Bulk Loading Hierarchical Mixture Models for Efficient Stream Classification

Philipp Kranen, Ralph Krieger, Stefan Denker, and Thomas Seidl

Data Manangement and Exploration Department
RWTH Aachen University, Germany
{kranen,krieger,denker,seidl}@cs.rwth-aachen.de

Abstract. The ever growing presence of data streams led to a large number of proposed algorithms for stream data analysis and especially stream classification over the last years. Anytime algorithms can deliver a result after any point in time and are therefore the natural choice for data streams with varying time allowances between two items. Recently it has been shown that anytime classifiers outperform traditional approaches also on constant streams. Therefore, increasing the anytime classification accuracy yields better performance on both varying and constant data streams. In this paper we propose three novel approaches that improve anytime Bayesian classification by bulk loading hierarchical mixture models. In experimental evaluation against four existing techniques we show that our best approach outperforms all competitors and yields significant improvement over previous results in term of anytime classification accuracy.

1 Introduction

With an abundance of streaming data due to widely deployed sensors or other data gathering devices, analysis of streaming data such as stream classification has recently received much attention in data mining research. Algorithms that work on data streams have to cope with the limited and often also varying amount of computation time. Traditionally they got for a certain task a fixed time budget which was known in advance, i.e. they were tailored to the specific application. These budget algorithms can neither provide a results in less time nor exploit additional time to improve their result. In contrast, so called anytime algorithms can provide a result after a very short initialization, improve their result incrementally when more time is available and hold the most recent result ready at any time. In data mining anytime solutions have been proposed for many tasks such as clustering [10], top-k processing [2] and classification [5,16].

Anytime algorithms are the natural choice for varying data streams since they flexibly exploit all available time to improve the quality of their result. Recently it has been shown in [12] that also on constant data streams anytime classifiers can improve the classification accuracy over that of traditional budget approaches. With their superiority on varying and constant data streams, applications for anytime classifiers are numerous and range from industrial applications, such as

M.J. Zaki et al. (Eds.): PAKDD 2010, Part II, LNAI 6119, pp. 325–334, 2010.
© Springer-Verlag Berlin Heidelberg 2010

machine monitoring, over sorting tasks to robotics and health applications. [11]. Our focus is on improving the performance of anytime Bayesian classification. Improving the anytime accuracy together with the results from [12] leads to better classification results on both constant and varying data streams.

2 Related Work

Classification aims at determining the class label of unknown objects based on training data. Different classification approaches are discussed in the literature including nearest neighbor classifiers, decision trees or support vector machines. Bayes classifiers constitute a statistical approach that has been successfully used in numerous application domains. Another classification approach is represented by Bayesian classification using kernel density estimation [9]. Especially for huge data sets the estimation error using kernel densities is known to be very low and even asymptotically optimal [3].

Anytime classification is real time classification up to a point of interruption. In addition to high classification accuracy as in traditional classifiers, anytime classifiers have to make best use of the limited time available, and, most notably, they have to be interruptible at any given point in time. This point in time is usually not known in advance and may vary greatly. Anytime classification has for example been discussed for support vector machines [5] or nearest neighbor classification [16].

For Bayesian classification based on kernel densities an anytime algorithm called Bayes tree has been proposed in [15]. The Bayes tree is a balanced tree structure and is basically an extension of the R-tree [8]. It stores in each entry a pointer and a minimum bounding rectangle (MBR) and additionally a cluster feature representing the corresponding subtree. In [15] the tree is constructed trough iterative insertion, i.e. no optimization is performed with respect to overlapping or quality of the resulting mixture densities. In this paper we propose several methods for bulk loading mixture densities in the Bayes tree and show in experimental evaluation that they outperform the results from [15].

3 Bulk Loading Mixture Densities

Before going into detail on the different bulk loading approaches in Sections 3.2 and 3.3 we briefly review Bayesian classification and describe the structure and working of the Bayes tree proposed in [15].

