University of Michigan – Dearborn
Department of Computer and Information Science
CIS 481: Computational Learning
(Winter 2023)
Instructor: Luis E. Ortiz

Project 1: (Generalized) Linear Regression, Model Selection (via Cross Validation), and Model Evaluation; Application: Polynomial Curve-Fitting Regression for Working-Age Data

Overview

In this project students will apply polynomial curve-fitting for regression learning based on *root-mean-squared error (RMSE)* on a data set of U.S.A Working-Age Population Data. [1] Students will use *cross-validation (CV)* for model selection. Students will use CV to select the optimal value for the degree of a low-degree polynomial to use on all the training data. Students will use a *test set* to evaluate the RMSE of the optimal polynomial models students selected at the end of learning process.

Specifically, recall that the RMSE that a hypothesis h achieves on the *hold-out dataset*

$$D_{\text{ho}} = \left\{ \left(x_{\text{ho}}^{(1)}, y_{\text{ho}}^{(1)} \right), \ldots, \left(x_{\text{ho}}^{(m_{\text{ho}})}, y_{\text{ho}}^{(m_{\text{ho}})} \right) \right\}$$

of m_{ho} examples during each fold of CV is given by

$$\text{RMSE}(h; D_{\text{ho}}) \equiv \sqrt{\frac{1}{m_{\text{ho}}} \sum_{l=1}^{m_{\text{ho}}} \left(y_{\text{ho}}^{(l)} - h\left(x_{\text{ho}}^{(l)} \right) \right)^2} \,.$$

As stated in the previous paragraph, in this project, the *hypothesis function* h would correspond to some *polynomial* from the different hypothesis classes under consideration depending on the polynomial *degree*. Students will need to *compute and record the RMSE* for *each* h found for *each* corresponding hypothesis class on *each* fold of CV. Then students will need to *compute the average RMSE* that each hypotheses h's, for *each* hypothesis class, achieved *over the different folds.* For each case of the low-degree polynomial up to degree 12, students will *select the hypothesis-class model achieving the minimum average RMSE*. Such a hypothesis class corresponds to the *best CV model class* for your final hypothesis. Finally, students will *apply the ML curve-fitting algorithm* on *all* the training data, using the best model class they found using CV.

U.S.A Working-Age Population Data

You are given a dataset of an indicator of working-age population in the U.S.A. through time. The working-age population is defined as those aged 15 to 64. This indicator measures the share of the working-age population in the total population for a number of years between 1970 and 2021, not necessarily consecutive. The *only input attribute* is the *year*. The *output* is the *(numerical)*

[1] `https://data.oecd.org/pop/working-age-population.htm`

indicator of the working-age population for the given input year. The files `train.dat` and `test.dat` contain the *training* and *test* datasets, respectively. Each consists of *two columns.* The *first column* corresponds to the *calendar-year values (input)* and the *second column* is the *indicator of the working-age population (output)* (**NOTE**: The examples are not chronologically sorted by the year input.)

Mean Squared Error for Regression using Polynomial Hypothesis Classes

For the working-age data, we have a single real-valued input feature (the normalized year) and a single real-valued output (the normalized working-age indicator), so that the domain, or feature space, and the range, or output space, are both one-dimensional. Note that, for numerical reasons, the input-output example pairs must be appropriately normalized, as described above.

In this project, you will use polynomials up to degree d as the hypothesis class. Hence, each hypothesis h in our class has the form $h(x) = \sum_{i=0}^{d} w_i x^i = w_0 + w_1 x + w_2 x^2 + \ldots + w_d x^d$.

Recall that, if the training data is of size m, and we denote the lth example as $\left(x^{(l)}, y^{(l)}\right)$, the mean squared-error function is

$$\mathrm{Err}(\mathbf{w}) = \frac{1}{m} \sum_{l=1}^{m} \left(y^{(l)} - \sum_{i=0}^{d} w_i \left(x^{(l)} \right)^i \right)^2 .$$

In this project, you will consider several values for the degree of the polynomial $d = 0, 1, \ldots, 12$.

Data Scaling/Normalization for Robust Learning

As presented and discussed during lectures, students must “normalize” the data in order to help the learning algorithm output a hypothesis in a more numerically robust and accurate way. Students must use the same simple normalization algorithm, called “standard scaling,” discussed during lectures in the similar context of the climate data; that is, apply a simple linear transformation to the input and output values separately that leads to the average of the values being 0 and the (empirical, unbiased estimator of the) standard deviation being 1. Recall that errors must still be evaluated in the original output space, not the transformed/normalized output space.

The ML Regression Algorithm Black-box

Students are encouraged to implement the ML learning algorithm for regression presented, derived, illustrated, and discussed during lecture, or write code that uses appropriate existing libraries to perform the learning tasks. Note that if you use existing libraries, you must make sure to use the appropriate methods and call them with the appropriate parameters to produce *exactly the same results that you would have obtained if you had coded the entire process yourself*; that is, you must understand exactly how the methods in the libraries you use work and what they are producing when you call them.

