Exercises: Introduction to machine learning

September 6, 2017

Henning Christiansen
Roskilde University
`http://www.ruc.dk/~henning/`, `henning@ruc.dk`

1 Linear regression: Data, models, errors and outliers

It is suggested that you work in small groups of two or three students, and no more. If possible, make sure that there is at least one in each group who is familiar with Processing. Discuss how to solve the problems together in the groups, and make sure that you all get the code to run on your own computer. If you need to check what a certain Processing function is doing, you can (likely) find it here:

`https://processing.org/reference/`

1.1 Getting started

Find on moodle, the file `ProcessingCodeExercisesSept7.zip` and unpack it. If you don't have Processing installed already, do so as explained at `https://processing.org`. Now, test the `trainingModels` program on the data file supplied, called `potatoprices.csv`:

1. Open the `ExercisesIntroML` directory; inside that, open the `trainingModels` directory;
2. Open the program in Processing by double clicking `trainingModels.pde`. (ignore or close the strange, empty window that also pops up).
3. Run the program by clicking the triangle in the upper left corner.
4. If the yellow window that pops up does not fit into your computer screen, use the Processing editor to change the settings in `size(−,−)` function and try again.

1.2 Understand the program

Now you should walk through the important parts of the program in a top down manner; the details concerning screen coordinates and that stuff in the `plotting` part are not important. It is suggested that you proceed in the following way (but do it in another way, if you wish).

- The main program in the `setup` and `draw` functions; guess from the comments what the instructions mean.[1]
- Next, the `abstractModel` part. Very little, but essential code, so be sure to understand it.
- Now to the meat of it, the `linearRegression` part. It is not essential that you understand the linear regression algorithm, but try to get some idea of what is is doing (and not necessarily why).

1.3 Add noise removal to the program

Now you are going to extend the program. You should now use the other datafile, `potatopricesNoised.csv`; as the name says, there are added some point that may be regarded as noise.

The course note "A gentle introduction to machine learning" explains (p. 13 at the bottom) a way to incorporate noise removal. It works by first training a model on the entire data, then use that model for removing noisy points, and training again to obtain a new and final model.

Implement this with the facilities available, and you should plot the points (as done in the given program), and:

- show both the preliminary and the final model in the plot,
- in the plot, mark the points that are removed, by a separate colour, so you can check that it looks right.

NB: You may need Processing's `removeRow()` function and `plot(`x`,`y`,`*color*`)` which is included in the source code.

2 Iterative parameter learning

Section 6 in the course note explains how you can train in principle any type of model by an iterative method; the pseudo-code is shown pp. 19–20.[2]

[1]If you are not familiar with Processing, ask someone who is, about the purposes of `setup()`, `draw()` and `noLoop()`.

[2]There is a potential misunderstanding in the text, p. 19. It reads "Draw a circle with radius ε ...", but it is a bad idea to actually plot such a circle. Read it instead as "Imagine a circle ...".

Your task is to implement and test this method by adding some code to an alternative version of the program that we used in the previous question. The program is in the folder `IterativeTrainingModels` which is included in the files you downloaded.

This program includes facilities for drawing error surfaces, and it has two classes of models you can use for your testing, the linear models we have used already, class `LinearModel`, and a new class `TrickyModel`; the latter is constructed with the sole purpose of demonstrating the problematic phenomena of local minima and plateaus.

Part of the code is already included, showing the most important functions that are new, compared with the previous program. Both model classes have two parameters, referred to as a and b as before, both in the interval $[0;1]$.

You can inspect the little code that is given already in main window (within `setup()` and `draw()`). We explain a few details:

- `plotErrorSurface(myData,` *model-type*`)`, where *model-type* is given by one of the constants `LINEARMODELS` and `TRICKYMODELS`, draws the error surface as a coloured contour map.

- The variables `a` and `b` are intended to serve as the "crawling point" called p in the pseudocode in the course note.

- For given values of `a` and `b`, the average error for the model (instantiated by `a`, `b`) is calculated as follows.

  ```
  float error = new LinearModel(a,b).avgError(myData);
  ```

  (This is for the linear models; for the tricky ones write "`new TrickyModel` $\cdots$" instead.)

- Use, in each iteration, `markParameterPoint(a,b)` to draw a black dot, indicating the current position of (a,b).

- Use the function `delay(10)` (or with another number) in each iteration of the `draw()` function to slow down, so you will be able to see the point moving.

- When your stop criterion is met, you can stop the iterations by the function `noLoop()`.

Warning: The calculation of the error surface may take very long time, and it grows with the size of the window as given by the `size(−,−)` function (for each pixel, the entire data sample is traversed to calculate the error shown by a colour). It is, thus, important, that you do this only once, in the `setup()` function.

1. Run the program as it is given and see what happens.

2. Add the necessary code to `draw()` that implements the iterative learning methods, and test it for the linear model class.

3. Now, change it such that you use the tricky models instead. If it works as expected, try running it several times to see if you can provoke it to get stuck in a local minimum or a plateau.