3.1 Bayesian Classification and the Bayes Tree

Given a set of classes C and an object space X a classifier is a function G that assigns the class label $G(x)$ to an object $x \in X$. Based on a statistical model of the distribution of class labels the Bayes classifier assigns to an object x the class c_i with the highest posterior probability $P(c_i|x)$. With Bayes rule it holds:

$$G(x) = \underset{c_i \in C}{argmax}\{P(c_i|x)\} = \underset{c_i \in C}{argmax}\{P(c_i) \cdot p(x|c_i)\}$$

In the Bayes tree the class-conditional density $p(x|c_i)$ is estimated using Gaussian mixture models in the inner nodes and Gaussian kernel estimators at the leaf level. Iterative refinement of mixture components enables anytime kernel density estimation for efficient and interruptible classification. The general idea of the Bayes tree is a hierarchy of mixture densities stored in a multidimensional index. Each level of the tree stores a complete model of the entire data at a different granularity. The node entries consist of pointers to a subtree and a single Gaussian representing the objects in the subtree. All objects stored in the leaves of the Bayes tree are d-dimensional kernels. The mean μ_s and the variance vector σ_s^2 can be computed from the cluster features.

Answering a probability density query uses a complete model which is available at each level of the tree. Besides these full models, the Bayes tree allows for local refinement of the model (to adapt flexibly to the query) and thus provides models composed of coarser and finer representations. The current mixture model components, i.e. their corresponding entries, are stored in a *frontier*. The current entries in the frontier have to represent each stored object exactly once. This is made sure by removing the entry e_s that is refined from the frontier and adding its child entries $e_{soj}, j = 1 \ldots \nu_s$ instead. The probability density for a query object is then calculated with respect to the current frontier.

For tree traversal best first descent using a probabilistic priority measure has proven to yield the best results in [15]. One Bayes tree is built per class, therefore several improvement strategies have been proposed to decide which tree has the right to refine its model in the next time step. Extensive experiments showed that refining the k most probable classes (qbk) in turns yielded the best results throughout. $k = \min\{2, \lfloor \log(m) \rfloor\}$, where m is the number of classes, showed the best performance on all tested data sets. For more details please refer to [15].

3.2 Machine Learning and Statistical Approaches

Our goal in this work is to improve the performance of the Bayes tree. The accuracy of the Bayes tree results is based on the quality of the mixture densities stored in its entries. The iterative insertion performed in [15] does not consider the quality of the resulting Gaussian components. We develop and evaluate several bulk loading approaches that try to overcome this shortcoming and improve the quality of the mixture densities.

Goldberger. Since the Bayes tree is a statistical approach to classification we looked for statistical methods to create a smaller mixture model from a given mixture model. Starting bottom up with a mixture model that contains a kernel estimator for each training set item we create successively coarser models that represent good approximations.

Our first statistical approach is based on [7] and is called Goldberger in the following. The Goldberger approach assumes two initial mixture models f and g to be given, where f is the finer model with r components and g an approximation with s components, hence $r > s$. Each component is assigned a weight and is specified by its mean and covariance matrix. To measure the quality of the approximation [7] defines the distance between two mixture densities as:

Definition 1. *Let* $f = \sum_{i=1}^{r} \alpha_i f_i$ *and* $g = \sum_{j=1}^{s} \beta_j g_j$ *be two mixture densities containing* r *and* s *Gaussian components* f_i *and* g_j *with their respective weights* α_i *and* β_j*. The distance between* f *and* g *is then defined using the Kullback-Leibler divergence* KL *[4] as follows*

$$d(f,g) = \sum_{i=1}^{r} \alpha_i \cdot \min_{j=1}^{s} \{KL(f_i, g_j)\}$$

The optimal model $\hat{g}$ reducing f to s components is $\hat{g} = \arg\min_g(d(f,g))$. Since there is no closed form to compute $\hat{g}$, a local optimum is computed iterating the following two steps until the distance $d(f,g)$ does no longer decrease. Therein $\pi(i) : \{1 \dots r\} \to \{1 \dots s\}$ is a mapping function that assigns each component in f to a component in g.

