Scalable Extreme Deconvolution

James A. Ritchie
School of Informatics
University of Edinburgh
james.ritchie@ed.ac.uk

Iain Murray
School of Informatics
University of Edinburgh
i.murray@ed.ac.uk

Abstract

The Extreme Deconvolution method is a mixture-model based approach original intended to perform density estimation on astrometry datasets. Newer much larger datasets are available to which the existing method for fitting the model cannot scale to. We propose two extensions to the extreme deconvolution method based on an online variation of the EM algorithm, and direct optimisation of the log-likelihood via SGD. We demonstrate that these methods provide faster fitting of the model, whilst being able to scale to much larger datasets.

1 Introduction

2 Background

$$\mathcal{L}(\theta) = \sum_i \log \sum_j \alpha_j \mathcal{N}(\mathbf{w}_i \mid \mathbf{m}_j, T_{ij}) \tag{1}$$

3 Methods

3.1 Online Expectation-Maximisation

The original method for fitting XD models used a modification of the Expectation-Maximisation (EM) algorithm for mixture models [2]. For the extreme deconvolution case, the expected sufficient statistics that need to be tracked are q_{ij}, $q_{ij}\mathbf{v}_i$, and $q_{ij}\mathbf{v}_i\mathbf{v}_i^T$. The M-step comprises normalisation of the sums of expected sufficient statistics to get the mixture component parameters.

Cappé and Moulines [1] proposed an online variation of EM for latent data models. Their method involves computing a stochastic approximation to the sums of expected sufficient statistics. At each iteration t, the sums of expected sufficient statistics s_t are computed over a minibatch, and the stochastic estimate $\hat{s}_t$ updated as

$$\hat{s}_t = (1 - \lambda)\hat{s}_{t-1} + \lambda s_t \tag{2}$$

where λ is a sufficiently small step-size. λ should theoretically decrease at each iteration according to the Robbins-Monro conditions, but in practice we found that using a constant step-size was sufficient. After each update of $\hat{s}_t$, we update the parameters.

3.2 Stochastic Gradient Descent

An alternative to EM-based methods is to optimise the log-likelihood directly. The log-likelihood has some constraints on its parameters, namely that the weight a_j sum to 1 and that the covariances V_j are positive definite. Directly fitting it with standard gradient-based optimisers requires a transformation

Preprint. Under review.

Table 1: Validation and test log-likelihoods for the Gaia data subset.

Method	Validation	Test
Existing EM	-	-
Online EM	-	-
SGD	-	-

of the parameters to remove the constraints. The mixture weights α_j can be parameterised by taking the softmax of an unconstrained vector $\mathbf{z}$,

$$\alpha_j = \frac{e^{z_j}}{\sum_k^K e^{z_k}}. \tag{3}$$

We parametrise the covariances V_j via the Cholesky decomposition,

$$V_j = L_j L_j^T \tag{4}$$

where L_j is a lower-triangular matrix. Note that for a standard GMM we could bypass forming the covariance altogether and evaluate the log PDF of the Normal distribution directly with the Cholesky factor. For the extreme deconvolution model, this cannot be done, as we need to form the covariance Tij for every datapoint. In practice we found that using the Adam optimisation algorithm eliminates the need to do this [4].

4 Experiments

We implemented both of our methods in PyTorch. The use of PyTorch allows us to take advantage of its automatic differentiation framework and in-built optimisers for the SGD method, as well as allowing us to write code targeting both CPU and GPU-based computation.

To evaluate our methods, we used a subset of rows from the Gaia DR2 source table [3]. We selected the 5 primary astrometric features, along with the BP-RP colour and mean flux in the G-band. In total there were approximately 2 million rows. Where data were missing, we set the field to zero, the variance to a large value. This is only a small fraction of the full dataset size, but this allows us to fit the training data into memory, a requirement for use with the original implementation of extreme deconvolution. We used the same number of components $K = 128$ for each model, following In practice we would want to select a value of K by cross-validation.

Initialisation of all models was performed by first running a minibatch version of the K-means clustering algorithm on the dataset to get estimates of the weights and means [5]. Initial covariances were set to the identity matrix. Early stopping was done based on a held-out validation set comprising 10% of the rows. Final model performance was evaluated on another held-out test set also comprising 10% of the rows.

5 Discussion

References

[1] O. Cappé and E. Moulines. On-line expectation–maximization algorithm for latent data models. 71(3):593–613. ISSN 1467-9868. doi: 10.1111/j.1467-9868.2009.00698.x. URL `https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2009.00698.x`.

[2] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. 39(1):1–38. ISSN 0035-9246. URL `https://www.jstor.org/stable/2984875`.

[3] Gaia Collaboration. Gaia Data Release 2 - Summary of the contents and survey properties. 616:A1. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201833051. URL `https://www.aanda.org/articles/aa/abs/2018/08/aa33051-18/aa33051-18.html`.

[4] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. URL `http://arxiv.org/abs/1412.6980`.

[5] D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web - WWW '10*, page 1177. ACM Press. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772862. URL http://portal.acm.org/citation.cfm?doid=1772690.1772862.