CS3920 Assessed Coursework Assignment 2

November 26, 2024

This assignment must be submitted by 16 December 2024, 14:00. Feedback will be provided by 23 January 2025.

Learning outcomes assessed

Be able to use and implement machine-learning algorithms, with the SVM, neural networks, and cross-conformal prediction algorithms as examples. Have an understanding of ways to apply the ideas and algorithms of machine learning in industry.

Instructions

The coursework assignment must be completed strictly individually. You should not use in your submission any downloaded code or existing implementations of the learning algorithms (however, you may use `scikit-learn` functions). The submission is entirely electronic. You should submit one file (a Jupyter notebook) via the module's Moodle page.

Give your Jupyter notebook a reasonable name, such as `CS3920_A2.ipynb`. It should contain your code for the methods you have implemented, your numerical results, and comments (as explained below). Ideally, include your module code ("CS3920") in the file name; if not, please mention it prominently inside the notebook.

The file that you submit cannot be overwritten by anyone else, and it cannot be read by any other student. You can, however, overwrite your submission as often as you like, by resubmitting, though only the last version submitted will be kept. Submissions after the deadline will be accepted but they will be automatically recorded as late and subject to College Regulations on late submissions (see below). Please note that all your submissions will be graded anonymously; your name should not appear anywhere in your submission.

Use of AI The use of generative AI tools (e.g., Microsoft Copilot, ChatGPT, Grammarly) is not permitted for this assignment. More information regarding the use of generative AI in your studies can be found here.

Academic misconduct Coursework submissions are routinely checked for academic misconduct (working together, copying from sources, etc.). Penalties can range from a 10% deduction of the assignment mark, zero for the assignment, or the case being referred to a Senior Vice-Principal to make a decision (for repeat offences). Any use of generative AI tools constitutes academic misconduct. Further details can be found **here**.

Extensions University extensions are available for this assignment. If you are unable to meet the assignment deadline, please read the documentation here.

Extenuating circumstances If you submit an assessment and believe that the standard of your work was substantially affected by your current circumstance then you can apply for Extenuating Circumstances. Details on how to apply for this can be found **here**. It is essential you read the accompanying documentation on the above link carefully.

Late submission In the absence of acceptable extenuating cause, late submission of work will be penalised as follows:

- for work submitted up to 24 hours late, the mark will be reduced by ten percentage marks;
- for work submitted more than 24 hours late, the maximum mark will be zero.

Coursework

This assignment requires the ability to use the SVM method and pipelines in `scikit-learn`, and also either an implementation of a cross-conformal predictor for classification or the ability to use neural networks in `scikit-learn`. All code (in Python) and text should be submitted as a single Jupyter notebook. Please leave all output produced by the system (i.e., do not remove the contents of cells like "Out [1]:"; if you do, you will lose marks).

Datasets

This assignment uses two datasets, `wine` (available in `scikit-learn`) and the `USPS dataset` of handwritten digits. The latter dataset can be downloaded from the web site of Hastie et al. (2009):

- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Web site for *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Second Edition, `https://web.stanford.edu/~hastie/ElemStatLearn/`.

Click on the `Data` button on the left and go to the end (`ZIP code`). You will need to guess the format of the dataset; since these are handwritten digits, the labels are digits 0–9; the other entries are the pixels of their images. (Each sample is represented by a line in the file; this single line is very long and will be split into several on your computer screen.)

Tasks

Do the following for both datasets (you might prefer to start with the smaller `wine` dataset):

1. Load the data set into Python using, e.g., `load_wine` or `genfromtxt`, as appropriate. In the case of the `USPS dataset`, merge the original training and test sets into one dataset.

2. Divide the dataset into a training set and a test set. You may use the function `train_test_split`. Use your birthday in the format DDMM as `random_state` (omit leading zeros if any).

3. Using cross-validation and the training set only, estimate the generalization accuracy of the SVM with the default values of the parameters. You may use the function `cross_val_score`.

4. Find the test error rate of the SVM with the default values of parameters, compare it with the estimate obtained in the previous task (task 3), and write your observations in a markdown cell of your Jupyter notebook.