- Regroup: update π: $\pi(i) = \arg\min_{j=1}^{s} \{KL(f_i, g_j)\}$
- Refit: for each component g_j recompute weight β_j, mean μ_j and covariance matrix Σ_j as follows
  - $\beta_j = \sum_{i,\pi(i)=j} \alpha_i$
  - $\mu_j = \frac{1}{\beta_j} \sum_{i,\pi(i)=j} \alpha_i \mu_i$
  - $\Sigma_j = \frac{1}{\beta_j} \sum_{i,\pi(i)=j} \alpha_i \left(\Sigma_j + (\mu_i - \mu_j)^2 \right)$

We devise a bulk loading technique based on [7] as follows. To initialize the mixture g we compute a first mapping π_0 by assigning $0.75 \cdot M$ components from f to one component in g according to the z-curve order of their mean values. M is given through the fanout, which in turn is dictated by the page size. When no more changes occur in step 2, the resulting components g_j are converted to Bayes tree nodes containing the entries f_i with $\pi(i) = j$. Since the final π might map more than M components from f to a single component in g, we investigated several strategies to restrict the fanout to the given boundaries. First we reformulated the regroup step into an integer linear program with constraints regarding the resulting fanout. However, for realistic problem sizes, this approach took way too long to compute a complete bulk loading. Hence, we decided for a post processing after the mapping π was computed, which splits the nodes that contain too many entries. Therefore two representatives are computed by moving the mean along the dimension a with the highest variance σ_a by an $\epsilon = \sigma_a/2$ in both direction. A Gaussian is placed over the two representatives and the mapping of the entries to the representatives is computed as in the regroup step. If a node contains too few entries it is merged with the node closest to it in terms of the Kullback-Leibler divergence.

Virtual sampling. The second approach, called virtual sampling, uses the work presented in [17] and does not rely on the KL divergence. The virtual sampling approach assumes a given mixture model $f = \sum_{i=1..r} \alpha_i \cdot f_i$ containing r components and computes a coarser mixture model $g = \sum_{j=1..s} \alpha_j \cdot g_j$ with $s < r$ components. The components $f_i = \mathcal{G}(x, \mu_i, \sigma_i)$ constitute multivariate Gaussian normal distributions with their respective weight α_i (analogue for g_j). To

derive an algorithm the following model is utilized: the mixture g can be computed using samples $R_1 \dots R_r$ from each component in f with $R = \cup_{i=1..r} R_i$ and $|R_i| = \alpha_i \cdot |R|$. Assuming independence of the sample points from different components in f yields the assumption that they can be assigned to different components in g while samples from the same f_i are likely to be assigned to the same g_j. Based on this assumption hidden variables z_{ij} are introduced that indicate for each component f_i its assignment to the corresponding g_j. While the z_{ij} are binary during initialization, they can take values between 0 and 1 during the iterations. The hidden variables are used in a modified Expectation Maximization algorithm to compute the coarser mixture g as follows (superscripts ^{f} and ^{g} are added for readability to indicate the origin of the components):

- Expectation: $z_{ij} = \dfrac{\left[\mathcal{G}(\mu_i^f, \mu_j^g, \Sigma_j^g) e^{-\frac{1}{2} trace\{(\Sigma_j^g)^{-1} \Sigma_i^f\}}\right]^{|R_i|} \cdot \alpha_j^g}{\sum_{k=1}^{s} \left[\mathcal{G}(\mu_i^f, \mu_k^g, \Sigma_k^g) e^{-\frac{1}{2} trace\{(\Sigma_k^g)^{-1} \Sigma_i^f\}}\right]^{|R_i|} \cdot \alpha_k^g}$
- Maximization:
  - $\alpha_j^g = \frac{1}{r} \sum_{i=1}^{r} z_{ij} \qquad \mu_j^g = \frac{\sum_{i=1}^{r} z_{ij} |R_i| \mu_i^f}{\sum_{i=1}^{r} z_{ij} |R_i|}$
  - $\Sigma_j^g = \frac{1}{\sum_{i=1}^{r} z_{ij} |R_i|} \left[\sum_{i=1}^{r} z_{ij} |R_i| \Sigma_i^f + \sum_{i=1}^{r} z_{ij} |R_i| \left(\mu_i^f - \mu_j^g\right)^2 \right]$

