EEL6825: Class Project

1. Introduction

A large part of your grade will be determined by the quality of your class project. You can do an individual project or team up with one or at most two other students in the class. Note, however, that two-person or three-person projects are expected to be two or three times as much work, respectively, as an individual projects. Important deadlines follow (these deadlines apply to all students, including FEEDS students):

1. ***Project description***: e-mail a description of your proposed project (no more than one single-spaced pdf page) to the TA Seth Mcneill <mcnese@ufl.edu> by ***midnight of November 6, 2003 (Thursday)***. Your description should include information on your particular application, where/how you will obtain data (e.g. from web, etc.), preliminary thoughts on feature extraction for your application, which methods/algorithms you intend to apply to your classification/recognition problem, what software you intend to use (e.g. Matlab, Mathematica, C/C++, etc.), and links to any references that you consulted in formulating your project idea.
2. ***Progress report***: a progress report on your project is due by ***midnight of November 25, 2003 (Tuesday)***. Methods of submission are discussed further below. By this time, you should have obtained data, have experimented with feature extraction methods, and have begun classification analysis of your data.
3. ***Final report***: a final report on your project is due by ***midnight of December 7, 2003 (Sunday)***. If incomplete or of insufficient quality, you may submit addenda to your final report through ***midnight of December 19, 2003 (Friday)***. Methods of submission are discussed further below. Upon submission of final reports on December 7, 2003, your project will be evaluated and ranked in terms of quality. ***If your project ranks in the top 10 (for the class), you will receive an A for the course***

All late submissions will be assessed a late penalty.

2. Reports

The progress report due on November 25, 2003 and the final report due on December 7, 2003 may be submitted in two different ways:

1. E-mail a link to a web page that contains your progress report/final report to nechyba@mil.ufl.edu. This method is preferred, since it allows you to link video and audio data (if applicable) to your project web page.
2. E-mail a pdf file of your progress/final report to nechyba@mil.ufl.edu. Be sure that the file size of your pdf file is less than 2Mb.

Your final report must include the following components:

1. A concise description of the problem (application).
2. A summary of previous solutions to the problem. Your report should have at least a few references to papers in the literature that you have read to help you implement your project.
3. A detailed description of your solution to the problem, including what software was used in your solution.
4. Classification/recognition results.
5. Discussion and analysis of your results and how your solution differs from previous work.
6. Ideas for how your project could be extended or improved if you had more time.
7. Appendices should include any detailed derivations or other information too detailed for the main body of the report. You should also include any code that you wrote to implement your project. Do not include code that was provided to you by the instructor or obtained through the web; simply mention that you used such code and reference it, if applicable.

3. Grading criteria

Your projects will be evaluated on the following criteria (in no particular order): (1) novelty and difficulty of your application, (2) complexity and number of algorithms implemented, (3) correctness of results, and (4) depths of discussion/analysis. Here are some examples of work that will tend to enhance your project grade:

1. Compare classification performance for multiple algorithms. For example, implement and compare performance of a Gaussian-based classifier, a histogram/VQ-based classifier and a mixture-of-Gaussian classifier.
2. Analyze classification performance of a single algorithm for different configurations. For example, (1) compare performance of Gaussian classifiers with scalar covariances *vs.* full covariance matrices; (2) compare performance of histogram classifiers with varying number of bins, uniform quantization and VQ-based quantization; (3) compare performance of an HMM-based classifier by varying the number of states, number of observables, etc.
3. Implement/investigate complex classifiers (e.g. neural networks, HMMs), including ones that we may not have covered in class, as part of your project.
4. Investigate classification performance for different feature extraction methods (e.g. color *vs.* texture, etc.).

Note that the above items are just a few examples of how you can enhance the complexity and depth of your project. In building a pattern classifier, the designer (i.e. you) is faced with a large number of design choices. To what extent you explore these choices will in large measure determine your final project grade (and, consequently, your final course grade).

One final note: above, one of the criteria listed is "correctness of results." This *does not* refer to the ultimate classification performance for your project. If you pick a difficult classification problem and do not obtain perfect classification results, that will not count against you, especially if your results are accompanied by a thoughtful discussion of what might have gone wrong. Instead, "correctness of results" means that you followed sound procedures in your implementation and testing of one or more classifiers. For example, you *must* separate your data into a training data set and a test (or cross-validation) data set, where the training data set is used to estimate the parameters of your classifier(s), and the test data is set is used to test the classification performance of your classifier(s).

4. Programming resources

While you are free to use your choice of programming environment to implement your project, some code for specific algorithms has been provided to you. In addition to Mathematica notebooks that have already been provided as part of the regular distribution of course materials, C source code is also available for the following algorithms: (1) vector quantization, (2) mixture-of-Gaussian (EM) modeling, (3) discrete-output HMMs and (4) multilayer neural networks. You are also free to use any source code for these and other algorithms that you find available on the web.

5. Data resources

One of the important factors in the success of your project is that you are able to obtain adequate data for your application. The web is a rich source of such data, and you are encouraged to look for suitable data on the web. If your project requires that you collect data yourself, be sure that you have a means to do so (e.g. speech/voice data). For outstanding project ideas requiring that you collect image/video data yourself, we can arrange to collect such data in our lab. Before you make such a request, however, please spend some time on the web to see if you can find already available data first. Below are just a few links to web sites with data archives suitable for this course:

1. ***UCI Machine Learning Repository Content Summary***: http://www.ics.uci.edu/~mlearn/MLSummary.html
2. ***Computer Vision Test Images***: http://www-2.cs.cmu.edu/~cil/v-images.html
3. ***Google directory***: http://directory.google.com/Top/Computers/Artificial_Intelligence/Machine_Learning/Datasets/