction in semiconductor manufacturing. Furthermore, we propose an effective algorithm named *H-MOTE* for solving this framework, which is guaranteed to converge. The time complexity of *H-MOTE* is linear with respect to the number of examples as well as the size of the weight tensor. Experimental results show the superiority of *H-MOTE* over state-of-the-art techniques on both synthetic and real data sets.

Introduction

In many real applications, data come in the form of tensors, or multi-dimensional arrays. For example, in semiconductor manufacturing, each recipe process usually has multiple steps. During each step, we could observe process variables such as temperature, pressure and gas flow per unit time. Therefore, to predict the wafer quality, the input data are naturally expressed as third-order tensors (the three dimensions or modes are steps, seconds within a step, and observed process variables, or features) or second-order tensors if we use the summary statistics for each process variable in a single step instead of the instantaneous measurements. Another example is in computer vision, where images can be modeled as second-order tensors, and image sequences can be modeled as third-order tensors (Wang, Chen, and Xu 2011). Much existing work on dealing with tensor data converts tensors into one-dimensional vectors, and applies the rich methodology for vector inputs to build the model,

either for classification or for regression (Mitchell 1997; Wasserman 2009). However, by converting tensors into vectors, we tend to lose much information embedded in the structure of tensors, such as the feature correspondence in different steps in the example of semiconductor manufacturing, or the neighborhood information of a pixel in the example of computer vision.

To maximally exploit such valuable structural information embedded in the input tensors, in this paper, we address the problem of predictive modeling with tensor inputs by directly operating on the tensors. To this end, we propose a general optimization framework, which predicts the output (or its probability of coming from each class) based on the inner product between the input tensor and a weight tensor. The weight tensor is then estimated in a hierarchical way. To be specific, we assume that the weight tensor has a low-rank approximation, and the Candecomp/Parafac (CP) decomposition (Kolda and Bader 2009) of the low-rank tensor can be further approximated based on prior information from various sources, e.g., different domain experts. This framework is motivated by wafer quality prediction in semiconductor manufacturing, where the input tensors have two dimensions (using summary statistics for each process variable in a single step instead of the instantaneous measurements): steps and features. On one hand, the features in a single step, or the same feature across different steps, tend to have similar values in the weight tensor, which leads to the assumption of low-rank approximation for the weight tensor; on the other hand, different domain experts may have various opinions regarding the relative importance of certain steps and certain features on predicting wafer quality, and we need to leverage their prior knowledge in order to improve the performance of the predictor, especially when the labeled set is small. Our proposed framework combines these two factors with the prediction loss in the objective function, which leads to an optimal solution for the weight tensor in the linear model.

Furthermore, we propose an effective algorithm for solving the optimization framework named *H-MOTE*. It is based on block coordinate descent, which is guaranteed to converge to a local optimum since the objection function in the general framework has unique minimum in each coordinate block. Experimental results on synthetic and semiconductor manufacturing data sets demonstrate the good performance of *H-MOTE* compared with state-of-the-art techniques.

The main contributions of this paper are as follows:

1. an optimization framework, modeling the predictive task with tensor inputs in a hierarchical way;
2. an algorithm (*H-MOTE*), solving the optimization framework;
3. proof and analysis, showing the quality, convergence and scalability of the proposed algorithm.

The rest of the paper is organized as follows. In Section 2, we briefly review the related work. The general optimization framework is presented in Section 3, followed by the introduction of *H-MOTE* algorithm in Section 4. We show some experimental results in Section 5 with discussion. Finally, we conclude the paper in Section 6.

Related Work

In this section, we briefly review related work on wafer quality prediction in semiconductor manufacturing and predictive modeling with tensor inputs.

Wafer Quality Prediction

In semiconductor manufacturing, wafers have to go through hundreds of processes to become a final IC device. Each process follows a given recipe that defines detailed fabrication steps and settings of the process variables. In recent years, virtual metrology (VM) has received a lot of attention in semiconductor industry, which builds models to predict wafer quality based on historical measurements of wafer quality and corresponding process variables. The predicted wafer quality can then prompt feedback control in a timely fashion, detect fault wafers early, and improve productivity by reducing actual metrology frequency of wafer quality.

For this purpose, researchers have built statistical models such as multiple regression with feature selection (Kang et al. 2011; Lynn et al. 2009), partial least squares (Khan, Moyne, and Tilbury 2008), SVM regression (Kang et al. 2011), and artificial neural networks (Chang et al. 2006; Su et al. 2008) based on one-dimensional vectors converted from the input tensors. However, as discussed in Section , this conversion tends to lose useful information embedded in the structure of input tensors. For example, for a certain recipe process, it may be the case that the process variables in Step 12 have key impact on the wafer quality. These types of prior knowledge cannot be naturally incorporated into the statistical model based on vectorized tensors. On the other hand, although the real-time process variables can be represented as multivariate time-series, for VM applications, the objective is to predict the wafer quality, rather than to forecast the process variables. Therefore, time-series analysis is not most appropriate for this purpose.

To the best of our knowledge, our work is the first to build a tensor-based model for wafer quality prediction, which incorporates prior knowledge from various sources in a principled way.

