Multiway Classification with Real Data

Matthew Asisgress

2025-10-27

In this demonstration, we apply R package **cpfa** (Asisgress, 2025) to the MNIST dataset (LeCun, Cortes, and Burges, 1998; LeCun et al., 2002)—showing how to use the package to distinguish between digits of 2 and 3, a binary classification problem. We further apply the package to the Fashion MNIST dataset (Xiao, Rasul, and Vollgraf, 2017)—distinguishing among images of tops, trousers, and sandals, which is a multiclass classification problem.

A.) MNIST

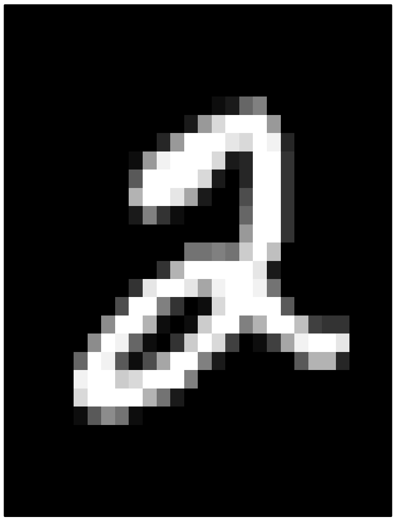
The MNIST dataset consists of 70,000 grayscale images of handwritten digits from 0 to 9. We use R package **dslabs** (Irizarry and Gill, 2025) to download the dataset. We subset to only 2000 images from the original training set. The images include 980 images of the digit 2 and 1020 images of the digit 3.

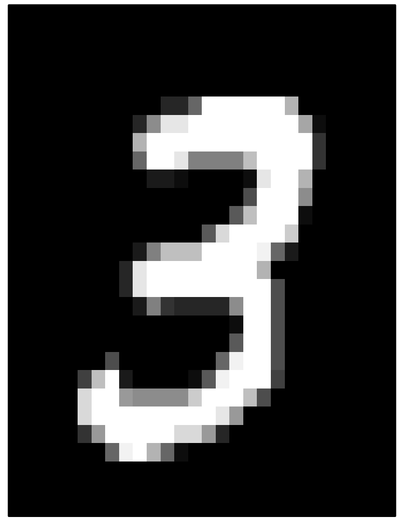
Download

```r
# load library
library(dslabs)

# download MNIST data and subset training set to digits of 2 or 3
mnist <- read_mnist()
inde <- which(mnist$train$labels %in% c(2, 3))
images <- mnist$train$images[inde, ]
labels <- mnist$train$labels[inde]

# restructure data into a three-way array and prepare labels
X0 <- array(images, c(nrow(images), 28, 28))
y0 <- as.factor(as.numeric(as.factor(labels)) - 1)

# subset to first 2000 images
ind <- 1:2e3
X <- X0[ind, , ]
y <- y0[ind]
```

We plot and examine an example of the digit 2 and the digit 3.

```r
# plot example of digits 2 and 3
par(mfrow = c(1, 2))
pdigit <- function(imat) {
  m <- t(apply(imat, 2, rev))
  image(m, col = gray(seq(0, 1, 0.05)), xaxt = "n", yaxt = "n")
}
for (i in 0:1) {pdigit(t(X[which(y == i)[1], , ]))}
```





