Multilinear Multitask Learning

Bernardino Romera-Paredes BERNARDINO.PAREDES.09@UCL.AC.UK
Department of Computer Science and UCL Interactive Centre, University College London, UK

Min Hane Aung M.AUNG@UCL.AC.UK
Department of Computer Science and UCL Interactive Centre, University College London, UK

Nadia Bianchi-Berthouze N.BERTHOUZE@UCL.AC.UK
UCL Interactive Centre Division of Psychology & Language Sciences, University College London, UK

Massimiliano Pontil M.PONTIL@CS.UCL.AC.UK
Department of Computer Science and Centre for Computational Statistics and Machine Learning, University College London, UK

Abstract

Many real world datasets occur or can be arranged into multi-modal structures. With such datasets, the tasks to be learnt can be referenced by multiple indices. Current multitask learning frameworks are not designed to account for the preservation of this information. We propose the use of multilinear algebra as a natural way to model such a set of related tasks. We present two learning methods; one is an adapted convex relaxation method used in the context of tensor completion. The second method is based on the Tucker decomposition and on alternating minimization. Experiments on synthetic and real data indicate that the multilinear approaches provide a significant improvement over other multitask learning methods. Overall our second approach yields the best performance in all datasets.

1. Introduction

The principal assertion in Multitask Learning (MTL) is that the combined learning of multiple related tasks can outperform learning each task in isolation, see for example (Argyriou et al., 2008a; Baxter, 2000; Caruana, 1997; Romera-Paredes et al., 2012) and references therein. In doing this, MTL allows for common information shared between the tasks to be used in the learning process, which leads to better generalization if the tasks are related (Ando and Zhang, 2005; Baxter, 2000; Maurer, 2009; Maurer and Pontil, 2012; Maurer et. al, 2013). For example, consider a task to be a regression problem in predicting restaurant ratings by a specific restaurant critic, given a restaurant as an input query. If we then include ratings from N critics, this will lead to N tasks. MTL methods will learn all of the regression functions that model all N tasks together by exploiting the common trends among all of the restaurant raters as well as the individual preferences.

Traditional MTL methods do not consider any additional inherent structure in the dataset and therefore the referencing of the tasks is simplified to a single index i; in the above example this would refer to the raters ranging from 1 to N. However, it is clear that a loss of information would arise if these methods were applied to datasets that are defined by multiple indices. For example, if our restaurant critics rated M separate aspects of each restaurant, this would give rise to a second index j ranging from 1 to M. This 2-dimensional indexing information would be lost in a traditional MTL approach. In this paper, we propose to extend the method developed in (Argyriou et al., 2008a) to consider the inherent structures in such datasets by preserving multi dimensional indices (or modes) which are associated with the tasks. To this end, we propose the use of multilinear models as a natural underpinning to represent this structural information. We will refer to our proposed framework as Multilinear Multitask Learning (MLMTL).

The use of multilinear models in previous applications

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

has been shown to be effective in determining separate underlying factors in data. In (Tenenbaum & Freeman, 2000) the authors introduced a bilinear model as a basis to represent the relationships in data with two modes. A common application for this bilinear model is to decouple two mode data for classification purposes; for example in (Mpiperis et al., 2008) the authors separate facial identity factors and expression factors to classify an emotional state and subject identity. By using bilinear models, the bi-factor interactions were more accurately represented. By extension, multilinear models in general have been used to separate more than two modes in similar datasets. For example (Vasilescu & Terzopoulos, 2002) decompose natural facial images according to four modes: identity, expression, head pose and lighting condition. Multilinear models have also been extended to account for different assumptions in the data. In (Vasilescu & Terzopoulos, 2005), the authors propose multilinear independent component analysis, where the factors extracted for each mode are not only uncorrelated but also statistically independent.

In this paper we form a multilinear model by structuring the weight parameters of all tasks into a tensor. This is a departure from the aforementioned studies, where the multilinear decomposition was applied directly to the input data, obtaining unsupervised learning models which can be seen as higher-order generalizations of principal (or independent) component analysis. The tensor representation allows us to account for the multi-modal interactions between the tasks. In addition, our approach allows one to *make predictions even in absence of training data for one or more of the tasks,* thereby providing a useful tool for transfer learning, see e.g. (Argyriou et al., 2008b). For example, the vector of parameters for the (i,j)-task can still be estimated provided that training data are available for at least one task (i,k), $k \neq j$ and one other task (ℓ, j), $\ell \neq i$.

In order to formulate MLMTL, we follow a complexity regularization approach which encourages low rank matricizations (Kolda & Bader, 2009) of the weight tensor. However, the regularization term will cause a non convex minimization problem. Therefore, the first of our learning approaches involves a convex relaxation of the original minimization problem. This solution is based on several recent studies which have showed that the use of the trace norm of tensors provides close convex approximations of similar minimization problems (Gandy et al., 2011; Liu et al., 2009; Signoretto et al., 2011; Tomioka, 2010). For our second approach we investigate a different strategy that makes use of an alternating minimization scheme for Tucker decomposed components of the original weight tensor. In summary, the main contributions of this paper are:

- The extension of multitask learning to account for multi-modal relationships among tasks using multilinear models;
- The introduction of an alternating minimization algorithm for MLMTL which implements the Tucker decomposition;
- A framework for transfer learning with multilinear models.

The remainder of this paper is organized as follows. In Section 2, we describe the key concepts from multilinear algebra that are needed to formulate our learning problem. In the following two sections, we describe two alternative ways to obtain solutions to the proposed problem: Section 3 describes a convex relaxation of the learning problem, whereas Section 4 presents an alternating minimization algorithm for Tucker decomposition. In Section 5 we compare the proposed tensor based methods with respect to their matrix based MTL counterparts. Additionally, non MTL baseline models are also compared. Finally, in Section 6 we conclude with a discussion of the results obtained.