Predictive Modeling with Tensor Inputs

Compared with the rich literature on predictive modeling with vector inputs (Mitchell 1997; Wasserman 2009), there

has not been as much effort on dealing with tensor inputs. Existing methods for predictive modeling with tensor inputs can be roughly categorized into the following 2 groups: dimensionality reduction and supervised tensor learning (classification and regression).

Dimensionality reduction for tensor data is also called tensor embedding. It aims at finding the intrinsic local geometrical structure of the tensor space by learning a lower dimensional tensor subspace (with the same order) (He, Cai, and Niyogi 2005), which has been successfully applied in computer vision tasks, such as face recognition (He, Cai, and Niyogi 2005; Dai and Yeung 2006; Tao et al. 2008; Li et al. 2008) and gait recognition (Tao et al. 2006; 2007; Li et al. 2008), as well as network anomaly detection and sensor measurements (Sun et al. 2008). For example, the authors in (He, Cai, and Niyogi 2005) propose the TSA algorithm for structured dimensionality reduction, which explicitly takes into account the manifold structure of the image space; the authors in (Dai and Yeung 2006) propose several tensor embedding methods, which allow the relationships between dimensions of a tensor representation to be efficiently characterized; for gait recognition, the authors in (Tao et al. 2007) first build a set of Gabor based human gait appearance models, and then use GTDA to seamlessly incorporate the object structure information as a natural constraint; the authors in (Tao et al. 2008) generalize the Bayesian principal component analysis (BPCA) to tensors; the authors in (Li et al. 2008) propose DLLE and its tensorized version, which generalize LLE (Roweis and Saul 2000) to enforce the separability between different classes; and the authors in (Sun et al. 2008) introduce a general framework named ITA, which efficiently computes a compact summary for high-order and high-dimensional data, as well as reveals the hidden correlations.

On the other hand, in supervised tensor learning, a classifier (or regressor) is directly built based on the tensor inputs instead of their vectorized version. For example, in (Hochreiter and Obermayer 2004), the authors introduce P-SVM, which minimizes a scale-invariant capacity measure under a new set of constraints, and develop a fast optimization algorithm based on SMO (Platt 1998); in (Tao et al. 2005), the authors establish a supervised tensor learning framework, within which conventional learning machines such as SVM and MPM can be generalized to tensors; and in (Cai, He, and Han 2006), the authors propose STM, which finds a maximum margin classifier in the tensor space, and TLS, which finds a minimum residual sum-of-squares classifier.

Our proposed method belongs to supervised tensor learning. In our method, the weight tensor of the underlying linear model is built in a hierarchical way: the weight tensor is first approximated using a low-rank tensor; the Candecomp/Parafac (CP) decomposition (Kolda and Bader 2009) of the low-rank tensor is then estimated based on prior information from various sources. Compared with the methods proposed in (Tao et al. 2005) and (Cai, He, and Han 2006), they can be seen as special cases of our model in the sense that they only use rank-one weight tensors, whereas in our model, the rank of the optimal weight tensor can be more than one. Compared with P-SVM (Hochreiter and

Obermayer 2004), we do not need to construct the data matrix, which involves complex interaction between the ‘row’ objects and the ‘column’ objects, and our method can be easily generalized to higher order tensors, whereas P-SVM can only be applied on second-order tensors (matrices). Furthermore, compared with all the existing methods in this group, our method is able to incorporate prior information from various sources in a principled way, whereas existing methods cannot leverage this information.

Optimization Framework

In this section, we propose the general optimization framework for hierarchical modeling with tensor inputs. First we introduce the notation used throughout this paper; then we provide some background on CP decomposition and tensor rank; based on the above introduction, we present the objective function; and finally, we interpret the objective function from different perspectives.

Notation

Suppose that we are given N training examples $\{\mathcal{X}_n, y_n\}$, $n = 1, \dots, N$, where $\mathcal{X}_n \in \mathbb{R}^{d_1 \times d_2 \times \dots \times d_K}$ is a K -dimensional array, or K^{th} -order tensor, and $y_n \in \mathbb{R}$ is the response variable for regression problems, or $y_n \in \{-1, 1\}$ is the class label for classification problems. Notice that for \mathcal{X}_n , K is the dimension of this array or the number of modes of this tensor, and d_k is the number of elements along the k^{th} dimension, $k = 1, \dots, K$. Therefore, the total number of input features is $\prod_{k=1}^K d_k$. When $K = 1$, the input \mathcal{X}_n is a vector, and the problem is reduced to regular regression or classification; when $K = 2$, \mathcal{X}_n is a matrix; when $K > 2$, \mathcal{X}_n is a K^{th} -order tensor. In this paper, we focus on the cases where $K > 1$. For such problems, we can always convert the input tensor into a vector by concatenating the fibers along different modes, which are defined by fixing the indices of all the modes but one.¹ Then we can apply the well-established techniques for dealing with vector inputs to predict the value of y_n . However, in this way, we lose the rich information embedded in the structure of \mathcal{X}_n . Therefore, in this paper, our goal is to predict the value of y_n by making use of the structure of \mathcal{X}_n .

