Multiway Classification with Real Data

Matthew Asisgress

2025-10-25

In this demonstration, we apply the R package **cpfa** (Asisgress, 2025) to the MNIST dataset (LeCun, Cortes, and Burges, 1998; LeCun et al., 1998). We show how to use the package to distinguish between the digits of 2 and 3, a binary classification problem.

MNIST

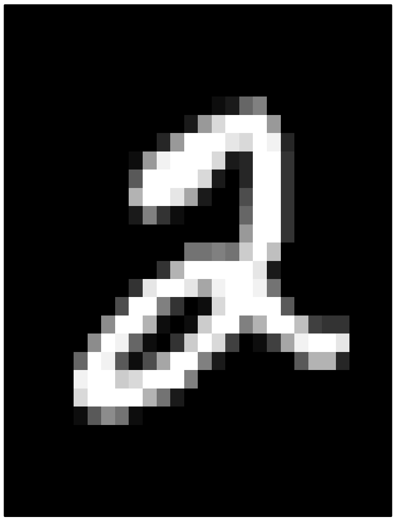
The MNIST dataset consists of 70,000 grayscale images of handwritten digits from 0 to 9. We use the R package **dslabs** (Irizarry and Gill, 2025) to download the dataset. We subset to only 1000 images from the original training set. The images include 495 images of the digit 2 and 505 images of the digit 3.

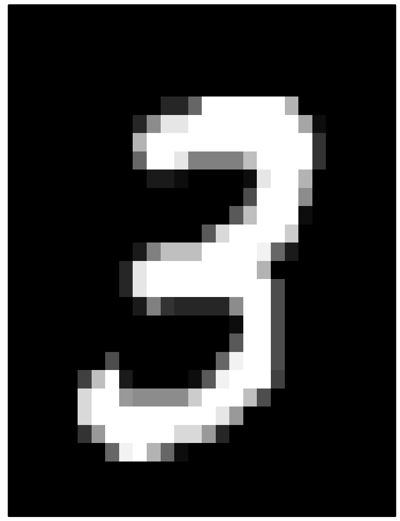
Download

```
# 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 1000 images
ind <- 1:1e3
X <- X0[ind, , ]
y <- y0[ind]
```

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

```
# plot example of digit 2 and digit 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")
}
pdigit(t(X[which(y == 0)[1], , ])); pdigit(t(X[which(y == 1)[1], , ]))
```