2. Formulation

We begin by introducing some notation. We let $\mathbb{N}$ be the set of natural numbers and, for every $k \in \mathbb{N}$, we use $[k]$ to denote the set of integers up to and including k. Let $N \in \mathbb{N}$ and choose $p_1, \dots, p_N \in \mathbb{N}$. An N-order tensor, $\boldsymbol{X} \in \mathbb{R}^{p_1 \times \cdots \times p_N}$, is a collection of real numbers $(\boldsymbol{X}_{i_1,\dots,i_N} : i_n \in [p_n], n \in [N])$. Vectors are 1-order tensors and will be denoted by lower case letters, e.g. a or b; matrices are 2-order tensors and will be denoted by upper case letters, e.g. A or B. Boldface letters, e.g. $\boldsymbol{X}, \boldsymbol{W}$, will be used to denote tensors of order higher than two. We use the symbol ":" in the lower-script indices to denote sub-arrays within a matrix or a tensor, e.g. if $A \in \mathbb{R}^{p_1 \times p_2}$, then $A_{:,i} \in \mathbb{R}^{p_1}$ denotes the i-th column of A, for every $i \in [p_1]$. For the sake of clarity, colons at the beginning of superscripts are omitted so that $A_i := A_{:,i} \in \mathbb{R}^{p_1}$.

Mode-n fiber is a vector composed of the elements of a tensor obtained by fixing all indices but one, corresponding to the n-th mode. This notion is a higher order analogue of columns (mode-1 fibers) and rows (mode-2 fibers) in matrices. For example, in a 3-order tensor $\boldsymbol{X} \in \mathbb{R}^{p_1 \times p_2 \times p_3}$, the set of vectors of the form $\{\boldsymbol{X}_{i_1,:,i_3} \in \mathbb{R}^{p_2} : i_1 \in [p_1], i_3 \in [p_3]\}$ are mode-2 fibers of tensor $\boldsymbol{X}$.

Mode-n matricization is the process of rearranging all the elements of a tensor $\boldsymbol{X} \in \mathbb{R}^{p_1 \times \cdots \times p_N}$ into a matrix. Specifically, if $n \in [N]$, the mode-n matricization, denoted as $X_{(n)}$, is obtained by arranging the mode-n fibers of the tensor so that each of them is a column of $X_{(n)} \in \mathbb{R}^{p_n \times J_n}$, where we define $J_n := \prod_{k \neq n} p_k$.

Mode-n product is the product of a tensor $\boldsymbol{X} \in \mathbb{R}^{p_1 \times \cdots \times p_N}$ with a matrix $A \in \mathbb{R}^{J \times p_n}$, denoted by $\boldsymbol{X} \times_n A$. The result is a new tensor of size $p_1 \times \cdots \times p_{n-1} \times J \times p_{n+1} \times \cdots \times p_N$, where each mode-$n$ fiber is multiplied by A, that is

$$(\boldsymbol{X} \times_n A)_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{p_n} \boldsymbol{X}_{i_1,\ldots,i_N} A_{j,i_n}.$$

Rank$_n$ is the dimension of the space spanned by the mode-n fibers, that is the rank of the mode-n matricization of the tensor, $\text{rank}_n(\boldsymbol{X}) := \text{rank}(X_{(n)})$. If $\boldsymbol{X}$ is a matrix, this is the usual definition of rank, since $\text{rank}_1(\boldsymbol{X}) = \text{rank}_2(\boldsymbol{X}) = \text{rank}(\boldsymbol{X})$. For higher order tensors, however, $\text{rank}_n(\boldsymbol{X})$ varies with n.

We are now ready to describe the learning problem. We consider a set of T linear regression tasks, each of which is represented by a vector $w_t \in \mathbb{R}^d$, $t \in [T]$. Each task is associated with two or more modes (recall the restaurant rating example described in the introduction). We regard the $d \times T$ matrix $[w_1, \ldots, w_T]$ as the mode-1 matricization, $W_{(1)}$, of the tensor $\boldsymbol{W} \in \mathbb{R}^{p_1 \times \cdots \times p_N}$. Thus $p_1 = d$, $T = J_1 = \prod_{n=2}^{N} p_n$ and the index t can be identified by the multi-index $(i_2, \ldots, i_N) \in [p_2] \times \cdots \times [p_N]$. For each task t we sample the underlying regression problem m_t times, obtaining a set of input/output observations, where $(x_i^t, y_i^t) \in \mathbb{R}^d \times \mathbb{R}$, $i = 1, \ldots, m_t$. We also use the shorthand notation for the data term

$$F(\boldsymbol{W}) = \sum_{t=1}^{T} \sum_{i=1}^{m_t} L(\langle x_i^t, w_t \rangle, y_i^t) \tag{1}$$

where $\langle \cdot, \cdot \rangle$ is the inner product in $\mathbb{R}^d$, L is a prescribed loss function, e.g. the square error $L(z, y) = (z - y)^2$.

We estimate the regression vectors as the solution of the joint optimization problem

$$\min \{F(\boldsymbol{W}) + \gamma R(\boldsymbol{W})\} \tag{2}$$

where γ is a positive parameter which may be chosen by cross validation. The regularizer R encourages common structure between the tasks. In particular, our goal is to encourage the tensors $\boldsymbol{W}$ which have a simple structure in the sense that they involves a small number of "degree of freedoms". To this end, a natural choice is to consider the sum of the ranks of the matricizations of the tensors. Specifically, we let

$$R(\boldsymbol{W}) = \frac{1}{N} \sum_{n=1}^{N} \text{rank}_n(\boldsymbol{W}). \tag{3}$$

3. Convex Relaxation

Finding a convex relaxation of $R(\cdot)$ has been the objective of recent works (Gandy et al., 2011; Liu et al., 2009; Signoretto et al., 2011). All of them agree to use the trace norm for tensors as a good convex proxy. This is defined as the average of the trace norm of each matricization of $\boldsymbol{W}$,

$$\|\boldsymbol{W}\|_{\text{tr}} = \frac{1}{N} \sum_{n=1}^{N} \|W_{(n)}\|_1 \tag{4}$$

where $\| \cdot \|_1$ is the trace norm of a matrix, namely the ℓ_1-norm of the singular values. Note that in the particular case of 2-order tensor, (4) coincides with the usual notion of trace norm of a matrix. The above observation motivates us to consider the convex problem

$$\min_{\boldsymbol{W}} \{F(\boldsymbol{W}) + \gamma \|\boldsymbol{W}\|_{\text{tr}}\}. \tag{5}$$