For future reference, throughout this paper, we use lower-case letters to denote scalars; boldface lower-case letters to denote vectors; and calligraphic upper-case letters to denote tensors. Let $\mathcal{T}_1, \mathcal{T}_2 \in \mathbb{R}^{d_1 \times d_2 \times \dots \times d_K}$ denote two tensors. Define $\langle \mathcal{T}_1, \mathcal{T}_2 \rangle$ to be the inner product between \mathcal{T}_1 and \mathcal{T}_2 , which is the sum of the products of their corresponding elements. Furthermore, define the norm of tensor $\|\mathcal{T}_1\| = \sqrt{\langle \mathcal{T}_1, \mathcal{T}_1 \rangle}$.

CP Decomposition and Tensor Rank

In our framework, the rank- R approximation of the weight tensor is decomposed into a sum of vector outer products based on CP decomposition (Kolda and Bader 2009). The

CP decomposition factorizes a tensor into a sum of component rank-one tensors (Kolda and Bader 2009). For example, given third-order tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, we would like to write it by

$$\mathcal{T} = \sum_{r=1}^R \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

where R is a positive integer, $\mathbf{a}_r \in \mathbb{R}^{d_1}$, $\mathbf{b}_r \in \mathbb{R}^{d_2}$, $\mathbf{c}_r \in \mathbb{R}^{d_3}$, and ‘ \circ ’ denotes vector outer product. For the ease of future explanation, we refer to $\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$ as the r^{th} component of \mathcal{T} , $r = 1, \dots, R$.

The rank of a tensor \mathcal{T} is defined as the smallest number of rank-one tensors that generate \mathcal{T} as their sum (Kolda and Bader 2009). In other words, in the above equation, the smallest value of R that satisfies the equality is the rank of \mathcal{T} . In particular, when $R = 1$, for K^{th} -order tensors, we can decompose them into the outer product of K vectors.

Objective Function

In our proposed framework, we predict the value of y_n using a linear model, such as the linear regression model for regression problems and logistic regression model for classification problems. Therefore, in this linear model, we have a weight tensor $\mathcal{C} \in \mathbb{R}^{d_1 \times d_2 \times \dots \times d_K}$, which is the same size as \mathcal{X}_n . The main idea of this framework is to model the weight tensor in a hierarchical way, i.e., we first approximate the weight tensor using a low-rank tensor, whose CP decomposition is in turn estimated based on prior information from various sources.

To be specific, we minimize a loss function $L(y_n, \langle \mathcal{X}_n, \mathcal{C} \rangle)$ summed over all the training examples. For example, $L(\cdot, \cdot)$ can be the squared loss in regression, or the logistic loss in classification. Here we require that $L(\cdot, \cdot)$ is convex with respect to the second argument. Based on the tensor structure, we assume that the weight tensor \mathcal{C} can be approximated by a rank- R tensor with CP decomposition $\sum_{r=1}^R \mathbf{a}_{1r} \circ \mathbf{a}_{2r} \circ \dots \circ \mathbf{a}_{Kr}$, where R is equal to the number of sources where we could obtain domain knowledge, e.g., R domain experts, and $\mathbf{a}_{kr} \in \mathbb{R}^{d_k}$ is the weight vector for the k^{th} mode in the r^{th} component. Therefore, $\|\mathcal{C} - \sum_{r=1}^R \mathbf{a}_{1r} \circ \mathbf{a}_{2r} \circ \dots \circ \mathbf{a}_{Kr}\|^2$ should be small. Intuitively, each weight vector \mathbf{a}_{kr} reflects the importance of the k^{th} mode of the input tensors in the r^{th} component, and \mathbf{a}_{kr} ($r = 1, \dots, R$) collectively measure the contribution of the k^{th} mode of \mathcal{X}_n to the output y_n . For example, when $K = 2$ and $R = 1$, \mathcal{C} is a matrix, and $\mathcal{C}(i, j)$ should be close to $\mathbf{a}_{11}(i) \times \mathbf{a}_{21}(j)$, where $\mathcal{C}(i, j)$ is the element of \mathcal{C} in the i^{th} row and j^{th} column, $\mathbf{a}_{11}(i)$ is the i^{th} element of \mathbf{a}_{11} , and $\mathbf{a}_{21}(j)$ is the j^{th} element of \mathbf{a}_{21} . Furthermore, to estimate the weight vector \mathbf{a}_{kr} , we need to leverage the domain knowledge from R different sources, e.g., domain experts. To be specific, for each \mathbf{a}_{kr} , we assume that it is close to vector $\mathbf{a}_{kr0} \in \mathbb{R}^{d_k}$, which is given to us a priori from the r^{th} source, $r = 1, \dots, R$.

¹Note that the vectorized version of the tensor may not be unique due to different orderings of the fibers.

Putting everything together, we minimize the following.

$$\begin{aligned} f(\mathcal{C}, \mathbf{a}_{kr}, k = 1, \dots, K, r = 1, \dots, R) \\ = \sum_{n=1}^N L(y_n, \langle \mathcal{X}_n, \mathcal{C} \rangle) + \gamma_0 \|\mathcal{C} - \sum_{r=1}^R \mathbf{a}_{1r} \circ \mathbf{a}_{2r} \circ \dots \circ \mathbf{a}_{Kr}\|^2 \\ + \sum_{k=1}^K \sum_{r=1}^R \gamma_{kr} \|\mathbf{a}_{kr} - \mathbf{a}_{kr0}\|^2 \end{aligned} \quad (1)$$

where γ_0 and γ_{kr} ($k = 1, \dots, K, r = 1, \dots, R$) are positive parameters that balance among different terms. In particular, the relative values of $\gamma_{1r}, \dots, \gamma_{Kr}$ reflect our confidence in using prior knowledge to approximate the weight vector in each mode of the r^{th} component: the bigger the value of γ_{kr} , the more confident we are about this approximation.

