Vertex finding by sparse model-based clustering

R Frühwirth[1], K Eckstein[1] and S Frühwirth-Schnatter[2]

[1] Institute of High Energy Physics, Austrian Academy of Sciences, Vienna

[2] Institute for Statistics and Mathematics, Vienna University of Economics and Business

E-mail: rudolf.fruehwirth@oeaw.ac.at

Abstract. The application of sparse model-based clustering to the problem of primary vertex finding is discussed. The observed z-positions of the charged primary tracks in a bunch crossing are modeled by a Gaussian mixture. The mixture parameters are estimated via Markov Chain Monte Carlo (MCMC). Sparsity is achieved by an appropriate prior on the mixture weights. The results are shown and compared to clustering by the expectation-maximization (EM) algorithm.

1. Introduction

Primary vertex finding is an important step in the data analysis of collider experiments. In the case of the LHC experiments, there is in each bunch crossing a large background of so-called pile-up events which are superimposed to the signal event, i.e. the one that has triggered the recording of the bunch crossing. Finding the correct assignment of the charged tracks to their respective production vertices is crucial for the subsequent steps of vertex fitting and kinematic fitting. The distribution of the primary vertices is very narrow in the transverse directions, but much wider in the direction along the beams, which by convention is called the z direction. Finding the primary vertices can therefore be formulated as a one-dimensional clustering problem, namely finding clusters of primary tracks in the z direction. In a perfect clustering, each found cluster of tracks corresponds to a single primary vertex. In reality, a cluster produced by the clustering may contain tracks from several vertices, for instance when the clustering fails to resolve two close vertices and merges them into a single cluster.

In this paper two methods are compared: sparse model-based clustering [1] and the EM algorithm [2], an iterative implementation of maximum-likelihood estimation. In both methods the observations are assumed to be drawn from a mixture model. Model-based clustering has the advantage that it can include additional available information via prior densities. More specifically, prior knowledge on the following quantities can be used: the number of clusters or vertices; the cluster size, i.e. the number of tracks per vertex; and the spread of a cluster in the z direction. In the sparse version clusters are allowed, even encouraged, to be empty. The EM algorithm is less flexible, as the number of clusters is fixed, although clusters can effectively merge. The distribution used to model the clusters is in principle arbitrary; in the case of vertices a normal model is both adequate and easier to implement than other distributions.

2. Data

Proton-proton collisions were simulated using PYTHIA 8 [3], an event simulation engine for high-energy physics. The center of mass energy was set to 13 TeV, the one of the LHC run in 2015. The detailed settings can be obtained from the authors on request.

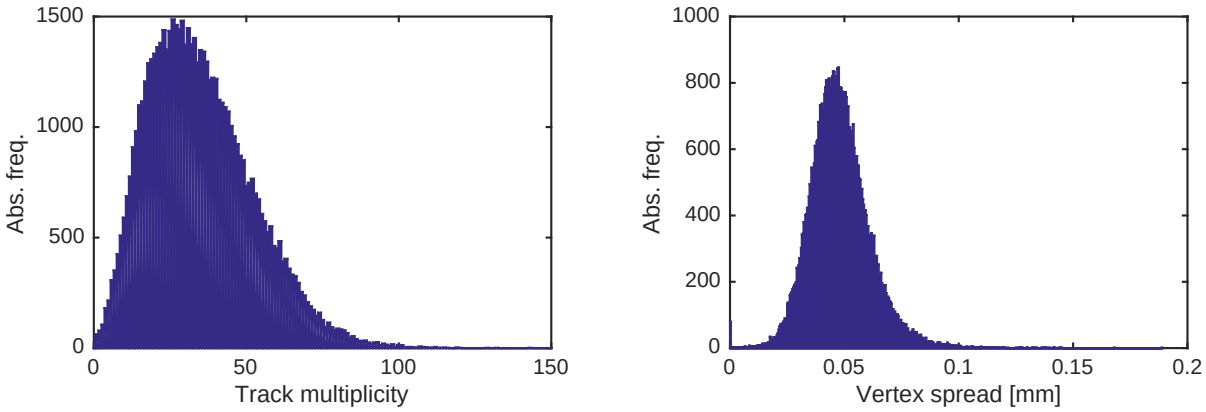

Figure 1. Left: distribution of the charged track multiplicity M_v per vertex, summed over all simulated vertices; right: distribution of the spread s_v of all simulated vertices.