The above equations are independent of the actual samples R_i and can be computed directly from the mixture components in f, hence *virtual sampling*. To use the described bottom up method for bulk loading we have to provide an initialization for the hidden variables z_{ij}. The initialization of the mixture g is done as in the goldberger approach described above. After getting the final values for z_{ij} from the virtual sampling algorithm, we assign each f_i to that g_j with the maximum z_{ij} for all j. Moreover, the result has to comply with the fanout parameters m and M of the Bayes tree. This is achieved through merging and splitting of the resulting components g_j. If a component g_j is assigned less than m components f_i, these components from f are assigned to the $g_{j'}$ with the second highest $z_{ij'}$. If more than M components f_i are assigned to one g_j, g_j is duplicated while moving the resulting two means in opposite direction along the dimension with the highest variance. The respective f_i are reassigned to the more probable candidate according to the density of their mean μ_i. After merging and splitting the corresponding mixture parameters are adapted following the above equations.

EMTopDown. Besides the above mentioned bottom up approaches we implemented a top down approach that recursively splits the training set into several clusters. In contrast to the previous approach, where Gaussian components were merged and mapped, we now operate solely on the data objects. More precisely, we start by applying the EM [6] algorithm to the complete training set. The desired number M of resulting clusters is always set to the fanout which is again given through the page size. If the EM returns less than m clusters, the biggest resulting cluster is split again such that the total number of resulting clusters is at most M. In the rare case that the EM returns a single cluster, this cluster is

split by picking the two farthest elements and assigning the remaining elements to the closest of the two. Finally, if a resulting cluster contains more than L objects (the capacity of a leaf node), the cluster is recursively split using the procedure described above. Otherwise the items contained in that cluster are stored in a leaf node, its corresponding entry is calculated and returned to build the Bayes tree. The EM approach may result in an unbalanced tree, which differs from the primary Bayes tree idea. However, as we will see in the experimental section, the results show that this is not a drawback but even leads to better anytime classification performance.

3.3 Data Base Driven Approaches

Since the Bayes tree extends the R-tree, we employ traditional R-tree bulk loading algorithms for comparison. We implemented two types of space filling curves, namely Hilbert curve and z-curve. We briefly describe the Hilbert curve approach, the z-curve bulk loading works analogously. The bulk loading according to the Hilbert curve is a bottom up approach where in the first step the Hilbert value for each training set item is calculated. Next the items are ordered according to their Hilbert value and put into leaf nodes w.r.t. the page size. After that the corresponding entry for each resulting node is created, i.e. MBR, cluster features (CF) and the pointer. These steps are repeated using the mean vectors as representatives until all entries fit into one node, the root node. Theory on creating multidimensional Hilbert curves can be found in [1], for implementation guide lines see [13]. Additionally we implemented the partitioning approach presented in [14] that is called sort-tile-recursive. The basic idea is to build a hierarchy of rectangles which have, at the same level of the hierarchy, approximately the same expansion in each dimension. For details please refer to [14].

4 Experiments

The three proposed bulk loading techniques *Goldberger*, *virtual sampling* and *EMTopDown* are compared to the existing R-tree bulk loading approaches *Hilbert*, *z-curve* and *STR* and the previous results from [15] (called *Iterative* in the graphs since it performs iterative insertion of objects). We also used the same settings as in [15], i.e. we use the same data sets, performed 4-fold cross validation and show the classification accuracy after each node averaged over the four folds. We used global best descent and the *qbk* improvement strategy as they showed the best results in [15]. Please note that the bulk loading is done per fold once and offline and the resulting classifier is then used on the data stream. Since our focus is on anytime classification, we do not study the time performance of the bulk loading algorithm but the performance of the resulting classifier, i.e. its anytime classification accuracy.

The top left part of figure 1 shows the results for the pendigits data set. The Goldberger approach fails to improve the accuracy over the iterative insertion for the first 50 nodes. After that it performs slightly better, but cannot increase the accuracy by more than 1%. Virtual sampling performs worst on this data set.

