Project 4

AdaBoost and Support Vector Machines for Unbalanced Data Sets

Assigned: Friday, November 16
Single Due Date (same for everyone): Friday, December 7, 23:59:59

Overview

In this project, you will make use of the LIBSVM public domain library for Support Vector Machines (SVMs), developed by Chang and Lin at National Taiwan University. This library provides software support for SVMs, with source code available for C++ and Java, and interfaces available to MATLAB, Python, Perl, Lisp, and several other programming languages/environments.

Here's the link to the LIBSVM library: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

In this project, you will use the LIBSVM library to perform classification tasks with three public-domain datasets. You will also implement AdaBoost.M1 (the multi-class version of AdaBoost), and explore its effectiveness for classifying unbalanced data using SVMs as the component classifiers.

Installing LIBSVM

First, follow the instructions on the LIBSVM website for downloading the LIBSVM software. If you want to interface a language other than C++ or Java to LIBSVM, select the appropriate link near the bottom of the above website. Note that the README file with the installation provides a lot of helpful information for using LIBSVM. Additionally, the documentation provided with this library gives several examples for using the SVM library. See also the following documentation, which is available on the LIBSVM website:

- *A Practical Guide to Support Vector Classification*, by Hsu, Chang, and Lin: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
- *LIBSVM: a Library for Support Vector Machines*, by Chang and Lin: http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf

Getting familiar with LIBSVM

To help get up to speed with LIBSVM, I recommend following the examples in Appendix A of the above "Practical Guide". The datasets are available to you online from the LIBSVM directory, already formatted into the form expected by LIBSVM, here:
http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

As a learning exercise, you should be able to recreate the results given in this Appendix. Note that these examples illustrate the importance of properly scaling the data and selecting the best parameters C and γ.

Datasets

The datasets you will be using for this project are all public-domain datasets. They have also already been scaled and formatted for LIBSVM (by the providers of LIBSVM), so you don't have to do any scaling or reformatting to prepare your data. We have also segmented out some of the data for testing. So, even though you may find similar datasets online, the 3 datasets below are the ones you should use:

I. Liver-disorders (2 classes, 6 features, 300 training instances, 45 test instances):

- liver-disorders_scale_train (training data set): http://web.eecs.utk.edu/~parker/Courses/CS425-528-fall12/liver-disorders_scale_train
- liver-disorders_scale_test (testing data set): http://web.eecs.utk.edu/~parker/Courses/CS425-528-fall12/liver-disorders_scale_test

II. Glass (6 classes, 9 features, 180 training instances, 34 test instances)

- glass_scale_train (training data set): http://web.eecs.utk.edu/~parker/Courses/CS425-528-fall12/glass_scale_train
- glass_scale_test (testing data set): http://web.eecs.utk.edu/~parker/Courses/CS425-528-fall12/glass_scale_test

III. Vowel (11 classes, 10 features, 528 training instances, 462 testing instances)

- vowel_scale_train (training data set): http://web.eecs.utk.edu/~parker/Courses/CS425-528-fall12/vowel_scale_train
- vowel_scale_test (testing data set): http://web.eecs.utk.edu/~parker/Courses/CS425-528-fall12/vowel_scale_test

Part 1: Using LIBSVM "grid.py" to find ideal C and γ values

Using a series of steps similar to those shown in Appendix A of the "Practical Guide", train an SVM using the tools of LIBSVM on each of the 3 datasets, using "grid.py" to find the best values of C and γ. You should use both a loose grid search and a fine grid search (as discussed in section 3.2 of the "Practical Guide"). Capture the plot of the grid search for each dataset for both the loose and fine grid searches [note that a png file with this information is output automatically for you by grid.py]. For each of these 3 datasets, compare the overall training accuracy of the default settings of LIBSVM with those that use the "best" C and γ values.

Part 2: Implement AdaBoost.M1 with SVMs as Component Learners

Implement AdaBoost.M1, except extend it to vary the value of γ to obtain different SVM component learners, as shown in Table 2 (algorithm AdaBoostSVM) of the Li, Wang, and Sung paper (*AdaBoost with SVM-based component classifiers*). Note that in the Li, Wang, and Sung paper, the σ parameter is the same thing as the γ parameter in LIBSVM. For the component SVM learners, use the ideal "C" value obtained from your earlier grid search, but vary the value of σ/γ, per the AdaBoostSVM algorithm. Keep track of the number of component SVM learners that your approach generates.

Part 3: Generate a "test mode" from your AdaBoost.M1 implementation

In order to test your AdaBoost.M1 implementation, you need to write an additional module that uses the component SVM hypotheses generated by AdaBoost.M1, and then classifies new input instances. Thus, for this part of the project, you will write a stand-alone program that uses the component learners generated from your AdaBoost.M1 algorithm, and then uses these learners to classify new instances that are provided by the user from a test file. The manner in which the user specifies the input file of test cases is up to you; either define it in a README, or as a clear command-line option.

Data to Gather

- For each dataset, include a graph of the grid search (from grid.py) results, for both the coarse grid search and the fine grid search.
- Include a table of the “ideal” C and γ values for each of the 3 datasets, which you derived from your coarse and fine grid searches.
- Report on the number of SVM component learners generated for each dataset by the AdaBoost.M1 algorithm (Part 2).
- For each dataset, compare the accuracies after training with: 1) the default LIBSVM parameters, 2) the “ideal” C and γ values from grid.py (Part 1), and 3) training using AdaBoost.M1 (Part 2). Your results should not only show overall performance for all classes combined, but also the performance for each class. Include a confusion matrix for each type of learning for each dataset.
- Analyze your results by comparing and contrasting grid.py’s performance (in Part 1) with AdaBoost.M1’s performance (in Part 2). For example, you should discuss which approach did better, and why you think it did better. Were the results consistent for each dataset? Did AdaBoost.M1 help? Why or why not, in your opinion? Is there any correlation between the degree to which the data is unbalanced and the effectiveness of SVMs (alone) vs. AdaBoost.M1? Other types of analyses of your results along these lines are encouraged.