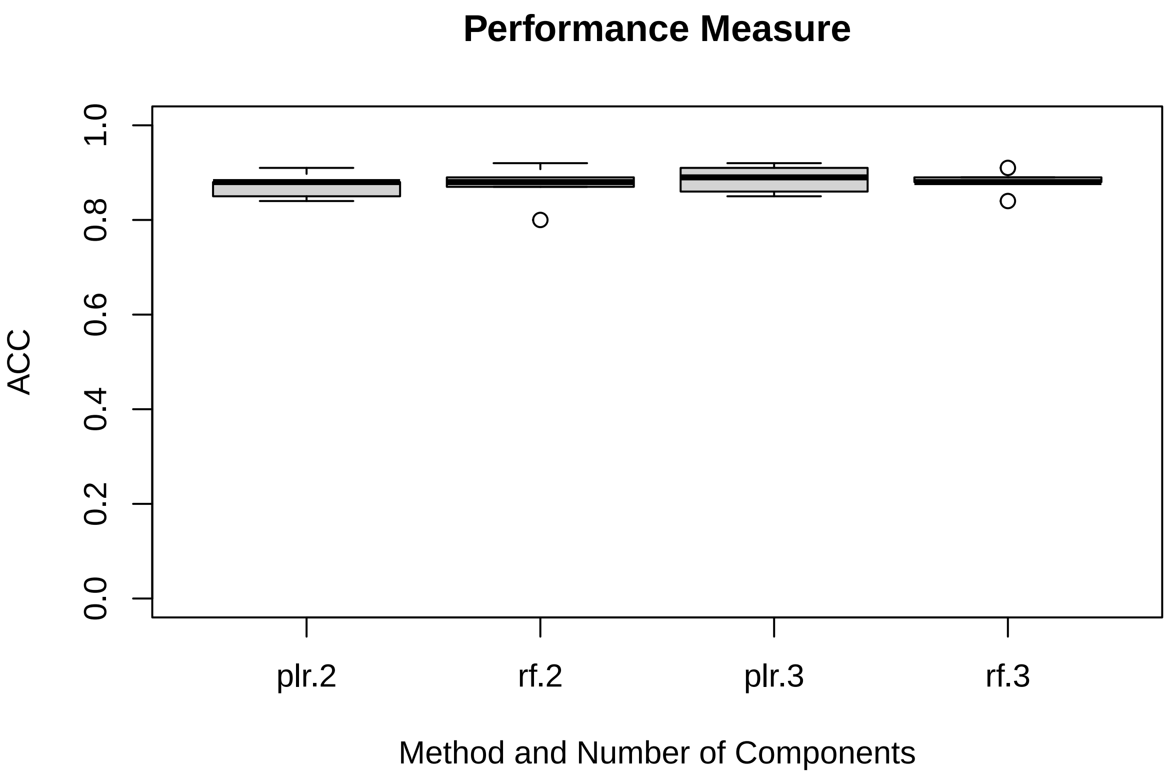
Analysis

We load the 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. Note that the Parafac2 model can be used for ragged tensors. 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`. 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). 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).

We next 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 other options and for references associated with these classifiers. 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 will contain a proportion of 0.9 of the full input data while the outer testing set will contain 0.1 of the data. 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 Parafac 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. We also examine, averaged across outer train-test splits, the optimal tuning parameters chosen (i.e., that minimized misclassification error) for each classifier.

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

```
##            err  acc
## fac.2plr 0.12 0.88
## fac.2rf  0.12 0.88
```

```
## fac.3plr 0.11 0.89
## fac.3rf  0.12 0.88
```

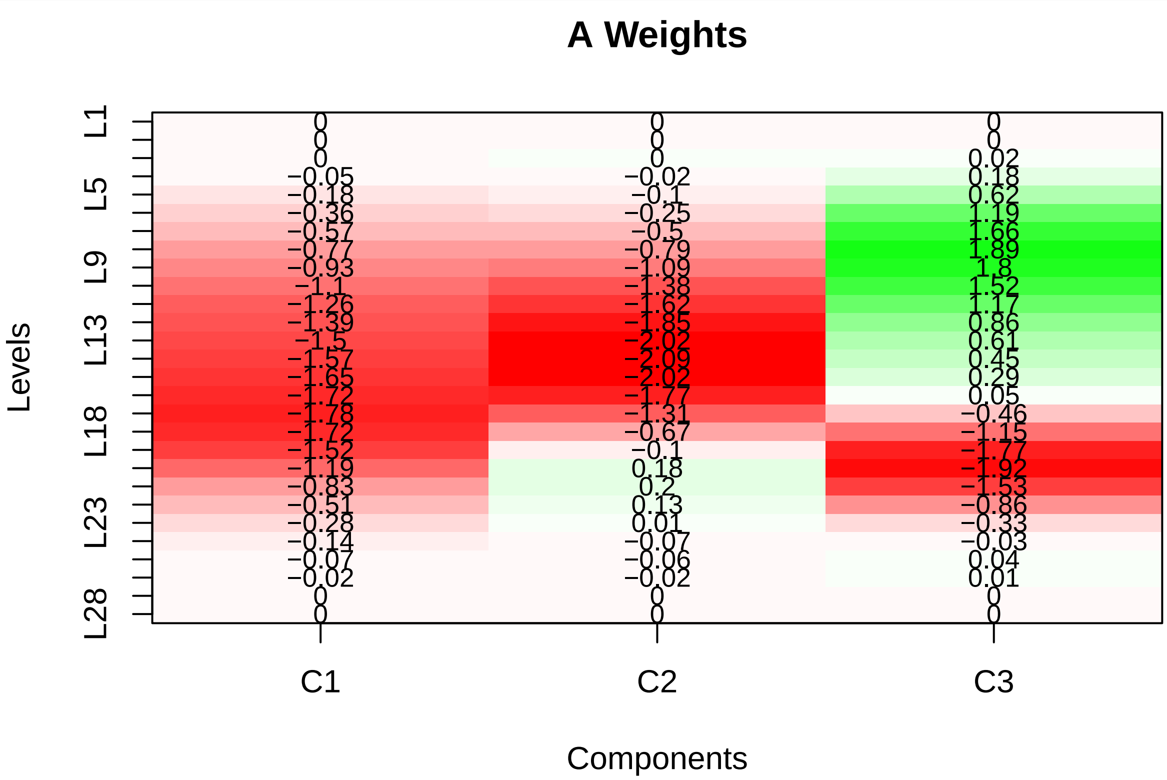
```
# examine optimal tuning parameters averaged across train-test splits
outputR$mean.opt.tune
```

