Mnist logistic classification

Data Tidying and Reporting. UC3M

Marcos Crespo

2024-02-12

Introduction

This document is the submission for the Task 1 in *Data Tidying and Reporting* course for *MSc in Statistics for Data Science at Carlos III University of Madrid*. In this task we are asked to make a ridge logistic model for classification while working with a given dataset. The goals of the task are several that will be conveniently indicated in the following sections. In addition, an appealing and concise report is mandatory, since it is one of the main goals of this course. Finally it is also required that the text is self-explanatory, meaning that any reader without being enrolled in the course may be able to read and understand the document.

Data insights

The MNIST database is a widely used dataset in the field of machine learning and computer vision. It stands for *Modified National Institute of Standards and Technology* database. It consists of a large collection of handwritten digits. Each observation consists of the label (the intended written number), the writer id and the $28 * 28$ grey scale grid values for the image itself. This version of the dataset includes approximately 30,000 training images and 30,000 testing ones. These images are normalized and centered, making them a standard benchmark for evaluating the performance of some algorithms in tasks such as digit recognition and classification[1]. A possible representation of a 0 can be seen in Figure 1.

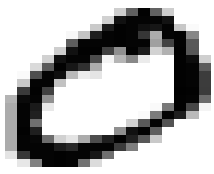

Figure 1: 0 as represented in Mnist

In the first task, we are asked to implement a ridge logistic model for classifying the digits 4 and 9, so our next step is to filter only the data with labels 4 and 9. (Wickham et al. 2023)

[1]For some application and references see (Wikipedia contributors 2024). Also for another application in clustering classification see my TFG.

```r
train <- train_nist %>%
  filter(digit %in% c(4, 9))
# drop the spare levels
train$digit <- droplevels(train$digit)
```

The model

(García-Portugués 2023) Ridge regression is a method employed to estimate coefficients within multiple regression models when independent variables display high correlation. Its application spans various disciplines such as econometrics, chemistry, and engineering.

The characteristics of the ridge regression are useful in the Mnist situation. If we look at the column 1 and 2 of the grey scale matrix, its s.d. is 0. So the collinearity is NA. Probably, since they are top left pixels and the images are centered, a lot of white pixels (0's) are expected. This may result in high collinearity even though the pixels wont affect each other. On the other hand, if a pixel is coloured, it makes sense that the nearby pixels are also coloured because of the drawing.

The ridge model can be found in `glmnet` (Tay, Narasimhan, and Hastie 2023) package as a subcase for $\alpha = 0$. We will be trying to fit ridge a model, with a cross-validated-chosen λ penalty. **Standardization of the data wont be necessary since all of the inputs are in the same (grey) scale.**

This can be done with the following code[2]:

```r
# 10-fold cross-validation. Keep the seed for reproducibility
set.seed(12345)
lambdaGrid <- 10 ^ seq(log10(45193.6), log10(0.1), length.out = 150)

kcvRidge <-
  cv.glmnet(
    x = as.matrix(train$px),
    y = train$digit,
    alpha = 0,
    standardize = FALSE,
    family = "binomial",
    nfolds = 10,
    lambda = lambdaGrid
  )
```

The optimal λ that the function gets is the minimum $\lambda_{min} = 18.93$ or the lowest *1se* $\lambda_{1se} = 83.64$. Note that this values are not in the limits of the grid 0.1, 4.51936×10^4. We proceed with the λ_{1se} and fit a model.

```r
ridgeMod <-
  glmnet(
    x = as.matrix(train$px),
    y = train$digit,
    alpha = 0,
    family = "binomial",
    standardize = FALSE ,
    lambda = 83.64
  )
```

We can now **plot the coefficients** in order to get some insights about them. We exclude the intercept because it is in a much different magnitude.

It seems clear in Figure 2. how the central pixels tend to have bigger coefficients than the first ones. This is because of the number being drawn in the centre of the grid.

[2]This computation takes time so the specific value isn't computed in knit-time.

