TensorTest2D: Fitting Generalized Linear Models with Matrix Covariates

by Ping-Yang Chen, Hsing-Ming Chang, Yu-Ting Chen, Jung-Ying Tzeng, and Sheng-Mao Chang

Abstract The **TensorTest2D** package provides the means to fit generalized linear models on second-order tensor type data. Functions within this package can be used for parameter estimation (e.g., estimating regression coefficients and their standard deviations) and hypothesis testing. We use two examples to illustrate the utility of our package in analyzing data from different disciplines. In the first example, a tensor regression model is used to study the effect of multi-omics predictors on a continuous outcome variable which is associated with drug sensitivity. In the second example, we draw a subset of the MNIST handwritten images and fit to them a logistic tensor regression model. A significance test characterizes the image pattern that tells the difference between two handwritten digits. We also provide a function to visualize the areas as effective classifiers based on a tensor regression model. The visualization tool can also be used together with other variable selection techniques, such as the LASSO, to inform the selection results.

1 Introduction

Tensors are multidimensional arrays and are increasingly encountered in practices due to the burgeoning development of high throughput technology, e.g., brain image data (Zhou et al., 2013) and multi-omics data (Chang et al., 2021). Within the framework of regression analysis, tensor-structured data can play a role in the response variable, the explanatory variable, or in both. Some available R packages, such as **TRES** and **MultiwayRegression**, consider tensor regression with general tensor structure. The package **TRES** (Wang et al., 2020) provides tools to perform regression analysis with a tensor envelope structure in the tensor regression model, and the output of which includes p-values for the regression coefficients. **TRES** aims at variable selection via significance tests. The package **MultiwayRegression** (Lock, 2018, 2019) performs L_2 penalized tensor regression which is useful for predictive modeling but not for the identification of important variables. Both the **TRES** and the **MultiwayRegression** consider regression models with continuous outcome variables only. Compared to existing R packages, the proposed package **TensorTest2D** (Chen et al., 2021) considers a generalized linear model (GLM) with matrix-structured predictors and a scalar outcome, and it can be used for outcome prediction or testing.

There are four main functions in **TensorTest2D**. The function `tensorReg2D()` is designed to provide estimates of regression coefficients and their standard deviations, as well as the p-value for testing whether a regression coefficient is significantly different from zero. The function `summary()` organizes the above information into an output table. The function `plot()` can be used to visualize the locations of the predictors significantly affecting the response variable in the predictor matrix. Finally, the function `predict()` can be used to predict the response values using the conditional mean given a specific predictor matrix.

The rest of this paper is arranged as follows. First, we describe a regression model with tensor predictors under GLM. Next, we illustrate the main functions in package **TensorTest2D** using two examples, and illustrate the relevancy of using low-rank tensor regression. The first example focuses on association testing, where we apply tensor regression to identify genomic variables that affect the drug sensitivity for lung cancer treatment. The second example is for classification, where we apply logistic tensor regression to model the relationship between a binary response variable and images of handwritten digits in the MNIST database. We also use significance testing of the image predictors to identify locations of an image that play important roles in distinguishing between two different handwritten digits. The datasets being used in these two examples are included in the package **TensorTest2D** — users can use `data(omics)` to load the dataset of the first (association) example and `data(mnist_mp2c2)` to load the dataset of the second (classification) example.

2 Generalized tensor regression model

In this work, we consider tensor regression under the GLM framework and extend the inference procedure of tensor parameters in Chang et al. (2021) from continuous responses to binary and count responses. Suppose that there exists a dataset consisting of n independent triplets, $\{y_i, \mathbf{w}_i, \mathbf{X}_i\}$, $i = 1, \ldots, n$, where y_i is a scalar outcome, $\mathbf{w}_i$ is a d-dimensional covariate vector and $\mathbf{X}_i$ is a $P \times Q$ matrix predictor. Without loss of generality, we assume hereafter $P \leq Q$. For two matrices, say $\mathbf{X}_1$ and

$\mathbf{X}_2$, of the same size, define the dot product of these two matrices as $\mathbf{X}_1 \circ \mathbf{X}_2 = \sum_{p=1}^{P} \sum_{q=1}^{Q} X_{1,pq} X_{2,pq}$ where $X_{j,pq}$ is the (p,q)th entry of the matrix $\mathbf{X}_j$, $j = 1, 2$. The GLM with an order-2 tensor predictor can therefore be defined as

$$g\left(E(y_i)\right) = \mathbf{w}_i^\top \boldsymbol{\beta} + \mathbf{X}_i \circ \mathbf{B}, \tag{1}$$

where $g(\cdot)$ is the link function, $\boldsymbol{\beta} \in \mathbb{R}^d$ and $\mathbf{B} \in \mathbb{R}^{P \times Q}$. If there are $P \times Q$ unconstrained parameters in $\mathbf{B}$, the above representation is equivalent to a glm with $d + PQ$ covariates, including the intercept. In **TensorTest2D**, we implement linear regression with identity link, Poisson regression with log link, and logistic regression with logit link.

When PQ is relatively small, one can vectorize the matrix $\mathbf{X}_i$ so that (1) can be expressed as a conventional GLM with $d + PQ$ covariates. When PQ is large, one can explore the matrix structure of predictors and consider a low-rank tensor GLM so as to reduce the number of parameters of interest while retaining the variable-specific resolution. The main idea of tensor GLM is to model $\mathbf{B}$ by a low-rank-constrained $\mathbf{B}$ so that $\mathbf{B}$ can be fully specified using fewer unconstrained parameters. See Hung and Wang (2012) and Chang et al. (2021) for more detail. Take the MNIST handwritten image classification as an example, where handwritten images are recorded in 10×10 matrices. When there is no constraint on $\mathbf{B}$, the number of parameters of interest is $PQ = 100$. On the other hand, if we restrict the rank of $\mathbf{B}$ to be r, the number of unconstrained parameters is $(P + Q) \times r - r^2$ that takes a value of 19, 36, 51, and 64 when $r = 1$, 2, 3, and 4, respectively. The reduction in the number of parameters is significant when r is small.