Interpreting the Objective Function

In this subsection, we interpret the objective function in Equation (1) from different perspectives.

If $L(y_n, \langle \mathcal{X}_n, \mathcal{C} \rangle)$ is the squared loss for regression problems or logistic loss for classification problems, it reflects the negative log-likelihood of the n^{th} example, and Equation (1) can be interpreted from a probability perspective. To be specific, if the prior distribution of \mathcal{C} is normal with mean $\sum_{r=1}^R \mathbf{a}_{1r} \circ \mathbf{a}_{2r} \circ \dots \circ \mathbf{a}_{Kr}$ and variance $\frac{1}{2\gamma_0}$ for each element, and the prior distribution of \mathbf{a}_{kr} is normal with mean \mathbf{a}_{kr0} and variance $\frac{1}{2\gamma_{kr}}$ for each element, then Equation (1) is the posterior probability of \mathcal{C} and \mathbf{a}_{kr} ($k = 1, \dots, K, r = 1, \dots, R$) given the data (up to a constant). Therefore, by minimizing Equation (1), we can find the MAP estimates of the weight tensor \mathcal{C} as well as the weight vectors \mathbf{a}_{kr} ($k = 1, \dots, K, r = 1, \dots, R$).

On the other hand, traditional ridge regression and logistic regression for vector inputs can be seen as special cases of the proposed optimization framework. To see this, simply fix \mathbf{a}_{kr} ($k = 1, \dots, K, r = 1, \dots, R$) to be 0 vectors, and we have the same objective function as in ridge regression or logistic regression for vector inputs. In this way, when we minimize the original objective function with respect to both the weight tensor and the weight vectors, the second term on the right hand side of Equation (1) can be seen as a new regularizer which encourages a low-rank approximation of the weight tensor \mathcal{C} instead of shrinking it to 0 as in ridge regression and logistic regression. As we will see in Section , the use of this new regularizer effectively prevents over-fitting, especially when the labeled data is very scarce.

H-MOTE Algorithm

In this section, we introduce the *H-MOTE* algorithm (Hierarchical MOdeling with Tensor inputs) for calculating the weight tensor \mathcal{C} that minimizes Equation (1), analyze its performance in terms of quality, convergence, and time complexity, and elaborate on a special case of *H-MOTE* for squared loss in regression problems.

Analysis of Equation (1)

Notice that the function f is not jointly convex with respect to \mathcal{C} and \mathbf{a}_{kr} , $k = 1, \dots, K, r = 1, \dots, R$. However, if we

fix \mathbf{a}_{kr} , and minimize f with respect to \mathcal{C} , we have

$$\begin{aligned} f_{\mathbf{a}_{kr}}(\mathcal{C}) &= \sum_{n=1}^N L(y_n, \langle \mathcal{X}_n, \mathcal{C} \rangle) \\ &+ \gamma_0 \|\mathcal{C} - \sum_{r=1}^R \mathbf{a}_{1r} \circ \mathbf{a}_{2r} \circ \dots \circ \mathbf{a}_{Kr}\|^2 \end{aligned}$$

Notice that the third term on the right hand side of Equation (1) is not dependent on \mathcal{C} . It is easy to see that $f_{\mathbf{a}_{kr}}(\mathcal{C})$ is convex in \mathcal{C} , given that $L(\cdot, \cdot)$ is convex with respect to the second argument, and

$$\begin{aligned} \frac{\partial f_{\mathbf{a}_{kr}}(\mathcal{C})}{\partial \mathcal{C}} &= \sum_{n=1}^N l(y_n, \langle \mathcal{X}_n, \mathcal{C} \rangle) \mathcal{X}_n \\ &+ 2\gamma_0 (\mathcal{C} - \sum_{r=1}^R \mathbf{a}_{1r} \circ \mathbf{a}_{2r} \circ \dots \circ \mathbf{a}_{Kr}) \end{aligned} \quad (2)$$

where $l(\cdot, \cdot)$ is the partial derivative of $L(\cdot, \cdot)$ with respect to the second argument.

Similarly, if we minimize f with respect to \mathbf{a}_{ls} , keeping \mathcal{C} and \mathbf{a}_{kr} fixed, $k \neq l, r \neq s$, we have

$$\begin{aligned} f_{\mathcal{C}, \mathbf{a}_{kr}, k \neq l, r \neq s}(\mathbf{a}_{ls}) &= \gamma_0 \|\mathcal{C} - \sum_{r=1}^R \mathbf{a}_{1r} \circ \mathbf{a}_{2r} \circ \dots \circ \mathbf{a}_{Kr}\|^2 \\ &+ \gamma_{ls} \|\mathbf{a}_{ls} - \mathbf{a}_{ls0}\|^2 \end{aligned}$$