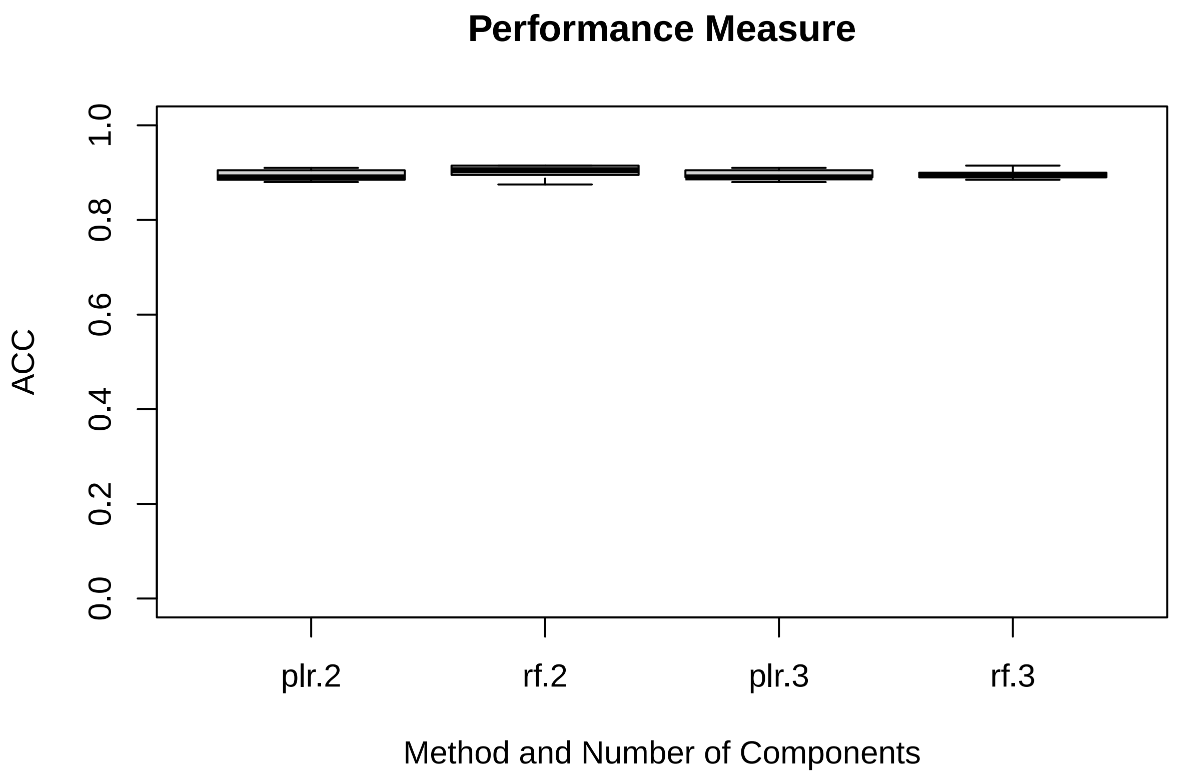
Analysis

We load R package **cpfa** and initialize the tensor model. First, the data array is a regular three-way array; so we set `model <- "parafac"` to use a Parafac model (Harshman, 1970). Second, we initialize the number of components to fit for the model by setting `nfac <- c(2, 3)` in order to fit both a two-component Parafac model and a three-component Parafac model. We set `nstart <- 10` to allow for 10 random starts in the Parafac alternating least squares algorithm fit by R package **multiway** (Helwig, 2025), upon which R package **cpfa** depends. Third, we specify the constraint desired for each array mode using `const`. Fourth, we use `cmode <- 1` to specify that the classification mode is the first mode of the input array (i.e., the mode connected to the class labels). Note that numerous constraint options are available; after loading **cpfa**, type `const()` in the R console to access a constraint options list provided through R package **CMLS** (Helwig, 2025).

Next, we initialize classification methods. First, we use `method = c("PLR", "RF")` to employ penalized logistic regression (PLR) and random forest (RF) classifiers for this problem. See `help(cpfa)` for information on additional classification methods available. Second, we specify that the problem is a binary classification problem by setting `family <- "binomial"`. Third, we use 10-fold cross-validation (CV) in our inner training, setting `nfolds <- 10`; and fourth, we set `nrep <- 5` to perform five outer train-test splits of our data. Fifth, we set `ratio <- 0.9` to specify that each outer training set contains a proportion of 0.9 of the full input data while the outer testing set contains a proportion of 0.1. Finally, we specify ranges for tuning parameters alpha (PLR), the number of trees (RF), and node size (RF), wrapping them into a list called `parameters`.

```r
# load library
library(cpfa)

# set seed
set.seed(500)

# initialize model
model <- "parafac"
nfac <- c(2, 3)
nstart <- 10
const <- c("uncons", "uncons", "uncons")
cmode <- 1

# initialize classification
method <- c("PLR", "RF")
family <- "binomial"
```

```r
nfolds <- 10
nrep <- 5
ratio <- 0.9

# initialize tuning parameters
alpha <- seq(0, 1, length = 8)
ntree <- c(400, 600, 800, 1000)
nodesize <- c(4, 8, 16, 32)
parameters <- list(alpha = alpha, ntree = ntree, nodesize = nodesize)

# implement train-test splits with inner k-fold CV to optimize classification
outputR <- cpfa(x = X, y = y, model = model, nfac = nfac, nstart = nstart,
                const = const, cmode = cmode, method = method, family = family,
                nfolds = nfolds, nrep = nrep, ratio = ratio,
                parameters = parameters, type.out = "descriptives",
                seeds = NULL, plot.out = TRUE, parallel = FALSE,
                verbose = FALSE)
```



