# Bayesian K-Means as a "Maximization-Expectation" Algorithm

Max Welling * Kenichi Kurihara †

**Abstract**

We introduce a new class of "maximization expectation" (ME) algorithms where we maximize over hidden variables but marginalize over random parameters. This reverses the roles of expectation and maximization in the classical EM algorithm. In the context of clustering, we argue that the hard assignments from the maximization phase open the door to very fast implementations based on data-structures such as kd-trees and conga-lines. The marginalization over parameters ensures that we retain the ability to select the model structure. As an important example we discuss a top-down "Bayesian k-means" algorithm and a bottom-up agglomerative clustering algorithm. In experiments we compare this algorithm against a number of alternative algorithms that have recently appeared in the literature.

## 1 Introduction

K-means is undoubtedly one of the workhorses of machine learning. Faced with the exponential growth of data, researchers have recently started to study strategies to speed up k-means and related clustering algorithms. Most notably, methods based on kd-trees [7, 6, 11, 10] have been very successful in achieving orders of magnitude improved efficiency.

Another topic of intense study has been to device methods that automatically determine the number of clusters from the data [7, 4]. One of the most promising algorithms in this respect is based on the "variational Bayesian" (VB) paradigm [1, 3]. Here distributions over cluster assignments and distributions over stochastic parameters are alternatingly estimated in an EM-like fashion. Unfortunately, it is difficult to apply the speed-up tricks mentioned above to this algorithm, at least without introducing approximations.

One of the main goals in this paper is to propose a modification of VB-clustering that combines model selection with fast implementations. The technique we propose is an instance of a new class of "ME algorithms" that reverses the roles of expectation and maximization in the EM algorithm. Alternatively, it can be viewed as a special case of the VB framework where expectation over hidden variables is replaced with maximization.

In the context of clustering we discuss a ME algorithm that is very similar to k-means but uses a full covariance and an upgraded "distance" to penalize overly complex models. We also derive an alternative agglomerative clustering algorithm. Both algorithms can be implemented efficiently using kd-trees and conga-lines respectively. Experimentally, we see no performance drops relative to VB but at the same time we demonstrate significant speedup factors. All software is publicly available at *http://mi.cs.titech.ac.jp/kurihara/bkm.html*

## 2 Maximization-Expectation Algorithms

Consider a probabilistic model, $p(x, z, \theta)$, where $x$ and $z$ are observed and a hidden random variables , and $\theta$ is a set of parameters (which are assumed random as well). Given a dataset $\mathcal{D}$, a typical task in machine learning is to compute posterior or marginal probabilities such as $p(z|\mathcal{D})$, $p(\theta|\mathcal{D})$ or $p(\mathcal{D})$. It is not untypical that exact expressions for these quantities can not be derived and approximate methods become necessary. We will now review a number of approaches based on alternating model estimation.

A standard approach is to represent the distribution using samples like a Gibbs sampler. Instead of sampling, one can fit factorized variational distributions to the exact distribution, i.e. $p(\theta, z|\mathcal{D}) \approx q(\theta)q(z)$. This "variational Bayesian" (VB) approximation [1, 3] alternatingly estimates these distributions by minimizing the KL-divergence, $KL[q(\theta)q(z)||p(\theta, z|\mathcal{D})]$ between the approximation and the exact distribution. This results in the updates,

$$q(\theta) \propto \exp(\mathbb{E}[\log p(\theta, z, \mathcal{D})]_{q(z)})$$
$$\leftrightarrow q(z) \propto \exp(\mathbb{E}[\log p(\theta, z, \mathcal{D})]_{q(\theta)}).$$

Instead of maintaining a distribution over parameters, one could decide to estimate a MAP value for $\theta^*$ by the expectation-maximization algorithm (EM) to alternatingly compute,

$$\theta^* = \underset{\theta}{\text{argmax}}\ \mathbb{E}[\log(p(\theta, z, \mathcal{D})]_{q(z)} \leftrightarrow q(z) = p(z|\theta^*, \mathcal{D}).$$

