Homework 5 - Expectation Maximization

Machine Learning - CSCI 780 - Spring 2010 - Prof. Rosenberg

Due May 10 at 11:59pm

Problem 1) (100 pts.) Implement Expectation Maximization with Gaussian Mixture Models.

You may use python, java or c++ to perform this. Note: The msvc implementation of "c++" does not conform to any standard that would suggest that it is c++.

The program should take 2 parameters.

- A training file – comma separated values posted to the course webpage
- A m value – the number of clusters.

Your code should run using the following command line format

```
./em clustering.csv <m>
java -cp <CLASSPATH> EM clustering.csv <m>
./em.py clustering.csv <m>
```

Your code should output the final likelihood of the data given the converged GMM parameters for a given value of d. If you like, your code may also describe the locations (means) of the clusters, as well as the covariance matrices so you can plot these in your write up.

Deliverables

- All source code and libraries required for your project.
- A README/Report file whose contents are described below.

Expectation Maximization Experiments

- Evaluate the model likelihood, AIC and BIC using at least 20 values of m – the number of clusters.
- Evaluate the model likelihood, AIC and BIC re-using at least 10 values of m.

- Discuss the results of this experiment – What is the best value of m? Do likelihood, AIC and BIC agree? How quickly does the training converge? Does this change as a function of m? If so, how? Do you see any effect of the random initialization on repeated runs using the same value of m?

The README can be in any electronic format you like and should minimally include the following

- A list and description of every file included in the submission – including which matrix library (if any) you use.
- A description of how your code is compiled.
- A high level description of the task – here: Expectation Maximization on Gaussian Mixture Models. This should be about a paragraph long, and doesn't necessarily need a lot of mathematical notation. It should be understandable to a reasonably informed and proficient coder.
- A report of the experiments – as described above. Graphs may be particularly helpful for this assignment
- Any other points of interest – running time, complexity, unique qualities of your implementation, etc.

Grading:

- **10 points - Compilation** Each file must compile without error or warning into an executable as described above. (Note: for python, no points are awarded for compilation, but execution is worth 20 points.)
- **10 points - Execution** Each executable must run without error or warning on valid input using the command line parameters described above. (Note: for python, no points are awarded for compilation, but execution is worth 30 points.)
- **10 points - README** Does your README documentation completely and accurately describe the linear regression algorithm, as well as how to compile and run your project.
- **15 points - Report** Within your README or external Report have you reported the model likelihood using at least the appropriate number of settings of m, the number of clusters? Have you thoroughly discussed the experiments?
- **7 points - Within Code Documentation** Is the code documented for obvious understanding of the use, preconditions, and postconditions of each function?

- **8 points - Style** Are functions and variables given self-explanatory names? Are functions used to aid intelligibility of the code? Are functions used to reduce repeated blocks of code? Is indentation, spacing, use of parentheses, use of braces consistent, and sensible? For example, if you use brackets on the same line at the start of a block, always do so. If you place a brace on the line following the start of the brace, always do so. If you put a space between variables and operators, e.g. `if (i == j)`, always do so. So, `if (i == j) i = j+k;` is bad. It should be `if (i == j) i = j + k;` or if you prefer `if (i==j) i=j+k;`. You will be graded on consistency in these decisions, not on any particular style.

- **15 points - Correctness** Is the algorithm implemented correctly?