Given a pre-specified rank $r = R$, one can model $\mathbf{B}$ with a low-rank constraint by setting $\mathbf{B} = \mathbf{B}_1 \mathbf{B}_2^\top$, where $\mathbf{B}_1 \in \mathbb{R}^{P \times R}$, $\mathbf{B}_2 \in \mathbb{R}^{Q \times R}$ and $R \leq P$ (Zhou et al., 2013; Chang et al., 2021). The low-rank tensor regression model is therefore

$$g\left(E(y_i)\right) = \mathbf{w}_i^\top \boldsymbol{\beta} + \mathbf{X}_i \circ \left(\mathbf{B}_1 \mathbf{B}_2^\top\right). \tag{2}$$

Additional constraints on $\mathbf{B}_1 \mathbf{B}_2^\top$ are needed to ensure parameter identifiability, see Zhou et al. (2013) for example. We adopt in this package the constraints considered by Chang et al. (2021), that leaves the total number of unconstrained parameters to be $d + (P + Q)R - R^2$. Denote $\boldsymbol{\eta}$ as the vector which collects all unconstrained parameters in (2). According to the theory of GLM, the score function and the Fisher information matrix with respect to the model are

$$\sum_{i=1}^{n} \frac{\partial \mu_i}{\partial \boldsymbol{\eta}} (y_i - \mu_i) \quad \text{and} \quad \sum_{i=1}^{n} \frac{\partial \mu_i}{\partial \boldsymbol{\eta}} V_i \left(\frac{\partial \mu_i}{\partial \boldsymbol{\eta}}\right)^\top,$$

respectively, where $\mu_i = E(y_i)$ and $V_i = Var(y_i)$. However, we are interested in estimating $\mathbf{B} = \mathbf{B}_1 \mathbf{B}_2^\top$ and testing whether entries of $\mathbf{B}$ are nonzero. The derivation of the sampling distribution of $\hat{\mathbf{B}}$ is omitted here, for the process is similar to that in Chang et al. (2021) and it does not need to be reproduced here again to misdirect readers' attention. As the true rank of $\mathbf{B}$ is unknown, following Chang et al. (2021), we use the Akaike information criterion (AIC) to determine the optimal rank.

Before giving a brief description of how Chang et al. (2021) place constraints on tensor regression parameterization, we wish to emphasize that the process of estimating for $\partial \mu_i / \partial \boldsymbol{\eta}$ is in typical not exactly simple. It is known that the matrix factorization (decomposition) $\mathbf{B} = \mathbf{B}_1 \mathbf{B}_2^\top$ is not unique because, for every invertible matrix $\mathbf{O} \in \mathbb{R}^{R \times R}$, $\mathbf{B} = \left(\mathbf{B}_1 \mathbf{O}^{-1}\right)\left(\mathbf{O}\mathbf{B}_2{}^\top\right)$. Write

$$\mathbf{B}_1 = \begin{bmatrix} \mathbf{B}_{11} \\ \mathbf{B}_{21} \end{bmatrix},$$

where $\mathbf{B}_{11} \in \mathbb{R}^{R \times R}$ and $\mathbf{B}_{21} \in \mathbb{R}^{(P-R) \times R}$, and assume $\mathbf{B}_{11}$ is invertible. One way to ensure the uniqueness of the matrix factorization is to force $\mathbf{O} = \mathbf{B}_{11}$, and thus

$$\mathbf{B}_1 \mathbf{B}_2^\top = \begin{bmatrix} \mathbf{B}_{11} \\ \mathbf{B}_{21} \end{bmatrix} \mathbf{O}^{-1} \mathbf{O} \mathbf{B}_2^\top = \begin{bmatrix} \mathbf{B}_{11}\mathbf{O}^{-1} \\ \mathbf{B}_{21}\mathbf{O}^{-1} \end{bmatrix} \tilde{\mathbf{B}}_2^\top = \begin{bmatrix} \mathbf{I}_R \\ \tilde{\mathbf{B}}_{21} \end{bmatrix} \tilde{\mathbf{B}}_2^\top,$$

where $\tilde{\mathbf{B}}_{21} = \mathbf{B}_{21}\mathbf{O}^{-1}$ and $\tilde{\mathbf{B}}_2 = \mathbf{B}_2 \mathbf{O}^\top$. Consequently, the unknown parameter matrices are $\tilde{\mathbf{B}}_{21} \in \mathbb{R}^{(P-R) \times R}$ and $\tilde{\mathbf{B}}_2 \in \mathbb{R}^{Q \times R}$ with a total of $(P - R) \times R + Q \times R$ unknown parameters. We believe that an exact formula for $\partial \mu_i / \partial \tilde{\boldsymbol{\eta}}$ can not be found prior to the work by Chang et al. (2021) in the case when $\tilde{\boldsymbol{\eta}} = (vec(\tilde{\mathbf{B}}_{12})^\top, vec(\tilde{\mathbf{B}}_1)^\top)^\top$, and the same formula is used in **TensorTest2D**.

3 Data analysis examples

In this section, we present examples of real data analysis by using the package **TensorTest2D**. The main function, `tensorReg2D`, in our **TensorTest2D** package is used for following data analysis. The inputs are the response vector, `y`, covariates matrix `X`, collecting tensor data, and vector `W`, collecting adjustment information such as age and gender. The key configurable parameters are the rank of $\mathbf{B}$, `n_R`, and the type of response variable, `family`. The `tensorReg2D` handles three types of generalized regression problems. For continuous response, set `family = "gaussian"` and it fits the linear regression model based on identity link function. If the responses are binary, by setting `family = "binomial"`, it runs logistic regression modeling through the logit link. When the response variable is non-negative integer, the log link corresponding to poisson regression is used by setting `family = "poisson"`.
The function `tensorReg2D()` returns a list object which includes the following variables: `b_EST` represents the coefficient vector $\hat{\beta}$; `b_SD` represents the the corresponding standard deviation vector and `b_PV` the p-value vector; `B_EST` represents the coefficient matrix $\hat{\mathbf{B}}$ for the image effect, `B_SD` the standard deviations of the coefficient estimates and `B_PV` the matrices of p-values; the output `IC` contains the AIC and BIC values for the purpose of model selection.
See `?tensorReg2D` for more details of the configuration and the output values.

