Java final assignment

Exploring Java Language through a practical work

In this assignment, you will have 2 csv files to use :

- mnist_test.csv
- mnist_train.csv

This java practical work relies on no other prerequisites than having taken the java lectures.

First part: Reading data from csv

1. Open the mnist_train.csv file (not with excel!). This csv file contains 785 columns and 60 002 thousand lines! That is a huge set of data. Therefore, we need a programming language to analyze its content.

🛠 A. First task, extract raw data:

a. **Create a proje**ct in your IDE and import the 2 files in this project, then **create a small program** that can read all the lines from the csv files**. Print the 2 first** lines in the console.

b. For each line, you will have this kind of format:

```
5,0,0,0,0,0,0,0,0,0,0, … ,254,0
```

Modify your program to take in account this format **and get an array of String** from this line. You can take advantage of the split function.

c. Convert this array of String into an **array of double** (beware, you have one line to define the headers)

2. This file contains only 2 data.
   - On the first column, one can find a number, ranging from 0 to 9.
   - On the remaining columns, there is a “flat” matrix, a 2d matrix which has been written on a single line.

🛠 B. Second task, reshaping data

a. **Add some logic** to your existing code, to transform the array of double extracted from the remaining 784 columns to a 2d square matrix (28 x 28)

b. **Write a method** “showMatrix” that prints a matrix in the console.
In our case, the matrix contains values that range from 0 to 255, which is not so indicative while printed in the console.
To improve the rendering, print “xx” in the console if the value is above 100, else print “..”.
   - **Use this showMatrix method on the matrix extracted from line 23**
   - **Print** the first column of the same line. What do you notice?

⚒ C. Refactor code to match java standards

a. Refactor your code to build 2 classes :

- An **Image** class that will contain 2 fields : “label” and “dataMatrix”, with the appropriate types.
- An **ImageCsvDAO** that will contain a function getAllImages(), and will contain all the code you have written to read data from a csv file. Be careful, the file name can vary, so make it variable in this service class.

Second part: toward machine learning, from scratch (and for beginners 😊 !)

The considered data set is in fact the MNIST dataset, that contains handwritten digits. This dataset is a standard case of study for new beginners in machine learning.

As the goal here is not to train you on ML, but to find an interesting way to practice java, mathematics necessary to achieve that will be really limited to simple formula (like knowing how to calculate a distance between 2 points or how to calculate an average from a list of values.

As you have noticed, the first column contains what we call a “label”, which indicates what is supposed to represent the associated image (the 28x28 matrix).

To introduce briefly what is (statistical) machine learning: it is a process where a program will determine statistical characteristics of a dataset containing well known labels to take decisions, like classification decisions.

In our case, we will use a very basic approach to take decisions :

- For each digit, we’ll compute the “average representant” (also called centroid) of each digit image.
- Providing a new digit image, we’ll compare the distances from this image to each digit centroid, and we’ll decide that this image will be classified as the digit from which the average representant is the closest in terms of distance.

⚒ D. Analysing the distribution of the dataset :

1. Write a “calculateDistribution” method, that for each digit, will assign the total count of occurrences in the data set

⚒ E. Calculating the average representant :

1. write a “trainCentroids” method, that will take a list of “Image”
2. This method should return a data structure associating a digit to its centroid.
3. The centroid matrix contains the average of all the images, index by index.

⚒ F. Performing your first classification:

1. Load the file “mnist_test.csv” under the form of a list of Image instances
2. Isolate only the 10 first “0” occurrences.

3. For each instance, calculate the distance between this instance and every centroid defined from previous task. Are you satisfied by the result?

Hint : the distance can be defined as the square root of the sum of each module (absolute difference) of index-to-index values of the 2 considered matrix. Once you have that value, you have to take the minimum.

⚒ G. Refactor code to match java standards

1. Write a class “CentroidClassifier” containing the “trainCentroids” method
2. Refactor the point “3” of the previous task to a method named “predict” that will return the predicted digit providing a image.

Third part: Evaluating the model

The model evaluation is mandatory to know what confidence you can have in a machine-learning based system.

We define 4 core metrics :

- True positives, example: “the prediction is 5 and the label is actually 5.”
- True negatives, example: “the prediction is ‘not 5’ and the label is 4”.
- False positives, example: “the prediction is 5 and the label was actually 4”
- False negatives, example : “the prediction is ‘not 5’ but the label was actually 5”.

⚒ H. Implement classification performance assessment.

1. As our model (CentroidClassifier), will aggregate all the results from all the centroid comparisons, you have to find a simple way to implement the above description for model performance.
2. Find a complete (per centroid) implementation of the above
3. Display the confusion matrix of this solution. Confusion matrix

⚒ I. Improve the model

1. Change your strategy to replace average with standard deviation in the centroid calculation.