

# Classify the glass samples using Gaussian Mixture Models (GMM)

Ville Sillanpää, k84338 - Lauri Viitanen, 338853  
*ville.sillanpaa@aalto.fi - lauri.viitanen@aalto.fi*

April 13, 2015

# 1 Introduction

In this project we analyze a glass data set using both unsupervised and supervised machine learning approaches. The glass dataset [1] has been provided by the US Forensic Science Service and is available for download from the UCI machine learning repository. The dataset consists of samples for 6 types of glass (class label 4 is missing in the current data set); defined in terms of their oxide content (i.e. Na, Fe, K, etc). In particular, the dataset contains 214 samples characterized by 9 features and the samples have been categorized into 6 different classes of glass.

The goal of the project is to classify the glass samples using Gaussian Mixture Models (GMM) in unsupervised and supervised approaches. In the unsupervised approach we use GMMs to find clusters from the full data set and compare the clustering results with the ground-truth class labels that are provided with the data set. In the supervised approach we fit a separate GMM for each class label and evaluate the classification performance using the fitted models on the test set. The dataset has been randomly divided into training (75 %) and test sets (25 %) keeping an equal proportion of different class instances in the training and test sets. In addition, we compare the performance of the GMM model with the k-nearest neighbors classifier.

## 2 Methodology

This section describes the methods used in the analysis. Unsupervised and supervised approaches are described in their own subsections.

Unsupervised classification was done using Gaussian Mixture Model with Dirichlet Process prior (sometimes referred to as Infinite Gaussian Mixture model). Supervised classification was done using Gaussian Mixture Model (GMM) with no prior distributions for model parameters.

### 2.1 Unsupervised Approach

Dirichlet Process Gaussian Mixture Model (DP-GMM) was selected for unsupervised classification, as it does not require user to specify the number of clusters beforehand. Instead, the model allows structure of data define how many clusters the model ends up with. This is a major asset when compared to traditional GMM, as it is often difficult to assess how many clusters an 'optimal' model has. This property is especially of use in this analysis, where we don't really know anything about the data beforehand.

The specification of the model is as follows. Remainder of this subsection is brief adaptation from Wood et al. (2006) [2]. For more elaborate description of model with appropriate referencing to previous work leading into this model specification, see [2].

Our model assumes that our dataset comes from the following generative model:

$$\begin{aligned} c_i | \pi &\sim \text{Multinomial}(\cdot | \pi) \\ y_i | c_i = k, \theta &\sim N(\cdot | \theta_k) \end{aligned} \quad (1)$$

Where  $C = \{c\}_{i=1}^N$  indicate which class each observation belongs to,  $\theta = \{\theta_k\}_{k=1}^K = \{\mu, \Sigma_k\}$  are parameters that characterize each class distribution,  $Y = [y_1, \dots, y_N]$  are observations and  $\pi = \{\pi_k\}_{k=1}^K = P(c_i = k)$  are class membership distribution parameters.

We model this data generation process with the following model (assuming  $K$ , the number of classes, is finite):

$$\begin{aligned} \pi | \alpha &\sim \text{Dirichlet}(\cdot | \frac{\alpha}{K}, \dots, \frac{\alpha}{K}) \\ \theta &\sim G_0 \end{aligned} \quad (2)$$

Where  $\theta \sim G_0$  is shorthand for

$$\begin{aligned} \Sigma_k &\sim \text{Inverse-Wishart}_{v_0}(\Lambda_0^{-1}) \\ \mu &\sim N(\mu_0, \Sigma_k / \kappa_0) \end{aligned} \quad (3)$$

The parameters of prior distributions Normal, Inverse-Wishart and (symmetric) Dirichlet encode our beliefs about parameters of the system. Under this model posterior distribution is

$$\begin{aligned} &P(C, \theta, \pi, \alpha | Y) \\ &\propto P(Y | C, \theta, \pi) P(\theta | G_0) \prod_{i=1}^N P(c_i | \pi) P(\pi | \alpha) P(\alpha) \end{aligned} \quad (4)$$

The idea is to sample from this distribution assuming that  $K \rightarrow \infty$ , which means that there are potentially infinite number of classes and that our finite data manifests a finite but unknown amount of them. The sampling is done using Gibbs sampling, which is possible because our prior distributions are all conjugate. Below are the conditional distributions that we sample from sequentially. See Wood (2006) for more detailed derivations and references:

$$\begin{aligned} &P(\theta | C, \pi, \alpha, \theta_{-k}, Y) \\ &\propto \prod_{c_i=k} P(y_i | c_i, \theta_k) P_{G_0}(\theta_k) \end{aligned} \quad (5)$$

$$\begin{aligned} &P(c_i = k | C_{-i}, \pi, \alpha, \theta, Y) \\ &\propto P(y_i | c_{-i}, \theta) P(c_i | C_{-i}) \end{aligned} \quad (6)$$

$$P(c_i|C_{-i}) = \begin{cases} \frac{m_k}{i-1+\alpha}, & \text{if } k \leq K_+ \\ \frac{m_k}{i-1+\alpha}, & \text{if } k \geq K_+ \end{cases} \quad (7)$$

Where  $m_k$  is the number of observations in class  $k$ .