Notice that the first term on the right hand side of Equation (1) is not dependent on \mathbf{a}_{ls} . It is easy to see that $f_{\mathcal{C}, \mathbf{a}_{kr}, k \neq l, r \neq s}(\mathbf{a}_{ls})$ is convex in \mathbf{a}_{ls} , and

$$\begin{aligned} \frac{\partial f_{\mathcal{C}, \mathbf{a}_{kr}, k \neq l, r \neq s}(\mathbf{a}_{ls})}{\partial \mathbf{a}_{ls}} &= 2\gamma_0 (\alpha_{ls} \mathbf{a}_{ls} - \beta_{ls} + \tau_{ls}) \\ &+ 2\gamma_{ls} (\mathbf{a}_{ls} - \mathbf{a}_{ls0}) \end{aligned} \quad (3)$$

where $\alpha_{ls} = \langle \mathbf{a}_{1s} \circ \dots \circ \mathbf{a}_{(l-1)s} \circ \mathbf{a}_{(l+1)s} \circ \dots \circ \mathbf{a}_{Ks}, \mathbf{a}_{1s} \circ \dots \circ \mathbf{a}_{(l-1)s} \circ \mathbf{a}_{(l+1)s} \circ \dots \circ \mathbf{a}_{Ks} \rangle$, and β_{ls}, τ_{ls} are d_l -dimensional vectors. For β_{ls} , its i^{th} element $\beta_{ls}(i) = \langle \mathcal{C}_{l=i}, \mathbf{a}_{1s} \circ \dots \circ \mathbf{a}_{(l-1)s} \circ \mathbf{a}_{(l+1)s} \circ \dots \circ \mathbf{a}_{Ks} \rangle$. Here $\mathcal{C}_{l=i} \in \mathbb{R}^{d_1 \times \dots \times d_{l-1} \times d_{l+1} \times \dots \times d_K}$ is a $(K-1)^{\text{th}}$ -order tensor. Its elements are equal to \mathcal{C} with the index of the l^{th} dimension fixed at i . For τ_{ls} , its i^{th} element $\tau_{ls}(i) = \langle \mathcal{T}_{lsi}, \mathbf{a}_{1s} \circ \dots \circ \mathbf{a}_{(l-1)s} \circ \mathbf{a}_{(l+1)s} \circ \dots \circ \mathbf{a}_{Ks} \rangle$. Here $\mathcal{T}_{lsi} \in \mathbb{R}^{d_1 \times \dots \times d_{l-1} \times d_{l+1} \times \dots \times d_K}$ is a $(K-1)^{\text{th}}$ -order tensor, and $\mathcal{T}_{lsi} = \sum_{r \neq s} \mathbf{a}_{lr}(i) (\mathbf{a}_{1r} \circ \dots \circ \mathbf{a}_{(l-1)r} \circ \mathbf{a}_{(l+1)r} \circ \dots \circ \mathbf{a}_{Kr})$.

Therefore, setting Equation (3) to 0, we have the following optimal vector \mathbf{a}_{ls}^* that minimizes $f_{\mathcal{C}, \mathbf{a}_{kr}, k \neq l, r \neq s}(\mathbf{a}_{ls})$.

$$\mathbf{a}_{ls}^* = \frac{\gamma_0 \beta_{ls} - \gamma_0 \tau_{ls} + \gamma_{ls} \mathbf{a}_{ls0}}{\gamma_0 \alpha_{ls} + \gamma_{ls}} \quad (4)$$

Algorithm Description

Based on the above discussion, in this paper, we make use of block coordinate descent method to find the optimal solution to Equation (1). The convergence of block coordinate descent is guaranteed since the objective function has unique minimum in each coordinate block (Luenberger 1973). The proposed *H-MOTE* algorithm is shown in Algorithm 1. It works as follows. In Step 1, we initialize vector \mathbf{a}_{kr} to be \mathbf{a}_{kr0} ; between Step 2 and Step 9, we alternatively update

weight tensor \mathcal{C} and weight vectors \mathbf{a}_{kr} ($k = 1, \dots, K$, $r = 1, \dots, R$) for T times.

During the test stage, given a tensor \mathcal{X} , we first calculate its inner product with the weight tensor \mathcal{C} , $\langle \mathcal{X}, \mathcal{C} \rangle$, which can be used to predict the output for regression problems, or transformed into probabilities via the logistic function for classification problems.

Quality and Convergence Analysis

With respect to the convergence of the proposed *H-MOTE* algorithm, we have the following lemma.

Lemma 1. If the number of iteration steps T is sufficiently large, *H-MOTE* will converge to a local optimum of the objective function in Equation (1).

Proof. Omitted due to lack of space.

Time Complexity

Assuming that the number of iteration steps needed for gradient descent in Step 3 of *H-MOTE* is upper bounded by T' , we have the following lemma demonstrating the time complexity of *H-MOTE*.

Lemma 2. The time complexity of *H-MOTE* is $O(T((T'(N + KR) + K^2R^2) \prod_{k=1}^K d_k))$.

Proof. Omitted due to lack of space.

Notice that in all our experiments, T and T' are always upper bounded by 50. Therefore, according to the above lemma, *H-MOTE* scales linearly with respect to the number of examples and the size of the weight tensor.

