Expectation Maximization of Gausian Mixture Models in VTK

Release 0.00

David Doria

September 21, 2010

Rensselaer Polytechnic Institute, Troy NY

Abstract

Expectation maximization (EM) is a common technique for estimating the parameters of a model after having collected observations of data generated by the model. We first explain the algorithm, then present our impelementation. We focus on estimation of the parameters of a Gaussian Mixture Model (GMM). The implementation is written in the VTK framework and is provided as a new class, vtkExpectationMaximization.

The code is hosted here: http://github.com/daviddoria/ExpectationMaximization for the time being.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3218]

Contents

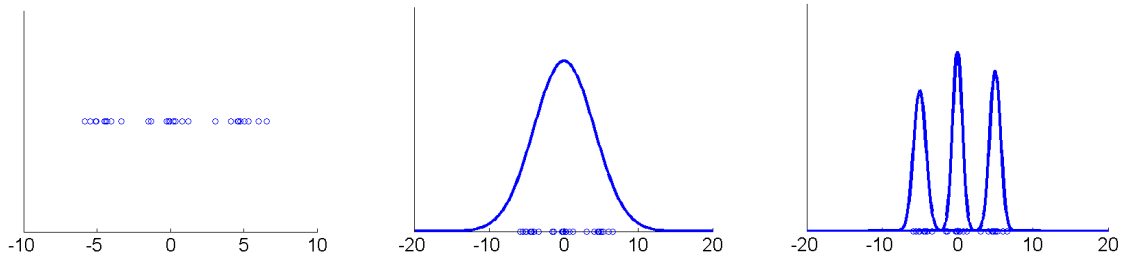
1 Introduction

We are often interested in observing data and then estimating the distribution(s) from which this data came. Many phenomenon cannot be modelel well by a single distribution. Rather than allow a free form analytic distribution, we often make the assumption that the observed data has come from a combination, or mixture, of multiple "common" distributions. A common example is a Gaussian Mixture Model (GMM), where we assume the observed data has come from a weighted sum of Gaussian distributions.

We start by introducing mixture models and maximum likelihood estimation (MLE) before getting to our explanation of EM and our impelementation.

2 Mixture Models

Imagine you have observed the one dimensional data shown in Figure 1(a). You may think it came from a single distribution as shown in Figure 1(b), or from multiple distributions as shown in Figure 1(c).



(a) Observed Data (b) Single Distribution Model (c) Multiple Distribution Model

Figure 1: 1D data and proposed distributions

There is often no "correct" answer, you must simply choose the model that appears to fit the data the best (in a qualitative sense, for now).

A mixture model is simply a fancy word for a weighed sum of distributions. The number of distributions in the mixture must be known, we will call it N. The weights are often refered to as the "mixing coefficients".

A generic way to write a mixture model is:

$$p(x) = p(x) = \sum_{k=1}^{N} \pi_k p_k(x) = \pi_1 p_1(x) + \pi_2 p_2(x) + ... + \pi_k p_k(x) \tag{1}$$

where the p_k's are the individual distributions that compose the mixture model and the π_i's are the mixing coefficients. The mixing coefficients must all be between 0 and 1, and the sum of the coefficients must be equal to 1. Formally,

$$0 \leq \pi_k \leq 1 \forall k \tag{2}$$

and

$$\sum_{k=1}^{N} \pi_k = 1 \tag{3}$$

2.1 Gaussian Mixture Models (GMM)

A GMM is simply a mixture distribution where all of the components are Gaussian distributions. The mixture distribution is still written as

