Learning Systems Project

Please read the full description of this project before starting to work on it.

The project can be done in a group or one of two students maximum. The official deadline to submit the report is **Monday 9 March 2020**. However, if you still need some time to finish writing your report, then extension of the deadline is possible (until the end of March) if you justify why it is needed.

In this project you will work on both regression and classification problems. A total of 6 datasets (3 for regression, 3 for classification) are provided to you. You can use machine learning libraries in Python such as scikit-learn: https://scikit-learn.org/stable/

1. Datasets Description

1.1. Estimating cetane number for diesel fuel (Regression)

Your task for this dataset is to see whether it is possible or not to estimate the cetane number for diesel fuel from the near infrared spectrum for the fuel.

There is a total of 245 observations. You receive 133 of these for training and validation, and 112 are kept for testing.

You are given the file ***cnDieselTrain.mat***, which contains three matrices: *cn-TrainX* (401 × 133), *cnTrainY* (1 × 133), and *cnTestX* (401 × 112). The first matrix (*cnTrainX*) contains the IR-spectrum for each sample, one column per sample. The second matrix (*cnTrainY*) contains the output value cetane number for each diesel fuel. The third matrix (*cnTestX*) contains the input (IR-spectra) for each sample in the test data set.

The spectrum has 401 channels (features), and the data must first be reduced in dimensionality, e.g. by transforming to a principal component basis (which is how it is done in the commercial application) or by selecting appropriate features.

1.2. Modeling the need for cooling in a H2O2 process (Regression)

Some background: This is a non-linear regression task. The process that generated the data is EKA chemicals Hydrogen Peroxide production. Data is available from 1997-06-01 thru midnight 1998-12-31, a total of 19 months. The interval between each measurement is 10 minutes and some periods of the data is missing e.g., where service have been conducted and some very fast transition. Lastly the data has been decimated to 30 min measurements by averaging over 3 samples. The output is the valve opening of a valve connected to one (out of two) heat exchangers. This opening is part of a feed-back control loop based on the temperature of the fluid that passes thru the heat exchanger. The goal with the control is to keep the temperature constant and the fluid in liquid format. In cases where a large amount of cooling is required (large opening of the valve) it's indicating that the fluid is in transition to gas form and this information is of particular significance. The goal of the modeling is to construct a model with all or part of the variables available to model the valve opening.

The data is supplied in a MATLAB file, ***ChemTrainNew.mat***, and consist of three matrices: *XtrainDS* (4466x65), *YtrainDS* (4466x1) and *XtestDS* (2971x65). The input matrix (*XtrainDS*) contains all variables to the process (the first column is time, which should not be considered as a feature) and the output matrix (*YtrainDS*) contain the valve opening. The third matrix (*XtestDS*) is test data consisting of all inputs from a time period that follow the 19 month training data.

1.3. Predicting power load (Regression)

The task is a nonlinear regression: To predict the power load for Puget Sound Power & Light Co. 24 hours in advance, at 8 in the morning, when the current day is a working day and tomorrow is a working day (to make the problem a little bit easier, because things look a little different when tomorrow is a holiday or when the current day is a holiday). To solve the problem, you get observations for the period January 1985 – October 1990 (in all seasons). To test your system, we have withheld data from the winter months (November– March) of 1990/1991 and 1991/1992. You will be given the input data corresponding to those months, without information about the correct output, and asked to provide predictions for them. The 15 input variables are:

1. The current power load (MW)
2. Average power load over the last 24 hours (MW)
3. Average power load over the last week (MW)
4. Peak power load during the last 24 hours (MW)
5. Peak power load during the last week (MW)
6. Forecasted temperature 24 hours ahead (Farenheit)
7. The current temperature (Farenheit)
8. Average temperature last 24 hours (Farenheit)
9. Average temperature last week (Farenheit)
10. Variance of the temperature over the last 24 hours (Farenheit2)
11. Variance of the temperature over the last week (Farenheit2)
12. Average forecasted temperature for the next 24 hours (Farenheit)
13. Variance of forecasted temperature for the next 24 hours (Farenheit2)
14. cos(2π*daynum) where daynum = (the number of the day in the year)/365
15. sin(2π*daynum)

These inputs are the result of quite a lot of variable selection so you can assume that these variables should all be used (but you can also check if feature selection would improve your results). The output that you shall predict is the load 24 hours ahead.