Example 1: Tensor regression for continuous response using CCLE dataset

The package **TensorTest2D** includes a data set, `omics`, which consists of a continuous response variable and 30 omics predictors that can be organized into a 3×10 matrix. The response variable is the drug sensitivity of vandetanib measured by log-transformed activity area. Vandetanib is a drug targeting gene EGFR for lung cancer treatment. The 30 omics predictors are the genomic information of 10 genes measured from 3 platforms: copy number variation (CNV), methylation and mRNA expression. Among the 10 genes, 7 of them (EGFR, EREG, HRAS, KRAS, PTPN11, STAT3, and TGFA) are involved in the protein-protein interaction network of EGFR (`https://string-db.org`) and the rest (ACTB, GAPDH, and PPIA) are arbitrarily chosen housekeeping genes with permuted entries and serve as negative controls. The included data, omics.RData, is a subset of the data set provided by cancer cell line encyclopedia (CCLE) project (Barretina et al. (2012); `https://sites.broadinstitute.org/ccle/`). Detailed pre-processing procedure for `omics` is available in (Chang et al., 2021). The data set `omics` can be loaded via the following syntax:

```
library(TensorTest2D)
data(omics)
# The size of the data P, Q, n
print(dim(omics$omics))

#> [1]  3 10 68
```

In the omics example, $\mathbf{w}_i$ only consists of intercepts and $\mathbf{X}_i$ being a $P \times Q$ matrix with $P = 3$ and $Q = 10$. As described, this matrix consists of expression values of 10 genes evaluated under three different platforms. For the reason of $R \leq \min\{P, Q\}$ (see Chang et al. (2021)), there are three possible tensor models, namely, the rank-1, the rank-2 and the rank-3 model, to describe the relationship between the outcome and the matrix predictors. The models with the smallest AIC value will be selected as the optimal one, and here the rank-1 model has the smallest AIC value. The rank-1 model identifies two important variables: EGFR under methylation platform (coefficient = -0.2416; p-value = 0.0022) and EGFR under CNV platform (coefficient = 0.2508; p-value = 0.0061). Those lines of code below this paragraph were used as an example to perform and print out the results of model fitting. The utility function `summary(omicsMdl)` shows the model structure, summary statistics about the residuals, and the table of significance tests for the coefficients. On top of the result table, the model structure `y ~ I + X` of this case is revealed where `I` is the intercept term and `X` is the matrix covariate. The names of the coefficients appear in the first column of the table. In addition to the `(Intercept)` and the terms of $\mathbf{w}$, `Xi.j` is the coefficient of the ith row and jth column of $\mathbf{X}$. If the row and the column names of $\mathbf{X}$ are specified, then the names of coefficients in $\mathbf{X}$ are `ROWi:COLUMNj`. Those values in the summary table can also be returned separately. Here, we print the attributes of the `tsglm` object separately for the estimated coefficients and their standard deviations, along with the p-values by the Wald test (see Wald (1943)).

```
set.seed(100) # Set seed for reproducibility
# Try from rank-1 to rank-3 models
omicsAIC <- numeric(3)
for (k in 1:3) {
  # Temporary storage for the rank-k model for withdrawing its AIC value
```

```
  omicsTmp <- tensorReg2D(y = omics$Y, X = omics$omics,
                          W = matrix(1, length(omics$Y), 1),
                          n_R = k, family = "gaussian",
                          opt = 1, max_ite = 1000, tol = 10^(-7) )
  omicsAIC[k] <- omicsTmp$IC[1] # AIC
}
sprintf('Rank-%d model is the best with smallest AIC = %4.4f', which.min(omicsAIC), min(omicsAIC))

#> [1] "Rank-1 model is the best with smallest AIC = -62.3135"

# Train the tensor regression model of rank 1
omicsMdl <- tensorReg2D(y = omics$Y, X = omics$omics,
                        W = matrix(1, length(omics$Y), 1),
                        n_R = which.min(omicsAIC), family = "gaussian",
                        opt = 1, max_ite = 1000, tol = 10^(-7) )

# Return the results of significance tests for all coefficients
summary(omicsMdl)

#> Call:
#> formula =  y ~ I + X
#>
#> Residuals:
#>     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
#> -1.31835 -0.29160  0.03354  0.00000  0.40356  1.06511
#>
#> Coefficients:
#>                 Estimate Std. Error  t value  Pr(>|t|)
#> (Intercept)     -0.06078    0.07044 -0.86285 0.3919672
#> cnv:EGFR         0.25078    0.08796  2.85098 0.0061255 ***
#> meth:EGFR       -0.24162    0.07511 -3.21673 0.0021740 ***
#> rna.rpkm:EGFR    0.08933    0.07857  1.13696 0.2604852
#> cnv:EREG        -0.00751    0.05094 -0.14743 0.8833301
#> meth:EREG        0.00724     0.0494  0.14648 0.8840812
#> rna.rpkm:EREG   -0.00268    0.01849 -0.14468 0.8854906
#> cnv:HRAS        -0.04866    0.05983 -0.81334 0.4195282
#> meth:HRAS        0.04689    0.06056  0.77425 0.4421012
#> rna.rpkm:HRAS   -0.01733    0.02956  -0.5865 0.5599385
#> cnv:KRAS         -0.0267    0.05067 -0.52699 0.6003203
#> meth:KRAS        0.02573    0.04913  0.52367 0.6026098
#> rna.rpkm:KRAS   -0.00951    0.01754 -0.54244 0.5897064
#> cnv:PTPN11       0.09193    0.06183  1.48669 0.1428065
#> meth:PTPN11     -0.08857    0.05093 -1.73886 0.0876539   *
#> rna.rpkm:PTPN11  0.03275    0.03289  0.99558 0.3238137
#> cnv:STAT3       -0.05747    0.05517  -1.0417 0.3021072
#> meth:STAT3       0.05537    0.04953  1.11792 0.2684585
#> rna.rpkm:STAT3  -0.02047     0.0257 -0.79672 0.4290423
#> cnv:TGFA         0.05049    0.06361  0.79367 0.4307973
#> meth:TGFA       -0.04864    0.05903 -0.82399 0.4135018
#> rna.rpkm:TGFA    0.01798    0.02625  0.68526 0.4960557
#> cnv:ACTB        -0.03107    0.05212  -0.5961 0.5535539
#> meth:ACTB        0.02993    0.05128  0.58371 0.5618033
#> rna.rpkm:ACTB   -0.01107    0.02339 -0.47307 0.6380374
#> cnv:GAPDH       -0.04123    0.04963 -0.83059 0.4097953
#> meth:GAPDH       0.03972    0.04737  0.83856 0.4053450
#> rna.rpkm:GAPDH  -0.01469    0.02221 -0.66128 0.5111932
#> cnv:PPIA        -0.04299    0.06373 -0.67454 0.5027957
#> meth:PPIA        0.04142    0.05989  0.69153 0.4921444
#> rna.rpkm:PPIA   -0.01531    0.02711 -0.56477 0.5745259
#> ---
#>  Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Estimated coefficients
print(round(omicsMdl$B_EST, 3))
```