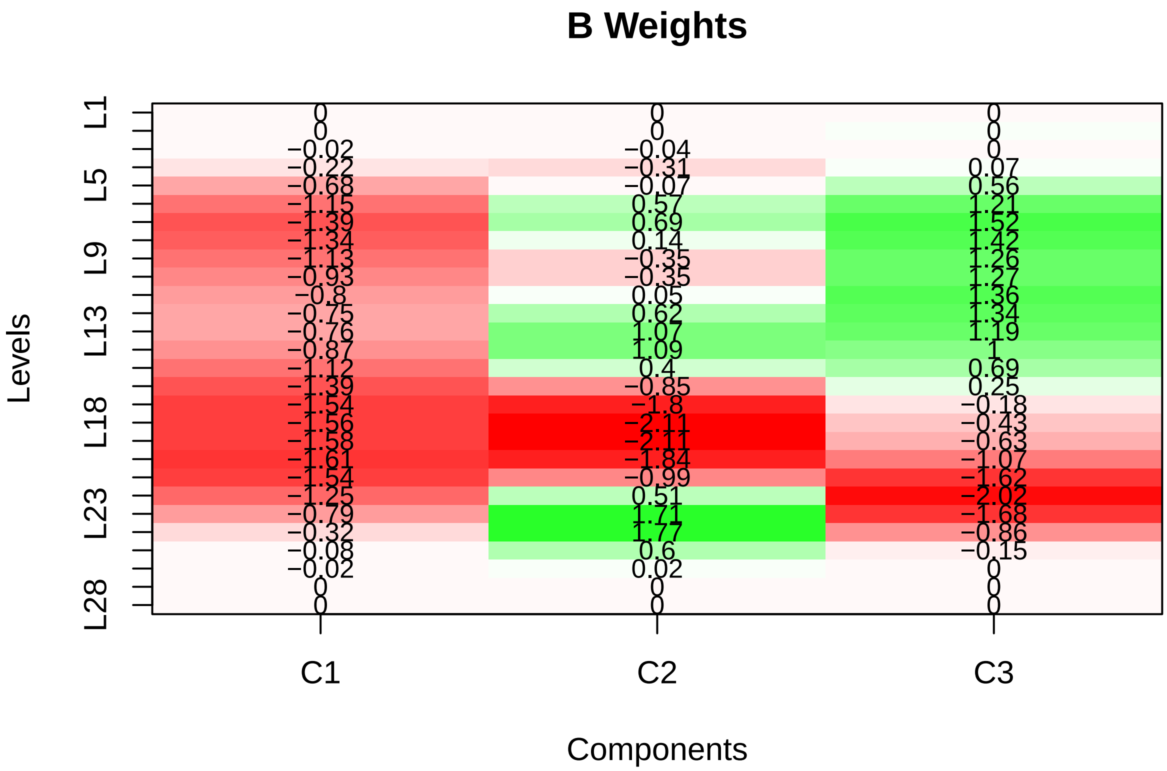
```
##   nfac     alpha   lambda gamma cost ntree nodesize size decay rda.alpha delta
## 1    2 0.1714286 4.032271    NA   NA   760     32.0   NA    NA        NA    NA
## 2    3 0.2000000 5.321921    NA   NA   640     18.4   NA    NA        NA    NA
##   eta max.depth subsample nrounds
## 1  NA        NA        NA      NA
## 2  NA        NA        NA      NA
```

We can see average optimal tuning parameters for each classifier. Note that PLR optimized lambda internally.

Next, we use the function `plotcpfa` to plot the component weights for the A and B modes of the data array. (unfinished, updates planned in November 2025)

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

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





References

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

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.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324. https://doi.org/10.1109/5.726791.

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/.

(unfinished, updates planned in November 2025)