This turns out to be a special case of the VB formalism

*Dept. of Computer Science University of California Irvine, CA, USA

†Dept. of Computer Science Tokyo Institute of Technology, Japan

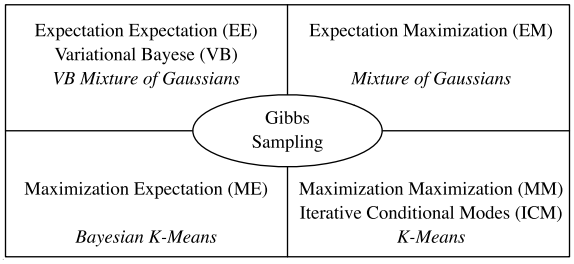


Figure 1: Four faces of alternating model learning: EE, EM, ME and MM algorithms

where $q(\theta) = \delta(\theta, \theta^*)$. Finally, one can also choose to use point-estimates for both $\theta$ and $z$, which is known as "iterative conditional modes" (ICM). In this case we iteratively maximize the posterior distributions, $p(\theta|z, \mathcal{D})$ and $p(z|\theta, \mathcal{D})$ which is equivalent to,

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\ p(z^*, \theta, \mathcal{D}) \leftrightarrow z^* = \underset{z}{\operatorname{argmax}}\ p(z, \theta^*, \mathcal{D}).$$

In figure 1 we have depicted the 4 algorithms discussed so far, Gibbs sampling, VB, EM, and ICM. A particular choice among these methods often represents a trade-off between computational efficiency and accuracy, where ICM is the most efficient and VB and Gibbs sampling the least efficient of these methods. On the other hand, VB has the potential of model selection. Viewing VB as an extension of EM, we could classify VB as an "expectation-expectation" (EE) algorithm while ICM should be interpreted as a "maximization-maximization" (MM) algorithm. This paper is about a new class of algorithms which we call "maximization-expectation" algorithms where we maximize over hidden variables but take expectations over parameters,

$$q(\theta) = p(\theta|z^*, \mathcal{D}) \leftrightarrow z^* = \underset{z}{\operatorname{argmax}}\ \mathbb{E}[\log(p(\theta, z, \mathcal{D})]_{q(\theta)}.$$

We can interpret the ME algorithm as an approximation of the VB formalism where we use $q(z) = \delta(z, z^*)$. The main motivation behind this idea is to propose a class of algorithms that is computationally efficient on the one hand but has the necessary ingredients for model selection on the other. In particular for the clustering example under consideration, we will show that efficient data-structures such as kd-trees can be employed with relative ease. Although this paper focuses on the "Bayesian K-Means" algorithm we like to stress that our ideas are rather more general than this.

## 3 The Hard Assignment Approximation

Let $\mathbf{x}_n,\ \{n = 1, ..., N\}$ be a vector of IID continuous-valued observations in $D$ dimensions. For each data-case we define a cluster assignment variable, $z_n \in [1, ...K]$. We assume that the data has been generated following a Gaussian a mixture of model,

$$p(\mathbf{x}, z) = \int \mathrm{d}\boldsymbol{\mu}\mathrm{d}\Omega\mathrm{d}\boldsymbol{\alpha}\ \mathcal{N}(\mathbf{x}|z, \boldsymbol{\mu}, \Omega)\ \mathcal{M}(z|\boldsymbol{\alpha}) \\ \mathcal{D}(\boldsymbol{\alpha}|\phi_0)\ \mathcal{N}(\boldsymbol{\mu}|\mathbf{m}_0, \xi_0\Omega)\ \mathcal{W}(\Omega|\eta_0, B_0). \tag{3.1}$$

Here, $\mathcal{N}(\cdot)$ is a normal distribution, $\mathcal{M}(\cdot)$ a multinomial (discrete) distribution, $\mathcal{D}(\cdot)$ a Dirichlet distribution and $\mathcal{W}(\cdot)$ a Wishart distribution. The priors depend on some (non-random) hyper-parameters, $\mathbf{m}_0, \phi_0, \xi_0, \eta_0, B_0$ .

