Practical 25: Preparing data for machine learning

ADS 2

Semester 2, 2020/21

```
## Loading required package: viridisLite
```

We expect this practical to take around 1-2 hours to complete, but don't worry if it takes longer. Please make use of the Slack discussion boards and Friday afternoon student hours if you have questions.

Handwritten digit recognition

One "classic" in machine learning is the recognition of hand-written digits. The problem is the following: Given a large enough sample of hand-written digits (0-9) together with labels of what they represent (0-9), can a machine be trained to correctly classify previously unseen hand-written digits?

In today's practical, we are not concerned with an actual machine learning algorithm. Instead, we are going to look at how to read in, prepare, and look at the data.

Get the data set(s)

We are going to use the MNIST (Modified National Institute of Standards and Technology) dataset.

It can be downloaded in .csv format from this page: https://pjreddie.com/projects/mnist-in-csv/

You will see that there is a "train set" and a "test set". Of course, if you were to train and then evaluate a machine learning method, you would need both datasets. For this practical, since we are only interested in general data wrangling, you can get away with just downloading one of them. Ideally, you would use the "train set", because that's what you'll need for training. But if you are limited in terms of bandwidth or storage, use the "test set" instead: It is smaller.

If even that takes too long to download, get the file "really_tiny_dataset.csv" from the Practical Folder in Blackboard Learn. It has only the 20 first samples of the train set. (It is a good idea to try out your methods on a small data set anyway, so that you can see quickly whether your approach works)

Import the data

Read in the .csv file dataset. This particular dataset does not have column headings in the first row, so use header=FALSE.

This is a large dataset (how large?), so for now just look at the first 10 rows and 10 columns. You will see that the first column has a number between 0 and 9. This is the label, i.e. the true number represented by the data point in that row.

Then there are 784 more columns, with a lot of (but not exclusively) 0 entries. Those are pixel intensities for every pixel in the image that stores the handwritten number: the image has 28×28 pixels, so therefore there are 784 light intensity values in each row.

If you weren't working with the "really tiny dataset" in the first place, now is a good time to limit yourself to only a small dataset. For instance, you can save the first 20 rows as a new dataset, run all subsequent steps on this small dataset, check it works, and then go back and analyse the big dataset.

Reshaping the dataset

With our smaller dataset, we now want to get it into a shape where it is easier to plot and to work with. At the moment each row is one sample, that is one complete 28x28 image and the label that goes with it.

An easier format to deal with would be if each row was a single pixel. For each of them, you would want to know:

- What sample that pixel came from
- What its intensity is
- The label of the digit that this pixel is a part of
- the "how many-eth" pixel it is in its image, more precisely the x and y coordinates of the pixel in the image