5. Create a pipeline for SVM involving data normalization and `SVC`, and use grid search and cross-validation to tune parameters `C` and `gamma` for the pipeline, avoiding data snooping and data leakage. You may use the `scikit-learn` class `GridSearchCV` (along with other `scikit-learn` classes). Experiment with different ways of doing normalization (such as `StandardScaler`, `MinMaxScaler`, `RobustScaler`, and `Normalizer`). Which ways are appropriate for either dataset? (The answer, which should be written in your Jupyter notebook, may depend on the results that you obtain for the next task.)

6. Fit the `GridSearchCV` object of task 5 to the training set and use it to predict the test labels. Write the resulting test error rate in your Jupyter notebook.

7. Implement a cross-conformal predictor. You may use the `KFold` class for splitting into folds (start from 5 or 10 folds). For computing the conformity scores for each fold, you may use one of the `GridSearchCV` objects that you created in task 5 in combination with the `decision_function` method (see Section 3 of Lab Worksheet 9 for examples). Run your cross-conformal predictor on the two datasets, training it on the training set and testing on the test set.

   - To check its validity, produce a calibration curve, plotting the percentage of errors made on the test set[1] vs the significance level $\epsilon \in [0, 1]$.
   - Compute the average false p-value on the test set.

8. An alternative to implementing a cross-conformal predictor is to experiment with a neural network. Perform tasks 3–6 for the `scikit-learn`

[1] As usual, a cross-conformal predictor makes an error if its prediction set fails to cover the true label. The percentage of errors is the number of errors made on the test samples divided by the size of the test set.

class `MLPClassifier`, as described in Lab Worksheet 8, Section 1. It is up to you to decide which parameters to fit using grid search. And in this task (and only in this task) it is OK to ignore the warnings (if any).

Some tasks might be hard to separate (such as 5 and 6), and it's OK to implement several tasks in the same code cell.

Details

The cross-conformal predictor is described in Chapter 9. For other useful information and code, see Lab Worksheet 9 (e.g., the class `KFold` is described in Section 2 and plotting calibration curves is discussed in Section 5).

You are not allowed to use any existing implementations of the cross-conformal and related predictors, but you are allowed to use `scikit-learn` functions.

Make sure to include the following in your Jupyter notebook (for both datasets) if you are implementing a cross-conformal predictor:

- Your estimate of the generalization accuracy of SVM with the default values of the parameters (task 3) and the test error rate for the SVM with the default values of the parameters (task 4).
- The test error rate of your `GridSearchCV` object in task 6 for at least two different ways of normalization (including `Normalizer`).
- The average false p-value and calibration plot for your cross-conformal predictor.

Therefore, there should be at least $2 \times (2 + 2 + 1) = 10$ numbers and two plots in your Jupyter notebook.

If you are experimenting with a neural network, you should include the following numbers (for both datasets and both classifiers):

- Your estimate of the generalization accuracy with the default values of the parameters (task 3) and the test error rate with the default values of the parameters (task 4).
- The test error rate of your `GridSearchCV` object in task 6 for at least two different ways of normalization (including `Normalizer`).

Therefore, there should be at least $2 \times 2 \times (2 + 2) = 16$ numbers.

Marking criteria

To be awarded full marks you need both to submit correct code and to obtain correct results on the given data sets. Even if your results are not correct, marks will be awarded for correct or partially correct code. Ideal completion of tasks 1–6 will give you 60%. The remaining 40% will be awarded for your implementation of a cross-conformal predictor. In you are experimenting with a neural network instead of implementing a cross-conformal predictor, you will get 85% for an ideal submission; an ideal submission will involve selecting more than one parameter for the neural network.

Removing the output produced by the system (or submitting a .py file instead of a Jupyter notebook) may result in losing up to one half of the marks.

Extra marks

There are several ways to get extra marks (at most 10%) that will be added to your overall mark (the sum will be truncated to 100% if necessary). Extra marks will be given for:

- A systematic investigation of the optimal number of folds for the cross-conformal predictor (taking into account both validity and efficiency).
- Getting rid of warnings for `MLPClassifier` might give you a couple of extra marks.
- Any other interesting observations about the methods or the datasets (discuss these in your Jupyter notebook).

Appendix Unusual features of `scikit-learn`

- When using the method `decision_function` in `SVC` please avoid applying it to one test sample; it may give wrong results. It is safe to apply it to your whole test set at once.
- `GridSearchCV` runs only on one thread by default. If you set the variable `n_jobs = -1` in `GridSearchCV` then it allows the function to run on all threads of your CPU (`n_jobs = -2` runs on all but one thread etc.). It may improve the run-time considerably.