We restrict the full posterior distribution to have a factorized form, $p(z, \boldsymbol{\theta}|\mathcal{D}) \approx q(\boldsymbol{\theta})\ \delta(z, z^*)$ i.e. we maintain a distribution over parameters, but settle for a point-estimate of the assignment variables. Minimizing the KL-divergence between this approximate posterior and the exact posterior, $KL[q(\boldsymbol{\theta})\delta(z, z^*)||p(z, \boldsymbol{\theta}|\mathcal{D})]$, over $q(\boldsymbol{\theta})$ we find,

$$q(\boldsymbol{\mu}, \Omega, \boldsymbol{\alpha}) = \mathcal{D}(\alpha|\{\phi_c\}) \prod_{c=1}^{K} \mathcal{N}(\boldsymbol{\mu}_c|\mathbf{m}_c, \xi_c\Omega_c)\ \mathcal{W}(\Omega_c|\eta_c, B_c)$$

where

$$\begin{aligned} \xi_c &= \xi_0 + N_c & \mathbf{m}_c &= \frac{N_c\bar{\mathbf{x}}_c + \xi_0\mathbf{m}_0}{\xi_c} \\ \eta_c &= \eta_0 + N_c & \phi_c &= \phi_0 + N_c \\ B_c &= B_0 + N_cS_c + \frac{N_c\xi_0}{\xi_c}(\bar{\mathbf{x}}_c - \mathbf{m}_0)(\bar{\mathbf{x}}_c - \mathbf{m}_0)^T \end{aligned} \tag{3.2}$$

and where $N_c$ denotes the number of data-cases in cluster $c$, $\bar{\mathbf{x}}_c$ is the sample mean of cluster $c$, and $S_c$ is its sample covariance.

The main simplification that has resulted from the approximation is that this distribution now factorizes over the clusters. Using this, we derive the following bound on the log evidence (or "free energy")[1],

$$\mathcal{F}(z,K) = \sum_{c=1}^{K} \Big[ \frac{DN_c}{2}\log\pi + \frac{\eta_c}{2}\log\det(B_c) \\ -\frac{\eta_0}{2}\log\det(B_0) + \frac{1}{2}\log\frac{\xi_c}{\xi_0} - \log\frac{\Gamma_D(\frac{\eta_c}{2})}{\Gamma_D(\frac{\eta_0}{2})} \\ -\log\frac{\Gamma(\phi_c)}{\Gamma(\phi_0)} + \frac{1}{K}\log\frac{\Gamma(N + K\phi_0)}{\Gamma(K\phi_0)}\Big] \tag{3.3}$$

where $\Gamma_D(x) = \pi^{\frac{D(D-1)}{4}} \prod_{i=1}^{D} \Gamma(x + \frac{1-i}{2})$ and $\Gamma(\cdot)$ the Gamma function. Our task is to minimize Eqn.3.3 jointly over assignment variables $z_n,\ n = 1..N$ and $K$, the number of clusters.

[1] $\mathcal{F}(z, K) = -\mathbb{E}[p(\mathcal{D}, \theta, z')]_{q(\theta)\delta(z',z)} - H[q(\theta)] = -\log p(\mathcal{D}) + KL[q(\theta)\delta(z', z)||p(\theta, z'|\mathcal{D})] \geq -\log p(\mathcal{D})$

## 4 "Bayesian K-Means" and "Agglomerative Bayesian Clustering"

In this paper we study two approaches to minimizing the cost Eqn.3.3, a Bayesian variant of k-means (BKM) and an agglomerative Bayesian clustering algorithm.

From Eqn.2.1 we see that the ME algorithm iteratively maximizes $\mathbb{E}[\log(p(\boldsymbol{\theta}, z, \mathcal{D})]_{q(\boldsymbol{\theta})}$, where $p(\boldsymbol{\theta}, z, \mathcal{D})$ can be derived from Eqn.3.1 and $q(\boldsymbol{\theta})$ is given by Eqn.3.2. This leads (after some algebra) to the following iterative "labelling cost",

$$\mathcal{C}_{\text{BKM}} = \sum_n \gamma_{z_n}(\mathbf{x}_n) \qquad \text{with} \tag{4.4}$$