You are given the file ***PowerTrainData.mat***, which contains the following matrices: *powerTrainInput* (15 × 844), *powerTrainOutput* (1 × 844), *powerTrainDate* (1 × 844), and *powerTestInput* (15 × 115). The *powerTrainDate* is the date for the training observations (in MATLAB datenum format).

1.4. Thyroid disease (Classification)

The task here is to tell if a particular set of measurements comes from a person who is normal or suffers from being hypothyroid or hyperthyroid (i.e. 3 output categories).

There are 7200 observations representing patients. You are given 5000 of these, and 2200 are withheld for testing. There are 21 variables (there is no information on what these represent). You are given the file ***thyroidTrain.mat***, which contains the matrices *trainThyroidInput* (5000 × 21), *trainThyroidOutput* (5000 × 3), and *testThyroidInput* (2200 × 21). The first matrix, *trainThyroidInput*, contains the input patterns for the training data. The second matrix, *trainThyroidOutput*, contains the outputs coded in a "1-out-of-3" fashion (i.e. as a one-hot-vector). That is, the outputs are coded as (1,0,0), (0,1,0), or (0,0,1). The third matrix, *testThyroidInput*, contains the inputs for the test data.

1.5. Breast cancer (Classifciation)

The task is a classification task: To tell if a patient has a benign or malign breast cancer, based on image features from a Fine Needle Aspiration (FNA). The diagnosis test is done in the following way:

- An FNA is taken from the breast mass. This material is then mounted on a microscope slide and stained to highlight the cellular nuclei. A portion of the slide in which the cells are well-differentiated is then scanned using a digital camera and a frame-grabber board.
- The user then isolates the individual nuclei using an image processing software.
- When all (or most) of the nuclei have been isolated, values for each of ten characteristics of each nuclei are computed, measuring size, shape and texture. The mean, standard error and extreme values of these features are computed, resulting in a total of 30 nuclear features for each sample.

The ten nuclei characteristics are:

(a) radius (mean of distances from center to points on the perimeter)
(b) texture (standard deviation of gray-scale values)

(c) perimeter
(d) area
(e) smoothness (local variation in radius lengths)
(f) compactness (perimeter2/area - 1.0)
(g) concavity (severity of concave portions of the contour)
(h) concave points (number of concave portions of the contour)
(i) symmetry
(j) fractal dimension (“coastline approximation” - 1)

There are 569 observations, of which 400 are provided to you for training. You are given a file, ***cancerWTrain.mat***, which contains the matrices *cancerTrainX* (30×400), *cancerTrainY* (1×400), and *cancerTestX* (30×169). The output is coded as 0 = benign and 1 = malign.

1.6. Electrocardiograms (Classification)

In this dataset, the task is to tell if a patient suffers from Transmural Ischemia (TI) or not, based on the signal from a 12 channel electrocardiogram (ECG). The 12 ECG channels are called V1, V2, V3, V4, V5, V6, aVL, I, -aVR, II, aVF, and III. There are 300 observations: 150 control subjects (both healthy subjects and subjects suffering from heart infarction, but not TI), and 150 subjects that suffer from TI. For each subject you are given 26 features for each ECG channel, i.e. 26×12 = 312 features per subject. The features are 26 features for each one of the 12 channels in the ECG (26 × 12 = 312). The 26 features are denoted I dur, Q dur, R dur, S dur, Rp dur, Sp dur, K dur, I ampl, Q ampl, R ampl, S ampl, Rp ampl, Sp ampl, K ampl, QRS area, QRS dur, Tmaxampl, Tminampl, timeTmax, timeTmin, ST ampl0, ST ampl20, ST ampl40, ST ampl60, ST ampl80, ST ampl100. The “dur” features are the durations (in time) between different parts of the ECG (different points on the ECG are denoted P, Q, R, S, T...etc.). The “ampl” are the amplitudes at these points. The “QRS area” variable denotes the area of the QRS peak complex. The “timeTmax” and “timeTmin” are the times for the maximum and the minimum signal in the ECG. The “ST amplXX” are the amplitudes at different points in the “ST” interval (which are generally thought to be important for doing the classification). The most important features, believed by the physicians, are the times, i.e. features 19–26 for each channel. These correspond to inputs 19–26, 45–52, 71–78, and so on.

