Pattern Recognition, Spring 2015
Project 1: Handwritten Math Symbol Classification

Prof. Zanibbi

Due Date: 11:59pm Friday April 3, 2015

This project will be completed in teams of two students. For this first part of the project, you will create Python programs for classifying individual handwritten mathematical symbols.

Submission: One student from each team should submit the final write-up as a .pdf file, and the python code, README, parameter files and result files in a separate .zip file. Make sure that both student names appear in the write-up and source code files.

Teams should sign up to indicate their team members ASAP using this Doodle poll: `http://doodle.com/9nhbb5wh8v4n2pg5`.

Grading

Project 1 is graded out of 100 points (16.7% of final grade for the course).

40 points Project write-up

60 points System implementation and results

- 5 README explaining how to run your classifiers
- 10 Data split (2 folds), LgEval metric (`00_Summary`) and .html confusion files (produced by the LgEval `confHist` program)
- 20 Preprocessing and features
- 5 Nearest Neighbor classifier (1-NN control)
- 20 Your classifier (including files containing learned classifier parameters) - **cannot be a k-NN classifier**

Bonus Points: There are two opportunities for bonus points in the project.

1. +5 / +3 / +2 - for top 3 teams with the highest recognition rate
2. +5 / +3 / +2 - for top 3 teams with the highest recognition rate for the *control* classifier (i.e. the best features for the 1-NN classifier)

Rankings will be determined by running your classifier using a separate test set. **Your classifier may be run using the parameters learned from the complete training set, so you need to include this in your .zip file.**

Data Set and Tools

For the first project, you will construct classifiers for the CROHME 2014 competition training data set. The data set is available through MyCourses, along with a paper describing the competition, and the evaluation tools needed for the project (in particular, LgEval).

- Online visualizer for CROHME InkML files `http://saskatoon.cs.rit.edu/inkml_viewer/` (also available through CROHMELib).
- Notes on CROHME .inkml file format (NOTE: see LgEval README for updated .lg eval format) `http://www.cs.rit.edu/~dprl/CROHMELib_LgEval_Doc.html`

You will need to construct/re-use your 1-NN classifier, and a second classifier that **uses the same features as your 1-NN classifier.** You are welcome to write code for your own classifier or use/modify code from open source python libraries such as:

- scikit-learn: `http://scikit-learn.org/stable/` (various methods)
- orange: `http://orange.biolab.si/doc//reference/C45Learner.htm` (Quinlan's C4.5)
- libsvm: `http://www.csie.ntu.edu.tw/~cjlin/libsvm` (widely used SVM library)
- pybrain: `http://pybrain.org` (neural networks, deep learning, etc.)

You must acknowledge the source of any code in your source code and write-up.

Implementation and Results

Include any library files and parameter files needed to run your classifiers in your .zip file, and check that your classifier will run using just the contents of the .zip file before submission. Your code must contain a README explaining how to run all of your classifiers using provided parameter files.

You will implement two classifiers in Python, as two separate programs:

1. The control algorithm. This will be a 1-NN classifier - you may modify your implementation from Assignment 1 for this.
2. Your chosen classifier, capable of reading learned parameters from a file (e.g. saved using the python `pickle` or related modules (*use binary encodings to avoid massive files!!)).

Each classifier should take a CROHME .inkml file as input (format: `http://www.cs.rit.edu/~dprl/CROHMELib_LgEval_Doc.html`), and produce a label graph (.lg) file as output (see README for the LgEval library - **use the Object-Relationship format**). Only labels for objects (symbol stroke groups) need to be defined in the .lg file.

Reading .ink files: to read stroke and symbol data (e.g. the segmentation of strokes into symbols), use the built-in **ElementTree XML library**:

`http://docs.python.org/2/library/xml.etree.elementtree.html`

to easily obtain stroke coordinate lists and segmentation information.

Data Split: You should randomly partition files into two folds, such that each fold has roughly the same percentage of symbols from each class as in the original data set (i.e. try to maintain the prior probabilities within each fold). **The training set should contain roughly 2/3 of the files, and the test set the remaining 1/3. Note:** some classes may only have 1 or 2 instances - do not worry about representing them properly.

Results: Your 1-NN and other classifier should be trained on the training fold, and then evaluated on both the training and test folds (i.e. you will have 4 sets of results in your write-up). You should record all recognition rates and confusion matrices from evaluation runs: provide the `00_Summary` file produced by the LgEval `evaluate` program, and the .html output from the `confHist` program using graph size '1.' See the LgEval README for descriptions of these and other tools in the library, and/or try running these programs without arguments.

For the bonus, train your classifier using *all* of the training data, and save the necessary parameter files.

In the write-up, report on the recognition rates obtained by LgEval, and discuss the confusions made by each classifier in the `confHist` .html file. **You should also comment on the variation in performance between training and testing data sets.**

Write-Up

Your write-up should be at most six pages long, and contain the following sections:

1. **Design:** Indicate which properties of the data and other information or assumptions informed your choice of pre-processing, features and (non-control) classifier type. You do not need to define the classification task.
2. **Preprocessing:** summarize computations performed before measuring features (e.g. normalization, whitening, scaling, resampling stroke points, etc.).
3. **Features:** a clear description of the features used by your two classifiers. For features more complex than counting, provide your own mathematical definition for the feature (do *not* copy and paste definitions directly from .pdf files or other sources). This section should also describe any dimensionality reduction (e.g. PCA or LDA) applied to features.
4. **Classifier:** Describe the classification algorithm that you chose for the project. *You do not need to describe 1-NN.* Make sure that key parameters of the classifier for training and execution are discussed (including their values) along with any modifications that you made.

5. **Results:** Present the recognition results for the 1-NN classifier and your own algorithm, as described above. Note that the confusion matrices should be summarized (e.g. compare the most frequent confusions made by the two classifiers), as there are over 100 classes in the CROHME data set. **Make use of figures (e.g. plots, bar graphs) to compare metrics between classifiers.**
6. **Discussion:** provide possible explanations for patterns of success and error observed in your results, and suggest improvements for your features and classifier.
7. **References:** provide references that informed the design of your features, classifier or analysis. These should be cited in the document in an appropriate reference format (e.g. ACM or IEEE).