685.621 Algorithms for Data Science

SANDESH IYER

Homework 4

Assigned at the start of Module 7

Due at the end of Module 9

* 1. For the first problem, I ended up using the built in scikit learn Gaussian Mixture model. This model utilizes the expectation maximization algorithm.
  2. The algorithm is an iterative algorithm and start at an initial estimate theta and then iteratively updates theta value until convergence is detected.
  3. Each iteration essentially consists of the E-step and the M-step.
  4. The algorithm is started by initializing it with a set of initial parameters and then conducting E-step or by starting with a set of initial weights and then doing an M-step.
  5. The initial parameters or weights can be chosen via a heuristic method such as a k-means algorithm to cluster the data at first and then define weights based on these k-means memberships.
  6. For this algorithm we can detect convergence by computing the value of the log-likelihood of each iteration and stopping when it doesn’t appear to be changing in a significant manner from an iteration to the next.
  7. In the code, the model is fit to run for 50 iterations and prints out the iteration number, mean, covariance, weights for iterations < 6.
  8. The code is currently configured to run for 3 clusters but can be changed by changing n\_components parameter to handle ‘n’ cases.
  9. The program also prints the maximum amount of iterations the gaussian mixture model took to find convergence in the dataset. This convergence number can be lesser than the maximum number of iterations 50 or even greater depending on the dataset.
  10. For this dataset, the model converged in second iteration and continues to show convergence until the 5th iteration based on the printed iterations that is. See problem1.log file for test run results.
  11. The input dataset is extremely small and is very easy for the gaussian mixture model to identify maximum likelihood of the parameters.

1. See problem2.log for parameter values.
2. To generate observations,
   1. Used numpy multivariate normal command to generate 300 observations for x,y and z based on mean and covariance provided.
   2. The x,y,z arrays where then transformed to be within the min and max values given in question using MinMaxScaler command from sklearn preprocessing library
   3. Following this, similar method implemented for problem 1 and 2 was implemented for this and Max iterations, mean, covariance and weights were derived. See problem3.log file for results.

Approach for 3 class classifier

* 1. This problem is straightforward if thought about for a while. The combination of an EM algorithm and the bayes classifier can be seen in the implementation of a gaussian naïve bayes classifer.
  2. The naïve bayes classifiers are essentially a collection of classification algorithms based on bayes theorem. It’s a family of algorithms where every pair of features being classified is independent of each other.
  3. Bayes theorem provides a way to calculate probability of a hypothesis given prior knowledge
  4. Bayes theorem: P(h|d) = (P(d|h) \* P(h)) / P(d)
  5. The classical naïve bayes model can be extended to the real-values attributes mostly by assuming a gaussian distribution. This is called Gaussian Naïve Bayes.
  6. This method is easy as we only need to estimate the mean and standard deviation from training data which is where the EM algorithm plays a big part.
  7. To summarize the distribution, the mean and standard deviation are calculated from EM and this can be stored in addition to the probabilities of each class for each input variable for each class.
  8. The probabilities for the new x values are calculated using the gaussian probability density function (PDF).
  9. During prediction, these parameters can be plugged into the Gaussian PDF with a new input for the variable and the Gaussian PDF would provide an estimate of probability of the new input value for that class. See below formula.
  10. pdf(x, mean, sd) = (1 / (sqrt(2 \* PI) \* sd)) \* exp(-((x-mean^2)/(2\*sd^2)))
  11. Then we can plug the probabilities into the above equation to make predictions with real valued inputs.

Code and analysis

1. The code does exactly whats described above with the scikit learn libraries to fit GaussianNB model on the iris dataset. The classification graph is saved as problem4.jpg. Please take a look.
2. For this problem, the three class classifier is based on the sepal width and the sepal length features of the flowers.
3. From the classification image we can see that the yellow dots represent the versicolor species, the green dots represent the virginica species and the blue dots represent the setosa species.
4. The boundary (classification) is drawn to show how good/bad the classification of the three classes are.
5. We notice that there are three distinct regions for each class (majority of points in one area/region for each class). The setosa classification is very good given that almost no blue dots exist in other regions.
6. However, we notice that the green and orange dots overlap with some being in the region of the other while still at a minimum extent. The classifier still did a good job classifying the classes but there are still a few outliers for these classes which have potential to be wrongly classified.