When $N = 2$ problem (5) is equivalent to the one proposed in (Argyriou et al., 2008a). However, if $N > 2$, problem (5) is more difficult to solve due to the *composite* nature of the regularizer (4). To explain this observation, we introduce N auxiliary tensors $\boldsymbol{B}_n \in \mathbb{R}^{p_1 \times \cdots \times p_N}$, $n \in [N]$, each of which represents a version of the original tensor $\boldsymbol{W}$. With this notation, problem (5) can be reformulated as

$$\min_{\boldsymbol{W}, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_N} \left\{ F(\boldsymbol{W}) + \gamma \sum_{n=1}^{N} \left\| (B_n)_{(n)} \right\|_1 \quad \text{s.t} : \boldsymbol{B}_n = \boldsymbol{W},\ n \in [N] \right\} \tag{6}$$

where all the trace norm regularizers on the auxiliary matrix are related through the equality constraints. As noted by Gandy et al. (2011) and Signoretto (2013) problem (6) can be solved by the alternating direction method of multipliers (ADM) (see e.g. Bertsekas & Tsitsiklis, 1989). This optimization strategy allows problem 6 to be decoupled into independent subproblems which no longer have interdependent trace norm constraints. This decoupling is achieved by introducing a set of Lagrange multipliers $\mathbf{C}_n$, $\forall n \in \{1, \ldots, N\}$. The resultant augmented Lagrangian function is (Bertsekas & Tsitsiklis, 1989)

$$\mathcal{L}(\boldsymbol{W},\mathbf{C},\boldsymbol{B}) = F(\boldsymbol{W}) + \sum_{n=1}^{N}\Big(\gamma \left\|(B_n)_{(n)}\right\|_1 - \langle \mathbf{C}_n, \boldsymbol{W} - \boldsymbol{B}_n\rangle + \frac{\beta}{2}\left\|\boldsymbol{W} - \boldsymbol{B}_n\right\|_{\mathrm{Fr}}^2\Big) \quad (7)$$

for some $\beta > 0$, where the inner product between tensors is defined as the regular inner product between the vectorized form of the tensors. Appendix A describes an algorithm to solve problem (7).

The main advantage of this approach is that it always obtains the global solution of problem (4). However the fact that the outputs of the algorithm are the weight vectors themselves leads to two important drawbacks. First, transfer learning is not possible straight from the model. This is because the factors (see equation (8) below) are learned implicitly but one cannot have access to them under this approach. Therefore, if we want to add a new entity (e.g. a new restaurant in the example described in the introduction), the whole algorithm needs to be run again from scratch. The second drawback is related to memory necessities. In some problems, dealing with the whole weight tensor $\boldsymbol{W}$ can be problematic since this can be very large. Furthermore, this approach needs to keep $N+1$ versions of the tensor in memory so the total memory needed to run the algorithm is $\mathrm{O}\big((N+1)\prod_{n=1}^{N} p_n\big)$, which can be unfeasible for many problems. Finally, note that this approach is not optimizing the original problem but a convex approximation of it. To overcome these shortcomings, we propose a new method in the following section.

4. Obtaining Local Solutions of the Original Problem

In this section, we describe an alternative method which encourages low rank representations of the tensor using the Tucker decomposition, (see e.g. Kolda & Bader, 2009). It is defined as

$$\boldsymbol{W}_{i_1,\ldots,i_N} = \sum_{j_1=1}^{k_1}\cdots\sum_{j_N=1}^{k_N} \boldsymbol{G}_{j_1,\ldots,j_N} A^{(1)}_{i_1,j_1}\cdots A^{(N)}_{i_N,j_N}$$

where $\boldsymbol{W} \in \mathbb{R}^{p_1\times\cdots\times p_N}$ is the tensor containing all weight vectors, $A^{(n)} \in \mathbb{R}^{p_n\times k_n}$, $n\in[N]$, are the factor matrices and $\boldsymbol{G} \in \mathbb{R}^{k_1\times\cdots\times k_N}$, which is called the core tensor, models the interaction between factors. Figure 1 depicts the Tucker decomposition of a 3 mode tensor. We can also express this decomposition in a more compact way using the matricization and product operators,

$$\boldsymbol{W} = \boldsymbol{G} \times_1 A^{(1)} \cdots \times_N A^{(N)}. \quad (8)$$

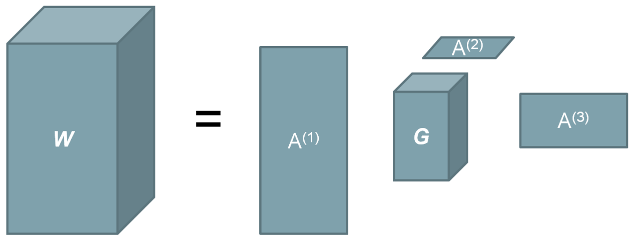

Figure 1. Tucker decomposition of a 3 mode tensor.

We would like to minimize the error term $F(\boldsymbol{W})$ in equation (1), over tensors of the form (8). Note that the Tucker decomposition is invariant under multiplication and division of different factors by the same scalar. With the aim of avoiding this issue and reducing overfitting, we add Frobenius norm regularization terms to the components. The resultant problem is

$$\min_{\boldsymbol{G},\mathrm{A}^{(1)},\ldots,\mathrm{A}^{(N)}} H(\boldsymbol{G}, A^{(1)},\ldots,A^{(N)})$$

where we defined

$$H(\boldsymbol{G}, A^{(1)},\ldots,A^{(N)}) = F(\boldsymbol{G}\times_1 A^{(1)}\cdots\times_N A^{(N)}) + \alpha\left(\|\boldsymbol{G}\|_{\mathrm{Fr}}^2 + \sum_{n=1}^{N}\left\|A^{(n)}\right\|_{\mathrm{Fr}}^2\right) \quad (9)$$

and α is a regularization parameter. Although the regularization term is heuristic in nature, we will argue in Section 5 that it helps avoiding overfitting.

We attempt to solve problem (9) by alternating minimization, where in each step we fix all components but one and solve the resultant convex problem. We distinguish three different cases: minimizing over $\boldsymbol{G}$, over $A^{(1)}$ (the set of components for the input data), and over $A^{(n)}$ for any $n \in \{2,\ldots,N\}$.