$$\begin{aligned}\gamma_{z_n}(\mathbf{x}_n) =& \frac{\eta_{z_n}}{2}(\mathbf{x}_n - \mathbf{m}_{z_n})^T B_{z_n}^{-1}(\mathbf{x}_n - \mathbf{m}_{z_n}) \\ &+ \frac{1}{2}\log\det(B_{z_n}) + \frac{D}{2\xi_{z_n}} \\ &- \frac{1}{2}\sum_{d=1}^{D}\psi\left(\frac{\eta_{z_n}+1-d}{2}\right) - \psi(\phi_{z_n}),\end{aligned} \tag{4.5}$$

where $\psi(x) = \frac{\partial \log\Gamma(x)}{\partial x}$ is the digamma function. We recognize $\gamma_{z_n}(\mathbf{x}_n)$ as the Mahalanobis distance plus some constant correction term. BKM thus alternates updating assignment variables $z_n$ to minimize $\mathcal{C}_{\text{BKM}}$ and recalculating the quantities in Eqn.3.2 similar to the classical k-means algorithm. To search over different numbers of clusters we need to introduce cluster split and merge operations[2].

An alternative approach to minimize Eqn.3.3 is bottom-up agglomerative clustering. At first, assign a separate cluster to every data-case. At each iteration we search for the best pair of clusters to merge which maximally decrease the objective. This Bayesian agglomerative algorithm stops when it has inferred the number of clusters in the data (see also [5]).

To improve computational efficiency we adapted methods based on kd-trees [7, 6]. In particular, we need to compute the minimal and maximal distance between a cluster and a hyper-rectangle, where "distance" is defined by Eqn.4.5. While the minimal distance is a standard constrained QP, the maximal distance must be replaced by the following bound to maintain tractability: replace $B_c^{-1}$ in Eqn.4.5 by $\lambda_c^{\max} I$ with $\lambda_c^{\max}$ the largest eigenvalue of $B_c^{-1}$. Using this, the problem becomes axis aligned and can be separately solved in each dimension. For agglomerative clustering we have used "Conga-Lines" [2] to reduce time complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2\log(N))$. We emphasize that the application of these data-structures is restricted to hard-clustering algorithms if we do not allow approximations.

**Top-Down Bayesian K-Means**

1 *Initialize:*
1i Set hyper-parameters $\{B_0, \xi_0, \eta_0, \phi_0, \mathbf{m}_0\}$.
1ii Assign all data-cases to a single cluster $\{z_n\} = 1$.
2 *Repeat split operations until no more improvement is possible:*
2i Use heuristic to rank clusters for splitting.
2ii Split highest ranked cluster.
2iii *Run Bayesian k-means updates until convergence:*
2iiia Update quantities $\{B_c, \xi_c, \eta_c, \phi_c, \mathbf{m}_c\}$ using Eqn.3.2.
2iiib Update assignment variables using Eqn.4.5.
2iv *Accept or Reject Split*
2iva If the free energy decreased: accept split and goto 2i.
2ivb Otherwise: reject split, remove candidate from list and goto 2ii.
3 *Merge two clusters into one cluster:*
3i Use heuristic to rank pairs of clusters for merging.
3ii Merge highest ranked cluster.
3iii Run Bayesian k-means updates until convergence (see 2iia & 2iib above)
3iv *Accept or Reject Merge*
3iva If the free energy decreased: accept merge and goto 2.
3ivb Otherwise: reject merge, remove candidate from list and goto 3ii.

**Bottom-Up Agglomerative Clustering**

1 *Initialize:*
1i Set hyper-parameters $\{B_0, \xi_0, \eta_0, \phi_0, \mathbf{m}_0\}$.
1ii Assign all data-cases to a separate cluster $\{z_n = n\}$.
2 *Repeat merge operations until no more improvement is possible:*
2i Update quantities $\{B_c, \xi_c, \eta_c, \phi_c, \mathbf{m}_c\}$ using Eqn.3.2.
2ii Find the pair of clusters that generates the largest decrease in the free energy in Eqn.3.3.
2iii Merge closest pair by assigning their data-cases to a single cluster and goto 2i.