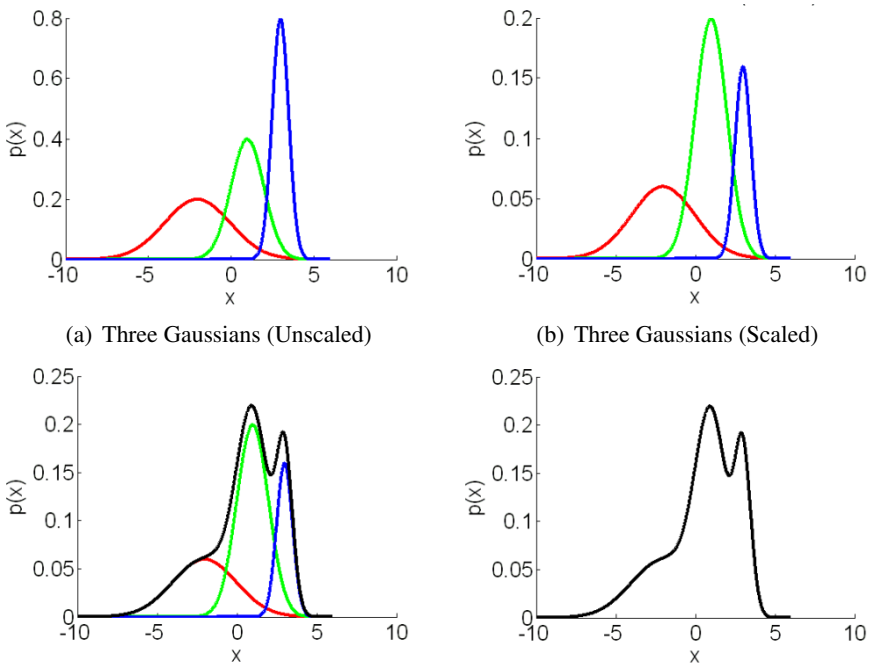
$$p(x) = \sum_{k=1}^{K} \pi_k p_k(x)$$

but in the case of GMMs all of the p_k's are of the form

$$p_k(x) = \frac{1}{\sigma_k \sqrt{2\pi}} \exp\left(-\frac{(x-u_k)^2}{2\sigma^2}\right)$$

An example of a GMM is shown in Figure 2. The parameters are:

- $u_1 = -2, \sigma_1 = 2, \pi_1 = .3$
- $u_2 = 1, \sigma_2 = 1, \pi_2 = .5$
- $u_3 = 3, \sigma_3 = .5, \pi_3 = .2$



(a) Three Gaussians (Unscaled)

(b) Three Gaussians (Scaled)

(c) Mixture of Gaussians

(d) Composite distribution

Figure 2: Gaussian Mixture Model

3 Maximum Likelihood Estimation (MLE)

Note: MLE is an extremely important topic that many whole books have been written about. This is the short version.

MLE is a form of parametric density estimation. The goal is to find the maximum of the joint distribution of all of the observations over the parameter space

$$MLE = \arg\max_{\theta} p(x_1, x_2, ..., x_n|\theta) = \arg\max_{\theta} \prod_{i=1}^{N} p(x_i|\theta) \tag{4}$$

where θ is a vector of parameters of the distribution and N is the number of observations.

To simplify the mathematics, the log of the function is often taken. The log function is monotonic, so the maximum of the original function is equal to the maximum of the log of the function.

Since the log of a product is a sum, we can write:

$$MLE = \arg\max_{\theta} \sum_{i=1}^{N} log(p(x_i)) \tag{5}$$

In the general case (an arbitrary distribution), this optimization must be performed numerically.

3.1 MLE Simplification for Gaussians

When the distribution you would like to find the parameters of is a Gaussian, an analytic solution is possible. When we take the log of the distribution, we obtain a very simple result. This is because the Gaussian function is an exponential function, and the log function and exponential functions are inverses.

Writing the Gaussian distribution as:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{6}$$

we are trying to solve

$$\begin{aligned} MLE &= \arg\max_{\theta} \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right) \\ &= \arg\max_{\theta} \sum_{i=1}^{N} log\left[\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)\right] \\ &= \arg\max_{\theta} \sum_{i=1}^{N} \left[log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) - \frac{(x_i-\mu)^2}{2\sigma^2}\right] \end{aligned}$$

In this case, our parameter vector θ is

$$\theta = \begin{pmatrix} \mu \\ \sigma \end{pmatrix} \tag{7}$$

Taking the derivative of the likelihood function with respect to μ and maximizing (ie. setting the result of the differentiation to 0 and solving for μ), we find the MLE of the mean is the average of the values of the observations:

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^{N} x_n \tag{8}$$

Differentiating with respect to σ^2, we find the MLE of the variance to be

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^{N} (x_n - \hat{\mu})^2$$

4 Expectation Maximization (EM)

The name Expectation Maximization is very confusing. It is not a method for finding the expected value of a maximization function or for maximizing an expectation function as the name may imply. Rather, it is a two step process for finding MLE solutions. The steps are refered to as "The E Step" (expectation) and "The M Step" (maximization).

4.1 Initialization

We must start with a guess of the distributions. Often, nothing is known so we simply generate random values for all of the parameters. In the case of GMMs, we generate random values for the means and

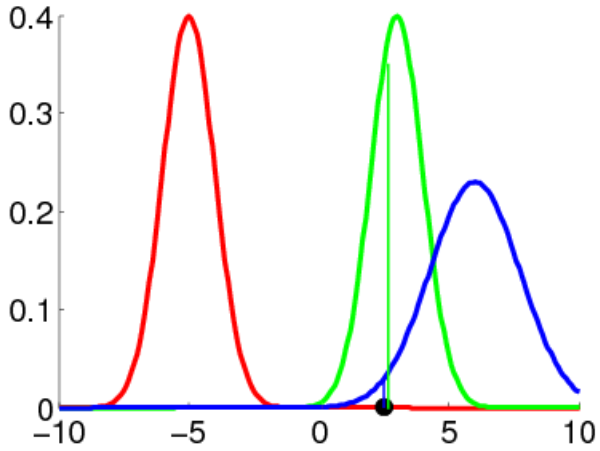
variances of the distributions. We must also guess the mixture coefficients. If nothing is known, we set them all to $\frac{1}{N}$ where N is the number of distributions in the model.