Results

We examine classification performance metrics of error (`err`) and overall accuracy (`acc`) for each model and for each classifier, looking at their median across outer train-test splits. We also examine, averaged across train-test splits, the optimal tuning parameters chosen (i.e., those that minimized misclassification error in the inner 10-fold CV) for each classifier.

```r
# examine classification performance measures - median across train-test splits
outputR$descriptive$median[, 1:2]
```

```
##             err   acc
## fac.2plr 0.110 0.890
```

```
## fac.2rf  0.095 0.905
## fac.3plr 0.110 0.890
## fac.3rf  0.105 0.895
```

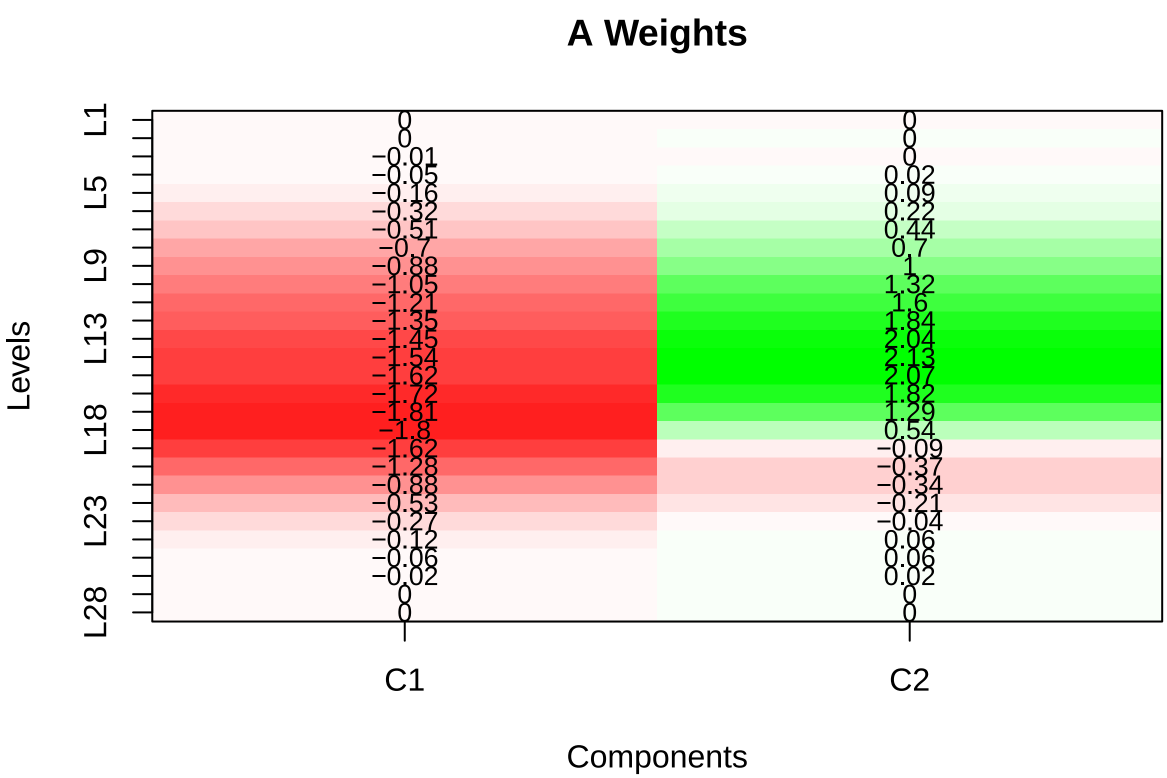
```r
# examine optimal tuning parameters averaged across train-test splits
outputR$mean.opt.tune
```