## 5 Experimental Results

In our first experiment we compared Bayesian K-means (BKM) with G-means [4] , K-means+BIC [8] , mixture of Gaussians + BIC and variational Bayesian learning for mixtures of Gaussians (VBMG) [1, 3]. K-means+BIC, mixtures of Gaussians+BIC and VBMG use the same split and merge strategy as BKM. These algorithms were tested on a number of synthetic datasets which sampled from 10 Gaussians (i.e. $K$=10). They were generated similarly to [4][3]. Table 1 shows the results of the experiments. It should come as no

[2] We have followed the approach taken in the SMEM algorithm [9]. Our criteria to rank clusters are also the same as [9]. When we split, we initialize the new clusters using the heuristic described in [4].

[3] For each synthetic dataset, we sampled K centroids and stds. such that no two clusters are closer than $\tau$ × (the sum of two stds.) / 2. After that, we multiplied the data within each cluster with a random matrix to give them full covariance structure.

surprise that BKM, VBMG and MoG+BIC outperform G-means and k-means+BIC since the latter assume isotropic covariance. Between BKM, VBMG and MoG+BIC it seems hard to discern significant difference in performance. Note however that there are no speedup techniques available for VBMG and only an approximate speedup technique for MoG+BIC [6]. In contrast, the kd-tree speedup is exact.

Next, we evaluated agglomerative Bayesian clustering (ABC) against the following traditional agglomerative clustering methods: single, complete and average linkages. We ran these algorithms on a number of handwritten digits datasets: Pendigits, CEDAR dataset, MNIST dataset and "3 digits" dataset[4]. Following [4], we applied a random projection to MNIST to reduce the dimension to 50. For each dataset, we created 10 new datasets consisting of 100 randomly chosen data-cases. Dendrograms were evauated using dendrogram purity[5]. Table 2 shows the results. On CEDAR, MNIST and 3 digits, ABC built the highest purity dendrograms.

Finally, we compared the naive BKM implementation against efficient implementations using kd-trees and Conga-Lines. For the kd-tree experiment we varied one parameter at a time and used default values of $N = 20000$, $D = 2$, $\tau = 3$ and $K = 5$ to sample data. A node in a kd-tree is declared a leaf if the number of data-cases in the node is less than 1000 (the remaining points are treated individually). In the experiments with Conga-Lines, we vary $N$ and use $D = 2$, $\tau = 3$ and $K = 10$. Figures 2 show the speedup factors. Speedup increases very fast with $N$ (factor of 67 at $N = 80000$), moderately fast with $\tau$ and decreases surprisingly slow with dimension (factor of 3.6 in 256 dimensions at $N = 20000$). BKM using Conga-Lines is only slightly faster than the naive implementation due to large overhead but this clearly becomes more significant with growing $N$.

## 6 Conclusion

The contributions of this paper are twofold. Firstly a new class of algorithms is introduced that reverses the roles of "expectation" and "maximization" in the traditional EM algorithm. This combines the potential for model selection with fast implementations . Secondly, we have implemented and studied one possible application of this idea in the context of clustering. Explicit algorithms were derived for bottom-up agglomerative Bayesian clustering and top-down Bayesian K-means clustering and experimentally tested.

Although we have explored these ideas in the simplest possible context, namely clustering, the proposed techniques seem to readily generalize to more complex models such as HMMs. Whether the efficiency gains can also be achieved in this setting remains to be investigated.

[4] These datasets have 10 true classes (digits 0-9). Pendigits, CEDAR and MNIST have 7984, 7000 and 60000 data-cases and 16, 64 and 784 dimensions respectively. "3 digits" is a dataset which only contains the "0"s, "2"s and "4"s from MNIST.

[5] When a dendrogram and all correct labels are given, pick uniformly at random two leaves which have the same label $c$ and find the smallest subtree containing the two leaves. The dendrogram purity is the expected value of (#leaves with label $c$ in subtree)/(#leaves in the subtree). For each class, if all leaves in the class are contained in a pure subtree, the dendrogram purity is 1.