4.2 Expectation - “The E Step”

In this step we compute how “responsible” each distribution is for producing each observed point. The responsibility that the k^{th} distribution takes for explaining an observation x is the value of the weighted k^{th} distribution function evaluated at x divided by the sum of all of the weighted distributions evaluated at x

$$\gamma_k = \frac{\pi_k p_k(x)}{\sum_{j=1}^{K} \pi_j p_j(x)}$$

An example is shown in Figure 4.2.



Here, the responsibility of the red distriubtion is very low (almost zero), the responsibility of the blue distribution is very small, and the responsibility of the green distribution is very large.

4.3 Maximization - “The M Step”

In this step, we estimate the distribution parameters using the responsibilities computed in the E step to maximize the likelihood of the data. In the case of GMMs, we can simply use the analytic estimators described in Section 3.1.

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} x_n \tag{9}$$

$$\hat{\sigma}_k^2 = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} (x_n - \hat{\mu}_k)^2 \tag{10}$$

Here, N_k is the effective number of points assigned to the k^{th} distribution. That is,

$$N_k = \sum_{n=1}^{N} \gamma_{nk} \tag{11}$$

γ_{nk} is the responsibility taken by the k^{th} distribution for the n^{th} observation.

At the end of this step, the new mixing coefficients are computed to be

$$\pi_k = \frac{N_k}{N} \tag{12}$$

5 Implementation

5.1 Filter style

We have chosen to place the code in a filter with zero inputs and zero outputs. This doesn't allow the user to do anything with the output of the filter in the VTK sense, but it provides an interface that VTK users are accustomed to in the way that parameters of the algorithm are set, etc.

5.2 Implementation Details

Model Class

We have created a Model class that stores the parameters of a model as well as its mixing coefficient.

- Evaluate(double) - this function computes the value of the model at a particular point. That is, modelk.Evaluate(x) is the equivalent of $p_k(x)$
- WeightedEvaluate(double) - this function computes the value of the model at a particular point weighted by its mixing coefficient.That is, modelk.Evaluate(x) is the equivalent of $\pi_k p_k(x)$

5.3 Data storage

We store the distributions of the mixture model in

```
std::vector<Model> Models;
```

We store the observations in

```
std::vector<double> Data;
```

We keep track of the responsibilities in an $Data.size()$ x $Models.size()$ matrix:

```
vtkSmartPointer<vtkDenseArray<double> > Responsibilities;
```

6 Parameters

The parameters that can be set are:

- MaxIterations - the maximum number of times to iterate the EM algorith
- MinChange - If the difference between every paramter of every distribution from one iteration to the next is less than this value, we quit early and claim success.

7 Code Snippets

8 Model Initialization

The user must create an initial guess at the model. As discussed in Section 4.1, if nothing is known ahead of time, random values can be used for the model paramters and equal values used for the mixture coefficients:

```
// Initialize the model
std::vector<Model> models(3);

for(int i = 0; i < models.size(); i++)
  {
  Model model;
  model.SetMean(vtkMath::Random(-5, 5));
  model.SetVariance(vtkMath::Random(0, 3));
  model.SetMixingCoefficient(1./models.size());
  models[i] = model;
  }
```

9 Data preparation

The user must then get their observations into a vector of doubles. We suggest creating a function so that the following can be done:

```
std::vector<double> data = GetUserData(...);
```

10 Performing the EM

```
// Perform EM
vtkSmartPointer<vtkExpectationMaximization> expectationMaximization =
  vtkSmartPointer<vtkExpectationMaximization>::New();
expectationMaximization->SetData(data);
expectationMaximization->SetModels(models);
expectationMaximization->SetMinChange(1e-2);
expectationMaximization->Update();
```

11 Output and visualization

For convenience we have provided a function which outputs the properties of all of the distributions in the mixture model to the command line.

```
std::cout << "Initial models:" << std::endl;
OutputModelInfo(models);
```

```
std::cout << "Final models:" << std::endl;
OutputModelInfo(expectationMaximization->GetModels());
```

We have also provided a function to visualize the components of the model.

```
double domain=[-5, 5]; // the domain over which to plot the distributions
PlotModels(expectationMaximization->GetModels(), domain);
```