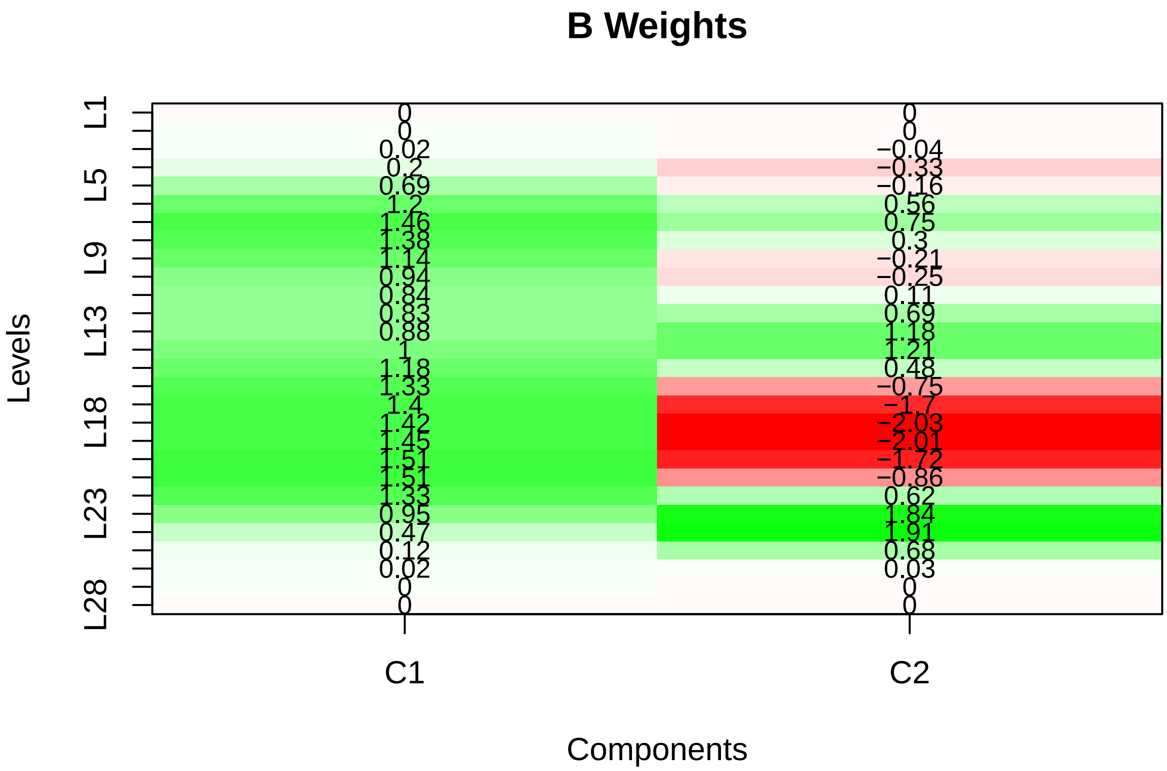
```
##   nfac      alpha   lambda gamma cost ntree nodesize size decay rda.alpha delta
## 1    2 0.11428571 2.350755    NA   NA   600     32.0   NA    NA        NA    NA
## 2    3 0.02857143 6.256281    NA   NA   680     10.4   NA    NA        NA    NA
##   eta max.depth subsample nrounds
## 1  NA        NA        NA      NA
## 2  NA        NA        NA      NA
```

We can see that classification accuracy values are close to 0.9 for both two-component and three-component models. The two-component Parafac model with the RF classifier performed best. In addition, we can see average optimal tuning parameters for each classifier. Note that PLR optimized lambda internally.

Next, we use the function `plotcpfa`. The function performs two tasks: (1) it fits the optimal model among those fit using function `cpfa`; and (2) it plots the estimated component weights from that model for the non-classification modes. Looking at the plots, we can search for relationships between the levels of mode A or B and the model components. In this case, mode A corresponds to the horizontal axis of the images (e.g., such as those images shown above) while mode B corresponds to the vertical axis.

```r
# set seed
set.seed(500)

# plot heatmaps of component weights for optimal model
results <- plotcpfa(outputR, nstart = 10, ctol = 1e-6, verbose = FALSE)
```





For the A mode, component weights are stronger between levels 5 and 24 for the first component. Looking at the B mode, weights are stronger in the middle between levels 4 and 25. Whether viewed from the A mode or B mode perspective, this first component appears to be a general component showing the presence of non-zero values, distinguishing empty outer parts of images from non-zero inner parts. The digits 2 and 3 have systematic differences in the presence of non-zero values in these areas.

Examining the second component for the A mode, weights are stronger near and around level 14 where there appears to be a greater presence of zero values in digit 3 compared to digit 2. A similar pattern exists in the B mode around levels 18 and 19 with more non-zero values for 2, compared to 3. Taken together between the two modes, this component could be identifying the closed circle present in the digit 2, which differs from the open or half circle seen in digit 3.

