EE 559 **Course Project Assignment** Posted: Thurs., 4/6/2017
Jenkins Due: Tues., 5/2/2017, 1:00 PM

Introduction

The goal of this project is to develop a pattern recognition system that operates on a given real-world dataset that is nontrivial to solve. In doing so, you will apply tools and techniques that we have covered in class. You will also confront and solve issues that occur in practice and that have not been covered in class. Discussion sessions will give you some tips and pointers for this; also piazza discussions will likely be helpful.

Each student will do his or her own project. In your final report you will describe the work that you have done, and where you describe the work of others, it must be cited and referenced as such. Plagiarism (copying information from elsewhere without crediting the source) of text, figures, or code will cause substantial penalty.

You will have significant freedom in designing what you will do for your project. You must cover various topics that are listed below (as "Required Elements"); the methods you use, and degree of depth you go into each topic, are up to you.

Everyone will base their work on the same dataset. Thus, collaboration and comparing notes on piazza may be helpful, and may make it more fun and engaging.

Dataset: German credit risk

There are 2 versions of this dataset.

It is recommended that you start with the first dataset (because it is simpler and requires less formatting and preprocessing). It has 9 features and 1000 samples. The features describe different aspects of each application for credit, such as applicant's age, checking account balance, etc. This is a 2-class problem in which the label is whether or not the applicant is a good credit risk or not. The dataset is posted on D2L as:

Proj_dataset_1.csv

This is a comma separated values (csv) file; it has one header row, then 1000 rows, one for each of the 1000 data points. The first column is a data-point index (0-999), the last column is the class label (1 for good credit risk, 2 for bad credit risk), and each of the other columns represents a feature. The dataset is from Kaggle (with class labels added), and is described at:

https://www.kaggle.com/uciml/german-credit

Once you have done some work with this dataset, you are welcome (but not required) to try adding features from the second dataset. The second dataset is posted on D2L as:

Proj_dataset_2.csv

Proj_dataset_2_documentation.pdf

This csv file is the original UCI dataset, in a more raw form. It has 20 features, and the same 1000 samples (in the same order) as the first dataset. As described in the Kaggle dataset description, the Kaggle (Proj_dataset_1) features were developed from the original UCI dataset (Proj_dataset_2), by reformatting the values into a more meaningful form (like integers, or descriptive categories like "little", "moderate", etc.). Some of the original features were combined, and some of the original features were discarded.

To use Proj_dataset_2, you will need to reformat any feature values from raw form (A21, A22, etc.) to a more useful form. You can compare with Proj_dataset_1 to see which features are included in both datasets. You can add some of the features missing in Proj_dataset_1 if you want to try using more features that may supply additional useful information. Or, you can use a reformatted version of Proj_dataset_2 directly with all its features if you prefer. The documentation posted on D2L gives a brief description. This dataset comes from the UCI Machine Learning Repository at:

https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29

Computer languages and available code

You may use Matlab, Python, or C/C++. (To use another language, please ask the TA or instructor first.)

Matlab and Python are recommended; Matlab is supported.

You may use any toolbox or libraries you prefer.

For Matlab users, some common choices include Pattern Classification Toolbox, LibSVM and PMTK.

For Python users, scikit-learn is the most common choice.

Be sure to state in your project report what languages and toolboxes/libraries you used, and what you coded up yourself. In the code that you turn in, please show code you wrote yourself in black, and code you got from elsewhere in (medium or dark) blue (or underlined if you prefer). Also indicate using comments where the code came from (e.g., provide a link). If you start from code taken from somewhere else, and modify it for your needs, then state so in the comment.

Required elements

- **Preprocessing** [**Hint:** Discussion 12 has provided tips; also prior homeworks.]
  - It is highly recommended to let Matlab (using the "import" button) or Python (using pandas) to handle csv parsing.
  - Consider the feature types: numerical, ordered categorical, or unordered categorical. Re-cast the representation of data as appropriate.
  - Missing data. If there is any missing data, decide how you will deal with it.
  - Normalization. Decide whether, and how, you will normalize the data, and which features will be normalized.