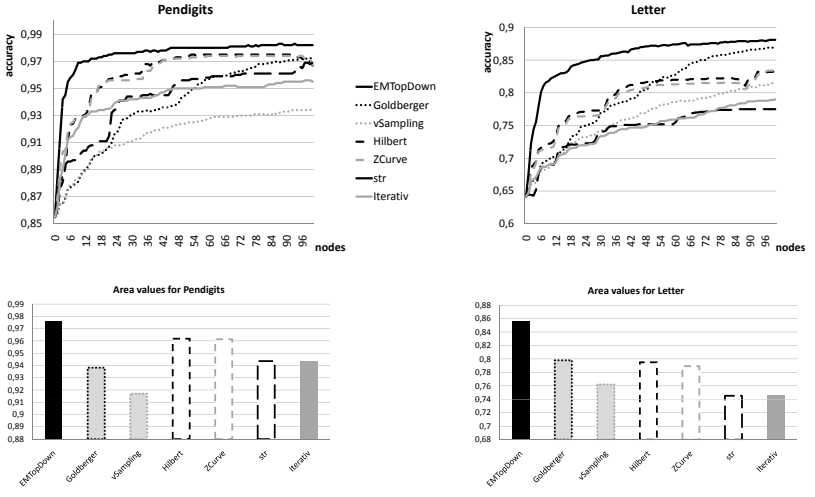

Fig. 1. Anytime classification accuracy and a ranking of all approaches according to the area for Pendigits (left) and Letter (right)

The Hilbert and z-curve bulkload yield comparable results, their corresponding curves show a steep increase similar to the iterative insertion and show better performance in most cases. After falling behind during the first nodes, STR performs equaly well compared to Iterative. The EMTopDown bulkload outperforms all other approaches and improves the accuracy over the iterative insertion constantly by 3% or more on this data set.

The performance of the Goldberger bulkload stayed below the iterative insertion in the majority of our experiments. Just on the Letter data set it improved the accuracy for larger time allowances (cf. Figure 1, right). For the first 40 nodes Goldberger and Iterative perform equally well, after that the accuracy of Iterative stays behind that of Goldberger. While the virtual sampling and STR bulkload shows similar performance to Iterative, Hilbert and z-curve (which are again in close proximity to each other) show constantly better accuracy than Iterative. The EMTopDown again constantly yields the best accuracy up to 13% better than the iterative insertion.

To facilitate an easier comparison between the different approaches we report the values for the normalized area under the anytime curves for Pendigits, Letter, Vowel, USPS and Verbmobil in Figures 1 (bottom) and 2 (top) respectively. Throughout the data sets Hilbert and z-curve show nearly the same performance, while z-curve is usually slightly behind Hilbert except for the USPS data set. STR ranges between these two and the iterative insertion; it is never better than the former and never beaten by the latter. Surprisingly both statistical approaches exhibit the same weakness as STR, i.e. they never outperform the z-curve bulk load (except Goldberger on Letter) and several times show even worse performance than iterative insertion. This is especially interesting since both approaches are initialized using the z-curve, however, only with $0.75 \cdot M$ entries per node (cf. Section 3.2). We discuss the reasons for this shortcoming of