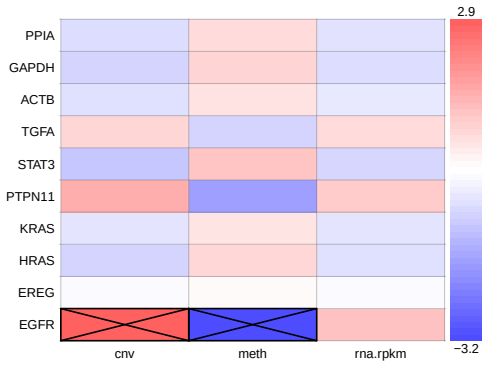

Figure 1: The image plot of the values for t-statistics of matrix covariate in the omics data. The effective pixels identified by the tensor regression model are marked out by the ☒ symbol.

```
#>            EGFR   EREG   HRAS   KRAS PTPN11  STAT3   TGFA   ACTB  GAPDH   PPIA
#> cnv       0.251 -0.008 -0.049 -0.027  0.092 -0.057  0.050 -0.031 -0.041 -0.043
#> meth     -0.242  0.007  0.047  0.026 -0.089  0.055 -0.049  0.030  0.040  0.041
#> rna.rpkm  0.089 -0.003 -0.017 -0.010  0.033 -0.020  0.018 -0.011 -0.015 -0.015

# The standard deviation of the coefficients
print(round(omicsMdl$B_SD, 3))

#>           EGFR  EREG  HRAS  KRAS PTPN11 STAT3  TGFA  ACTB GAPDH  PPIA
#> cnv      0.088 0.051 0.060 0.051  0.062 0.055 0.064 0.052 0.050 0.064
#> meth     0.075 0.049 0.061 0.049  0.051 0.050 0.059 0.051 0.047 0.060
#> rna.rpkm 0.079 0.018 0.030 0.018  0.033 0.026 0.026 0.023 0.022 0.027

# The p-values of the coefficients by the Wald test
print(round(omicsMdl$B_PV, 3))

#>           EGFR  EREG  HRAS  KRAS PTPN11 STAT3  TGFA  ACTB GAPDH  PPIA
#> cnv      0.006 0.883 0.420 0.600  0.143 0.302 0.431 0.554 0.410 0.503
#> meth     0.002 0.884 0.442 0.603  0.088 0.268 0.414 0.562 0.405 0.492
#> rna.rpkm 0.260 0.885 0.560 0.590  0.324 0.429 0.496 0.638 0.511 0.575
```

In our package **TensorTest2D**, the function `plot()` can be used to visualize the importance of the matrix predictor. The output is an $P \times Q$ heat map that the plotted values on it are controlled by the option `type`. If the unit of data varies across the rows or columns in X, it is suggested to choose the t-statistics of the coefficients (`type = "tval"`) instead of their values `type = "coef"`. In addition, the function `plot()` also marks the pixels with p-values smaller than a pre-determined significance level. Users can select the p-value adjusting method (see `help("p.adjust")`) by the option `method` and specify the significance level through the option `alpha`. We plot in Figure 1 the t-statistics of the coefficients in $\hat{\mathbf{B}}$, where red and blue colors represent the pixels of positive and negative values, respectively. For those coefficients with p-values less than `alpha`, their corresponding pixels are marked with the symbol, ☒, in this example, cnv:EGFR and meth:EGFR.

```
plot(x = omicsMdl, method = "none", alpha = 0.05, type = "tval",
     showlabels = TRUE, plot.legend = TRUE)
```

Example 2: Logistic tensor regression and classification using MNIST dataset

MNIST is a well-known benchmark database for image recognition in machine learning. It consists of over 60,000 training images and 10,000 testing images. For R users, one can obtain the image data from the **dslabs** package (Irizarry and Gill, 2019). For the purpose of demonstration, we reduce the size of MNIST image data by a max-pooling step with 2×2 block, as shown in the images on the left and in the middle of Figure 2. The original 28×28 images are thus pixelated into 14×14 max-pooled images. Because pixels at the edges and the corners of the max-pooled images take no information across almost all images, we removed those pixels and ended up with $P \times Q = 10 \times 10$ sub-images as illustrated in the image on the right of Figure 2. The mean image of the pre-processed training set for each label in the MNIST database is shown in Figure 3. These pre-processed images of 10×10 pixels are included in the package **TensorTest2D** and can be imported using the following commands:

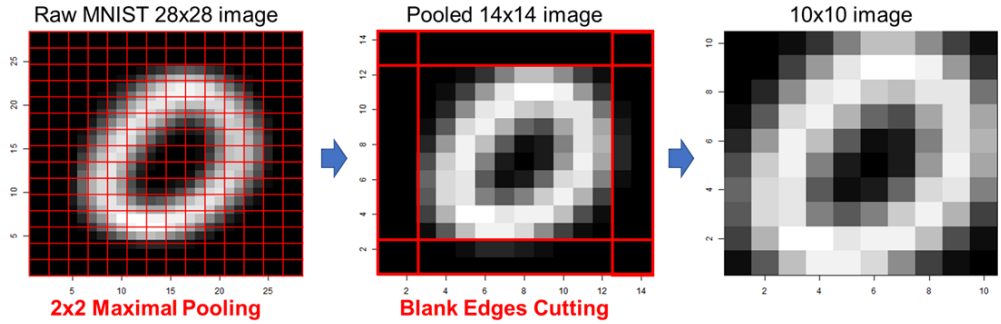

Figure 2: Data pre-processing for the MNIST dataset. First, the left subfigure shows the max-pooling step for reducing the image size. Next, the center subfigure shows the edge-cutting step for removing the noninformative pixels. Finally, the right subfigure shows the data pre-process result.

