gmm_diag and gmm_full: C++ classes for multi-threaded Gaussian mixture models and Expectation-Maximisation

Conrad Sanderson[1, 2, 4] and Ryan Curtin[3, 4]

[1]Data61, CSIRO, Australia
[2]University of Queensland, Australia
[3]Symantec Corporation, USA
[4]Arroyo Consortium

12 October 2017

Paper DOI: http://dx.doi.org/10.21105/joss.00365
Software Repository: https://github.com/conradsnicta/armadillo-gmm
Software Archive: http://dx.doi.org/10.5281/zenodo.1012627

Summary

Statistical modelling of multivariate data through a convex mixture of Gaussians, also known as a Gaussian mixture model (GMM), has many applications in fields such as signal processing, econometrics, and pattern recognition (Bishop 2006). Each component (Gaussian) in a GMM is parameterised with a weight, mean vector (centroid), and covariance matrix.

gmm_diag and *gmm_full* are C++ classes which provide multi-threaded (parallelised) implementations of GMMs and the associated Expectation-Maximisation (EM) algorithm for learning the underlying parameters from training data (Moon 1996)(McLachlan and Krishnan 2008). The *gmm_diag* class is specifically tailored for diagonal covariance matrices (all entries outside the main diagonal in each covariance matrix are assumed to be zero), while the *gmm_full* class is tailored for full covariance matrices. The *gmm_diag* class is typically much faster to train and use than the *gmm_full* class, at the potential cost of some reduction in modelling accuracy.

The interface for the *gmm_diag* and *gmm_full* classes allows the user easy and flexible access to the trained model, as well as control over the underlying parameters. Specifically, the two classes contain functions for likelihood evaluation, vector quantisation, histogram generation, data synthesis, and parameter modification, in addition to training (learning) the GMM parameters via the EM algorithm. The classes use several techniques to improve numerical stability and promote convergence of EM based training, such as keeping as much as possible of the internal computations in the log domain, and ensuring the covariance matrices stay positive-definite.

To achieve multi-threading, the EM training algorithm has been reformulated into a MapReduce-like framework (Dean and Ghemawat 2008) and implemented with the aid of OpenMP pragma directives (Chapman, Jost, and Pas 2007). As such, the EM algorithm runs much quicker on multi-core machines when OpenMP is enabled during compilation (for example, using the `-fopenmp` option in GCC and clang compilers).

The *gmm_diag* and *gmm_full* classes are included in version 7.960 of the Armadillo C++ library (Sanderson and Curtin 2016), with the underlying mathematical details described in (Sanderson and Curtin 2017). The

documentation for the classes is available online[1].

References

Bishop, Christopher. 2006. *Pattern Recognition and Machine Learning*. Springer.

Chapman, Barbara, Gabriele Jost, and Ruud van der Pas. 2007. *Using OpenMP: Portable Shared Memory Parallel Programming*. MIT Press.

Dean, Jeffrey, and Sanjay Ghemawat. 2008. "MapReduce: Simplified Data Processing on Large Clusters." *Communications of the ACM* 51 (1): 107–13. doi:10.1145/1327452.1327492.

McLachlan, Geoffrey, and Thriyambakam Krishnan. 2008. *The EM Algorithm and Extensions*. 2nd ed. John Wiley & Sons.

Moon, Todd K. 1996. "Expectation-Maximization Algorithm." *IEEE Signal Processing Magazine* 13 (6): 47–60. doi:10.1109/79.543975.

Sanderson, Conrad, and Ryan Curtin. 2016. "Armadillo: A Template-Based C++ Library for Linear Algebra." *Journal of Open Source Software* 1: 26. doi:10.21105/joss.00026.

———. 2017. "An Open Source C++ Implementation of Multi-Threaded Gaussian Mixture Models, K-Means and Expectation Maximisation." *arXiv:1707.09094*.

[1] http://arma.sourceforge.net/docs.html#gmm_diag