## Acknowledgments

We are grateful to the authors of [6] and [4] for sharing their code. Moreover, special thanks to Andrew Moore for answering questions about data-structures for fast implementations and Katherine Heller for sharing data.


## References


[1] H. Attias. A variational bayesian framework for graphical models. In *Neural Information Processing Systems 12*, 2000.

[2] David Eppstein. Fast hierarchical clustering and other applications of dynamic closest pairs. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[3] Z. Ghahramani and M. J. Beal. Variational inference for Bayesian mixtures of factor analysers. In *Neural Information Processing Systems*, volume 12, 2000.

[4] G. Hamerly and C. Elkan. Learning the $k$ in $k$-means. In *Neural Information Processing Systems*, volume 17, 2003.

[5] K. Heller and Z. Ghahramani. Bayesian hierarchical clustering. In *Twenty-second International Conference on Machine Learning*, 2005.

[6] A. Moore. Very fast EM-based mixture model clustering using multiresolution kd-trees. In *Neural Information Processing Systems Conference*, 1998.

[7] D. Pelleg and A. Moore. Accelerating exact k-means algorithms with geometric reasoning. In *Proc. of the 5th Int'l Conf. on Knowledge Discovery in Databases*, pages 277–281, 1999.

[8] D. Pelleg and A. Moore. $X$-means: Extending $K$-means with efficient estimation of the number of clusters. In *ICML*, volume 17, pages 727–734, 2000.

[9] N. Ueda, R. Nakano, Z. Ghahramani, and G.E. Hinton. SMEM algorithm for mixture models. *Neural Computation*, 12(9):2109–2128, 2000.

[10] J. Verbeek, J. Nunnink, and N. Vlassis. Accelerated variants of the em algorithm for gaussian mixtures. Technical report, University of Amsterdam, 2003.

[11] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *ACM-SIGMOD Int. Conf. Management of Data*, pages 103–114, 1996.

Table 1: Number of clusters estimated on synthetic data by G-means, K-means+BIC, mixture of Gaussians+BIC, Bayesian K-means and variational Bayesian learning. Results are averaged over 10 runs of the algorithm. True number of clusters is 10.