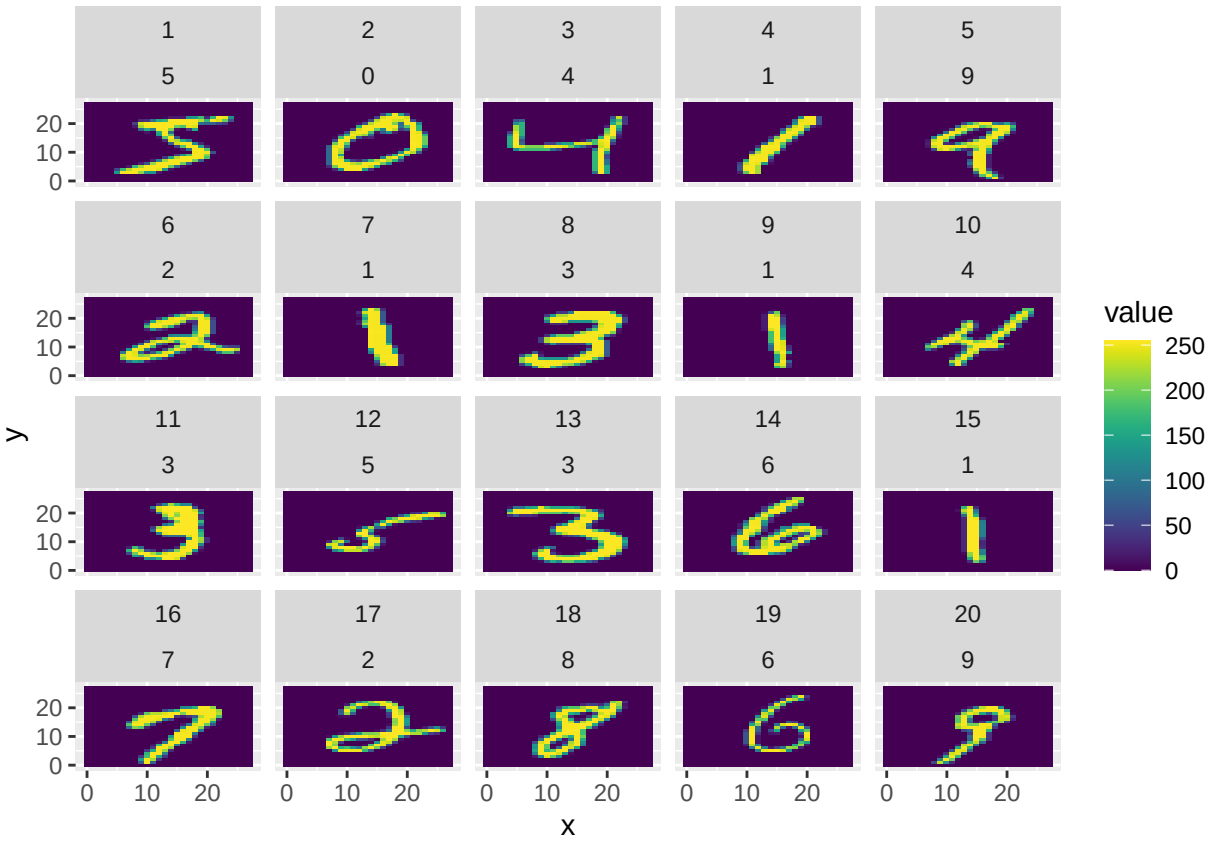
The first three steps are essentially a transformation from a wide to a long dataset. You have previously learned how to do this using the gather() function in the tidyr library. Remind yourself how to this.

The last step requires that you transform pixel number (a number from 1 to 784) into a row and column number. For this, you need to know that pixels are numbers sequentially: Pixels 1 to 28 represent the first row, then 29 to 56 the second row etc. Do a little bit of math to allow you to add an "x" and "y" column to your gathered dataframe to store the coordinates of the pixel.

Plot the data

Now that we have a tidy long dataset, we can use ggplot2 to plot it. Plot the x coordinate on the x axis, the y coordinate on the y axis, use geom_tile() and colour by pixel value. Use facet_wrap() to create one panel per sample. this should now look like actual numbers!

Here is what it looked like for me for the tiny (only 20 digits) dataset. (Yours may look a wee bit different, but the general idea should be the same)