Algorithm 1 *H-MOTE*: Hierarchical Modeling with Tensor Inputs

Input: \mathcal{X}_n , y_n , $n = 1, \dots, N$, γ_0 , γ_{kr} , \mathbf{a}_{kr0} , ($k = 1, \dots, K$, $r = 1, \dots, R$), the number of iteration steps T

Output: Weight tensor \mathcal{C} and weight vectors \mathbf{a}_{kr} ($k = 1, \dots, K$, $r = 1, \dots, R$)

- 1: Initialize $\mathbf{a}_{kr} = \mathbf{a}_{kr0}$ ($k = 1, \dots, K$, $r = 1, \dots, R$)
 - 2: **for** $t = 1$ to T **do**
 - 3: Update the weight tensor \mathcal{C} using gradient descent according to Equation (2)
 - 4: **for** $k = 1$ to K **do**
 - 5: **for** $r = 1$ to R **do**
 - 6: Update vector \mathbf{a}_{kr} according to Equation (4)
 - 7: **end for**
 - 8: **end for**
 - 9: **end for**
-

Case Study

In this subsection, we study a special case of *H-MOTE*, where the loss function is given by $L(y_n, \langle \mathcal{X}_n, \mathcal{C} \rangle) = (y_n - \langle \mathcal{X}_n, \mathcal{C} \rangle)^2$. In this way, we have $l(y_n, \langle \mathcal{X}_n, \mathcal{C} \rangle) = -2(y_n - \langle \mathcal{X}_n, \mathcal{C} \rangle)$. By setting Equation (2) to 0, we have a closed-form solution for the vectorized version \mathbf{c} of the weight tensor \mathcal{C} in each iteration step of *H-MOTE*,

$$\mathbf{c} = (\gamma_0 I_{\prod_{k=1}^K d_k} + \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n')^{-1} (\gamma_0 \sum_{r=1}^R \mathbf{b}_r + \sum_{n=1}^N y_n \mathbf{x}_n) \quad (5)$$

where $I_{\prod_{k=1}^K d_k}$ denotes the $\prod_{k=1}^K d_k \times \prod_{k=1}^K d_k$ identity matrix, \mathbf{x}_n denotes the vectorized version of \mathcal{X}_n , \mathbf{x}_n' denotes the transpose of \mathbf{x}_n , and \mathbf{b}_r denotes the vectorized version of $\mathbf{a}_{1r} \circ \mathbf{a}_{2r} \circ \dots \circ \mathbf{a}_{Kr}$. Notice that Equation (5) is very similar to ridge regression on the vectorized inputs except for the term $\gamma_0 \sum_{r=1}^R \mathbf{b}_r$, which reflects both the low-rank nature of the weight tensor as well as prior knowledge. As can be seen in the next section, it is particularly helpful for preventing overfitting when the labeled set size is small.

In this case, in Step 3 of *H-MOTE*, instead of updating \mathcal{C} using gradient descent, we simply apply Equation (5) to obtain \mathbf{c} , which, after rearranging the elements, will give us the current weight tensor \mathcal{C} .

Experimental Results

In this section, we demonstrate the performance of the proposed *H-MOTE* algorithm on both synthetic and real data sets. In particular, we aim to answer the following questions.

1. How does the performance of *H-MOTE* compare with the algorithms dealing with vectorized inputs?
2. How does the performance of *H-MOTE* compare with other supervised tensor learning algorithms?
3. How is *H-MOTE* affected by small perturbations in the parameters?
4. How fast does *H-MOTE* converge to a local optimum?
5. Will higher-rank tensors improve the performance?

To answer the first 5 questions, we fix $R = 1$ in *H-MOTE*, and test the following variants of *H-MOTE*: *H-MOTE1* with \mathbf{a}_{k10} ($k = 1, \dots, K$) set to zero; *H-MOTE2* with \mathbf{a}_{k10} ($k = 1, \dots, K$) given by a domain expert; *H-MOTE3* with \mathbf{a}_{k10} ($k = 1, \dots, K$) set using the output of *H-MOTE2*.

Synthetic Data

In this subsection, we answer the first question, and compare the three variants of *H-MOTE* with ridge regression for vectorized inputs (Ridge) (Wasserman 2009) on synthetic data sets. The data sets consist of 1000 randomly generated examples represented as second-order tensors. The outputs are obtained by first calculating the inner product between the input tensors and a rank-one weight tensor, and then adding Gaussian noise with increasing variance. The Root Mean Squared Error (RMSE) of 5-fold cross validation are shown in Figure 1, which are averaged over 50 runs.

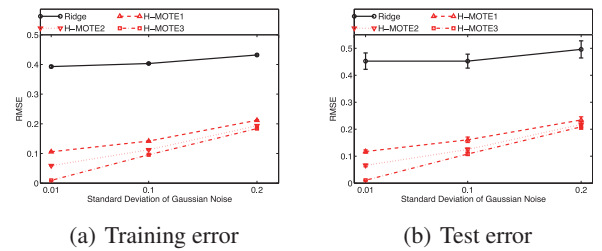


Figure 1: Comparison on synthetic data: *H-MOTE* methods perform better than ridge regression and prevent overfitting.