B.) Fashion MNIST

The Fashion MNIST dataset consists of 70,000 grayscale images of fashion objects within 10 different categories, indexed by class labels of 0 through 9 (Xiao, Rasul, and Vollgraf, 2017). We use R package **keras3** (Kalinowski, Allaire, and Chollet, 2025) to download the dataset. We subset to only 1000 images from the training set. We only use images within the categories of top (with a label of 0), trouser (1), or sandal (5). The images include 308 tops, 357 trousers, and 335 sandals.

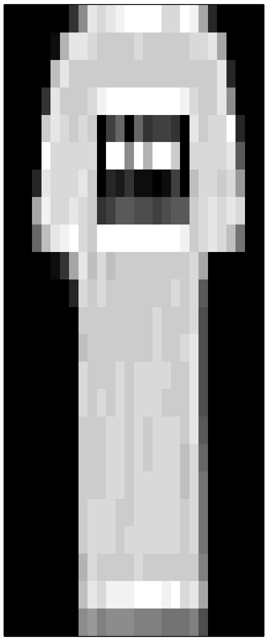
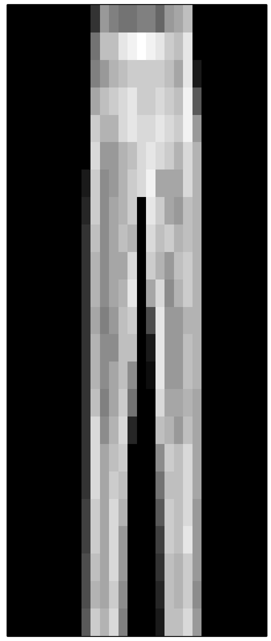
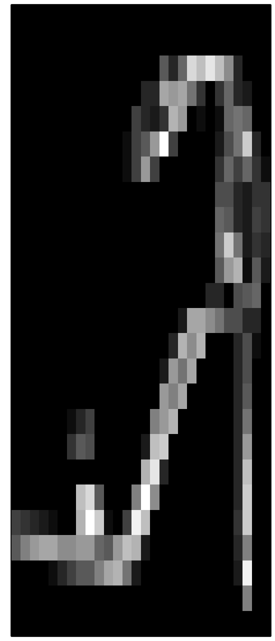
We remove a different number of horizontal levels from each image, randomly selecting a number of levels between three and six (inclusive) to remove for each image. The resulting array is ragged: mode A (the horizontal mode) contains a different number of rows for each image while mode B (the vertical mode) contains exactly 28 levels for all images. This removal is meant to simulate data corruption or sensor malfunction that could lead to incomplete images. To continue forward, we use a Parafac2 model (Harshman, 1972), which can be fit to a ragged array (i.e., one mode has a different number of levels, conditional on another mode). The Parafac2 model maintains properties similar to the Parafac model, including the intrinsic axis property (for details, see Harshman and Lundy, 1994, 1996).

Download

```
# load library
library(keras3)

# download Fashion MNIST data and subset training set to categories of 0, 1, 5
fmnist <- dataset_fashion_mnist()
inde <- which(fmnist$train$y %in% c(0, 1, 5))
X0 <- fmnist$train$x[inde, ,]
labels <- fmnist$train$y[inde]

# prepare labels by converting labels to class factor with levels of 0, 1, and 2
y0 <- as.factor(as.numeric(as.factor(labels)) - 1)

# subset to first 1000 images
ind <- 1:1e3
nimage <- length(ind)
Xp <- X0[ind, , ]
y <- y0[ind]

# permute the array to make the third mode the classification mode
X <- aperm(Xp, c(2, 3, 1))

# set seed
set.seed(500)

# remove rows from each image and restructure array into a list of images
nremove <- sample(3:6, nimage, replace = TRUE)
Xrag <- vector(mode = "list", length = nimage)
for (i in 1:nimage) {
   ranremove <- c(sample(1:28, nremove[i]))
   image0 <- X[-ranremove, , i]
   Xrag[[i]] <- image0
}
```

