Digit Recognition Perceptual Experiment

Jorge Hernández

This project is a Shiny app which is meant to work as a perceptual experiment and to use some basic Machine Learning algorithms. It is an experiment which allows to introduce various types of noises to the Mnist handwritten digit dataset and visualize random numbers. It also allows the user to try and guess which number is being represented, with his results being stored. It also allows the user to compare his answer with a few basic machine learning which can be set up in another tab.

The first part in the shiny app is loading the libraries, setting up the data and functions to add the noise. The data is provided already divided in a train and a test sample so there is no need for separation and the only transformation needed is making the written number a factor. The next step is setting up a colour palette between -1 and 1 representing between white and black to be able to visualize the numbers. The final step used before starting to program the app is setting up the functions. These modify a 16x16 pixel matrix by adding:

* A percentage of the image is covered in vertical black lines. These cannot overlap.
* A number of squares of a certain length. These can overlap.
* A zeroing noise. Sets up a percentage of the pixels to be either white or black.
* A white noise. Changes the value of a percentage of the pixels according to either a uniform distribution between -2 and 2, a normal distribution with mean 0 and sd 1 or making it either black or white. If the values are not between 1 and -1 they are set to the closest of either 1 or -1.

Next there is the actual app. It is a simple dashboard created with shiny dashboards which consists of 4 tabs which can be alternated through the sidebar.

The first tab is the Design tab which allows to set up all the noises for the experiment. The details for each noise are set in a conditional panel that expands upon checking an input. Depending on the type of the noise there are numeric or select input. These inputs are not used here but are being set up for the Play tab.

The next tab is the Bot tab. This tab allows adding one of three classification algorithms through a select input and change some parameters:

* A K-Nearest-Neighbours. Requires no training and the only parameter that is possible to modify in this experiment is the number of neighbours through a slider input.
* A Support- Vector- Machines. The only parameter made modifiable is the type of kernel used. This algorithm requires training. The train action button is linked to an observer that when clicked it stores the trained model in the reactive value cont$svm. This value is originally 0 so it will give an error later if the model is used without training. I am working on making it give a message to remind the user to train it if he tries to use it without being trained. Training can take long time. Currently the message under training fades when it is training the algorithm but I am working on a clearer system.
* A Random Forest. The only parameters modifiable are the number of trees and the size of the sample. Like SVM it requires training which is stored in the reactive value cont$rf. It has similar problems to SVM.

The next tab is the Play tab. In this tab we have a blank space where the number will be represented, a numeric input which allows us to write our guess, an action button which changes the number, an action button which selects our choice and another box in which the results is posted.

* The new number button should be the first one used. This button selects a random entry from the dataframe with the test sample. This button causes the reactive value cont$count2 to increase by 1. This number starts at 0 and is basically a counter of how many different numbers the user has seen. It is used later to add the results to a table in the final tab.
* Then the number should be visualized in the blank space. The number is visualized by converting the entry except the first value which represents the number into a 16x16 array and then modified by all the noise functions that the user selected in the Design tab. This 16x16 modified array is represented as 16x16 matrix in which each slot is a different shade of grey (there are far more than 50 in this experiment). This shows a picture of the handwritten number after going through all the noise.
* Then what the user does is write his guess of the number in the numeric input box and click the action button select. This action button triggers multiple actions:
  + It increases the reactive value cont$count by 1. It starts at 0 and basically works to know if the user answered before. This counter is reset when the user clicks the new number button.
  + If it is the first time he answers it checks whether he chose the right number. If he did, he is told he is right and the details of the experiment along with his choices are added in the row cont$count2 to the reactive value cont$tabla3 which is the table where results are stored for the final tab. If is not right, he is told the correct answer and the results are added similarly to the table.
  + If it is not the first time he answers as cont$count >1 he is told he already answered and which was the number. Nothing is added to the results table.
  + If the user has set up a classification bot he is told what the bot thinks it is. This transforms the 16x16 modified array back into a row of the dataframe which is stored in the reactive value cont$datos. Then it is used by whichever algorithm was chosen to predict the number and rendered into a text that says which number the bot thinks it is.
* After this, the user can click a new number to reset the cont$count and display a new number so he can try again.

The final tab is the results table. This tab is very simple as it only contains a datatable made from the dataframe cont$tabla3. It is missing the algorithm choice of number as it gave an error that closed the app if there was no training for the algorithm. The solution that I tried to implement has not worked yet but I will continue working on it. This datatable allows the user to check his results and performance and filter by them.

This is a complete description of the functionality and logic behind this perceptual experiment shiny app. I am planning on solving the mentioned problems as well as adding other classification algorithms and adding an action button to calculate the expected error of the selected algorithm after the noises have been applied. This app once totally completed should be useful to understand how different noises affect different algorithms as well as human perception