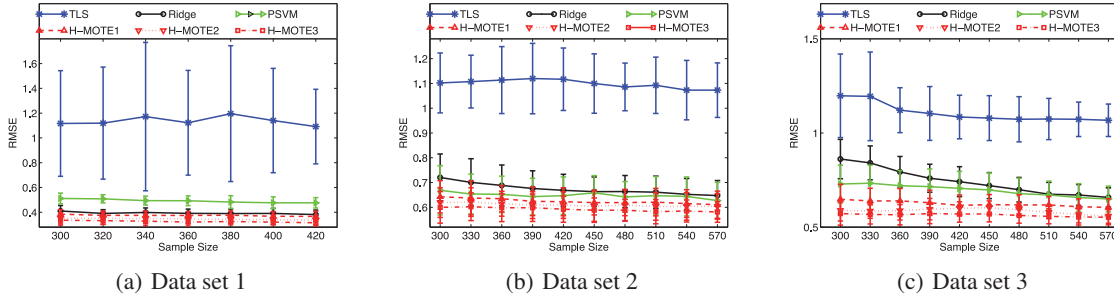


Figure 2: Comparison on real data: *H-MOTE3* has the lowest average RMSE

From this figure, we have the following observations. First, *H-MOTE* methods for tensor inputs are significantly better than ridge regression for vectorized inputs in terms of both the training error and the test error. Second, on the test set, the standard deviation of RMSE for *H-MOTE* methods is much smaller compared with ridge regression. Third, using ridge regression, the difference between the average RMSE on the training set and on the test set is larger than using *H-MOTE* methods, showing that *H-MOTE* methods are more robust to overfitting. Finally, as we increase the standard deviation of the Gaussian noise, the difference between *H-MOTE* methods and ridge regression is getting smaller.

Real Data

In this subsection, we test the performance of *H-MOTE* on three data sets collected from semiconductor manufacturing processes. The first data set corresponds to a process with a total of 7 steps, each having 17 process variables. The median of each process variable is obtained, which can be represented as a second-order tensor of size 7×17 or a vector of length 119. The second and third data sets correspond to a process with 10 steps, each having 15 process variables. For the second data set, we use the median of each process variable, which can be represented as a second-order tensor of size 10×15 or a vector of length 150; and for the third process, we use both the median and the variance, which can be represented as a second-order tensor of size 10×30 or a vector of length 300. Due to the low frequency of actual metrology, in 9-month time period there are totally 488 and 891 target measurements for the two processes. Before building VM prediction models, data are preprocessed to eliminate observations with missing values, measurement errors and outliers. The process variables and the output are normalized to have mean zero and standard deviation one.

Comparison with Different Algorithms On the real data sets, we compare the three variants of *H-MOTE* with the following three competitors: Ridge (ridge regression (Wasserman 2009)) for dealing with vectorized inputs; PSVM (Potential Support Vector Machine (Hochreiter and Obermayer 2004)) for supervised tensor learning; and TLS (Tensor Least Squares (Cai, He, and Han 2006)) for supervised tensor learning. For all these methods, the cross-validation results of RMSE are used for comparison, and the parameters

are chosen based on cross-validation in the training set only.

Next we answer the first two questions on the real data sets. The comparison results of the six methods are shown in Figure 2. For each training set size, we run the experiments 50 times, and report both the mean and the standard deviation. From these figures, we can see that the performance of *H-MOTE* methods is consistently better than Ridge, which takes vectorized inputs, and existing supervised tensor learning methods (PSVM and TLS). Comparing *H-MOTE2* and *H-MOTE1*, the prior information provided by the domain expert used in *H-MOTE2* helps improve the performance; and in *H-MOTE3*, by using the output of *H-MOTE2* as the prior, we further reduce the RMSE.

Robustness Study In this part, we answer the third question, and use the first data set to test *H-MOTE* under small perturbations of the parameters γ_0 , γ_{11} and γ_{12} . Figure 3 shows the RMSE vs. different values of γ_0 , which demonstrates the robustness of *H-MOTE*. The results of the other parameters are similar and omitted due to lack of space.

Convergence Rate In this part, we answer the fourth question using the first data set. We assume that the algorithm converges if the change of the estimated weight tensor \mathcal{C} and estimated weight vectors \mathbf{a}_{11} and \mathbf{a}_{12} is less than 10^{-7} . We let the algorithm run sufficiently large number of iterations to obtain the optimal value \mathcal{C}^* , \mathbf{a}_{11}^* and \mathbf{a}_{12}^* . Then at iteration t , we calculate the difference between the current estimates and their optimal value: $\delta_t = \sqrt{\|\mathcal{C}_t - \mathcal{C}^*\|^2 + \|\mathbf{a}_{11,t} - \mathbf{a}_{11}^*\|^2 + \|\mathbf{a}_{12,t} - \mathbf{a}_{12}^*\|^2}$, where \mathcal{C}_t , $\mathbf{a}_{11,t}$ and $\mathbf{a}_{12,t}$ are the estimated tensor and weight vectors at iteration t . In Figure 4, we plot the natural log of δ_t versus the iteration number t . We can see that *H-MOTE* converges at least exponentially.

Higher Rank Weight Tensors In this part, we answer the last question. In our framework, the rank R depends on the number of sources to obtain the domain knowledge. Next we test the performance of *H-MOTE* with $R = 2$, where one source of prior knowledge is from the domain expert, and the other is from *H-MOTE1* with $R = 1$. The comparison results of *H-MOTE* with $R = 1$ and $R = 2$ on the second data set are shown in Figure 5. We can see that bigger values of R are able to further improve the performance of *H-MOTE* in terms of the average RMSE of cross validation.