| $\tau$ | N | D | G-means | k-means+BIC | MoG+BIC | BKM | VBMG |
|---|---|---|---|---|---|---|---|
| 0.1 | 100 | 2 | 4.50±2.42 | 2.00±1.41 | 3.80±2.04 | 2.00±0.82 | 4.30±2.16 |
| | | 32 | 7.40±3.86 | 1.00±0.00 | 1.70±0.48 | 2.50±4.74 | 1.00±0.00 |
| | | 64 | 13.10±5.47 | 1.00±0.00 | 1.00±0.00 | 13.30±1.06 | 1.30±0.48 |
| | 1000 | 2 | 32.10±12.33 | 3.50±1.90 | 7.80±1.32 | 7.40±1.96 | 8.80±0.79 |
| | | 32 | 39.10±10.07 | 1.10±0.32 | 5.60±1.65 | 7.70±1.16 | 9.90±0.32 |
| | | 64 | 49.40±22.51 | 1.00±0.00 | 3.60±1.43 | 5.70±4.79 | 2.90±0.74 |
| | 5000 | 2 | 108.90±15.07 | 3.80±1.62 | 9.20±2.70 | 8.10±4.72 | 9.90±0.32 |
| | | 32 | 130.90±46.17 | 2.70±1.49 | 3.90±2.64 | 10.00±0.00 | 9.80±0.42 |
| | | 64 | 104.30±51.93 | 1.30±0.67 | 2.10±0.32 | 10.00±0.00 | 10.00±0.00 |
| 0.5 | 100 | 2 | 4.80±2.49 | 2.50±1.72 | 5.20±2.35 | 4.10±3.60 | 4.40±2.12 |
| | | 32 | 7.40±4.06 | 1.00±0.00 | 2.00±0.00 | 14.80±5.35 | 1.00±0.00 |
| | | 64 | 10.60±6.55 | 1.00±0.00 | 1.00±0.00 | 13.80±0.92 | 1.40±0.52 |
| | 1000 | 2 | 21.30±4.55 | 4.20±1.87 | 9.20±1.32 | 4.30±3.65 | 9.40±0.84 |
| | | 32 | 25.60±10.31 | 1.20±0.42 | 9.20±0.63 | 9.20±2.10 | 9.90±0.31 |
| | | 64 | 37.90±9.72 | 1.60±1.08 | 3.80±0.84 | 11.40±6.15 | 3.80±1.03 |
| | 5000 | 2 | 82.00±14.91 | 4.20±1.32 | 9.10±2.76 | 8.10±3.57 | 9.80±0.42 |
| | | 32 | 74.00±32.40 | 2.60±0.97 | 8.90±1.45 | 9.90±0.32 | 10.00±0.00 |
| | | 64 | 74.30±31.72 | 2.50±1.72 | 9.60±0.52 | 9.70±0.48 | 9.80±0.42 |
| 2 | 100 | 2 | 6.80±3.26 | 2.10±1.45 | 6.50±1.96 | 8.30±1.89 | 7.20±2.15 |
| | | 32 | 11.90±1.20 | 3.30±2.21 | 2.40±0.00 | 10.70±5.17 | 1.00±0.00 |
| | | 64 | 9.90±3.87 | 2.20±2.20 | 1.00±0.00 | 13.80±0.63 | 1.00±0.00 |
| | 1000 | 2 | 18.30±5.29 | 4.30±1.57 | 9.50±1.65 | 9.60±1.26 | 9.80±0.42 |
| | | 32 | 13.90±2.08 | 6.80±2.94 | 7.40±1.90 | 9.40±1.26 | 9.00±0.94 |
| | | 64 | 13.50±3.66 | 8.60±2.41 | 3.00±0.67 | 12.60±3.31 | 4.00±1.03 |
| | 5000 | 2 | 48.70±14.22 | 5.30±2.50 | 9.60±1.58 | 10.00±0.00 | 9.40±1.43 |
| | | 32 | 15.00±5.79 | 9.50±1.84 | 7.90±1.10 | 10.00±0.00 | 10.00±0.00 |
| | | 64 | 11.60±2.07 | 9.90±2.12 | 8.40±0.97 | 10.00±0.00 | 10.00±0.00 |

Table 2: Dendrogram purities on some digits datasets for various agglomerative clustering algorithms.

| Data Set | ABC | Single Linkage | Complete Linkage | Average Linkage |
|---|---|---|---|---|
| Pendigits | 0.701±0.085 | 0.691±0.055 | 0.653±0.055 | 0.707±0.054 |
| CEDAR | 0.424±0.069 | 0.297±0.051 | 0.402±0.055 | 0.403±0.039 |
| MNIST | 0.207±0.038 | 0.182±0.017 | 0.198±0.025 | 0.199±0.026 |
| 3 digits | 0.487±0.031 | 0.395±0.021 | 0.411±0.020 | 0.407±0.027 |

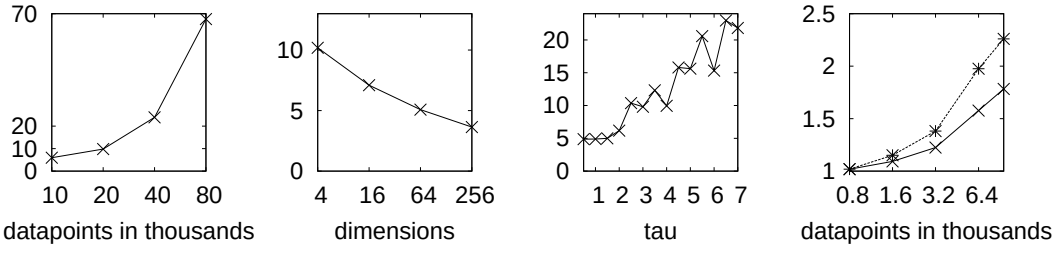


Figure 2: Speedup factors using fast data-structures. Three top figures show speedup factors for kd-trees with varying $N$, $D$ and $\tau$. The right bottom figure shows results for Conga-Lines. Here, $\times$ and $*$ denote overall speedup factor and speedup factor per iteration respectively.