CIT-644-Fall 2020 Scientific Computing : Coursework 1

Due Date : Sunday 27 December, 2020 (Hard Deadline - No Extensions)

Description

Develop a C++ class library comprising a collection of numerical techniques for scientific computing described in the following three parts.

Instructions

1. You are not allowed to use any external libraries other than plotting libraries.
2. This is a group coursework. So, you need to make a team of 3-4.
3. A .zip comes with this file that contains all the datasets mentioned in the following questions.
4. One member of the group should do the submission on Moodle. Also, you are required to add group members names and IDs to the report.
5. You will need to electronically submit the source code files of your implementation and a report clearly explaining your findings.
6. After submission, each team will have to demonstrate the program and have a detailed discussion with course TAs
7. Criteria for evaluation: understanding, correctness, functionalities, features, efficiency and utilization of object-oriented design principles.

Part 1 - Solving a System of Linear Equations

A. Gauss Elimination

Implement a C++ class for solving systems of linear equations using Gauss elimination with scaled partial pivoting. Your class should define a function `solve(A, b)` that takes in an `nxn` matrix `A` and an `n`-dimensional vector `b`, for any arbitrary size `n`, and returns a vector `x` that satisfies `Ax = b`.

Your class should also define a flag `valid_solution` as an instance variable. This flag should be set to false if the last system solved by the solver is ill-conditioned, and true otherwise. You always should check the value of this flag before using the solution computed by the solver in further computations.

Test your implementation by solving the following system:

```
2x1 + x2 - x3 = 0
x1 + 4x2 + 3x3 = 14
-x1 + 2x2 + 7x3 = 30
```

Submit a well-documented implementation along with the code that runs the solver on the given test case.

B. Gauss-Seidel

Repeat exercise `1-A` using Gauss-Seidel. Use initial values `x1 = x2 = x3 = 0`. Report the values of `x1, x2, x3` for the first 30 iterations of the algorithm. Print the numbers rounded to 4 digits after the decimal point.

Part 2 - Regression

A. Curve Fitting

Implement a C++ class for polynomial regression of single variable scalar functions `y = f(x)`. Your class should define the following functions:

- `fit(x, y, m)` - Where `x` and `y` are `n`-dimensional vectors, such that `(x[i], y[i])` represent the `i'th` point in the dataset. The function should compute the coefficients of an `m`'th order polynomial that fits the data and store them in an instance variable. When solving the system of equations, try both solvers implemented in part 1. Propose a criterion of convergence for Gauss-Seidel and report the number of iterations required for convergence for each of the following test cases and visualize the polynomials computed at iterations 0, 10, 20, ...
- `predict(x)` - A function that evaluates the computed `m`'th order polynomial for values in the vector `x`, and returns a vector containing the estimated `y` values. Use this function to plot the polynomial for `x` between `0` and `10` with step `0.1`.

Evaluate your regressor on the two datasets located at `datasets/part_2/a` directory and plot the results.

B. Multivariable Linear Regression

Implement a C++ class for 2D linear regression (plane fitting). Your class should define the following functions:

- `fit(X, y)` - Where `X` is an `nx2` matrix and `y` is an `n`-dimensional vector. Try both solvers like in part `2-A` and report the number of iterations required by Gauss-Seidel.
- `predict(X)`

Use your regressor to fit the house pricing dataset https://www.kaggle.com/ananthreddy/housing.

Consider using a 2-dimensional feature vector `x` by selecting two of the following features to fit the price value.

- lotsize
- bedrooms
- bathrooms
- stories

Try all possible combinations to select two out of these four features. For each combination, evaluate the mean squared error between the estimated and the ground truth prices. Which attributes matter

the most in the determination of the house price?

Part 3 - Interpolation

Implement two C++ classes, one for Newton's interpolation and the other for cubic spline interpolation (check course notes). You will be given 2 sample datasets and you need to increase the number of points in each dataset 2 and 4 times using both Newton's and cublic spline interpolation. You can use the defined elimination classes in part 1 if needed.

The two datasets can be found in `datasets/part_3` directory.

Notes

1. Newton's polynomial function order should be determined using the number of points given in the datasets.
2. Cubic spline intervals number should be determined by the number of points given in the dataset .

Your classes should define the following functions:

- `fit(x, y)` - Where `x` and `y` are `n`-dimensional vector, such that `(x[i], y[i])` represent the `i'th` point in the dataset.
- `interpolate(x)` - A function that returns the `y` value for values in the vector `x`, using the calculated polynomial/spline functions.

You need to deliver the following for both Newton's and cubic spline interpolation methods:

- A .csv file that contains the calculated coefficients for Newtons's polynomial function for each dataset.
- A .csv file that contains the computed coefficients for each interval in cubic spline for each dataset.
- A .csv file that contains both the original and new interpolated points for each dataset.
- 2 plots for each dataset before and after interpolating for 2 and 4 times.
- Your comment on the difference between Newton's and cubic spline performance on each dataset.