Each collision produces a vertex with a number of particles attached to it. Only charged particles in the final state with a transverse momentum of at least $700\,\mathrm{MeV}/c$ are retained. In addition, their polar angle θ is restricted to the interval $9^\circ \leq \theta \leq 171^\circ$, roughly corresponding to the acceptance of a typical vertex detector. The multiplicity of the remaining charged tracks per vertex is denoted by M_v. Its empirical distribution is shown in the left panel of Figure 1. In order to make use of the distribution in the clustering, it is smoothed by a kernel estimator.

The z-positions of the vertices are generated according to a normal distribution centered at the origin with standard deviation $\sigma = 65\,\mathrm{mm}$, corresponding to the luminous region in the LHC (see [4]). This standard deviation has been determined from $\sqrt{s} = 7\,\mathrm{TeV}$ data, but is about the same for $\sqrt{s} = 13\,\mathrm{TeV}$ collisions. The z-positions of the tracks belonging to a vertex are smeared by an additional statistical error due to multiple scattering in the beam tube plus the position and the extrapolation error from the innermost pixel layer. For each vertex, its spread s_v is computed as the r.m.s. of the z-positions of the tracks belonging to the vertex. The empirical distribution of s_v is shown in the right panel of Figure 1 and is characterized by its mean $\mu_s = 0.048\,\mathrm{mm}$ and its standard deviation $\sigma_s = 0.013\,\mathrm{mm}$.

To simulate an amount of pile-up similar to the one expected in the Phase I upgrade of LHC, the vertices are grouped into individual bunch crossings. Each bunch crossing consists of L superimposed vertices, where L is drawn from a Poisson distribution with mean $\lambda = 100$. As the average track multiplicity M_v per vertex is about 35, the average number of tracks per bunch crossing ist about 3500.

3. Clustering algorithms

3.1. Data preparation for clustering

Clustering produces groups of tracks, called clusters, that are compatible with the hypothesis that they belong to the same collision vertex. The input is the set of the z-positions z_i of all tracks in a bunch crossing. For a better performance, however, the clustering is not applied to all tracks in a bunch crossing, but rather to sections in z with boundaries that depend on the data. To find suitable boundaries, the z-positions of the tracks in a bunch crossing are filled into a 1D histogram with a bin width of $h = 1\,\mathrm{mm}$. Then the following three preparation steps are executed.

1. First, boundaries of *basic sections* (BS) are defined by empty bins. These BS usually contain between 1 and 3 vertices. Tracks belonging to the same vertex are always contained within a single BS, as there is a gap of at least 1 mm between two BS.

2. Next, as a trade-off between accuracy and speed, ten adjacent BS are merged into one *final section* (FS).

3. For each FS, the three most likely numbers of vertices K_1, K_2, K_3 are estimated. The likelihood of a value K is given by the probability of the vertex multiplicity $M = N/K$ in the empirical distribution $g(M_\mathrm{v})$, where N is the number of tracks in the FS.

The cluster finding then proceeds independently in each FS. This can be done in parallel, although this was not exploited in the present study.

3.2. Sparse model-based clustering

The sparse model-based clustering algorithm and its implementation is explained in detail in [1]. The basic idea is to start with a large number of clusters and let the algorithm find out how many clusters are needed in the most economical clustering.

The input to the clustering are the z-positions $z_i,\ i = 1, \ldots, N$, of the N tracks in the FS. The initial number of clusters K is the largest of $K_1 + K_0, K_2 + K_0, K_3 + K_0$ with $K_0 = 5$. The observations z_i are assumed to be drawn from a Gaussian mixture with K components and the following probability density function (pdf):

$$f(z_i|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K, \boldsymbol{\eta}) = \sum_{k=1}^{K} \eta_k \, \varphi_k(z_i|\boldsymbol{\theta}_k), \tag{1}$$

where $\varphi_k(z_i|\boldsymbol{\theta}_k)$ is the Gaussian pdf with component specific parameters $\boldsymbol{\theta}_k = (\mu_k, \sigma_k^2)$, and η_k is the weight of component k. Solutions that are sparse with respect to K are obtained by choosing an appropriate prior distribution for the weights $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_K)$. In this study the prior is a symmetric Dirichlet distribution with concentration parameter e_0 and the following pdf:

$$p(\eta_1, \ldots, \eta_K|e_0) = \frac{\Gamma(Ke_0)}{\Gamma(e_0)^K} \prod_{k=1}^{K} \eta_k^{e_0 - 1}. \tag{2}$$

Smaller values of e_0 tend to produce fewer clusters than larger values. The prior distribution of the component means is a Gaussian centered at the mean b of the observations. Its width depends on a hyperparameter N_0 that controls the amount of information in the prior relative to the information in the observations. The prior of the component variances is the inverse Gamma distribution $\mathcal{IG}(c, C)$, where the hyperparameters c and C are chosen such that the mean is μ_s and the variance is ${\sigma_s}^2$ (see Section 2 and Figure 1).