Minimizing over $\boldsymbol{G}$. Equation (9) can be minimized over $\boldsymbol{G}$ by noticing that

$$w_t = A^{(1)}G_{(1)}\left(A^{(N)}\otimes\cdots\otimes A^{(2)}\right)^\top e^t \quad (10)$$

where $\otimes$ denotes the Kronecker product and $e^t \in \mathbb{R}^T$ is a vector such that $e^t_t = 1$ and $e^t_s = 0$ for $s \neq t$. Here, we express the weight vector estimators in terms of the product of the first matricization of $\boldsymbol{G}$ with the other factor matrices. This leads to the convex problem $\min_{\boldsymbol{G}} \sum_{t=1}^{T}\sum_{i=1}^{m_t} L\left(x_i^{t\top} A^{(1)}G_{(1)}\left(A^{(N)}\otimes\cdots\otimes A^{(2)}\right)^\top e^t, y_i^t\right) + \alpha\|\boldsymbol{G}\|_{\mathrm{Fr}}^2$

which we can solve by gradient descent if L is differentiable. The gradient of H w.r.t $G_{(1)}$ is given by

$$\sum_{t=1}^{T}\sum_{i=1}^{m_t} L'_{i,t} A^{(1)\top} x_i^t e^{t\top}\left(A^{(N)}\otimes\cdots\otimes A^{(2)}\right) + 2\alpha G_{(1)}$$

where $L'_{i,t}$ is the derivative of L with respect to its first argument evaluated at $x_i^{t\top} A^{(1)} G_{(1)} \left(A^{(N)} \otimes \cdots \otimes A^{(2)}\right)^\top e^t$.

Finally, in order to obtain the tensor $\boldsymbol{G}$, we only need to invert the matricization operation.

Minimizing over $A^{(1)}$. In this case, we can reuse the equality (10) to minimize over $A^{(1)}$. This can be solved by gradient descent, where the gradient of H w.r.t. to $A^{(1)}$ is given by

$$\sum_{t=1}^{T}\sum_{i=1}^{m_t} L'_{i,t} x_i^t e^{t\top} \left(A^{(N)} \otimes \cdots \otimes A^{(2)}\right) G_{(1)}^\top + 2\alpha A^{(1)}.$$

Minimizing over $A^{(n)}$, $n \in \{2, \ldots, N\}$. This set of cases is more difficult to describe. In order to simplify the presentation we assume that $N = 3$ and $n = 2$, but the generalization to larger values is straightforward. First of all, it is useful to note that the 2-mode splits all tasks into p_2 sets, each of which has p_3 tasks. For every $\theta \in [p_2]$ we let $\mathcal{S}_\theta$ be the set of tasks indexed by $(\theta, \cdot)$. We rearrange the input data belonging to those tasks as

$$\widetilde{X}^\theta = \begin{bmatrix} X^{\theta,1} & 0 & \cdots & 0 \\ 0 & X^{\theta,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & X^{\theta,p_3} \end{bmatrix}, \widetilde{y}^\theta = \begin{bmatrix} y^{\theta,1} \\ y^{\theta,2} \\ \vdots \\ y^{\theta,p_3} \end{bmatrix}$$

where $\widetilde{X}^\theta \in \mathbb{R}^{p_1 p_3 \times M_\theta}$, $\widetilde{y}^\theta \in \mathbb{R}^{M_\theta}$, and M_θ is the number of instances of all tasks belonging to group $\mathcal{S}_\theta$, that is, $M_\theta = \sum_{t \in \mathcal{S}_\theta} m_t$. Then, we can write

$$\sum_{t=1}^{T}\sum_{i=1}^{m_t} L\left(x_i^{t\top} W_{(1)} e^t,\, y_i^t\right) = \sum_{\theta=1}^{p_2}\sum_{i=1}^{M_\theta} L\left(\widetilde{x}_i^{\theta\top} W_{(2)} e^\theta,\, \widetilde{y}_i^\theta\right)$$

$$= \sum_{\theta=1}^{p_2}\sum_{i=1}^{M_\theta} L\left(\widetilde{x}_i^{\theta\top} \left(A^{(3)} \otimes A^{(1)}\right) G_{(2)}^\top \left(A^{(2)\top}\right)_\theta,\, \widetilde{y}_i^\theta\right).$$

Notice that, unlike the previous cases, the columns of $A^{(2)\top}$ are decoupled, so we can solve instead p_2 simpler problems. The corresponding gradient of H with respect to $\left(A^{(2)\top}\right)_\theta$ is given by

$$\sum_{i=1}^{M_\theta} L'_{i,\theta} G_{(2)} \left(A^{(3)} \otimes A^{(1)}\right)^\top \widetilde{x}_i^\theta + 2\alpha \left(A^{(2)\top}\right)_\theta.$$

The local approach has a set of advantages derived from the explicit calculation of the factors. First, it allows for adding new factors in the setting without the necessity of relearning the previous factors, thereby allowing for transfer learning in a natural way. Second, the memory needed is $\mathrm{O}\left(\sum_{n=1}^N p_n k_n + \prod_{n=1}^N k_n\right)$ which can be much smaller than that of the convex approach, particularly if $k_n \ll p_n$ for some $n \in [N]$. The main drawback of this approach is that the solution obtained is a local optimum and there is no guarantee about how far this is from the global optimum.

5. Experiments

We have conducted a set of experiments on one synthetic dataset and two real world datasets. In this section, we present and analyze the results obtained. The predictive performances of the five methods outlined below are compared:

- Ridge Regression (RR): this model, chosen as a baseline, makes no assumption regarding the relationships among the tasks.
- Multitask Feature Learning (MTL-C): a convex MTL approach developed in (Argyriou et al., 2008a) which encourage all the tasks to share a common low dimensional representation of the data.
- Matrix Factorization (MTL-NC): a non convex MTL approach consisting of applying the matrix based counterpart to the proposed method described in Section 4.
- Convex Multilinear Multitask Learning (MLMTL-C): this approach, based on tensor trace norm regularization, is described in Section 3 and corresponds to an extension of MTL-C to multilinear algebra.
- Non-convex Multilinear Multitask Learning (MLMTL-NC): this is the approach proposed in Section 4.

The last two methods have been implemented using the Tensor Toolbox (Bader & Kolda, 2006). All methods have one hyper-parameter which needs to be tuned. This is always done by means of a validation set. The range of hyper-parameter values tried are 10^s, for $s = -3, -2, \ldots, 5, 6$. Preliminary experiments show that this range for s empirically contains the best solution for all approaches.