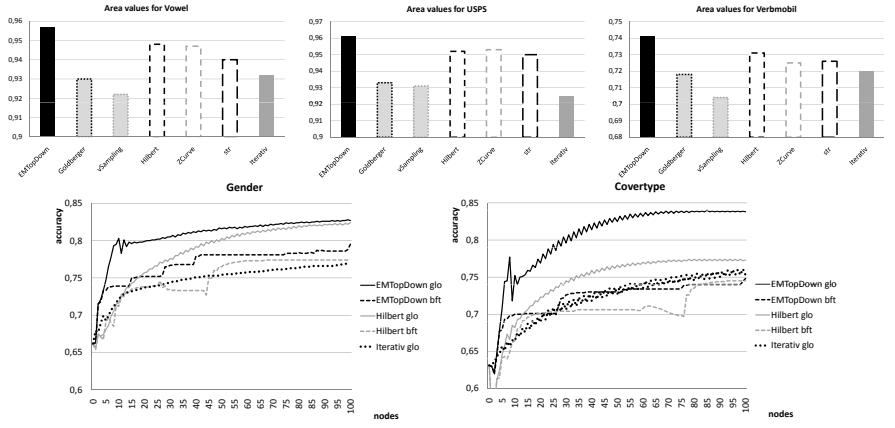

Fig. 2. Top: Comparison of all approaches on Vowel, USPS and Verbmobil data sets. Bottom: Anytime classification accuracy on Gender (left) and Covertype (right).

the statistical approaches at the end of the section where we analyze the structure of the resulting trees. Finally, the EMTopDown bulk load shows constantly the best performance on all data sets despite the unbalanced resulting trees. Again we defer the analysis to the end of the section and first discuss a different issue.

Figure 2 (bottom) shows the results for the gender and covertype data sets. For readability only the results for Hilbert and EMTopDown are shown. For both data sets $k = 2$ for the *qbk* improvement strategy (cf. Section 3.1). The graphs for EMTopDown and Hilbert using the global best descent (*glo*) show an oscillating behavior on both data sets. For comparison we recapitulated the breadth first traversal (*bft*), the results are plotted as well. As was found in [15], the global best descent performs better than breadth first traversal. However, the graphs for bft do not show the oscillating behavior mentioned above. Since $k = 2$, there is obviously a certain percentage of object whose class decision changes in favor of (or against) the tree which is currently refined. More precisely, these objects are likely positioned on the decision boundary between the two most probable classes. In global best descend refining mixture components close to the objects, and hence close to the decision boundary, affects the corresponding posterior probabilities more heavily than refinement of a farther component as in breadth first traversal. If we assume the oscillation to be a higher frequency added to a smooth underlying anytime curve, the percentage of these borderline objects corresponds to the amplitude. However, on balance, the oscillating behavior does not affect the superiority of the bulk loading over the iterative insertion.

To find reasons for the surprising ranking of the individual algorithms, we analyzed the structure of the resulting trees. Since more entries in a node correspond to more detailed information compared to less entries, we looked at the degree to which the nodes were filled in the different approaches. To this end we computed the average fanout per level of the trees from root to leaf. However, the resulting figures did not reveal any correlation to the found ranking of the approaches. For example, both space filling curve approaches always fill the nodes

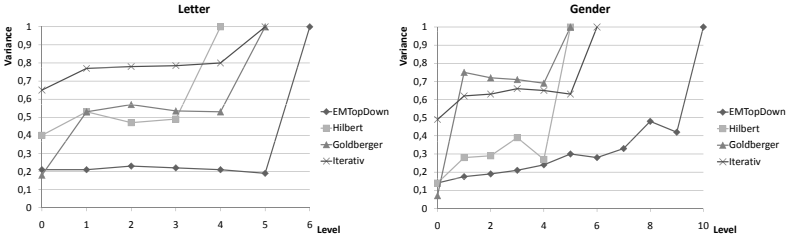

Fig. 3. Variances per level for Letter (left) and Gender (right)

to nearly 100% (except for the last one per level and the root), but the EMTopDown sometimes produces less than M entries for a given node.

We found a correlation between the performance of the algorithm and the variance of the mixture components in the resulting trees. More precisely, we calculated the average variance of all entries per level, Figure 3 shows the resulting numbers for Hilbert, Goldberger, EMTopDown and Iterative on the Letter and Gender data sets. The variances are normalized by the variance of the entire data set per class. Level 0 corresponds to the leaves, Level 1 is above the leaves etc. The trees resulting from different approaches can have different heights as can be seen in the graphs. Hilbert bulk load fills each node to 100% and consequently yields the smallest trees, while the unbalanced trees resulting from EMTopDown are up to twice as high on the Gender data set.