The clustering proceeds via data augmentation. For each observation z_i, a latent allocation variable S_i is created that takes integer values in the set $\{1, \ldots, K\}$. If $S_i = k$, z_i is assigned to cluster k, and the pdf of z_i conditional on the assignment is given by:

$$f(z_i|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K, S_i = k) = \varphi(z_i|\mu_k, \sigma_k^2), \quad \Pr(S_i = k|\boldsymbol{\eta}) = \eta_k. \tag{3}$$

Initial values of the allocation variables $\boldsymbol{S}$ are obtained from a tentative clustering by the simple and familiar k-means algorithm [5]. Estimation of the allocation variables is done via MCMC. A Gibbs sampler [6] generates a random sample from the posterior distribution of $\boldsymbol{S}$, given the data and all priors. It iterates the following steps:

(i) Simulate the weights η_k and the component specific parameters $\boldsymbol{\theta}_k = (\mu_k, \sigma_k^2)$ conditional on the current allocations, for $k = 1, \ldots, K$.

(ii) Sample the allocation variables $S_i, i = 1, \ldots, N$ from the discrete distribution on $\{1, \ldots, K\}$ defined by:

$$\Pr(S_i = k|z_i, \eta_k, \mu_k, \sigma_k^2) \propto \eta_k \varphi(z_i|\mu_k, \sigma_k^2). \tag{4}$$

(iii) Update the hyperparameters b, N_0, c, C according to the current cluster sizes.

The full details of the sampler including the updating formulas are spelled out in [1]. After generating the Markov chain, the clusters are identified by choosing the configuration of $\boldsymbol{S}$ with the largest posterior probability. Note that the number of clusters that are actually populated is likely to be smaller than the initial number K.

3.3. EM clustering

The EM algorithm [2] is an iterative method for finding the maximum likelihood estimates of the weights η_k and the component parameters $\boldsymbol{\theta}_k$ of the mixture in Eq. (1) for given K. In contrast to the sparse model-based clustering, the EM algorithm is not adaptive. To compensate for this, multiple runs with different numbers of components were performed, using the previously estimated most likely numbers of vertices for the algorithm, namely $K_1, K_2, K_3, \ldots, K_3 + 5$. Finally, the clustering with the smallest Bayesian information criterion (BIC) was selected.

4. Results

The two methods were compared by means of a simulation study with 600 bunch crossings, containing about 60 000 vertices and about 2 million tracks. The following runs were performed:

- Run MB1: Model-based clustering, $e_0 = 0.1$, long Markov chain (5000 draws)
- Run MB2: Model-based clustering, $e_0 = 1.0$, long Markov chain (5000 draws)
- Run MB3: Model-based clustering, $e_0 = 1.0$, short Markov chain (1000 draws)
- Run EM: EM algorithm with BIC

The performance of the clustering methods can be assessed by the difference $\Delta K = K_{\text{est}} - K_{\text{true}}$ of the estimated and the true number of clusters in a FS, and by the cluster purity, defined as the largest fraction of tracks in the cluster belonging to the same vertex. The frequency distribution of ΔK is shown in Fig. 2 for the four runs. The numerical results are summarized in Table 1. It shows the average, the standard deviation and the root of the mean squared error (r.m.s.e.) of ΔK, the average purity and the fraction of perfect clusters with 100% purity, i.e. clusters without contamination from tracks belonging to other vertices.

The run with the smallest r.m.s.e. is MB2, and the average number of clusters is very close to the true one. The shorter chain in MB3 still gives the correct average, but a slightly larger standard deviation and r.m.s.e. The standard deviation of ΔK in run MB1 is smaller than in run MB2, but there are too few clusters on average. The EM algorithm combined with the BIC overestimates the number of clusters, a feature described in the literature [7]. In all runs, more than 80% of the clusters are perfect; in nearly all of the other cases the cluster purity is between 50% and 100%, indicating the fusion of two very close primary vertices into a single cluster. This problem can be resolved to some extent in a subsequent adaptive vertex fit [8].

Table 1. Summary of the results.