Sample from this model is cluster membership status to all data points and distribution parameters to each existing cluster. The idea in the sampler is to go through all data points in turn, take their class status off and then sample a new class status to them from 6 with 7 allowing that a new cluster can be generated. If new cluster was generated, sample parameters to it from  $G_0$ . After this sample  $\theta$  using 5. This is done until log-likelihood of the model no longer increases substantially (which means that our samples are most likely from posterior and not from some intermediary state of the Markov Chain).

The sampler we used was written by Wood (2006) and is available online [3]. This sampler also treats  $\alpha$  as a random variable, and samples it for 7 from Gamma-distribution using Metropolis sampling.

We used prior parameters Wood’s algorithm had by default. So our prior distributions are  $N(0, \Sigma_k/1, )$ , *Inverse – Wishart*<sub>1</sub>(*diag*(0.3)), where *diag*(0.3) is a diagonal matrix with 0.3 on diagonal.

We ran into problems in initial testing runs, as the algorithm did not seem to find more than 1 cluster for the entire data. As this seemed wrong, we centered each variable to 0 and scaled their variance to 1. While we don’t know the root cause for the problems, we guess that it was due to the choice of priors (and some of our variables being quite far away from them). We chose this approach rather than rethinking the priors because it was more convenient to do.

## 2.2 Supervised Approach

In the supervised approach, the Gaussian mixture model is fitted to every class label separately. The model complexities range from 2 to 10 components. Class label 4 has no samples in the training data, class label 6 has only six samples and class label 5 has nine samples, meaning that using more than nine classes already drops the number of class labels from seven to four. Only classes 1 and 2 have more than 21 samples.

The GMM is fitted to the training data using the Expectation-Maximization (EM) algorithm. The training halts after 200 iterations or if the log likelihood has not changed enough (0.001 units) during the past few iterations (five). The resulting fitted Gaussians are then applied to the test data of the same classes, producing the log likelihood of the test data being produced by the fitted GMM.

VISUALIZATION HERE

THEN TALK ABOUT THE MODEL COMPONENT COUNT DISTRIBUTION

VISUALIZATION HERE

## 3 Results

This section describes the results. Results from unsupervised and supervised analysis are presented in their own subsections.

As a summary of results from unsupervised approach, we say that the model was able to identify only one cluster well. With remaining clusters the model had more mixed results. Details that lead to this conclusion are presented in the next paragraph.

### 3.1 Unsupervised Approach

A Gibbs sampling Markov chain of 5000 observations was ran. Below is the distribution of number of clusters estimated from the last 2500 samples of the chain.

As we can see, typically our model usually found 5 clusters with 6 and 2 being quite likely outcomes as well. The ground truth was that there were 7 clusters. So it seems that the model was unable to characterize this data in a manner that would reflect this.

Reasons for this become evident when we visualize the results. Below is a scatter plot from one of the runs. The visualization was produced with multidimensional scaling. PCA was tried first, but since the first two components explained only two thirds of variation, we chose to to MDS instead.

As you can see from ??, two dense clouds of points have been identified to belong to three clusters. This result was very common in runs that found more than two clusters. The runs that found only two clusters saw the two dense clouds as two separate clusters.

The points that are spread more sparsely were assigned to various clusters in various runs. Based on inspection of few samples the labeling of these data points was very arbitrary. Hence based on this model it is difficult to say what clusters the sparse points belong to or that how many true labels these sparse points represent.

To see how our model compares with ground truth, let's look at MDS-visualization with points given color codes based on true labels.

As we can see from ??, the lower dense region is more overlapping than what our model usually predicts. Moreover, quite a few of the more sparse points seem to belong into the label who is very densely concentrated in the bottom (yellow). Apparently the sparse region contains observations from two additional labels. They seem to overlap so much that it is difficult to see how a Gaussian mixture model would be able to find these two clusters correctly. Model and ground truth seem to agree the most about the upper dense cloud (label 7 in real data).

## 4 Discussion

[1] <http://archive.ics.uci.edu/ml/datasets/Glass+Identification>

[2] F. Wood, S.Goldwater & M.Black A Non-Parametric Bayesian Approach to Spike

Sorting, Department of Computer Science & Department of Cognitive and Linguistic Sciences, Brown University, Providence, RI, USA, 2006

[3] <http://www.gatsby.ucl.ac.uk/fwood/code.html>

## **Appendix A**

Matlab code for asdf.