We plot an example of a top, trouser, and sandal.

```
# plot example of top, trouser, and sandal
par(mfrow = c(1, 3))
pdigit <- function(imat) {
  m <- t(apply(imat, 2, rev))
  image(m, col = gray(seq(0, 1, 0.05)), xaxt = "n", yaxt = "n")
}
for (i in 0:2) {pdigit(Xrag[[which(y == i)[1]]])}
```

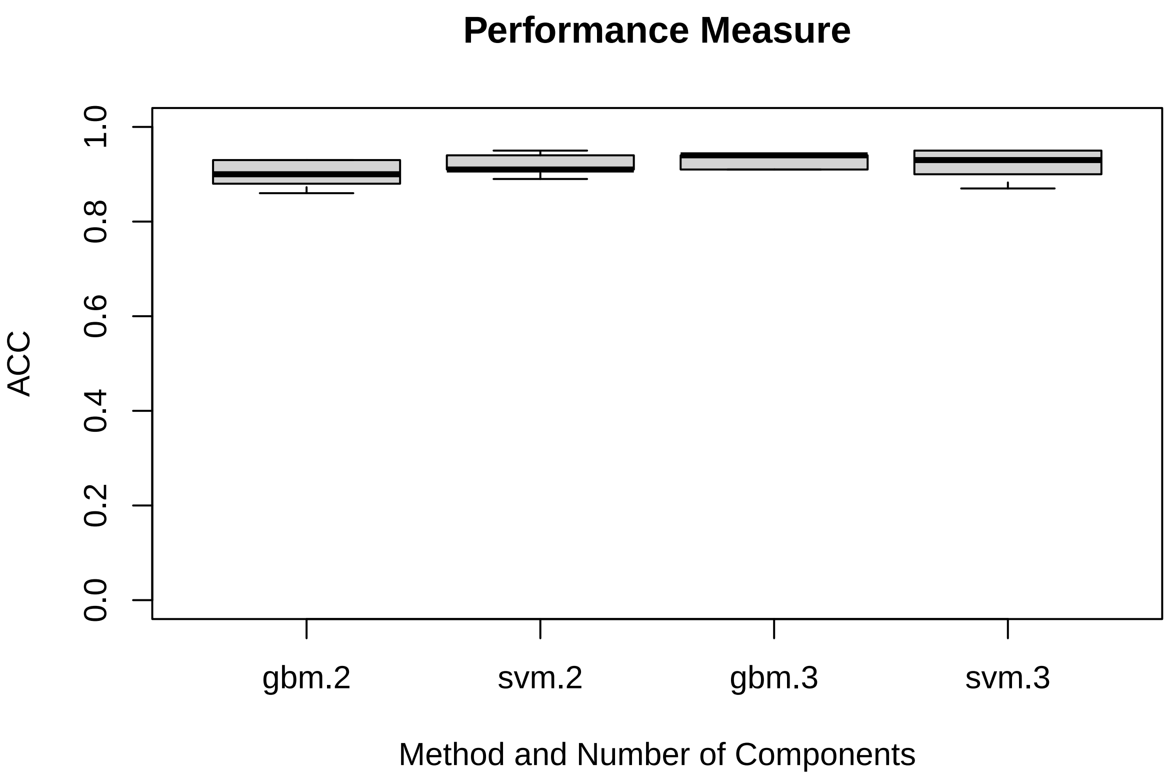
Analysis

We load R package **cpfa** and initialize the tensor model. First, the data array is a ragged, irregular three-way array; so we set `model <- "parafac2"` to use a Parafac2 model. Second, we initialize the number of components to fit by setting `nfac <- c(2, 3)`. Third, we set `nstart <- 10` for 10 random starts. Fourth, we specify the constraint for each mode using `const`. For Parafac2, we set the third mode to have non-negative weights.

