Voortgezet Programmeren (FEB23007-11)

3. Exercise

Deadline for submission: 2012-01-29 23:59 CET

Instructions

Include in each source file (in class documentation, @author) your names and student numbers. Submit the exercise as a zip file containing _only_ the source files in root of the zip. Submit via blackboard. Note that incorrectly submitted or non-compiling exercises are automatically awarded 0 points. Remember to document your code with Javadoc-annotations.

Exercise

Let us consider a data set n points $\{y_i, x_i\}_{i=1}^n$ where each observation includes a scalar response y_i and a vector of predictors (or regressors) x_i. Now the regressors x_i together with the intercept form the design matrix $\mathbf{X}$. By convention, the constant term is included in $\mathbf{X}$ by augmenting the set of regressors with a first element that is one for all data points, i.e. taking $x_{i1} = 1$ for alli=1,. . . ,n. The responses y_i form the vector $\mathbf{Y}$. Suppose we would like to find a linear equation $Y = X\beta + \epsilon$ that best describes the data set. We can estimate the multiple regression model using the ordinary least squares estimator (see `http://en.wikipedia.org/wiki/Ordinary_least_squares`) that minimizes ϵ, obtaining estimations of β ($\widehat{\beta}$) as

$$\widehat{\beta} = \left(\mathbf{X}'\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{Y} \, .$$

The estimated residuals can be computed as

$$\widehat{\epsilon} = \mathbf{Y} - \mathbf{X}\widehat{\beta} \, .$$

Note that now intercept is the first coefficient due to the augmented x_i's. Download the data set from `http://smaa.fi/tommi/courses/prog3/data/cocaine.dat`. The data set consists of observations of individual transactions from a study of cocaine sale in Northwestern California, US over the period 1984-91. The variables are: price (per gram in $, first column), quantity (grams / sale, second column), quality (purity %, third column) and trend (variable describing variation by time, 1984=1, 1991=8, fourth column). Variables quantity, quality, and trend together with the intercept form the design matrix $\mathbf{X}$, whereas price is the response vector $\mathbf{Y}$. Write a Java program that reads the data set, computes estimations $\widehat{\beta}$ and $\widehat{\epsilon}$, and displays $\widehat{\beta}$, $\widehat{\epsilon}$ and the sum of squared residuals. For making the program in an object-oriented manner, you should implement (at least) the following classes:

- `Matrix`: a class for modeling a rectangular matrix of numbers (doubles stored in dense format). `Matrix` should have at least methods with exactly the following signatures:
  - `public Matrix(int nRows, int nCols)`: a constructor for constructing a 0-matrix taking the number of rows and columns as the argument
  - `public Matrix(int size)`: a constructor for an $n \times n$ identity matrix

- public Matrix(Matrix from): a copy-constructor that makes a matrix has equal contents to from (but note that it has to have its own instance variables, i.e. not just referring to the ones of from)
- public int getNrRows(): gives the number of rows in the matrix
- public int getNrColumns(): gives the number of columns in the matrix
- public void setElement(int row, int column, double element): a mutator method for setting an element of the matrix
- public double getElement(int row, int column): an accessor method for obtaining an element
- public Matrix transpose(): a function for getting transpose of the matrix
- public Matrix multiply(Matrix other): matrix multiplication, returns this * other
- public Matrix subtract(Matrix other): matrix (component-wise) subtraction, returns this - other
- public Matrix inverse(): (square) matrix inverse. Computing inverse of a (square) matrix is not trivial, but can be accomplished with the Gauss-Jordan elimination algorithm (see http://en.wikipedia.org/wiki/Gauss%E2%80%93Jordan_elimination). We provide you with ready code for this downloadable from http://smaa.fi/tommi/courses/prog3/data/inversion.java.part. You may use this code directly within your class, but for it to work, the other methods need to have exactly the signatures given above. The code does not take into account the possible singularity of the Matrix (don't worry about it).

- Note that apart from the setElement, the above methods should not have any side effects. For all methods (including the constructors), document and assert their pre-conditions.

- DataSet: a class for modeling a data set. The class should have a constructor that takes a file name containing a data set as an input, and a method that gives a Matrix of column indices given as the parameter (i.e. Matrix asMatrix(int[] columnIndices)). If you want, you may reuse and improve your class from exercise 2. If you didn't complete the last weeks exercise, you can ask one of your colleagues for their DataSet to reuse.

- MultipleLinearRegression: a class for computing the regression model with a given DataSet and column indices of dependent and independent variables (i.e., the set of column indices belonging to the design matrix or index of the response column). The constructor should take at least a DataSet as a parameter. Note that the dependent variable shouldn't be one of the independent ones, and that all the variables should be valid ones for the data set: document and check.

- Main: a tester class that performs simple linear regression on the cocaine.dat data set, while using price as the dependent variable and the other columns as independent variables. This class should use DataSet and MultipleLinearRegression for loading the data and computing the regression, respectively. Print out the regression formula.

Hints

- The exercise is too large to code in "one piece". Make a plan what needs to be done, design the method signatures, and implement them one by one testing that each implementation works before continuing with the next one.

- Remember that the objects' internal representation can be different from what is given in the accessor methods, e.g. for the Matrix you could use row- or column-major representation together with a single-dimensional array (see http://en.wikipedia.org/wiki/Row-major_order).

- For obtaining an absolute value of a number or the minimum/maximum of 2 numbers, see java.lang.Math.

- For augmenting the design matrix with the intercept, make e.g. a procedure addColumnToLeft() to Matrix and then fill the added column with 1's.