5.1. Synthetic Data

In order to test the correctness of the implementation of the algorithms proposed and to investigate the performance, we create a synthetic dataset where the weight tensor is decomposable as described in equation (8). This dataset is generated as follows: we create a set of $T = 100$ tasks, organized in an $p_2 \times p_3$

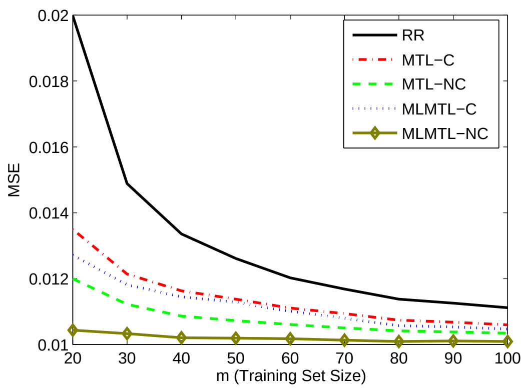

Figure 2. Synthetic dataset: Mean Square Error (MSE) comparison between Ridge Regression (RR), Multitask Feature Learning (Argyriou et al., 2008a) (MTL-C), Matrix Factorization MTL (MTL-NC), Convex Multilinear Multitask Learning (MLMTL-C) and Non-convex Multilinear Multitask Learning (MLMTL-NC).

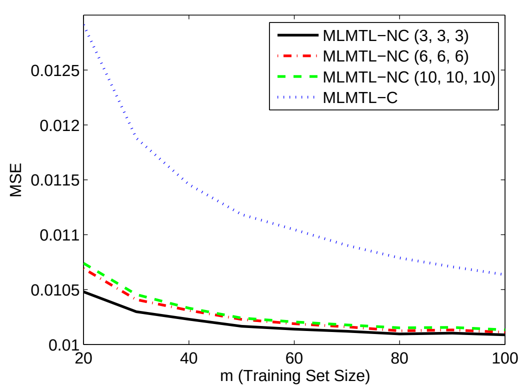

Figure 3. Synthetic dataset: Mean Square Error (MSE) comparison between Convex Multilinear Multitask Learning (MLMTL-C) and three versions of Non-convex Multilinear Multitask Learning (MLMTL-NC) (having different values for the ranks).

grid where $p_2 = p_3 = 10$ and the input data has dimensionality $p_1 = 10$. The tasks weight vectors can consequently be organized in an $N = 3$ mode tensor $\boldsymbol{W} \in \mathbb{R}^{p_1 \times p_2 \times p_3}$. Furthermore, this tensor has been generated so that $\mathrm{rank}_n(\boldsymbol{W}) = 3$, $\forall n \in [N]$. Particularly, any element in the tensor has been generated as $\boldsymbol{W}_{i_1,i_2,i_3} = \sum_{j_1,j_2,j_3=1}^{3} \boldsymbol{G}_{j_1,j_2,j_3} A^{(1)}_{i_1,j_1} A^{(2)}_{i_2,j_2} A^{(3)}_{i_3,j_3}$, $\forall i_1 \in [p_1], i_2 \in [p_2], i_3 \in [p_3]$, where all elements $A^{(1)}_{i_1,j_1}$, $A^{(2)}_{i_2,j_2}$, $A^{(3)}_{i_3,j_3}$, $\boldsymbol{G}_{j_1,j_2,j_3}$ are generated by random sampling from a Gaussian distribution $\mathcal{N}(0,1)$. For each task $t = (i_2, i_3)$, a set of m training instances $x^t_1, \ldots, x^t_m \in \mathbb{R}^d$ are sampled from $\mathcal{N}(0,1)$ and the labels are generated by the linear regression $y^t_i = w_t^\top x^t_i + \eta^t_i$, where $w_t = \boldsymbol{W}_{i_2,i_3}$ and η^t_i are sampled i.i.d. from $\mathcal{N}(0, 0.1)$. Similarly, a set of m_{val} and m_{test} instances and their corresponding labels are generated for each task for validation and testing purposes. The validation set is used to tune the regularization parameter for all approaches. Additionally for the factorization techniques (MTL-NC and MLMTL-NC), the number of factors for each mode has been fixed to the known values of the ranks.

The described experiment has been done for several values of m in order to investigate the effect of the number of training samples given. 20 trials have been executed for each value of m. The average results are shown in Figure 2, where we see that all MTL approaches perform better than ridge regression as expected. Furthermore, we see that among the convex approaches, MLMTL-C is slightly better than its matrix counterpart MTL-C although these differences are only significant for $m < 60$. Regarding the non-convex approaches, we see that MLMTL-NC obtains the best performance with a clear improvement with respect to all remaining approaches. Nevertheless, in the current setting, the non-convex approaches have advantage in that the ground truth ranks of the tensor are known for the synthetic dataset. To see how sensitive MLMTL-NC approach is with respect to incorrect values of the ranks, we have carried out a similar experiment where we compare MLMTL-C and three versions of MLMTL-NC, taking each one different values for the ranks.

The results are shown in Figure 3. The MLMTL-NC approaches with ranks $= (1,1,1)$ and ranks $= (2,2,2)$, which have ranks smaller than the true values, are not shown due to very poor performance[1]. As expected, the best approach is MLMTL-NC $(3,3,3)$ since in this case the ranks coincides with the actual ranks of the underlying tensor. However, we see that MLMTL-NC approaches with higher values of ranks perform quite similarly and in all of these cases there is an improvement with respect to MLMTL-C approach. This supports the hypothesis that MLMTL-NC approach is quite insensitive to the values of ranks, as long as they are an overestimation of the actual ones.

Finally, we empirically assess the computational efficiency of the approaches for different tensor dimensions. The results can be seen in Table 1. Both MLMTL approaches require more time for the learning process. Furthermore, we see that MLMTL-NC scales better than the convex counterpart as the size of the tensor increases.