Next, we initialize classification methods. First, we use `method = c("SVM", "GBM")` to use support vector machine (SVM) and gradient boosting machine (GBM). Second, we specify that the problem is a multiclass classification problem by setting `family <- "multinomial"`. Third, we use 10-fold cross-validation (CV) in our inner training, setting `nfolds <- 10`; and fourth, we set `nrep <- 5` to perform five outer train-test splits of our data. Fifth, we set `ratio <- 0.9` to specify that each outer training set contains a proportion of 0.9 of the full data while the outer testing set contains a proportion of 0.1. Finally, we specify ranges for tuning parameters gamma (SVM), cost (SVM), eta (GBM), maximum depth (GBM), subsample (GBM), and number of rounds (GBM), wrapping them into `parameters`.

```r
# load library
library(cpfa)

# set seed
set.seed(500)

# initialize model
model <- "parafac2"
nfac <- c(2, 3)
nstart <- 10
const <- c("uncons", "uncons", "nonneg")

# initialize classification
method <- c("SVM", "GBM")
family <- "multinomial"
nfolds <- 10
nrep <- 5
```

```r
ratio <- 0.9

# initialize tuning parameters
gamma <- c(0, 0.1, 1, 10); cost <- c(0.1, 1, 10, 100)
eta <- c(0.3, 0.7); max.depth <- c(1, 2)
subsample <- c(0.75, 0.9); nrounds <- c(100, 400)
parameters <- list(gamma = gamma, cost = cost, eta = eta, max.depth = max.depth,
                   subsample = subsample, nrounds = nrounds)

# implement train-test splits with inner k-fold CV to optimize classification
outputR2 <- cpfa(x = Xrag, y = y, model = model, nfac = nfac, nstart = nstart,
                 const = const, method = method, family = family,
                 nfolds = nfolds, nrep = nrep, ratio = ratio,
                 parameters = parameters, type.out = "descriptives",
                 seeds = NULL, plot.out = TRUE, parallel = FALSE,
                 verbose = FALSE)
```



Results

We examine classification performance metrics of error (`err`) and overall accuracy (`acc`) for each model and for each classifier, averaged across train-test splits. We also examine, averaged across train-test splits, the optimal tuning parameters chosen for each classifier.

```r
# examine classification performance measures - mean across train-test splits
outputR2$descriptive$mean[, 1:2]
```

```
##              err   acc
## fac.2svm 0.080 0.920
## fac.2gbm 0.100 0.900
## fac.3svm 0.080 0.920
```

```
## fac.3gbm 0.072 0.928
```

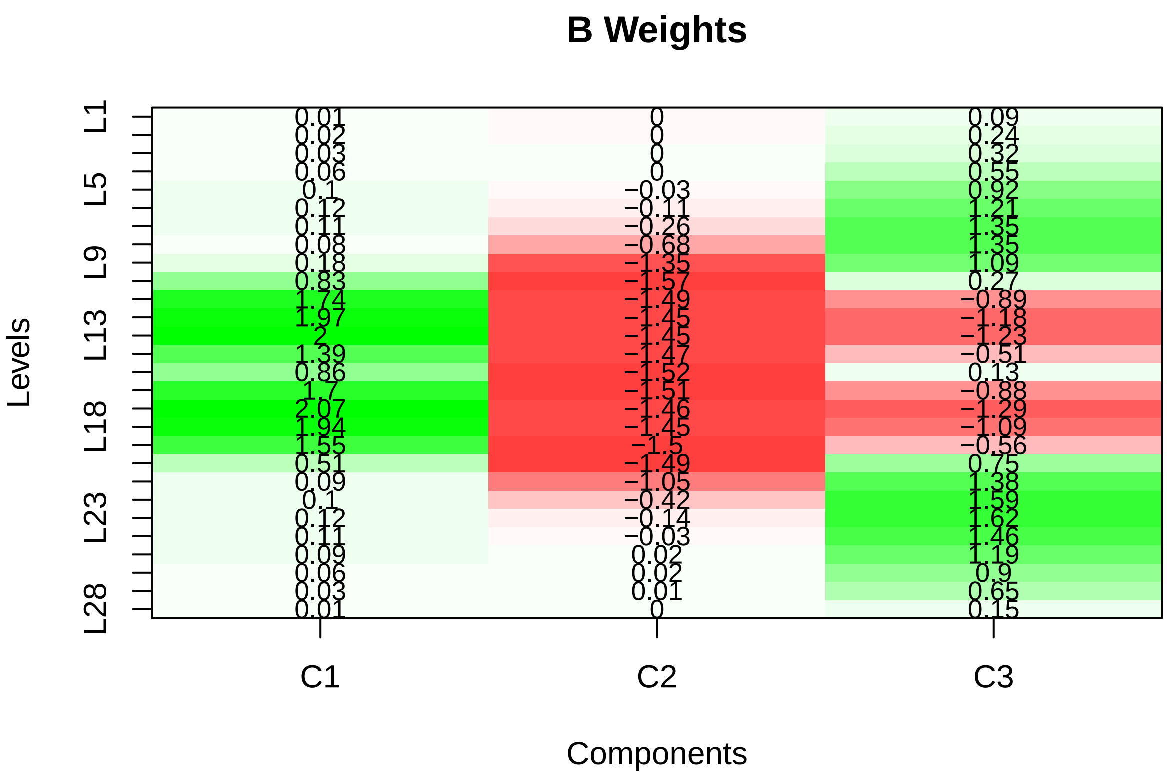
```r
# examine optimal tuning parameters averaged across train-test splits
outputR2$mean.opt.tune
```

```
##   nfac alpha lambda gamma cost ntree nodesize size decay rda.alpha delta  eta
## 1    2    NA     NA  6.40 26.2    NA       NA   NA    NA        NA    NA 0.54
## 2    3    NA     NA  0.64 40.6    NA       NA   NA    NA        NA    NA 0.46
##   max.depth subsample nrounds
## 1       1.6      0.84     100
## 2       1.6      0.81     100
```