As an alternative, students are provided a bash-shell script, named `learner_script`. The script is a wrapper for the 1-dimensional curve-fitting regression algorithm. This is how it works. Say that given some training data, we want to learn the optimal weights $\mathbf{w}^*$ for a polynomial of degree d. Suppose we would like to run the algorithm with $d = 9$. We first copy the training data into a file named `D.dat`, if using the bash-shell script, or into a file named , if using the Python script, and place it in the same directory that the learning-algorithm script will be executed. Then, execute the script as

```
learn_script '9'
```

which will produce a file named `w.dat`, with $d+1 = 10$ coefficient-weights, each placed in a separate line in the file, starting with w_0 in the first line, corresponding to the constant terms, and ending with w_9 in the last line (i.e., the tenth), corresponding to the coefficient of the 9-degree term of the polynomial. [2] As another example, for degree $d = 5$, you just need to replace `'9'` (given as input argument in the learning script call above), with `'5'`.

The actual regression algorithm is implemented in two different programming languages/environments: a Matlab-like programming language which should run in both Matlab and Octave, and Python. The wrapper assumes that you have installed either Matlab or Octave; [3] or Python. Note that the Python version requires Numpy and Scikit-learn. You need to edit the line in the script corresponding to the call to Matlab or Octave, or Python, and replace it with the correct command call and path to the Matlab or Octave, or the Python program installed in your system. Once you have specified a program and edited the script accordingly, the wrapper will call the corresponding program for you. I recommend you use the command-line version of Octave or Matlab, or Python, unless you are familiar enough with this systems, and bash-shell programming, to implement and execute everything directly on Octave or Matlab, or Python. *For students wanting to use the alternative black-box script provided for simple* 1*-dimensional polynomial regression, it is essential that they contact the instructor ASAP if they have trouble setting up and using the black-box script.*

Learning to Predict U.S.A. Working-Age Population Indicator

You will learn the best d-degree polynomial using 6-fold CV on the training data to select the optimal polynomial-degree value d to use from the set $\{0, 1, 2, 3, 4, 5, \ldots, 12\}$ (that is, $d = 0$ means using just a constant, $d = 1$ means standard linear regression, etc., up to $d = 12$ degree polynomial).

For CV, you must create each fold by splitting the examples in the training dataset using the *same order* as they appear in the data file. Specifically, the *first* fold corresponds to the examples with indexes $1, 2, 3, 4, 5, 6, 7$; the *sixth* fold corresponds to the examples with indexes $36, 37, 38, 39, 40, 41, 42$; and similarly for the indexes to examples in each of the other folds.

During each of the 6 folds of CV, you will obtain a RMSE value for each d value considered. Student must appropriately record those values so that they can evaluate/report the *average* of the RMSE values obtained for each d value during each of the 6 folds of CV.

Suppose d^* is the d value with the lowest RMSE after 6-fold CV. Then students must obtain the coefficient-weights of the d^*-degree polynomial using *all* the training data. Report the *coefficient-weights* for d^*, and the corresponding values of the *training and test* RMSE obtained for the resulting polynomials. Student must also plot all the training data along with the resulting polynomial curves for d^*, for the range of years 1968-2023 as input.

Once again, as with the ML regression algorithm described above, students are encouraged to write code for the CV process and the final learning, both including the data transformation/normalization described earlier, as weel as the final evaluation tasks from scratch; alternatively, students can use existing libraries or tools as long as they call those libraries or set up those tools in a way that would produce the same results as they would have obtained if they had coded the entire process themselves from scratch.

[2] Said differently, the file `w.dat` would encode the 9-degree polynomial $h(x) = w_0 + w_1\, x^1 + w_2\, x^2 + \cdots + w_9\, x^9$.

[3] GNU Octave is freely distributed software. See `http://www.gnu.org/software/octave/` for download information.

What to Turn In

You must submit the following (electronically via Canvas):

1. A **written report** (*in PDF*) that includes (1) the averages of the RMSE values obtained during the 6-fold CV for each case; (2) the optimal degree d^* obtained via the 6-fold CV; (3) the coefficient-weights of the d^*-degree polynomial learned on all the training data; (4) the training and test RMSE of that final, learned polynomial; (5) the plot containing all the training data along with the resulting polynomial curves for that final, learned polynomial with degree d^*, for the range of years 1968-2023 as input; and (6) a brief discussion of your findings and observations.

2. All your **code and executable** (as a tared-and-gziped compressed file), with instructions on how to run your program. A platform-independent standalone executable is preferred; otherwise, also provide instructions on how to compile your program. In general, the submitted program must be able to be executed as a standalone program from the command-line; and maybe only requiring an additional, previous compilation step, in which case, the compiler must also be excutable from the command-line. (*Student must use standard tools/compilers/etc., generally available for* **all** *popular platforms. Also, students* **must not** *submit source code that relies on or assumes that the resulting program will be run within a specific software-development IDE such as Microsoft's Visual Studio, Sun Microsystem's Eclipse, or Apple's Xcode.*)

Collaboration Policy: *While discussing general aspects of the project with peers is generally OK, each student must write and turn in their own report, code, and other required materials, based on the student's own work.*