Mia Watts April 25, 2024

**Probability and Applied Statistics Final Project**

**Java Plotter, Salter, and Smoother**

In creating a Java program for plotting, salting, and smoothing data, three formulas were used. The first is a polynomial equation, the second uses sine, and the third uses cosine. Each of these formulas is used to create and write a dataset to a .csv file. The code for each formula is similar minus the actual calculation, so some information after the first introduction will be omitted or briefly covered. The following is done within the programs:

* Plotting – Plots a polynomial, sine, and cosine equation by calculating results over an interval and writing the values to a .csv file to be graphed.
* Salting – Adds or subtracts a random value in a specific range to each data point calculated in the plotting program. The data points are read from the original plotter file. The addition or subtraction is index-determinant. What is meant by this is that the status (odd or even) of the index determines whether the salt value is added or subtracted to the original value. These salted values are placed into an array and are written to a new file that holds the salted values.
* Smoothing – The smoother represents a moving average program in which data points are taken from the left and right of a certain index and are averaged to “smooth” the data to as close as possible to the original form. This part of the program reads data from the salter file and writes the smoothed data values to a new smoother .csv file.

***Note: The polynomial function will be the one discussed in detail here because all other versions of the plotter, salter, and smoother are the same code, just with a different formula. Don’t worry! Results of all versions will be shown.***

**Java Plotter Results**

**Polynomial Version**

In the main class, there are two methods that pertain to the polynomial plotter. The first is a file writer that calculates the results across a certain interval then writes it to a .csv file, and the second is the equation. The equation is as follows:

* 0.1x2 – 0.5x – 2

Once the data is added to a result ArrayList, the information is written to a .csv file that can then be opened for further graphing. The interval can be changed, but for the purpose of this specific example, the program calculates all results with x-values ranging from -100 to 100. The method itself takes these parameters, so the user only needs to update them within the tester class.

After running the program from the tester class (i.e., passing a desired interval to the method), the polynomial results file is added to the IntelliJ project and can be opened using Excel. The user running the program will see all values in the first column of the file and can graph it from there. The graph looks like this when done:

**Sine Version**

Similar to those of the polynomial plotter, the sine plotter has the same writer method and has a formula method that calculates values from the following formula:

* sin(2x)

The information is placed into an ArrayList and is written in a .csv file where graphing can occur. When graphed, the formula looks like this:

Both the formatting of this graph and the cosine version of the plotter’s graph can be edited to look nicer by reducing the number of values that are calculated within the formula. However, for the case of this assignment, many values were used as per the instructions and to show more differences between salted and smoothed data iterations.

**Cosine Version**

Like the polynomial and sine plotters, there are two methods that pertain to the cosine version of the plotter. The first method is similar to that of the polynomial and sine method in that the values are calculated and are written to a file, and the second represents this formula:

* 3cos(x) – 5cos(2x) – 2cos(3x) – cos(4x)

After running the program, the results are shown in a .csv file and are graphed as follows:

**Java Salter Results**

**Polynomial Version**

In this section of the plotter, salter, and smoother part of the project, the salter will be explained. The salter reads the data added to the .csv files from the plotter method and will add or subtract a randomly generated salt value in a certain range depending on whether the index of the data point in the array is odd or even. If the current index in the iteration is even, the salt value is added to the original value, and if the current index in the iteration is odd, the salt value is subtracted from the original value. This practice of “salting” the data makes it unrecognizable from the original dataset.

As mentioned before, the salters read from the plotter file, add the values to an ArrayList, then write to a new file to add all salted values to a separate location once salting has been completed. The only parameter required for the salter methods is the name of a .csv file that will hold the salted values. The methods within the plotter class take care of the rest, including reading the original file by passing the filename into the method within the code. The results of the salter for the polynomial formula return the following graph:

**Sine Version**

The sine salter has the same methods as the polynomial salter. The only differences are a) the equation used, and b) the file that the salter method reads to obtain data. The file read in this case is the original sine file written in the sine plotter method. When graphed, the salted values can be visualized like this:

**Cosine Version**

The cosine salter has the same methods as the polynomial and sine salters. The only differences are a) the equation used, and b) the file that the salter method reads to obtain data. The file read in this case is the original cosine file written in the sine plotter method. The results are displayed in the following graph.

**Java Smoother Results**

**Polynomial Version**

To complete the plotter, salter, and smoother, a smoother was written to “smooth” the data in an attempt to get its visualization into a form that is reminiscent of the original graph. The smoother reads the data written to the salter file and calculates a moving average across the data points to partially remove the effects of the salting. After smoothing has been complete, the erratic data is, well, smoothed to show a form closer to its original.

There is both a smoother class and a tester class associated with the Java smoother program, much like the plotter and salter. The smoother class contains the three methods needed for the polynomial, sine, and cosine versions of the program. The method takes two parameters, one being the name of the file that will be read and the other being the window, or range, of values that will be taken from the left and right of a current index to form an average. The program checks to see what index is being returned. For example, if the index happens to be the first or last value in the array, only the values to the right or left respectively will be considered since there exist no values to the left of a value at the beginning of an array and so on.

Once the values have been smoothed and have been stored within a new array, they are written to a third .csv file that will hold all smoothed data from the program. In total, there are three files that are created within the plotter, salter, and smoother programs, one holding the plotted data, one holding the salted data, and the final one holding the smoothed data. After smoothing has been completed and data has been written to a new file, the resulting graph looks like this:

*Note: This is after smoothing with a window, or range, of four. Thus, only the four results to the left and to the right (if applicable) for each value are taken to calculate the moving average.*

**Sine Version**

The sine version of the smoother is similar to the polynomial version in how the code is written. The only real differences are the files read and written to when the program runs, as this part of the program reads the salted sine data and writes to a smoothed sine results file. The graph resulting from smoothing the salted data with a window of four is as follows:

As shown in the graph, the data is still quite erratic. However, when comparing the y-axis results of this graph to the y-axis results of the salted graph, the salted values that peaked at around 200-250 are now reduced to around 150. With more smoothing, the y-axis window would get closer to the original graph.

**Cosine Version**

The cosine version of the smoother is similar to the polynomial and sine versions in how the code is written. The only real differences are the files read and written to when the program runs, as this part of the program reads the salted cosine data and writes to a smoothed cosine results file. The graph resulting from smoothing the salted data with a window of four is as follows:

Once again, the points are still a bit erratic, but the y-value range has decreased. With more smoothing, the graph would be closer to that of the original.

**Final Graph Results**

**Polynomial Iterations**

**Sine Iterations**

**Cosine Iterations**