	$\Delta K = K_{\text{est}} - K_{\text{true}}$			Purity	
Run	average	std. dev.	r.m.s.e.	average	perfect
MB1	−0.98	1.43	1.73	0.95	0.83
MB2	−0.06	1.52	1.52	0.95	0.83
MB3	0.03	1.68	1.69	0.95	0.82
EM	1.47	3.15	3.47	0.93	0.81

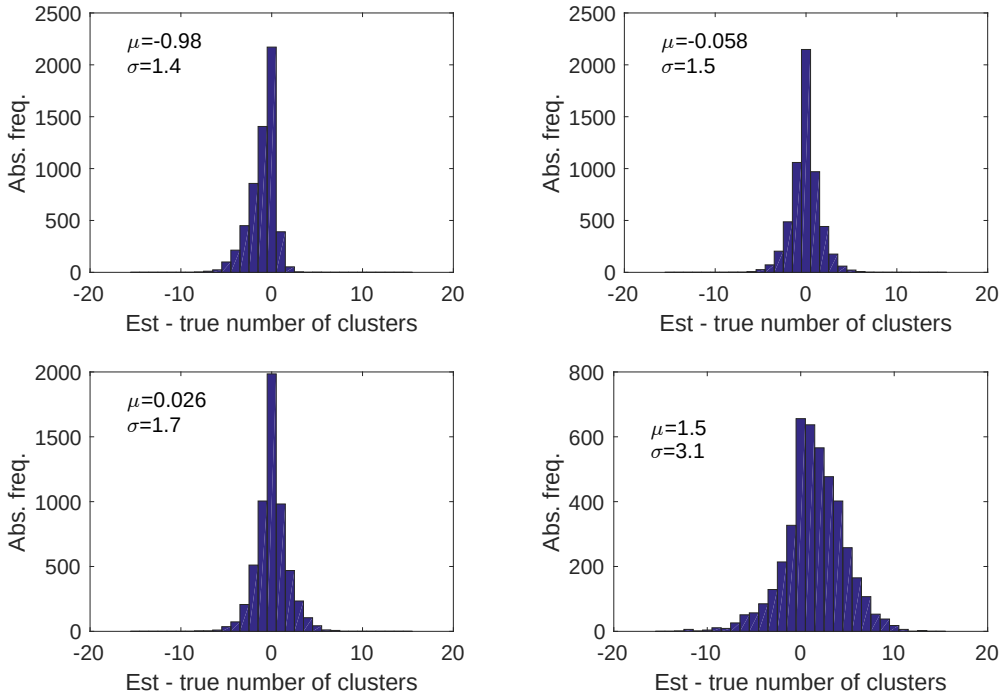

Figure 2. Estimated minus true number of clusters for runs MB1 (top left), MB2 (top right), MB3 (bottom left), and EM (bottom right).

5. Discussion and outlook

Model-based clustering has a number of attractive features: prior information on the number of tracks per vertex and the vertex spread is used; this information can be extracted from runs with low luminosity, where vertex finding is easy; only the expected (average) number of vertices depends on the luminosity; with a mimimal amount of tuning the correct average number of vertices is obtained, in contrast to the EM algorithm; and finer tuning of the parameters of the priors can be expected to give further improvements in the performance.

The main drawback is that sampling by MCMC is slow by HEP standards, so that model-based clustering is probably not suitable for standard vertex finding. Also, fine tuning of all hyperparameters is computationally intensive. The speed of the method is, however, less relevant in applications with fewer observations and fewer clusters, for instance ring finding in RICH detectors or cluster finding in calorimeters, where a lot of prior information on the cluster shape is available and can be incorporated in a natural manner. Further studies will have to concentrate on the optimal settings of the prior parameters and also on the robustness of the clustering against additional noise.

References

[1] Malsiner-Walli G, Frühwirth-Schnatter S and Grün B 2016 *Statistics and Computing* **26** 303–324
[2] Dempster A, Laird N and Rubin D 1977 *Journal of the Royal Statistical Society B* **39** 1–38
[3] Sjöstrand T, Mrenna S and Skands P 2008 *Computer Physics Communications* **178** 852–867
[4] Bartoldus R (for the ATLAS Collaboration) 2012 *Physics Procedia* **37** 2080–2088
[5] Anderberg M 1973 *Cluster Analysis for Applications* (New York: Academic Press)
[6] Tierney L 1994 *The Annals of Statistics* **22** 1701–1728
[7] Baudry J et al 2010 *Journal of Computational and Graphical Statistics* **9** 332–353
[8] Waltenberger W, Frühwirth R and Vanlaer P 2007 *Journal of Physics G: Nuclear and Particle Physics* **34** N343–N356