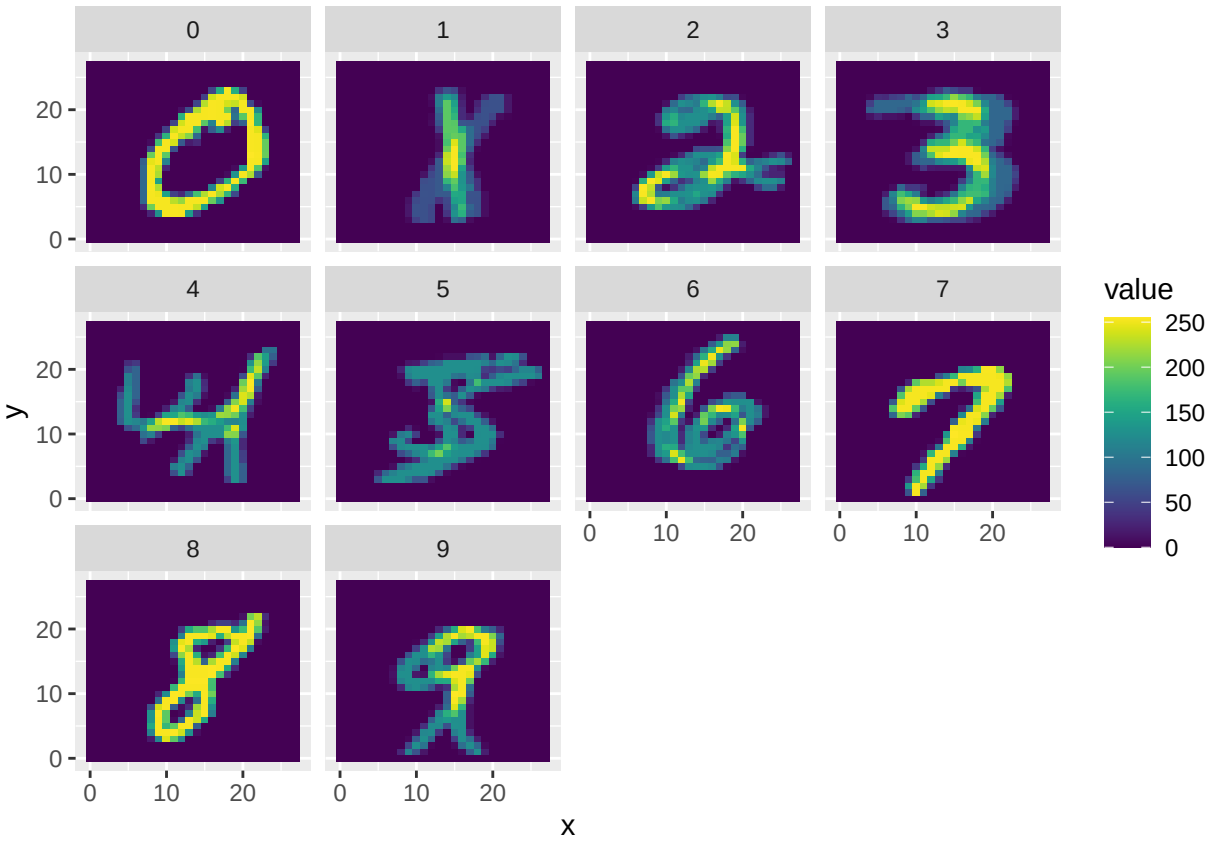
It may well be that the first time you try, your handwritten digits appear upside-down, as mirror images, or sideways rotated. If that is the case, go back and check how you computed the x and y coordinates to fix the problem.

What does the "average digit" look like?

In the small sample of only 20 digits, some appear more than once. For these, a reasonable question to ask is: What does the "average" version of each digit look like?

You may find the "aggregate" function useful here.

If you plot the average digit, what do you see? Bear in mind that this is a small data set, so you may not have enough instances of each digit to generate a meaningful "average". Rather, it looks a bit like all the instances of a digit have just been drawn "on top of each other". This does start to give you interesting information about the kind of variation you can expect within each label!



Move to the big dataset

Now that the transformation, plotting, and averaging work on your tiny data set, you can go back and apply the same methods to the bigger dataset, e.g. the entire "train set". The commands are essentially the same, but will take longer to run, some up to a few minutes.

The train set has sixty thousand samples, so plotting them all individually may not be a useful thing to do. But you may want to plot several examples of each digits. Also, now that we have so many samples, plotting the averages is a somewhat more satisfying experience.

Feature selection

Looking at your plots, start thinking about features that may be usable for distinguishing digits. For instance, what kind of things distinguish a "1" from a "0"? How is a "6" different from a "9"?

For instance, one useful thing to think about, for instance, are transitions. If you go along a row in an image of a 1, you will start with a bunch of dark pixels, then it will change to a few bright pixels, then quickly back to dark and stay dark until the end of the row. This is generally true at the row level, even if some ones are straight and others slant to the right or to the left.

Today, we are not asking you to automatically extract features, just to think about one or more potentially useful features. Useful here means to things: 1. They could be calculated from the image data we have without much trouble and 2. They are useful in distinguishing between digits.

Originally created by G. Luksys, 2020. Last edit by M.I. Stefan, April 2021