You are given a file, ***ECGITtrain.mat***, which contains the matrices *inputECGITtrain* (200 × 312), *outputECGITtrain* (200 × 1, i.e. a column vector), and *inputECGITtest* (100 × 312). The first matrix is the inputs for 200 training patterns. The *outputECGITtrain* vector is the target values for the 200 patterns, where 1 correspond to a TI pattern and 0 to a non-TI pattern. The last matrix contains the inputs for the 100 test patterns. The true outputs for the test data are not provided to you.

2. Instructions

You are free to do your own analysis and use any ML methods you want (for classification, regression, feature selection, dimensionality reduction etc.). However, we provide here some general instructions that can help you in your analysis:

- First, figure out how to do the principal component analysis (PCA) to visualize the data.
- Rank the features based on their importance and plot the most important ones.
- Get acquainted with the data. Do scatter plots and explore the relations between inputs and output to get a feel for how this relationship looks like.
- Check if data normalization is needed or not (depending on the dataset and models you use).
- Start by constructing simple linear models and see how well the output can be estimated (that is, estimate the generalization error using k-fold-cross-validation). Then you can gradually try more and more complex non-linear models. Each time, optimize the number of principal components (if you use PCA) or the number of selected features (if you use feature selection), i.e. determine how many components/features you need to get good generalization performance. It makes sense to use the MSE error to evaluate the performance of your regression models, and the classification error (or accuracy) to evaluate the performance of your classification models. When you compare several models, it makes sense to not only report the average estimate (e.g. error) of the 10-fold-cross-validation, but also the standard deviation of your estimates. For

each dataset, you can also test and report if the performances achieved by the two best models, are significantly different or not (i.e. if one is significantly better than the other). It can be wise to use both training and test set input data when computing the principal components (with PCA), to get better statistics.

- Hyperparameters tuning: Don’t forget to optimize the important hyperparameters of your models (e.g. regularization parameters etc.) based on k-fold-cross-validation.
- If a model performs significantly better (or worse) than the others on a specific dataset, then try to analyze and understand why.
- For each dataset, hand in a file containing the test results (i.e. predicted outputs for the test data) for your best model, so that we can check if it corresponds to your estimate of the generalization error/accuracy reported in your report. Note: the outputs corresponding to the test data are not provided to you in this project.

3. Report and presentation of results

You will present the results from your project in two ways: (1) A written report where the main conclusions are presented together with figures and tables supporting your conclusions. The deadline for this report is **Monday 9 March 2020** with a possible extension if justified. (2) An oral presentation, of 20 minutes on Week 11. The slides of your presentation should show the project results achieved so far as well as a state-of-the-art section which refers to existing research articles/papers related to your project (if any).

The report should be about 10 pages, including figures and tables, and can be structured as follows:

1. **Introduction**. Brief presentation of problem, 1 page.
2. **State-of-the-art**. Brief description of research papers doing work related to your project, 1 page.
3. **Methodology**. Brief listing of methods used, 1 page.
4. **Data**. Presentation of your dataset with important observations, 1-2 pages.
5. **Results and their interpretation**, 3-5 pages.
6. **Discussion**. Conclusions about your results and comparison to other researchers' results, 1 page.

When you are finished with your report, you should pack it (zip it) together with other important parts of your project (such as your Python code and the results achieved on the test datasets) ; then, upload it as a zip file to Blackboard. The idea being that someone else could unpack it and repeat the main steps in your analysis without rewriting everything.

4. Useful links and readings from the scikit-learn library

- ML models for classification and Regression:
  - https://scikit-learn.org/stable/supervised_learning.html#supervised-learning
- Feature Selection:
  - https://scikit-learn.org/stable/modules/feature_selection.html#feature-selection
- Principal Component Analysis (PCA) for dimensionality reduction:
  - https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
  - Examples: https://scikit-learn.org/stable/modules/decomposition.html#pca
- Model selection: https://scikit-learn.org/stable/model_selection.html
  - Cross-validation: https://scikit-learn.org/stable/modules/cross_validation.html
  - https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html
  - Hyper-parameters tuning: https://scikit-learn.org/stable/modules/grid_search.html
  - Model evaluation: https://scikit-learn.org/stable/modules/model_evaluation.html
  - Validation curves: https://scikit-learn.org/stable/modules/learning_curve.html