EE6550 Machine Learning HW3 Kernel SVM README

An Extension of HW2 linear SVM. This is a complete Support Vector Machine that not only support linear but also non-linear binary Classification, trained with Sequential Minimal Optimization.

- Author: Yu-Chun (Howard) Lo
- Email: howard.lo@nlplab.cc

User Manual

Dev Environment

- Developed under Anaconda 4.3.0 (x86_64).
- Require Numpy for matrix operations.
- Tested on Python 3.6.0.

File Structure

- `/dataset`: **(Important)** The program reads datasets from this folder and performs training and testing on the specified training and test data files. (See the *Dataset Format* section below)
- `/logs`: **(Important)** Output reports are stored in this folder.
- `/hypothesis`: **(Important)** Output hypotheses are stored in this folder. (There is **timestamp** at the end of the file name, which corresponds to the report)
- `svm.py`: Support vector machine (SVM) model.
- `utils.py`: Some utilities used by this program, such as loading dataset, normalize labels, etc.
- `main.py`: **(Important)** The main program. User should train a SVM by running this program.

Dataset Format

- Currently, the program only supports reading `.csv` file.
- The class label of each item should locate at the first column. The class labels should only be **binary**, e.g. `{+1, -1}`, `{1, 0}` or `{'+', '-'}`, etc.
- **(Important)** If you want to train your SVM with your own dataset, please be sure that you've followed the required format described above, and have placed your own training and test data files in the `/dataset` folder.

Getting Started

Train your SVM by running `python main.py` in terminal. Be sure that your terminal is under the same directory as `main.py`.

Note that we've set default values for required input arguments. Run `python main.py --help` to view input arguments information shown below.

```
usage: main.py [-h] [--header_filename HEADER_FILENAME]
               [--train_filename TRAIN_FILENAME]
               [--test_filename TEST_FILENAME] [--K K] [--C C]
               [--kernel_type KERNEL_TYPE] [--poly_degree POLY_DEGREE]
               [--rbf_sigma RBF_SIGMA] [--enable_heuristic]
               [--enable_kernel_cache] [--max_iteration MAX_ITERATION]

Binary support vector classifer.

optional arguments:
  -h, --help            show this help message and exit
  --header_filename HEADER_FILENAME
                        Training dataset with header csv. This is used to
                        output hypothesis. (Default:
                        "alphabet_DU_training_header.csv")
  --train_filename TRAIN_FILENAME
                        Training dataset csv. (Default:
                        "alphabet_DU_training.csv")
  --test_filename TEST_FILENAME
                        Training dataset csv. (Default:
                        "alphabet_DU_testing.csv")
  --K K                 Denotes for "K"-fold cross-validation for determine
                        the optimal hyper-parameters for SVM. (Default: None)
  --C C                 Parameter for penalty term. (Default: 0.1)
  --kernel_type KERNEL_TYPE
                        Kernel type to be used in SVM. Acceptable kernel type:
                        "linear", "poly", "rbf". (Default: None)
  --poly_degree POLY_DEGREE
                        Degree of the polynomial kernel function ("poly").
                        Ignored by all other kernels. (Default: 3)
  --rbf_sigma RBF_SIGMA
                        Sigma term in RBF (guassian). Ignored by all other
                        kernels. (Default: 0.5)
  --enable_heuristic    Whether use Platts heuristics to train SVM. (Defualt:
                        False)
  --enable_kernel_cache
                        Whether precompute kernel results. This can speed up
                        training but need time to initialize when data is
                        large. (Defualt: True)
  --max_iteration MAX_ITERATION
                        Max iteration for SMO training algorithm to avoid not
                        converging. (Defualt: 3000)
```

For Grading Session

Here we show some guides for different test scenarios:

- Place the training data file(e.g. `xxx_training.csv`), testing data file(e.g. `xxx_testing.csv`) and training header file(e.g. `xxx_training_header.csv`) in the `/dataset` folder before running `main.py` with specified `--train_filename`, `--test_filename` and `--header_filename`.
- For choosing a PDS kernel, for example:
    - For polynomial kernel, specify `--kernel_type="poly"` or maybe along with the free parameter `--poly_degree=3`.

    - For RBF kernel, specify `--kernel_type="rbf"` or maybe along with the free parameter `--rbf_sigma=0.5`.
- For performing K-fold cross-validation, for example, specify `--K=5`.
- All the required output information, such as class label mapping, cross-validation history, optimal hyper-parameters, etc., are stored at the `logs/` folder. Note that the log file name indicates what kernel type and number of K-fold you choosed, and when you run the program. This naming convension aims to help graders to choose which report to check after running the program.
- Note that the csv file name of SVM hypothesis in the `hypothesis/` folder is concatenated with the timestamp. This aims to let the graders know which hypothesis file to choose to test after running the program. (You may remove the timestamp for grading)
- Summing up, you may want to run the following commands in the different test scienarios:

```
(For 5-fold cross-validation on polynomial kernel)
>> python main.py --header_filename="xxx_training_header.csv"
--train_filename="xxx_training.csv" --test_filename="xxx_testing.csv" --K=5
--kernel_type="poly"

(For 5-fold cross-validation on RBF kernel)
>> python main.py --header_filename="xxx_training_header.csv"
--train_filename="xxx_training.csv" --test_filename="xxx_testing.csv" --K=5
--kernel_type="rbf"

(For specifying hyper-parameters when choosing polynomial kernel)
>> python main.py --header_filename="xxx_training_header.csv"
--train_filename="xxx_training.csv" --test_filename="xxx_testing.csv"
--kernel_type="poly" --C=1.0 --poly_degree=3

(For specifying hyper-parameters when choosing RBF kernel)
>> python main.py --header_filename="xxx_training_header.csv"
--train_filename="xxx_training.csv" --test_filename="xxx_testing.csv"
--kernel_type="rbf" --C=1.0 --rbf_sigma=0.5
```

Report

All required output information are stored at `logs/`. We've run on 24 different sets of hyper-parameters to select a set of optimal hyper-parameter.

- For polynomial kernel:
    - For 5-fold cross-validation results, check `logs/svm-poly-5-fold-[HH:MM:SS]`
    - For 10-fold cross-validation results, check `logs/svm-poly-10-fold-[HH:MM:SS]`
- For RBF kernel:
    - For 5-fold cross-validation results, check `logs/svm-rbf-5-fold-[HH:MM:SS]`
    - For 10-fold cross-validation results, check `logs/svm-rbf-10-fold-[HH:MM:SS]`

Note that the corresponding hypothesis csv file is indicated by the `HH:MM:ss` timestamp, which is the time that the program finished training.