Sequential Hierarchical Pattern Clustering

Bassam Farran, Amirthalingam Ramanan, and Mahesan Niranjan

School of Electronics and Computer Science, University of Southampton
Southampton, SO17 1BJ, United Kingdom
{bf06r,ar07r,mn}@ecs.soton.ac.uk

Abstract. Clustering is a widely used unsupervised data analysis technique in machine learning. However, a common requirement amongst many existing clustering methods is that all pairwise distances between patterns must be computed in advance. This makes it computationally expensive and difficult to cope with large scale data used in several applications, such as in bioinformatics. In this paper we propose a novel sequential hierarchical clustering technique that initially builds a hierarchical tree from a small fraction of the entire data, while the remaining data is processed sequentially and the tree adapted constructively. Preliminary results using this approach show that the quality of the clusters obtained does not degrade while reducing the computational needs.

Keywords: On-line clustering, Hierarchical clustering, Large scale data, Gene expression.

1 Introduction

Clustering as a tool in pattern analysis has a wide spectrum of applications: mining in large data warehouse environments [1], dynamic routing in optical networks [6], text classification [19] and codebook construction for bag-of-keypoint visual scene analysis problems [20] are examples of this. For pattern recognition in bioinformatics, the popular use of clustering is in gene expression analysis [4], where expression profiles of genes measured across different biological conditions are clustered. Genes that fall into the same cluster may be assumed to have common functional properties, such as acting under control of the same regulatory mechanism, or acting in tandem along a signalling pathway. Another example of clustering in bioinformatics is the analysis of protein sequences to assign putative function [5]. Repositories of protein sequences have seen massive growth in recent years [18].

As the number of sequenced proteins grows at a much faster rate than those whose structure is determined, or function characterised, automatically predicting function by clustering the sequence space is an active topic of interest.

While clustering algorithms and their performance characteristics have been studied extensively over recent years, a particular property of several of the new problems, including the bioinformatics problems, is their massive scale. In other areas, too, data mining examples with a million or more data points are

V. Kadirkamanathan et al. (Eds.): PRIB 2009, LNBI 5780, pp. 79–88, 2009.

becoming available (e.g. KDD Cup[1]). The UniProt database of proteins [18] now consists of over six million sequences. A matrix of pairwise similarity scores of all these proteins has a file size of 2.6GB. At these scales, classical clustering algorithms such as K-means or hierarchical clustering aren't straightforward to apply and a need for novel approaches arises.

Our approach to such very large scale clustering algorithms is to study sequential and constructive algorithms, much in the spirit of the resource allocating network by Platt [14] and its variants by Kadirkamanathan *et al.* [7], and Molina *et al.* [11]. Another online approach for large scale learning ($>$ 4M datapoints) was proposed by Farran *et al.* [2]. In recent work Ramanan *et al.* [15] developed a codebook design strategy for visual object categorization which uses a resource-allocating clustering approach. In this paper we investigate whether a formulation for hierarchical clustering by sequentially processing the data in a one-pass setting can be designed. Computational saving in such an approach will come from not having to evaluate all pairwise similarities between data items. This clearly is not possible for all the data as a measure of the scale of the distribution is required. Thus, the approach we take involves the construction of an initial tree of hierarchical clustering by processing a random subset of the data in batch mode. To a cluster structure formulated in this way, any further data may be sequentially included, adaptively changing the structure of the cluster tree, at a heavily reduced cost of similarity computations.

2 Previous Work

Clustering has been widely applied in bioinformatics in the past years. Hierarchical clustering techniques are useful for representing protein sequence family relationships [8]. Eisen *et al.* [4] applied a variant of the hierarchical average-linkage clustering algorithm to identify groups of co-regulated genome-wide expression patterns. Loewenstein *et al.* [10] applied agglomerative clustering to protein sequences that automatically builds a comprehensive evolutionary driven hierarchy of proteins from sequence alone. Frey *et al.* [5] applied an affinity propagation clustering technique to detect putative exons comprising genes from mouse chromosomes. While they claim lower computational cost in comparison to other algorithms, they do not include the cost of pairwise similarity computations. Since this is the most expensive stage in large scale problems, the claimed advantage is exaggerated. Our focus is on reducing the computational complexity arising from this cost.

El-Sonbaty *et al.* [17] proposed an on-line hierarchical clustering algorithm based on the single-linkage method that finds at each step the nearest k patterns with the arrival of a new pattern and updates the nearest seen k patterns so far. Finally the patterns and the nearest k patterns are sorted to construct the hierarchical dendrogram. While they claim their method is sequential, at the arrival of each data item they compute similarity to *all* the data seen previously. Thus there is little computational saving in their method, and it is equivalent to

[1] http://www.sigkdd.org/kddcup/index.php

re-training a new model at the arrival of new data. Contrast that with a truly sequential algorithm, where it is the model that is adapted, similar in fashion to the Kalman filter.