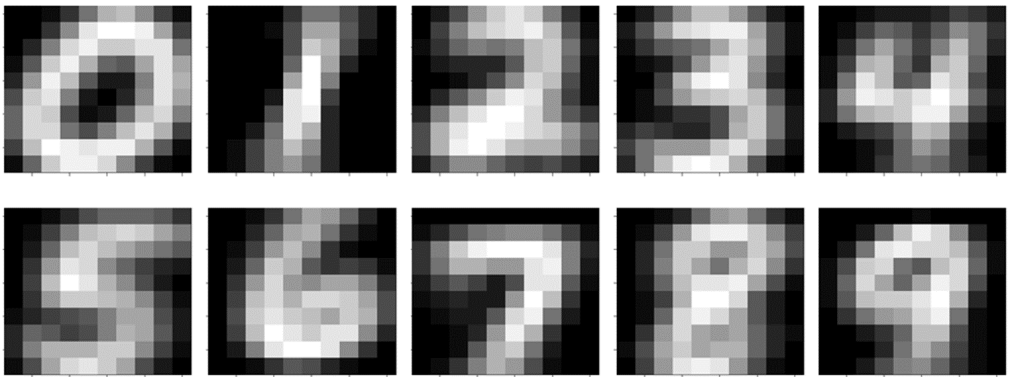

Figure 3: The mean plots of pre-processed images in the training dataset. The value at each pixel of the mean plot is the average grayscale value over the pre-processed images in the training dataset.

```
library(TensorTest2D)
data(mnist_mp2c2)
mnist_train <- mnist_mp2c2$train
mnist_test <- mnist_mp2c2$test
```

The aim of this data analysis is to recognize the digit for a given handwritten image using logistic regression. Here, we choose images of '2' and '5' for demonstration. Let $Y_i = 1$ if the ith image represents the digit '5', and $Y_i = 0$ if the ith image represents a '2'. The predictor matrix X_i here is a 10×10 matrix with its entries the pixel values of the handwritten image in grayscale. In the following, we first describe the data processing steps and provide the code being used to obtain our training data in the analysis. Our training data, `train_X`, is a $P \times Q \times n = 10 \times 10 \times 2000$ array, which contains subsets of 1,000 images of the digit '2' and 1,000 images of the digit '5' randomly chosen from the MNIST training set `mnist_train`. In this MNIST example, some pixels in the corners and on the edges take on the value zero across all handwritten images, which yields singularity when the alternating maximum likelihood algorithm is applied to the training data. To solve this problem, we can simply drop only those zero-valued pixels. However, doing so breaks the matrix form and hence low-rank model is no longer valid. Alternatively, we add independent standard normal noise to the images in our training data set that results in no significant harm to the prediction power, because the signal-noise ratio is high and the training set sample size is sufficiently large. Hereafter, we call images with random error the contaminated images.

```
library(abind)
# Draw image data of labels 2 and 5
x0_all <- mnist_train$image[,,which(mnist_train$label == 2)]
x1_all <- mnist_train$image[,,which(mnist_train$label == 5)]
# Random sampling from MNIST training set for each label
nSampleEach <- 1000
n0 <- dim(x0_all)[3]; n1 <- dim(x1_all)[3]
set.seed(2021)
s0 <- sample(1:n0, nSampleEach, replace = FALSE)
s1 <- sample(1:n1, nSampleEach, replace = FALSE)
```

```
# Normalizing image values into [-0.5, 0.5]
x0 <- x0_all[,,s0]/255 - 0.5
x1 <- x1_all[,,s1]/255 - 0.5
# Combine training data
train_X <- abind(x0, x1, along = 3)
# Add negligible noise for the images
# (so no constant zero values in one pixel over all covariate matrices)
set.seed(2021) # Set seed for reproducibility
train_n <- array((rnorm(prod(dim(train_X)), 0, 0.1)), dim(train_X))
train_Xn <- train_X + train_n # Contaminated images
# Define Y = 0 for label 2, and Y = 1 for label 5
train_y <- c(rep(0, dim(x0)[3]), rep(1, dim(x1)[3]))
```

In the package **TensorTest2D**, the function `tensorReg2D()` is also used for fitting the logistic tensor regression model to data:

$$\log \frac{Pr(Y_i = 1 \mid \mathbf{X}_i)}{Pr(Y_i = 0 \mid \mathbf{X}_i)} = \beta + \mathbf{X}_i \circ \mathbf{B}$$

Thus, a prediction for the digit presented in image $\mathbf{X}_i$ is

$$\hat{Y}_i = \underset{k \in \{0,1\}}{\arg\max}\, Pr(Y_i = k \mid \mathbf{X}_i) = I\{Pr(Y_i = 1 \mid \mathbf{X}_i) > 0.5\},$$