[1]MLMTL-NC $(2,2,2)$ approach obtains an error around 0.08 whereas the error of MLMTL-NC $(1,1,1)$ approach is above 0.16

Tensor dim.	[20, 20, 20]	[30, 30, 30]	[40, 40, 40]
RR	0.0163	0.0641	0.1541
MTL-C	0.2464	0.8751	2.4255
MTL-NC	0.4388	0.7801	1.2973
MLMTL-C	4.5105	21.1303	133.5753
MLMTL-NC	19.3252	30.6920	53.2175

Table 1. Execution time (s) of Ridge Regression (RR), Multitask Feature Learning (MTL-C), Matrix Factorization MTL (MTL-NC), Convex Multilinear Multitask Learning (MLMTL-C), and Non-convex Multilinear Multitask Learning (MLMTL-NC).

5.2. Real Data

In this section, we test the described approaches with two real world datasets. For both datasets we want to infer the weight tensor $\boldsymbol{W} \in \mathbb{R}^{p_1 \times p_2 \times p_3}$, where p_1 is the number of attributes, p_3 is the number of subjects involved in the data and p_2 is the number of tasks we want to learn for each subject.

In these experiments, we also compare with a version of each non-MLMTL approach that ignores the subject identifier index and groups all of the instances. This leads to only one generic impersonal predictor for each p_2 task. This is done with the objective of appraising the effect of discarding the information provided by one mode. The resultant approaches are denoted as GRR, GMTL-C and GMTL-NC.

Regarding the multilinear approaches, the value of each rank$_n$ for MLMTL-NC has been set to $\min(10, p_n)$ for both experiments. This value is deemed to be a safe overestimate of the true rank on these data. The results of the previous experiments, presented in Figure 3, show that overestimates have a minimal effect on the final performance.

5.2.1. Restaurant & Consumer Dataset

The Restaurant & Consumer Dataset (Vargas-Govea et al., 2011) contains data to build a restaurant recommender system where the objective is to predict consumer ratings given to different restaurants. Each of the $p_3 = 138$ consumers gave $p_2 = 3$ scores for food quality, service quality and overall quality. The dataset also contains $p_1 = 44$ various descriptive attributes of the restaurants (such as geographical position, cuisine type and price band). We consider this to be a regression problem where the objective is to predict the scores given the attributes of a restaurant as an input query. Since there are 138 consumers, this leads to a multitask problem composed of 138×3 regression tasks.

This experiment was conducted in a similar way to the synthetic dataset, so that the training, validation and test sets were randomly selected for each task. The process was repeated 20 times for each value of m and the average results are shown in Figure 4.

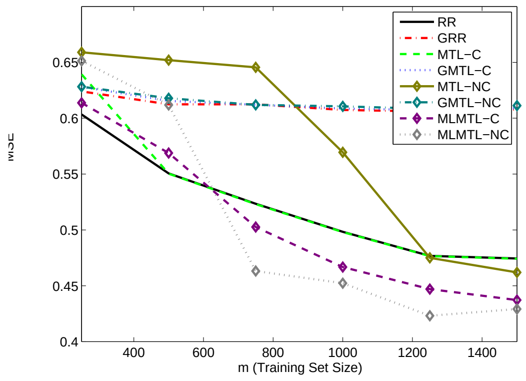

Figure 4. Restaurant & Consumer Dataset: Mean Square Error (MSE) comparison between [Grouped] Ridge Regression ([G]RR), [Grouped] Multitask Feature Learning ([G]MTL-C), [Grouped] Matrix Factorization MTL ([G]MTL-NC), Convex Multilinear Multitask Learning (MLMTL-C) and Non-convex Multilinear Multitask Learning (MLMTL-NC).

We observe that both MLMTL approaches outperform the remaining methods for $m \geq 750$. A set of paired t-test conducted between each pair of MLMTL and non-MLMTL shows that this improvement in the performance is significant obtaining always p-values below 0.01. This fact supports our hypothesis about the multi-modal relation among tasks and how MLMTL can take advantage of this over conventional MTL methods. We also checked the significance of the improvement observed between both MLMTL methods for $m \geq 750$, obtaining p-values below 0.025.

5.2.2. Shoulder Pain Dataset

In the second real world experiment we use the Shoulder Pain dataset (Lucey et al., 2011), which contains video clips of the faces of people who suffer from shoulder pain while performing active and passive exercises. For each frame of the video, the facial expression is described by a set of $p_1 = 132$ attributes (2D positions of 66 anatomical points). Each video is labelled frame by frame according to the physical activation of different set of facial muscles, encoded by the Facial Action Coding System (Ekman et al., 1978). This system defines a set of Action Units (AU) which refer to a contraction or relaxation of a determined set of muscles, e.g. AU6 is defined as the raising of a cheek. In this dataset, the intensity of an AU is expressed as a degree which ranges between 0 and 5. Our objective is to recognise the AU intensity level of $p_2 = 5$ different AUs for each of the $p_3 = 5$ different patients.

One common problem on this kind of data is that some subjects may not have shown any intensity for some AUs in the training set. For such AU/patient tasks traditional supervised learning approaches will not be effective. In contrast, MLMTL methods can naturally handle this scenario. Therefore, in this dataset we focus on assessing the performances of the methods in situations where no instances are provided to learn some of the tasks. The performances of the approaches are measured only on those tasks with no training instances, which we will refer to as target tasks hereafter.

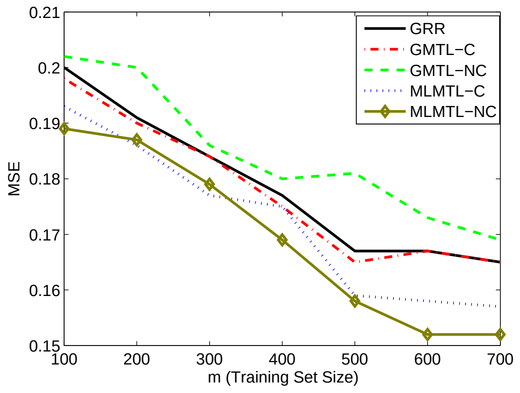

Figure 5. Shoulder Pain database: Mean Square Error (MSE) comparison between Grouped Ridge Regression (GRR), Grouped Multitask Feature Learning (GMTL-C), Group Matrix Factorization (GMTL-NC), Convex Multilinear Multitask Learning (MLMTL-C) and Non-convex Multilinear Multitask Learning (MLMTL-NC).