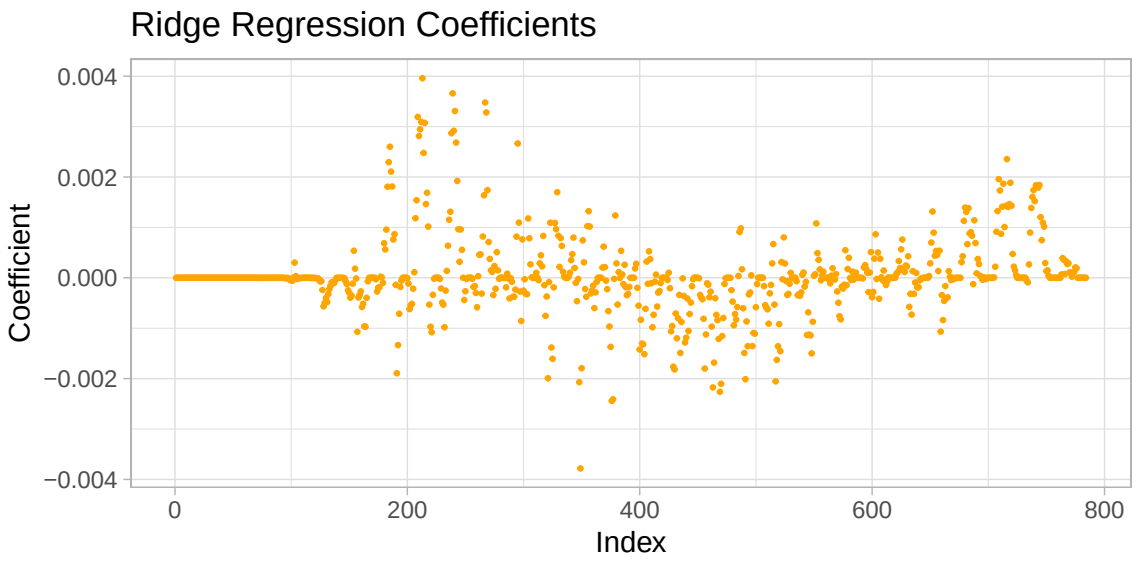

Figure 2: Coefficients plot

Predictions

We can now use the test set (properly transformed and filtered) in order to make some predictions using the model and measure the performance of the model. We are using a logistic binomial model, so the prediction is a estimated probability of the event being a certain class. Since we want a binomial output for computing the accuracy of the prediction, a cut-off for the probability is needed. The selected cut-off will be the naive 0.5.

```
# Predictions
predictions <- predict(ridgeMod, type = "response",
                       newx = as.matrix(test$px))

# Convert predicted probabilities to class labels (4 or 9)
predicted_classes <- ifelse(predictions > 0.5, 9, 4)

# Evaluate the accuracy
accuracy <- mean(predicted_classes == test$digit)
```

So the accuracy obtained in this model is 97.9351536% for the λ_{1se}.

Further work

In addition, we are asked to tackle the 45 classification problems of one versus another digit in a similar way to the previous approach and then report the results.

We have created the 45 necessary datasets and then using vectorization and different `apply()` functions we have computed the models and the global accuracy. No further explanation is given because of the optionality of the last task. The code we run is the following (3.5h runtime approx.) and is left here for reproducibility issues.

```
set.seed(1234)
lev <- 0:9
datasets_train <- vector("list", length = 45)
datasets_test <- vector("list", length = 45)
cont <- 1

# Generation of the datasets
for (i in lev) {
  for (j in lev) {
```

```
    if (i < j) {
      train_l <- train_nist %>%
        filter(digit %in% c(i, j))
      train_l$digit <- droplevels(train_l$digit)

      test_l <- test_nist %>%
        filter(digit %in% c(i, j))
      test_l$digit <- droplevels(test_l$digit)

      print(cont)
      datasets_train[[cont]] <- train_l
      datasets_test[[cont]] <- test_l

      cont <- cont + 1
    }
  }
}

# Vectorized generation of the models
models <- lapply(datasets_train, function(data) {
  x <- as.matrix(data[, 3])
  y <- data[, 1]
  cv.glmnet(
    x,
    y,
    alpha = 0,
    standardize = FALSE,
    family = "binomial",
    nfolds = 10
  )
})

# Vectorized predictions
predictions <- mapply(function(model, new_data) {
  x_new <- as.matrix(new_data$px)
  predict(model,
          newx = x_new,
          s = "lambda.1se",
          type = "response")
}, models, datasets_test)


# Vectorized class predictions and accuracy calculation
accuracy <- mapply(function(pred, mytest) {
  y_test <- as.vector(model.matrix(~ factor(mytest$digit))[, 2])
  class_predictions <- ifelse(pred > 0.5, 1, 0)
  mean(class_predictions == y_test)
}, predictions, datasets_test)
```

```
# Generation of the table
matrix_triang <- matrix(0, nrow = 10, ncol = 10)
matrix_triang[upper.tri(matrix_triang)] <- accuracy
rownames(matrix_triang) <- colnames(matrix_triang) <- 0:9

print(matrix_triang)
```

	1	2	3	4	5	6	7	8	9
0	0.9993876	0.9920558	0.9973571	0.9974579	0.9972532	0.9892221	0.9929215	0.9706697	0.9970850
1		0.9936174	0.9894537	0.9931328	0.9982650	0.9982644	0.9812852	0.9904495	0.9761231
2			0.9934662	0.9957039	0.9973641	0.9770561	0.9892869	0.9942775	0.9906977
3				0.9976759	0.9992197	0.9889558	0.9972016	0.9944351	0.9996759
4					0.9977204	0.9861549	0.9734711	0.9937725	0.9934913
5						0.9897419	0.9978838	0.9935854	0.9979757
6							0.9923990	0.9793515	0.9943219
7								0.9881625	0.9778689
8									0.9893617

Note how the previous code takes the base factor as de smallest digit. This makes sense since the datasets are created that way and also `class_prediction` loop is working the same way.

References

García-Portugués, E. 2023. *Notes for Predictive Modeling*. https://bookdown.org/egarpor/PM-UC3M/.

Tay, J. Kenneth, Balasubramanian Narasimhan, and Trevor Hastie. 2023. "Elastic Net Regularization Paths for All Generalized Linear Models." *Journal of Statistical Software* 106 (1): 1–31. https://doi.org/10.18637/jss.v106.i01.

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Wikipedia contributors. 2024. "MNIST Database — Wikipedia, the Free Encyclopedia." https://en.wikipedia.org/w/index.php?title=MNIST_database&oldid=1199732782.