where $I\{E\} = 1$, if E is true, and $I\{E\} = 0$, otherwise. To analyze the sampled data set, first, we feed the response variable, `train_y`, and the contaminated images, `train_Xn`, as inputs for model training. There is no auxiliary information available to further stratify the y_i's, we specify a constant vector `W = matrix(1, length(train_y), 1)` of length n, and if a rank $R = 4$ model is needed, we set `n_R = 4`. (In fact, the rank-4 model is the best model in this logistic tensor regression.)

```
# Train the logistic tensor regression model
lgMdl <- tensorReg2D(y = train_y, X = train_Xn,
                     W = matrix(1, length(train_y), 1),
                     n_R = 4, family = "binomial",
                     opt = 1, max_ite = 100, tol = 10^(-7) )
# Print model summary (not run)
#summary(lgMdl)
# Print the p-values of the estimates
cat("FDR-adjusted p-values of B_pq:\n")

#> FDR-adjusted p-values of B_pq:

round(matrix(p.adjust(as.vector(lgMdl$B_PV), method = "fdr"), 10, 10), 3)

#>        [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10]
#>  [1,] 0.263 0.830 0.114 0.243 0.019 0.480 0.595 0.629 0.491 0.830
#>  [2,] 0.558 0.552 0.098 0.491 0.263 0.948 0.137 0.816 0.491 0.953
#>  [3,] 0.923 0.927 0.029 0.012 0.999 0.017 0.204 0.050 0.471 0.008
#>  [4,] 0.648 0.541 0.004 0.019 0.029 0.204 0.004 0.655 0.541 0.156
#>  [5,] 0.293 0.491 0.055 0.004 0.381 0.004 0.101 0.024 0.081 0.006
#>  [6,] 0.110 0.491 0.954 0.491 0.648 0.948 0.954 0.865 0.491 0.825
#>  [7,] 0.023 0.652 0.889 0.019 0.489 0.491 0.110 0.188 0.042 0.029
#>  [8,] 0.137 0.706 0.830 0.491 0.244 0.145 0.491 0.491 0.889 0.889
#>  [9,] 0.655 0.034 0.655 0.977 0.083 0.114 0.019 0.629 0.706 0.471
#> [10,] 0.137 0.055 0.491 0.764 0.491 0.602 0.019 0.471 0.454 0.025
```

For binary classification problems, we can apply the function `plot()` of our package **TensorTest2D** in two ways. Similar to that Figure 1, we can make a plot first for the values of t-statistics for the pixels by using the `plot()` function as shown below this paragraph. Here, we adjust the p-values according to the approach in Benjamini and Hochberg (1995) by setting the parameter `method = "fdr"`. The resulting plot is shown in Figure 4, and most of the effective pixels can be found on the left half side of the plot.

```
plot(x = lgMdl, method = "fdr", alpha = 0.05, type = "tval",
     showlabels = TRUE, plot.legend = TRUE)
```

To understand which areas of an image that contribute the most information to classify between labels 2 and 5, we also add a meaningful background image by specifying an argument to the parameter

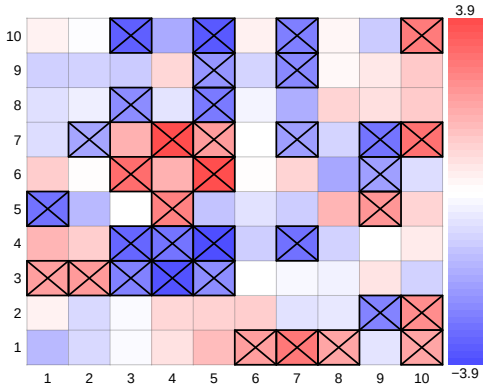

Figure 4: The image plot of the values for t-statistics of matrix covariate in the handwritten label data. The effective pixels identified by the logistic tensor regression model are marked out by the ⊠ marks.

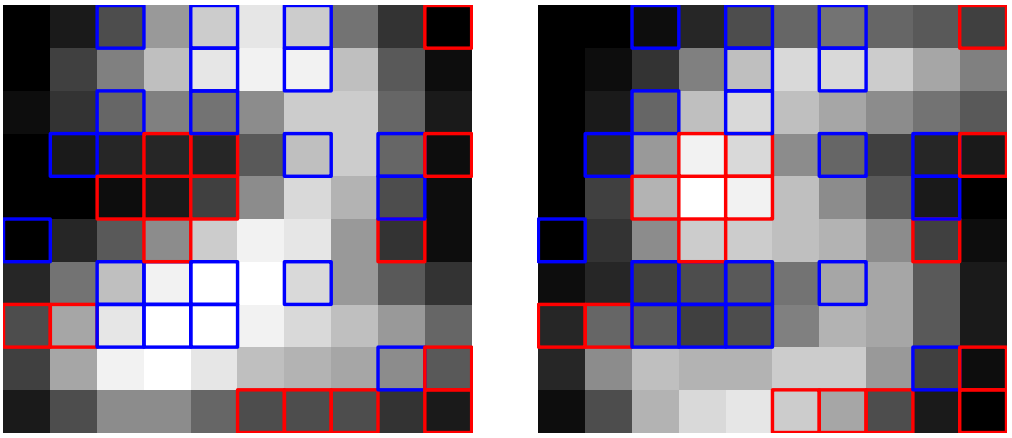

Figure 5: Effective pixels identified by the logistic tensor rgression model. The important pixels to discriminate between labels 2 and 5 are marked by red and blue frames, which indicate positive and negative coefficients, respectively.

`background` for the function `plot()`. In this example, we show separately the mean image of label 2 and the mean image of label 5 as the background image by assigning the value `xm0` or `xm1` to the parameter `background`. To adjust the visual style of the background image, one can assign the value `gray(0, 1, 0.05)` to the parameter `col` to create a grayscale colour map for contrast. Please refer to `help("image")` for more detail on the options available when creating a plot. Our resulting plots are shown in Figure 5. On top of both images, there are marks for the important pixels with red and blue frames. A red rectangle indicates that the corresponding estimate in $\hat{\mathbf{B}}$ has a significant positive coefficient and a blue one highlights a significant negative coefficient. In our example, important pixels are found to locate majorly at the curvy parts of 2 and 5.

```
xm0 <- xm1 <- matrix(0, dim(train_X)[1], dim(train_X)[2])
# Background image: mean image of label 2
for (k in 1:dim(x0)[3]) {
  xm0 <- xm0 + (1/nSampleEach)*x0[,,k]
}
# Background image: mean image of label 5
for (k in 1:dim(x1)[3]) {
  xm1 <- xm1 + (1/nSampleEach)*x1[,,k]
}
# Draw for visualizing effective pixels for both background images
par(mfrow = c(1, 2), mar = c(1, 1, 1, 1))
plot(x = lgMdl, method = "fdr", alpha = 0.05, background = xm0,
     showlabels = FALSE, plot.legend = FALSE, col = gray(seq(0, 1, 0.05)))
plot(x = lgMdl, method = "fdr", alpha = 0.05, background = xm1,
     showlabels = FALSE, plot.legend = FALSE, col = gray(seq(0, 1, 0.05)))
```