A set of $T_{target} = 2$ tasks are selected at random and the instances available for them are not used in the training process. Similarly, another set of tasks $T_{\text{val}} = 2$ are selected randomly for tuning the hyperparameters so that at the training stage, no instances from these tasks are used. Finally, m instances are used for the learning process of the remaining tasks and constitutes all of the information provided to the models to produce good estimators for the target tasks. Note that classic supervised learning approaches cannot learn predictors for tasks where there are no training instances. Therefore, we only compare with the grouped approaches (GRR, GMTL-C and GMTL-NC).

30 trials for different values of m were run, the averaged results are shown in Figure 5. The approaches based on tensors outperform their matrix based counterparts. A paired t-test shows that the improvement between MLMTL-NC and any other matrix approach is significant ($p < 0.01$) for all m. Also we see that MLMTL-NC generally outperforms MLMTL-C.

6. Discussion

In this paper, we have investigated two approaches for multilinear multitask learning. One being an adaptation of the low-rank tensor recovery strategy which employs a convex relaxation of the tensor decomposition problem. The second is based on an alternating minimization algorithm which optimizes the original non-convex problem together with a set of Frobenius norm regularizers to avoid overfitting. The sets of experiments carried out on both synthetic and real data support the hypothesis that employing multilinear methods in the described MTL scenarios is advantageous.

These approaches are useful in a multitask learning scenario where there is a priori information about how tasks are related among them, being these relations expressed as combinations of prescribed factors. This is the case for many real world datasets that contain multiple modalities. Even though such datasets are now commonplace, it is often seen that inter-task relationships are only exploited in one modality such as in stantard MTL. Furthermore, we have seen that multilinear models can obtain predictors for tasks which have no training instances, so long as there are enough training instances for other related tasks. This could potentially be of significant value in scenarios where specific instances in the data are missing or more difficult to gather.

The proposed methods use one hyperparameter to control the regularization over all matricizations of the tensor (γ in MLMTL-C and α in MLMTL-NC). An avenue for further study would be to assign a hyperparameter to each matricization regularizer, in order to trade-off the regularizing effect on each matricization. An interesting goal would be to find a way to tune these hyperparameters without any significant increase in computational expense.

Acknowledgments

This work was supported by EPSRC Grants EP/H016988/1, EP/H027203/1, and Royal Society International Joint Project Grant 2012/R2.

References

Ando, R.K. and Zhang, T. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Machine Learning Research*, 6:1817–1853, 2005.

Argyriou, A., Evgeniou, T., and Pontil, M. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

Argyriou, A., Maurer, A., and Pontil, M. An algorithm for transfer learning in a heterogeneous environment. *Proc. European Conf. Machine Learning*, pages 71–85, 2008.

Bader, B. W. and Kolda, T. G. Algorithm 862: MATLAB tensor classes for fast algorithm prototyping *ACM Transactions on Mathematical Software*, 32(4):635–653, 2006.

Baxter, J. A model for inductive bias learning. *J. of Artificial Intelligence Research*, 12:149–198, 2000.

Bertsekas, D.P. and Tsitsiklis, J.N. *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, 1989.

Caruana, R. Multi-task learning. *Machine Learning*, 28:41–75, 1997.

Ekman, P., Friesen, W. Facial Action Coding System: A Technique for the Measurement of Facial Movement. *Consulting Psychologists Press*, 1978.

Gandy, S., Recht, B., and Yamada, I. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27, 2011.

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

Kumar, A. and Daumé III, H. Learning task grouping and overlap in multitask learning. *International Conference on Machine Learning (ICML)*, 2012.

Liu, J., Musialski, P., Wonka, P., and Ye, J. Tensor completion for estimating missing values in visual data. *Proc. 12th International Conference on Computer Vision* (ICCV), pages 2114–2121, 2009.

Lucey, P. and Cohn, J.F. and Prkachin, K.M. and Solomon, P.E., and Matthews, I. PAINFUL DATA: The UNBC-McMaster Shoulder Pain Expression Archive Database. *IEEE Facial and Gesture (FG)*, pages 57–64, 2011.

van der Maaten, L. Audio-Visual Emotion Challenge 2012: A Simple Approach. *Workshop ICMI 12*, 2012.

Maurer, A. Transfer bounds for linear feature learning. *Machine Learning*, 75(3):327–350, 2009.

Maurer, A. and Pontil, M. Excess risk bounds for multitask learning with trace norm regularization. arXiv:1212.1496, 2012.

Maurer, A., Pontil, M., Romera-Paredes, B. Sparse coding for multitask and transfer learning. *International Conference on Machine Learning (ICML)*, 2013.

Mpiperis, I., Malassiotis, S., and Strintzis, M.G. Bilinear elastically deformable models with application to 3D face and facial expression recognition. *Proc. 8th International Conference on Automatic Face and Gesture Recognition*, pages 1–8, 2008.

Romera-Paredes, B., Argyriou A., Bianchi-Berthouze, N., Pontil, M. Exploiting unrelated tasks in multitask learning. *JMLR - Proceedings Track*, 22:951–959, 2012.

Signoretto, M., De Lathauwer, L., Suykens, J.A.K. Nuclear norms for tensors and their use for convex multilinear estimation, Technical Report, 2012.

Signoretto, M., Tran Dinh, Q., De Lathauwer, L., Suykens, J.A.K. Learning with tensors: a framework based on convex optimization and spectral regularization. *Machine Learning*, to appear.

Signoretto, M., Van de Plas, R., De Moor, B., and Suykens, J.A.K. Tensor versus matrix completion: a comparison with application to spectral data. *IEEE Signal Processing Letters*, 18(7):403–406, 2011.

Tenenbaum, J.B. and Freeman, W.T. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.

Tomioka, R., Hayashi, K., Kashima, H., Presto, J.S.T. Estimation of Low-Rank Tensors via Convex Optimization. 2010.

Vargas-Govea, B. and González-Serna, G. and Ponce-Medellín, R. Effects of relevant contextual features in the performance of a restaurant recommender system. *RecSys 11: Workshop on Context Aware Recommender Systems (CARS-2011)*, 2011.

Vasilescu, M. A. O. and Terzopoulos, D. Multilinear image analysis for facial recognition. *Proc. 16th International Conference on Pattern Recognition* (ICPR), pages 511–514, 2002.