3 Methodology

Our algorithm for sequentially updating a hierarchical tree is shown as pseudo code in Algorithm 1. We construct an initial hierarchical tree by computing all pairwise similarities between a small subset of the data, and then passing these to the Single-Round-MC-UPGMA algorithm [10].

Following the construction of the initial tree using Single-Round-MC-UPGMA, the remaining data is sequentially processed using Algorithm 1. Whenever a new pattern ($\mathbf{x}_i$) arrives for clustering, its similarity distance d to the *root* of the current hierarchical tree is computed. If d is greater than a predefined threshold (θ), a new root is created having the current pattern ($\mathbf{x}_i$) and the previous root as its children, and as a consequence the depth of the tree increases by one. The value of the new root is assigned with the arithmetic mean of all the leaf nodes. However, if d is less than θ, the nearest child of the current node is retrieved. If the distance of $\mathbf{x}_i$ to this child node is also smaller than θ then we continue to repeat finding the closest child until either the distance to the current node is greater than θ or we reach a leaf node. In either of the two cases, $\mathbf{x}_i$ is created as a sibling to the node under consideration, and $\mathbf{x}_i$'s value is propagated up the tree to update its ancestors.

Algorithm 1. Update the initial hierarchical tree in an online fashion

Input: Root of the initial tree (CurNode), the new pattern (NewNode), and the novelty threshold (θ)
Output: Updated hierarchical tree

```
simdist_CN ← similarity_distance(CurNode, NewNode)
if (simdist_CN ≤ θ) then
  (⋆) Children ← getChildrenOf(CurNode)
  if (Children == NULL) then
    Make NewNode as a sibling of CurNode and update ancestors
  else
    {CurNode has children}
    nearestNode ← min (similarity_distance(Children, NewNode))
    if (nearestNode ≤ thresh) then
      CurNode ← nearestNode
      Go to (⋆)
    else
      Make NewNode as sibling of CurNode and update ancestors
    end if
  end if
else
  Make NewNode as sibling of CurNode by creating a new root
end if
```

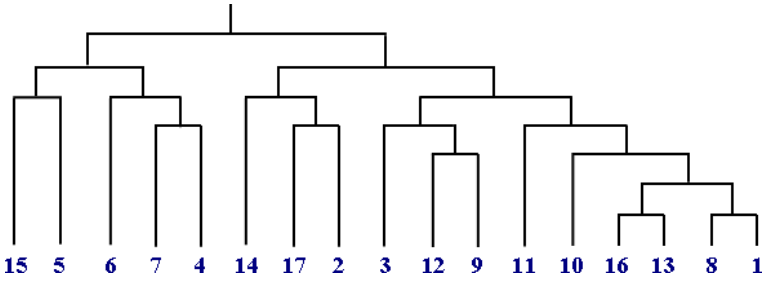

Fig. 1. Hierarchical tree constructed using Single-Round-MC-UPGMA on the entire capitals dataset

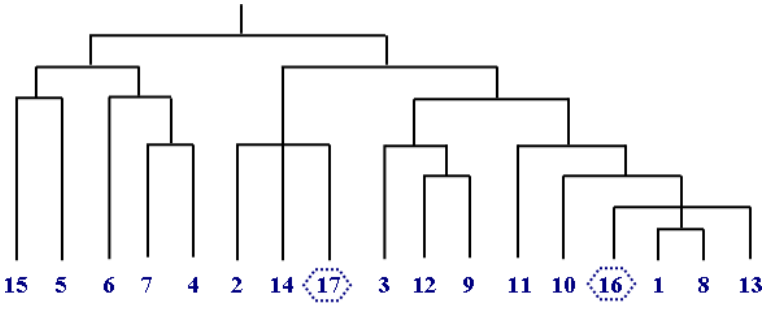

Fig. 2. Hierarchical tree constructed by our approach in an online fashion with the aid of an initial tree constructed by Single-Round-MC-UPGMA. The initial tree was constructed with the first 15 capitals in the dataset. Capitals Stockholm and Washington (depicted in dotted hexagons) were sequentially inserted using Algorithm 1.

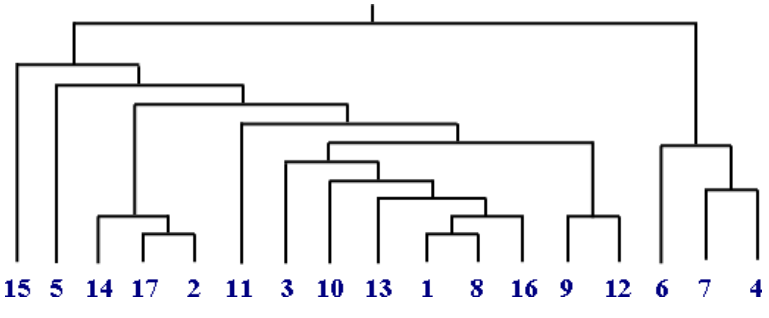

Fig. 3. Hierarchical tree constructed by the approach proposed in [17] on the entire capitals dataset

