CSE 555 Spring 2009 Project

Jason J. Corso
Computer Science and Engineering
University at Buffalo SUNY
jcorso@cse.buffalo.edu
Date Due 23 April 2009

*This document contains the descriptions of 3 projects. You should choose **one** of them (or do some other approved project). It is a programming-based project with a written report. You are free to choose your language (Matlab, C/C++ or Java). You should program and write alone but should discuss with classmates (away from the terminal).*

For none of these projects may you use ANY off-the-shelf library or tool other than for basic IO and basic computation.

The project is worth 100 pts equally divided between (1) implementation, (2) report, and (3) accuracies.

The report should be max 4 pages in the two-column IEEE format (`http://www.ieee.org/web/publications/authors/transjnl/index.html`). Unless you do something out of the ordinary (which is encouraged as long as it works!), you do not need to describe the methods, just the findings.

An extra-credit option is associated with each project for an additional X points (the actual project here is considered to be 100 points). This will let you dive into some more details, but it is more challenging (i.e., do the normal project first).

Project 1: AdaBoost Face Detection

1. Data: `http://www.cse.buffalo.edu/~jcorso/t/CSE555/project1-data.tar.gz`.
2. Implement the Viola and Jones Cascaded-AdaBoost method for face detection. You need to implement the full scale-varying search for testing the image itself.
3. Train a single AdaBoost classifier with 100 weak-learners on the provided data. What are the first ten weak-learners selected (draw them graphically in your report)? Can you explain why these may have been selected? Start with just a single weak-learner in our model and plot your accuracy on the training and testing data. Iteratively increase the number of weak-learners until you get to the full 100, continue to plot your accuracy for each. Explain your findings.
4. Quantify your accuracy (% correct) on the test dataset for three varying lengths of the cascade. Describe your findings. Is this what you expected?
5. Compute an ROC curve when you vary the final threshold on the best-performing cascade. Describe your findings.
6. Download an image from the Google or Flickr in response to the query "crowd" and run your code on it. Describe your findings.
7. Download an image from the Google or Flickr in response to the query "forest" and run your code on it. Describe your findings.

   Option (+25pts) Instead of the Cascade of AdaBoost, implement the (two-class) Probabilistic Boosting Tree and repeat the experiments.

Project 2: Mushroom Classification: Decision Trees on Categorical Data

1. Download the "Mushroom" dataset from the UCI Machine Learning Repository: `http://archive.ics.uci.edu/ml/datasets/Mushroom`. Split the data into a training and testing set: take the odd-indexed data as training and the even-indexed data as testing.
2. Implement the CART method for decision trees, including multi-way splits and a pruning process.
3. What features are selected at the top of the tree (say the first 2 levels)? Plot histograms of these features (for the whole data and for each group) and compare them to histograms of some features that were not selected. Describe your findings.
4. Compare both the resulting trees and the quantitative accuracy when you use the various *impurity* measures during tree learning (entropy, variance, misclassification). Prepare precision and recall scores of the findings. Describe your findings.
5. Quantify your accuracy on the training and testing data as you increase the height of the tree from 4 to however many give you a perfect classification on the training data. Describe your findings.

   Option (+20pts) Implement the CART, ID3 and C4.5 algorithms, repeat the experiments, and compare the trees and results.

Project 3: Pen-Gesture Recognition with Hidden Markov Models

1. Data: `http://www.cse.buffalo.edu/~jcorso/t/CSE555/project3-data.tar.gz`. This data has 5 files, one for each vowel of the alphabet. These are xml files and the format is self-explanatory. Again, take the odd indexed entries as training and the even ones as testing.
2. Implement a spatial clustering algorithm (e.g., K-Means) on the 2D training data.
3. Implement the standard Hidden Markov Model. The input will be the cluster index associated with each 2D point.
4. Train a separate HMM for each vowel (do separate clustering for each and then learn the HMM independently). Quantify accuracy against the training and testing set; for each candidate datum, compute its log-likelihood against each HMM and take the label of the HMM that gives the highest log-likelihood. Compute a confusion matrix of results. Describe your findings.
5. Vary the number of clusters in step and the number of hidden nodes (and their connectivity) in step and repeat the quantification. Do this for an additional three variants. Describe your findings.

   Option (+15pts) Implement Dynamic Time Warping and repeat the experiments.

How and what to submit: The entire project, including the report, should be tar'd and submitted via the CSE submit script.

1. Login to pollux (the code MUST work on the CSE machines).
2. Use "tar -cvf project.tar list-of-files-or-directories".
3. Then, type "submit_cse555 project.tar".

Include a README file to explain what is there, but only submit the code. If it is C/C++ code, you need to include a makefile that will compile on the CSE machines (it is thus encouraged to link to any libraries available there rather than submitting them). If it is Java code, you need to include an ant file that will compile on the CSE machines. In all cases, you need to include a script named "go" that will (once compiled) do everything requested in the assignment (in the case of C or Java, it is sufficient to generate image files of the plots and save them to reasonably named files). The script can be bash/tcsh for C/Java and should be a simple Matlab script for Matlab.