Vasilescu, M. A. O. and Terzopoulos, D. Multilinear independent components analysis. *Proc. 2005 Conference on Computer Vision and Pattern Recognition* (CVPR), pages 547–553, 2005.

A. Solution of ADM

The underlying algorithm to solve problem (8) is based on the ADM method, (see e.g. Bertsekas & Tsitsiklis, 1989). It consists of iteratively applying the update equations

$$\text{(a)}\quad \boldsymbol{W}^{[i+1]} \leftarrow \operatorname*{argmin}_{\boldsymbol{W}} \mathcal{L}\left(\boldsymbol{W}, \mathbf{C}^{[i]}\boldsymbol{B}^{[i]}\right)$$

$$\text{(b)}\quad \boldsymbol{B}_n^{[i+1]} \leftarrow \operatorname*{argmin}_{\boldsymbol{B}_n} \mathcal{L}\left(\boldsymbol{W}^{[i+1]}, \mathbf{C}^{[i]}, \boldsymbol{B}\right)$$

$$\text{(c)}\quad \mathbf{C}_n^{[i+1]} \leftarrow \mathbf{C}_n^{[i]} - \left(\beta\boldsymbol{W}^{[i+1]} - \boldsymbol{B}_n^{[i+1]}\right)$$

for $n = 1, \ldots, N$, where $\mathcal{L}$ is the augmented Lagrangian for problem (8) and is defined in equation (7).

We now discuss each of these steps in turn.

Minimizing over $\boldsymbol{W}$

In order to solve Step (a), we need to solve the problem

$$\min_{\boldsymbol{W}}\left\{ F(\boldsymbol{W}) - \sum_{n=1}^{N}\left(\langle \mathbf{C}_n, \boldsymbol{W} - \boldsymbol{B}_n\rangle + \tfrac{\beta}{2}\left\|\boldsymbol{W} - \boldsymbol{B}_n\right\|_{\mathrm{Fr}}^2\right)\right\}$$

which is equal to

$$\min_{\boldsymbol{W}}\left\{F(\boldsymbol{W}) - \left\langle \sum_{n=1}^{N}\mathbf{C}_n + \beta\boldsymbol{B}_n, \boldsymbol{W}\right\rangle + \frac{N\beta}{2}\left\|\boldsymbol{W}\right\|_{\mathrm{Fr}}^2 + c\right\},$$

for some constant c whose value is independent of $\boldsymbol{W}$.

Notice that the terms where the whole tensor $\boldsymbol{W}$ appears are both the square of its Frobenius norm and inner products with other tensors. By using the definition of the tensor inner products, it is easy to see that in both cases we can decouple the whole tensor $\boldsymbol{W}$ in terms of the fibers of its mode-1 unfolding, that is the original tasks weight vectors: $\langle \boldsymbol{Z}, \boldsymbol{W}\rangle = \sum_{t=1}^{T}\langle \boldsymbol{Z}_{:,t}, \boldsymbol{W}_{:t}\rangle, \forall \boldsymbol{Z} \in \mathbb{R}^{p_1\times\cdots\times p_N}$. Consequently, solving the above optimization problem is equivalent to solving the following $T = p_2p_3\ldots p_N$ minimization problems

$$\min_{w} \textstyle\sum_{i=1}^{m_t} L\left(\langle x_i^t, w_t\rangle, y_i^t\right) - \left\langle \left(\sum_{n=1}^{N}\mathbf{C}_n + \beta\boldsymbol{B}_n\right)_{(1),t}, w\right\rangle + \frac{N\beta}{2}\left\|w_t\right\|_{\mathrm{Fr}}^2, \tag{11}$$

for all $t \in \{1, \ldots, T\}$, where we use the notation $w_t = \hat{\boldsymbol{W}}_{(1),t}$. In particular, if we consider one half of the square loss function, then the solution to problem (11) has the close form

$$w_t = \left(X^tX^{t\top} + N\beta I\right)^{-1}\left[X^ty^t + \left(\sum_{n=1}^{N}\mathbf{C}_n + \beta\boldsymbol{B}_n\right)_{(1),t}\right]$$

where X^t is the $d \times m_t$ data matrix for task t, that is, the columns of X^t are the inputs x_i^t, $i = 1, \ldots, m_t$, and $y^t = (y_1^t, \ldots, y_{m_t}^t)^\top$.

Minimizing over $\boldsymbol{B}_n$

Minimizing equation (7) over $\boldsymbol{B}_n$ is equivalent to the problem

$$\min_{B_{n\,(n)}} \gamma\left\|B_{n\,(n)}\right\|_{1,1} - \langle C_{n\,(n)}, W_{(n)} - B_{n\,(n)}\rangle + \frac{\beta}{2}\left\|W_{(n)} - B_{n\,(n)}\right\|_{\mathrm{Fr}}^2$$

which is the same as the problem

$$\min_{B_{n\,(n)}} \frac{\gamma}{\beta}\left\|B_{n\,(n)}\right\|_{1,1} + \left\langle \frac{1}{\beta}C_{n(n)} - W_{(n)}, B_{n(n)}\right\rangle + \frac{1}{2}\left\|B_{n\,(n)}\right\|_{\mathrm{Fr}}^2 + Q_1$$

which in turn equals to

$$\min_{B_{n\,(n)}} \tfrac{\gamma}{\beta}\left\|B_{n\,(n)}\right\|_{1,1} + \tfrac{1}{2}\left\|B_{n\,(n)} - \left(\tfrac{1}{\beta}C_{n(n)} - W_{(n)}\right)\right\|_{\mathrm{Fr}}^2 + Q_2, \tag{12}$$

for some constant matrices $Q_1, Q_2 \in \mathbb{R}^{p_n\times J_n}$. The solution to problem (12) is given by (Gandy et al., 2011) as

$$\hat{B}_{n\,(n)} = \operatorname{shrink}\left(\tfrac{1}{\beta}C_{n(n)} - W_{(n)},\ \tfrac{\gamma}{\beta}\right),$$

where shrink (M, k) is a function that shrinks the eigenvalues of the matrix M by k. That is, given $M = U\Sigma V^T$, where Σ is a diagonal matrix containing the singular values of M, then shrink $(M, k) = US_k(\Sigma)V^T$, where $S_k(\Sigma) = \operatorname{diag}(\max\{\Sigma_{i,i} - k,\ 0\})$.