We see that accuracy values are between 0.9 and 0.928. The three-component Parafac2 model with the GBM classifier performed best. In addition, we can see average optimal tuning parameters for each classifier.

Next, we use function `plotcpfa`. Because the number of levels differs among images for mode A, we cannot inspect mode A weights easily. Function `plotcpfa` automatically excludes mode A weights from plots based on the Parafac2 model. Instead, we inspect only the component weights for mode B to search for relationships between the levels of B and the Parafac2 model components.

```r
# set seed
set.seed(500)

# plot heatmaps of component weights for optimal model
results2 <- plotcpfa(outputR2, nstart = 10, ctol = 1e-6, verbose = FALSE)
```



For the B mode (i.e., vertical axis), for the first component, weights are stronger between levels 11 to 14 and between 16 and 19. This first component might be capturing non-zero changes displayed at the center of the top images. The first component might help distinguish tops from trousers and sandals.

For the second component, weights are stronger between levels 6 to 23. This component could be a general component distinguishing non-zero values in the middle columns of images from zero values at the far left and far right columns. While the sandal has some non-zero values in the far left and far right columns, the top and trouser generally do not; so this second component might help distinguish the top and the trouser from the sandal.

For the third component, weights are stronger between levels 6 and 9 and between 22 and 25. This third component might be capturing zero values to the left and right of images, contrasting them with non-zero values in the middle. Interestingly, we see two negative bands between levels 11 and 14 and between 16 and 19; however, level 15 is positive. Level 15 might correspond to the empty space between the trouser legs while the negative bands might represent non-zero values in the legs. If so, this component might help distinguish trousers from tops or sandals.

These interpretations are limited. For all components, more work would need to be done to explore the estimated A mode weights (or C mode weights) to understand the components from other perspectives. Such work could clarify the components further and change the current interpretations.

References

Asisgress, M. (2025). cpfa: Classification with Parallel Factor Analysis. R package version 1.2-2, https://CRAN.R-project.org/package=cpfa.

Harshman, R. (1970). Foundations of the PARAFAC procedure: Models and conditions for an “explanatory” multi-modal factor analysis. UCLA working papers in phonetics, 16(1), 84.

Harshman, R. (1972). PARAFAC2: Mathematical and technical notes. UCLA working papers in phonetics, 22(3044), 122215.

Harshman, R. and Lundy, M. (1994). PARAFAC: Parallel factor analysis. Computational Statistics and Data Analysis, 18(1), 39-72.

Harshman, R. and Lundy, M. (1996). Uniqueness proof for a family of models sharing features of Tucker’s three-mode factor analysis and PARAFAC/CANDECOMP. Psychometrika, 61(1), 133-154.

Helwig, N. (2025). CMLS: Constrained Multivariate Least Squares. R package version 1.1, https://CRAN.R-project.org/package=CMLS.

Helwig, N. (2025). multiway: Component Models for Multi-Way Data. R package version 1.0-7, https://CRAN.R-project.org/package=multiway.

Irizarry, R., Gill, A. (2025). dslabs: Data Science Labs. R package version 0.9.0, https://CRAN.R-project.org/package=dslabs.

Kalinowski, T., Allaire, J., and Chollet, F. (2025). keras3: R Interface to ‘Keras’. R package version 1.4.0, https://CRAN.R-project.org/package=keras3.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (2002). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324.

LeCun, Y., Cortes, C., and Burges, C. (1998). The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

R Core Team (2025). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747.