CS425 – Undergraduate Instructions

Undergraduate Student Paper Guidelines

As part of your project, you must prepare a single 1-3 page document (in pdf format, using Latex and 2-column IEEE style) that presents the results listed in “Data to Gather”. The documentation you turn in must be in pdf format in a single file. Name your paper *yourlastname*-Project4-Paper.pdf.

Undergraduate Project Software

Your software submission should include the following:

- Your AdaBoost.M1 implementation, from Part 2
- Your AdaBoost.M1 testing program, from Part 3, including the learned component models.
- Sufficient instructions, makefiles, etc., needed to compile and run your project on the CS linux machines, including README files with instructions.

Undergraduate Grading

Your grade will be based on the quality of your project implementation and the documentation of your findings in the writeup. You should have a “working” software implementation, meaning that the learning algorithm is implemented in software, it runs without crashing, and performs the learning task. Your results description should be sufficient for the reader to tell that you have generated an AdaBoost.M1 learner.

Your grade will be based on:

- 75%: The quality of your final working code implementation, and software documentation. You should have a “working” software implementation, meaning that the learning algorithm is implemented in software, it runs without crashing, performs learning as appropriate for this project, and learns reasonably well. Your final code should have the training program and testing program, as outlined earlier in this document.
- 25%: Your paper, generated in Latex using IEEE styling, and submitted in pdf format – Figures and graphs should be clear and readable, with axes labeled and captions that describe what each figure/graph illustrates. The content should include all the results and discussion mentioned previously in this document.

Undergraduates: Turning in your project

You should submit your project using Blackboard by the deadline. This submission should include the following 2 parts:

1. Paper (in pdf, 2-column IEEE format, generated using Latex, named *yourlastname*-Project4-Paper.pdf)

2. All the programs, data files, include files, makefiles etc., needed to compile and run your project on the CS linux machines, including a README file that gives instructions on how to compile and run your code. Your main project code file should be called *yourlastname*-Project4.cc (or whatever extension is appropriate for the programming language you’re using). Pack your code together into a single tarball. When we unpack your files, we should be able to read your README file and follow the instructions to compile and run your code. We’ll be in a bad mood when grading your work if the instructions you provide do not work. So please test your instructions before you send your files to us.

CS528 – Graduate Instructions

Graduate Student Paper Guidelines

You must prepare a paper (3-6 pages) describing your project and results. Your paper must be formatted using Latex and the standard 2-column IEEE conference paper format, the same as in previous projects.

The paper you turn in must be in pdf format. Name your paper *yourlastname*-Project4-Paper.pdf. Your paper must include the following:

- An abstract of 200 to 300 words summarizing your findings.
- A brief introduction describing the learning task and your implementations.
- A detailed description of your experiments, with enough information that would enable someone to recreate your experiments.
- An explanation of the results, including all the data mentioned previously in this document under "Data to Gather", as well as your additional extensions (see below for suggestions). Use figures, graphs, and tables where appropriate. Your results should make it clear that the system has in fact learned.
- A discussion of the significance of the results.

The reader of your paper should be able to understand what you have done and what your software does without looking at the code itself.

Extensions for Grad Students

As always, graduate students are required to do more exploration of different parameter settings, or different techniques. Here are some possible avenues of exploration:

- Extend AdaBoost.M1 to not only vary the value of γ, but also to vary the value of C. The result will be a set of component SVM learners that correspond to points of the grid search used by grid.py. Explore the impact of this change.
- Explore different possible ranges of γ and/or C used in AdaBoost.M1.
- Explore the impact of the number of component learners on the performance of the overall learning system.
- Artificially undersample the training data, and use the regular SVM implementation (without AdaBoost), to see if it is possible to reduce the impact of the unbalanced data.
- Apply your AdaBoost.M1 algorithm to other datasets (e.g., on the LIBSVM website), and analyze the results.
- Investigate the feature selection tool in LIBSVM, and analyze its impact on learning performance.

Graduate Grading

Graduate students will be graded more strictly on the quality of the research and paper presentation. I expect a more thorough analysis of your results, and a working learning system. Your analysis should include further extensions selected from the above list, or other ideas you may have. The paper should have the "look and feel" of a technical conference paper, with logical flow, good grammar, sound arguments, illustrative figures, etc. As always, graduate students are expected to format their paper in standard IEEE conference format.

Please do not exceed 6 pages for your paper (and don't go smaller than 10 point font).

Turning in your project

You should submit your project using Blackboard by the deadline. This submission includes the following:

1. Paper (in pdf, 2-column IEEE format, generated using Latex, named *yourlastname*-Project4-Paper.pdf)

2. All the programs, data files, include files, makefiles etc., needed to compile and run your project on the EECS linux machines, including a README file that gives instructions on how to compile and run your code. Your main project code file should be called *yourlastname*-Project4.cc (or whatever extension is appropriate for the programming language you're using). Pack your code together into a single tarball. When we unpack your files, we should be able to read your README file and follow the instructions to compile and run your code. We'll be in a bad mood when grading your work if the instructions you provide do not work. So please test your instructions before you send your files to us.