EMTopDown and Hilbert show significantly smaller average variances compared to the iterative insertion. While the corresponding variances for Goldberger are smaller than those of Iterative for the Letter data set they are larger on the Gender data set. This is in line with the observed anytime classification performances. We found comparable similarities between the two measures on the other data sets. The average variances per level achieved by the EMTopDown bulk load were constantly amongst the lowest compared to all other approaches. This explains and underlines the superior performance of EMTopDown.

In general the EMTopDown shows the best results in terms of anytime classification accuracy on all tested data sets and continuously improves the accuracy over that of the previous results in [15] up to 13%. This proves the effectiveness of our bulk loading approach for hierarchical anytime classifiers.

5 Conclusion

We proposed three bulk loading approaches for hierarchical mixture models to improve Bayesian classification on data streams using the Bayes tree. We compared our approaches to the previously proposed iterative insertion [15] and three known R-tree bulk loading algorithms on a range of real world data sets. Experimental results showed that our novel EMTopDown bulk load constantly outperformed all other approaches and improved the accuracy by up to 13%. Surprisingly our two statistical approaches were outperformed by existing R-tree bulk loadings based on space filling curves. Further analysis attributed this

shortcoming to a structural property of the resulting Bayes trees. The results of the analysis were in line with the classification results found in the experiments confirming the superior performance of our new EMTopDown bulk loading in terms of anytime classification accuracy.

Acknowledgments. This work has been supported by the UMIC Research Centre, RWTH Aachen University.

References

1. Alber, J., Niedermeier, R.: On multi-dimensional hilbert indexings. In: 4th Annual International Conference on Computing and Combinatorics COCOON (1998)
2. Arai, B., Das, G., Gunopulos, D., Koudas, N.: Anytime measures for top-k algorithms on exact and fuzzy data sets. VLDB Journal 18(2), 407–427 (2009)
3. Bouckaert, R.: Naive Bayes Classifiers that Perform Well with Continuous Variables. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 1089–1094. Springer, Heidelberg (2004)
4. Chen, J.-Y., Hershey, J., Olsen, P., Yashchin, E.: Accelerated monte carlo for kullback-leibler divergence between gaussian mixture models. In: ICASSP (2008)
5. DeCoste, D.: Anytime interval-valued outputs for kernel machines: Fast support vector machine classification via distance geometry. In: ICML (2002)
6. Dempster, A.P., Laird, N.M.L., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society, Series B 39(1), 1–38 (1977)
7. Goldberger, J., Roweis, S.T.: Hierarchical clustering of a mixture model. In: NIPS (2004)
8. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: SIGMOD, pp. 47–57 (1984)
9. John, G., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: UAI. Morgan Kaufmann, San Francisco (1995)
10. Kranen, P., Assent, I., Baldauf, C., Seidl, T.: Self-adaptive anytime stream clustering. In: Proc. of the 9th IEEE ICDM (2009)
11. Kranen, P., Kensche, D., Kim, S., Zimmermann, N., Müller, E., Quix, C., Li, X., Gries, T., Seidl, T., Jarke, M., Leonhardt, S.: Mobile mining and information management in healthnet scenarios. In: Proc. of the 9th IEEE MDM (2008)
12. Kranen, P., Seidl, T.: Harnessing the strengths of anytime algorithms for constant data streams. DMKD Journal, ECML PKDD Special Issue 2(19) (2009)
13. Lawder, J.: Calculation of mappings between one and n-dimensional values using the hilbert space-filling curves. Technical Report JL1/00 Birkbeck College, University of London (2000)
14. Leutenegger, S.T., Edgington, J.M., Lopez, M.A.: Str: A simple and efficient algorithm for r-tree packing. In: ICDE, pp. 497–506 (1997)
15. Seidl, T., Assent, I., Kranen, P., Krieger, R., Herrmann, J.: Indexing density models for incremental learning and anytime classification on data streams. In: EDBT/ICDT (2009)
16. Ueno, K., Xi, X., Keogh, E.J., Lee, D.-J.: Anytime classification using the nearest neighbor algorithm with applications to stream mining. In: ICDM (2006)
17. Vasconcelos, N., Lippman, A.: Learning mixture hierarchies. In: NIPS (1998)