We use the function `predict()` to predict the label for the new images in the testing data set. The

input data must be a 3-dimensional array of size $P \times Q \times n^*$, where n^* is the number of testing images. We note here that, one need to reshape the $P \times Q$ matrix object into the 3-dimensional array by the R command `array(x, c(P, Q, 1))`. The function `predict()` returns the predictions in two ways. By setting the option `type = "link"`, it returns the values of the linear predictors; and by setting `type = "response"`, it returns the expected values of response variable. For example, for our logistic regression model, the predictions are log-odds (odds ratios on logarithmic scale) if `type = "link"` is chosen, or they are the predicted probabilities of $Y = 1$ if `type = "response"` is chosen.

```
# Normalize image values of the testing data into [-0.5, 0.5]
tx0 <- mnist_test$image[,,which(mnist_test$label == 2)]/255 - 0.5
tx1 <- mnist_test$image[,,which(mnist_test$label == 5)]/255 - 0.5
# Combine testing data and assign the vector of the true responses
test_X <- abind(tx0, tx1, along = 3)
test_y <- c(rep(0, dim(tx0)[3]), rep(1, dim(tx1)[3]))
# Print some predictions with different settings of type
pred_link <- predict(lgMdl, test_X, type = "link")
pred_prob <- predict(lgMdl, test_X, type = "response")
head(round(pred_link, digits = 2))
```

```
#>        [,1]
#> [1,]  -3.38
#> [2,] -16.24
#> [3,]  -4.93
#> [4,]  -5.38
#> [5,]  -9.94
#> [6,]  -6.41
```

```
head(round(pred_prob, digits = 4))
```

```
#>        [,1]
#> [1,] 0.0331
#> [2,] 0.0000
#> [3,] 0.0072
#> [4,] 0.0046
#> [5,] 0.0000
#> [6,] 0.0016
```

```
# Compute the prediction accuracy for the testing data
pred_test_y <- (pred_prob > .5)
cat(
  sprintf("Accuracy = %2.2f%%",
          100*sum(pred_test_y == test_y)/length(test_y)))
```

```
#> Accuracy = 96.10%
```

In addition, we provide a visualization tool that works with other methods in variable selection for 2D images. For example, the penalized regression via lasso (Tibshirani, 1996) is one of the popular approaches. Below are the codes that we implemented to train a LASSO model, including the use of the function `cv.glmnet()` (Friedman et al., 2010). The object `l1B` is the 10×10 array of LASSO estimates. Since LASSO tends to shrink small coefficients to zero, we treat those image pixels with zero-valued coefficients to be irrelevant to distinguish between images of 2 and 5. To visualize the effective pixels identified by LASSO, our package **TensorTest2D** provides the function `draw.coef()` to produce the marked image similar to that in Figure 5. Different from the function `plot.tsglm()`, users need to provide an input as the markers for effective pixels. The markers in our example here are the LASSO estimates, and by specifying `marks = l1B`, the pixels with non-zero coefficients are then marked. In addition, by specifying `markstyle = "bi-dir"`, as shown in Figure 6, the pixels marked out with red rectangles correspond to those of positive LASSO estimates, and the pixels with blue rectangles are those of negative LASSO estimates.

```
library(glmnet)
# Vectorize the hand-written images
xv <- t(sapply(1:dim(train_X)[3], function(k) as.vector(train_X[,,k])))
# Train the LASSO model using cross-validation
set.seed(2021) # Set seed for reproducibility
l1Mdl <- cv.glmnet(xv, train_y, family = "binomial", alpha = 1, standardize = FALSE)
# Draw the LASSO coefficients from the best model
```