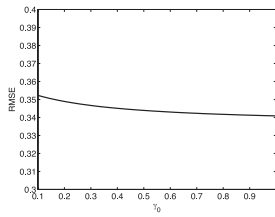


Figure 3: Impact of small perturbations in γ_0

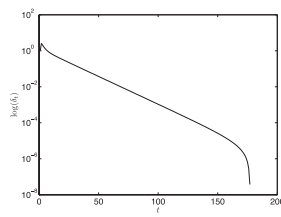


Figure 4: Convergence study of $H\text{-MOTE}$

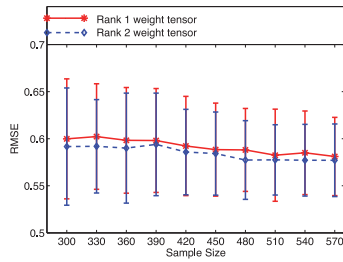


Figure 5: Comparison of $H\text{-MOTE}$ with different ranks

Conclusion

In this paper, we propose a general optimization framework for hierarchical modeling with tensor inputs, which is motivated by wafer quality prediction in semiconductor manufacturing. This optimization framework directly operates on the input tensors, and it is able to incorporate domain knowledge in a principled way. To solve the optimization framework, we propose an effective algorithm named $H\text{-MOTE}$ based on block coordinate descent. It converges to a local optimum, with linear time complexity regarding the total number of examples and the number of elements in the weight tensor. Experimental results on both synthetic and real data sets demonstrate the effectiveness of $H\text{-MOTE}$.

References

- Cai, D.; He, X.; and Han, J. 2006. Learning with tensor representation. Technical report, University of Illinois at Urbana-Champaign.
- Chang, Y.-J.; Kang, Y.; Hsu, C.-L.; Chang, C.-T.; and Chan, T. Y. 2006. Virtual metrology technique for semiconductor manufacturing. In *IJCNN*.
- Dai, G., and Yeung, D.-Y. 2006. Tensor embedding methods. In *AAAI*.
- He, X.; Cai, D.; and Niyogi, P. 2005. Tensor subspace analysis. In *NIPS*. MIT Press.
- Hochreiter, S., and Obermayer, K. 2004. Classification, regression, and feature selection on matrix data. Technical report, Technische Universität Berlin.

Kang, P.; Kim, D.; Lee, H.-J.; Doh, S.; and Cho, S. 2011. Virtual metrology for run-to-run control in semiconductor manufacturing. *Expert Systems with Applications* 38:2508–2522.

Khan, A. A.; Moyne, J. R.; and Tilbury, D. M. 2008. Virtual metrology and feedback control for semiconductor manufacturing processes using recursive partial least squares. *Journal of Process Control* 18:961–974.

Kolda, T. G., and Bader, B. W. 2009. Tensor decompositions and applications. *SIAM Review* 51(3):455–500.

Li, X.; Lin, S.; Yan, S.; and Xu, D. 2008. Discriminant locally linear embedding with high-order tensor data. *IEEE Trans. on Systems, Man, and Cybernetics* 38(2):342–352.

Luenberger, D. G. 1973. *Linear and Nonlinear Programming*. Massachusetts: Addison-Wesley, second edition.

Lynn, S.; Ringwood, J.; Ragnoli, E.; McLoone, S.; and MacGearailt, N. 2009. Virtual metrology for plasma etch using tool variables. In *Advanced Semiconductor Manufacturing Conference*.

Mitchell, T. M. 1997. *Machine Learning*. New York: McGraw-Hill.

Platt, J. C. 1998. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods*.

Roweis, S. T., and Saul, L. K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE* 290:2323–2326.

Su, Y.-C.; Lin, T.-H.; Cheng, F.-T.; and Wu, W.-M. 2008. Accuracy and real-time considerations for implementing various virtual metrology algorithms. *IEEE Trans. on Semiconductor Manufacturing* 21(3):426–434.

Sun, J.; Tao, D.; Papadimitriou, S.; Yu, P. S.; and Faloutsos, C. 2008. Incremental tensor analysis: Theory and applications. *TKDD* 2(3).

Tao, D.; Li, X.; Hu, W.; Maybank, S.; and Wu, X. 2005. Supervised tensor learning. In *ICDM*, 450–457. IEEE Computer Society.

Tao, D.; Li, X.; Maybank, S. J.; and Wu, X. 2006. Human carrying status in visual surveillance. In *CVPR*, 1670–1677.

Tao, D.; Li, X.; Wu, X.; and Maybank, S. J. 2007. General tensor discriminant analysis and gabor features for gait recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 29(10):1700–1715.

Tao, D.; Sun, J.; Shen, J.; Wu, X.; Li, X.; Maybank, S. J.; and Faloutsos, C. 2008. Bayesian tensor analysis. In *IJCNN*, 1402–1409.

Wang, Q.; Chen, F.; and Xu, W. 2011. Tracking by third-order tensor representation. *IEEE Trans. on Systems, Man, and Cybernetics* 41:385 – 396.

Wasserman, L. 2009. *All of Statistics*. New York: Springer-Verlag New York, LLC.