- **Feature-space dimensionality adjustment.**
  - Try reducing or expanding the dimensionality. Use a method (*e.g.*, model selection using cross-validation) to find a good dimensionality and feature set.
- **Training and classification.**
  - Try some of the methods we have covered in class. Try at least 3 different classification techniques; include both distribution-free and statistical classification.
- **Proper dataset (and subset) usage.**
  - Final test set, training set, validation sets, cross validation.
- **Interpretation.**
  - Interpret intermediate results and final results. Can you explain (or hypothesize) reasons for) what you observe?

Evaluation of performance.

- Randomly set aside 20% of the dataset as the test set (preserving percent representation of each class), or else use cross-validation for final error estimation. Be sure to describe in your final report the method you used, including how test set(s) were generated and used.
- Use accuracy (percent) and mean F1 score (an introduction is attached to the end of the assignment) as your main performance measures.

Tips

1. Be careful to keep your final test set uncorrupted, by setting it aside at the beginning, or by using cross-validation procedures appropriately.
2. If possible, it can be helpful to consider degrees of freedom (d.o.f.), and number of constraints, for example by using the rule of thumb of at least 3 to 10 times more constraints (samples) than d.o.f. But, this is easier to evaluate for some classifiers than for others; and yet for others, such as SVM, it doesn't directly apply.
3. It's good to start out with a baseline system and result to compare with. The choice of baseline is up to you. For example, it could be assignment based on priors only, or based on a simple classifier using the given feature set.

Grading criteria

Grading criteria will include: inclusion of required elements; understanding and interpretation (of approach, algorithms used, and results); technical soundness and final performance; quantity and quality of effort; and report write-up (clarity, conciseness, and completeness).

Final report

More detailed guidelines for the written report will be posted later.

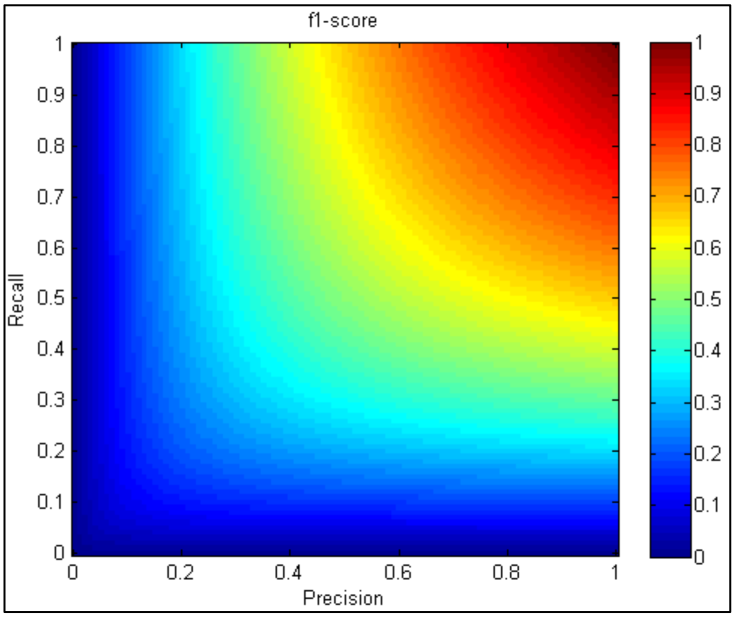
The F1 score

For each class, the F1 score is defined as:

$$\mathrm{F1} = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

where $precision$ and $recall$ are defined in Discussion 12.

The F1 score is in the range of $[0,1]$, the larger the better. It will have a large value when both $precision$ and $recall$ are relatively large, but much smaller when one of the two is very large and the other is very small. You can convince yourself with the plot below:



Matlab users can use the `confusionmat()` function to get the confusion matrix and calculate the scores from there.

Python users can use the `sklearn.metrics.classification_report()` function to get the scores directly.