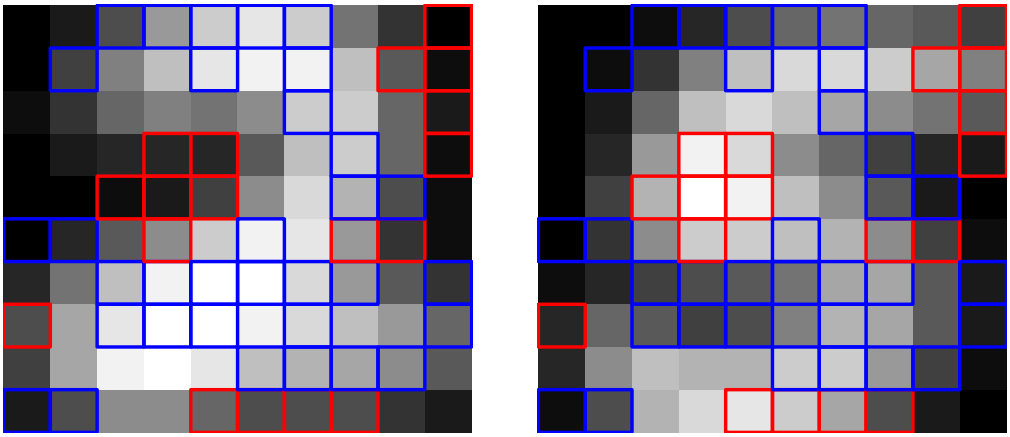

Figure 6: Effective pixels identified by the LASSO model. The important pixels to discriminate between labels 2 and 5 are marked by red and blue frames, which indicate positive and negative coefficients, respectively.

```
l1B <- matrix(l1Mdl$glmnet.fit$beta[,which.min(l1Mdl$cvm)], 10, 10)
# The LASSO estimates
print(round(l1B, digits = 3))
```

```
#>         [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]  [,10]
#>  [1,] -0.247  0.000  0.253  0.000 -0.409  0.000  0.000  0.000  0.000  0.000
#>  [2,] -0.559  0.000  0.000  0.000 -0.200  0.000  0.000  0.000 -0.691  0.000
#>  [3,]  0.000  0.000 -0.248 -0.513  0.000  1.287  0.000  0.000  0.000 -0.914
#>  [4,]  0.000  0.000 -1.526 -1.805  0.734  1.237  1.804  0.000  0.000 -0.254
#>  [5,]  0.142  0.000 -0.593 -0.795  0.000  1.280  0.929  0.000 -0.699 -0.738
#>  [6,]  1.625 -0.251  0.000 -1.429 -0.527  0.000  0.000  0.000  0.000 -0.592
#>  [7,]  0.592 -0.136 -0.668 -0.554  0.000  0.000  0.000 -0.406 -0.397 -0.285
#>  [8,]  0.074 -0.204  0.000 -0.757  0.265 -0.211 -1.082  0.000  0.000  0.000
#>  [9,]  0.000 -0.574  0.000  0.000  0.360 -1.558  0.000  0.000  0.638  0.000
#> [10,]  0.000  0.000 -0.628 -0.144  0.000  0.000  0.479  1.340  0.303  0.845
```

```
# Draw for visualizing effective pixels identified by LASSO for both background images
par(mfrow = c(1, 2), mar = c(1, 1, 1, 1))
draw.coef(img = xm0, marks = l1B, markstyle = "bi-dir", showlabels = FALSE,
          plot.legend = FALSE, grids = FALSE, col = gray(seq(0, 1, 0.05)))
draw.coef(img = xm1, marks = l1B, markstyle = "bi-dir", showlabels = FALSE,
          plot.legend = FALSE, grids = FALSE, col = gray(seq(0, 1, 0.05)))
```

4 Summary

Issues in estimation and test of hypothesis that emerged from fitting regression models with predictor variables that has a matrix form are of our major interest. Low-rank modelling can be applied to improve the efficiency of estimation. In this line, we developed the R package TensorTest2D to conduct tensor regression analysis within the framework of generalized linear models. In addition to model estimation and hypothesis testing, this package also includes a visualization tool that can be used to indicate the positions of effective or significant pixels when the tensor predictor is of image data type.

Bibliography

J. Barretina, G. Caponigro, N. Stransky, K. Venkatesan, A. A. Margolin, S. Kim, C. J. Wilson, J. Lehár, G. V. Kryukov, D. Sonkin, A. Reddy, M. Liu, L. Murray, M. F. Berger, J. E. Monahan, P. Morais, J. Meltzer, A. Korejwa, J. Jané-Valbuena, F. A. Mapa, J. Thibault, E. Bric-Furlong, P. Raman, A. Shipway, I. H. Engels, J. Cheng, G. K. Yu, J. Yu, P. Aspesi, M. de Silva, K. Jagtap, M. D. Jones, L. Wang, C. Hatton, E. Palescandolo, S. Gupta, S. Mahan, C. Sougnez, R. C. Onofrio, T. Liefeld, L. MacConaill, W. Winckler, M. Reich, N. Li, J. P. Mesirov, S. B. Gabriel, G. Getz, K. Ardlie, V. Chan, V. E. Myer, B. L. Weber, J. Porter, M. Warmuth, P. Finan, J. L. Harris, M. Meyerson, T. R. Golub,

M. P. Morrissey, W. R. Sellers, R. Schlegel, and L. A. Garraway. The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483:603–607, 2012. URL https://doi.org/10.1038/nature11003. [p155]

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: series B (Methodological)*, 57(1):289–300, 1995. [p159]

S.-M. Chang, M. Yang, W. Lu, Y.-J. Huang, Y. Huang, H. Hung, J. C. Miecznikowski, T.-P. Lu, and J.-Y. Tzeng. Gene-set integrative analysis of multi-omics data using tensor-based association test. *Bioinformatics*, 03 2021. URL https://doi.org/10.1093/bioinformatics/btab125. [p153, 154, 155]

M. Chen, S.-M. Chang, W. Lu, J.-Y. Tzeng, and P.-Y. Chen. *TensorTest2D: Fitting Second-Order Tensor Data*, 2021. URL https://CRAN.R-project.org/package=TensorTest2D. R package version 1.0.3. [p153]

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL http://www.jstatsoft.org/v33/i01/. [p161]

H. Hung and C.-C. Wang. Matrix variate logistic regression model with application to EEG data. *Biostatistics*, 1(14):189–202, 07 2012. URL https://doi.org/10.1093/biostatistics/kxs023. [p154]

R. A. Irizarry and A. Gill. *dslabs: Data Science Labs*, 2019. URL https://CRAN.R-project.org/package=dslabs. R package version 0.7.3. [p157]

E. F. Lock. Tensor-on-tensor regression. *Journal of Computationaland Graphical Statistics*, (27):638–647, 2018. doi: 10.1080/10618600.2017. [p153]

E. F. Lock. *MultiwayRegression: Perform Tensor-on-Tensor Regression*, 2019. URL https://CRAN.R-project.org/package=MultiwayRegression. R package version 1.2. [p153]

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: series B (Methodological)*, 58(1):267–288, 1996. [p161]

A. Wald. Tests of statistical hypotheses concerning several parameters when the number of observations is large. *Transactions of the American Mathematical Society*, 11 1943. URL https://doi.org/10.2307/1990256. [p155]

W. Wang, J. Zeng, and X. Zhang. *TRES: Tensor Regression with Envelope Structure and Three Generic Envelope Estimation Approaches*, 2020. URL https://CRAN.R-project.org/package=TRES. R package version 1.1.3. [p153]

H. Zhou, L. Li, and H. Zhu. Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502):540–552, 2013. doi: 10.1080/01621459.2013.776499. URL https://doi.org/10.1080/01621459.2013.776499. [p153, 154]

Ping-Yang Chen
Chimes AI
12F., No. 201-8, Dunhua N. Rd., Songshan Dist.,
Taipei City 105076, Taiwan
pychen@chimes.ai

Hsing-Ming Chang
Department of Statistics and Institute of Data Science, National Cheng Kung University
1 University Road,
Tainan 70101, Taiwan
nckuhmchang@ncku.edu.tw

Yu-Ting Chen
Department of Statistics, Purdue University
250 N. University St, West Lafayette,
IN 47907, United States of America
l50115011@gmail.com

Jung-Ying Tzeng
Department of Statistics and Bioinformatics Research Center, North Carolina State University

North Carolina State University
Raleigh NC, 27695, United States of America
jytzeng@ncsu.edu

Sheng-Mao Chang
Department of Statistics, National Taipei University
No. 151, University Rd., Sanxia Dist.,
New Taipei City 237303, Taiwan
Corresponding Author
smchang110@gm.ntpu.edu.tw