By adjusting θ, one can obtain different numbers of clusters at different levels of granularity. This threshold is seen as a hyperparameter in the algorithm and can be tuned in different ways. The specific way in which we set this is described in section 4. The leaf nodes of the hierarchical tree are the input patterns, and

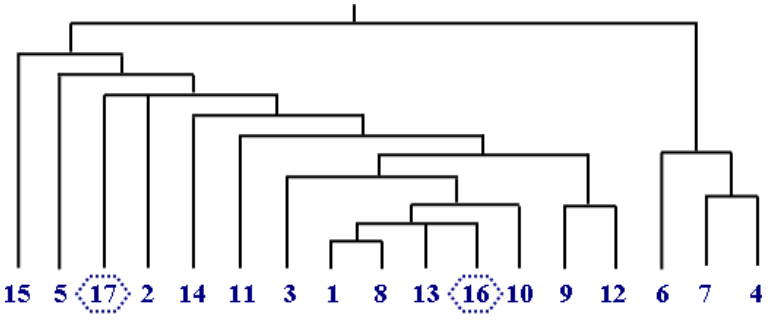

Fig. 4. Hierarchical tree constructed by our approach with the aid of an initial tree constructed using the method in [17] on the capitals dataset

each intermediate node (up to and including the root) contains the arithmetic mean of every leaf node it represents. Because of this, we need not traverse the entire tree during the update process.

We use the capitals dataset[2] for illustration and evaluation purposes of the proposed algorithm, as the structure of the clusters is known. We numbered the capital cities in the following way: Tallinn (1), Beijing (2), Berlin (3), Buenos Aires (4), Cairo (5), Canberra (6), Cape Town (7), Helsinki (8), London (9), Moscow (10), Ottawa (11), Paris (12), Riga (13), Rome (14), Singapore (15), Stockholm (16), Washington (17). In Fig. 1, we show the tree constructed using Single-Round-MC-UPGMA using the enire capitals data set, which is identical to the tree depicted on the data's website. Fig. 2 shows how our approach inserts the last two points into the tree, given that the first 15 points were used for the construction of the initial tree. Cutting at the second level gives us the exact same 4 clusters obtained by cutting the tree in Fig. 1 at the same level. To illustrate the importance of having a good initial tree, we use El-Sonbaty *et al.* [17]'s method on the entire capitals data set (shown in Fig. 3). Fig. 4 shows how Algorithm 1 inserts the last 2 points, given that the first 15 points were used for the initial tree using El-Sonbaty *et al.*'s method. Even though Algorithm 1 inserted the last 2 points correctly with respect to Fig. 3, the same 4 clusters obtained by the trees in Fig.1 and Fig.2 are not attainable due to the incorrect initial tree.

4 Experiments and Results

Evaluation of clustering algorithms is not straightforward. Unlike in supervised learning problems, there is no ground truth information available to verify whether the clusters obtained are correct or not. To evaluate our algorithm we used two bioinformatic datasets. The first is Eisen *et al.* [4]'s gene expression clusters consisting of ten clusters formed by their clustering algorithm. We make

[2] http://www.quretec.com/HappieClust/

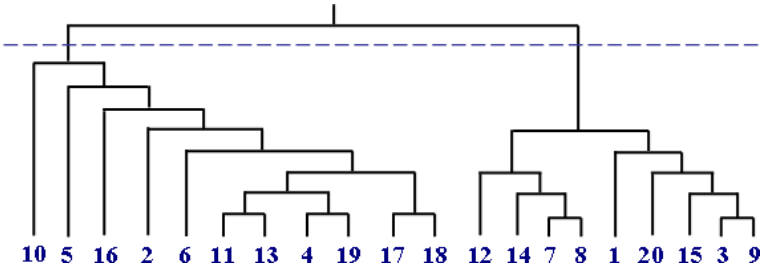

Fig. 5. Hierarchical tree constructed by the Single-Round-MC-UPGMA scheme on Eisen's data clusters labelled as B and D [4]. The tree was constructed by using the whole data of the selected two clusters. The dendrogram was cut at the root node (shown in dotted lines) to obtain two clusters.

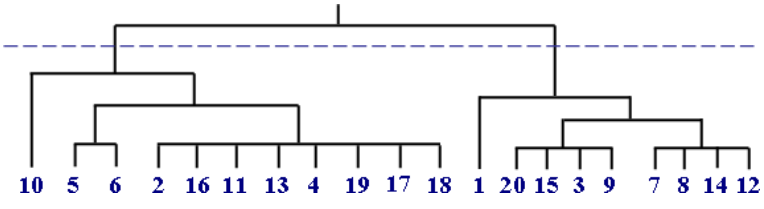

Fig. 6. Tree constructed by the proposed approach with the aid of an initial tree constructed by Single-Round-MC-UPGMA on Eisen's clusters labelled as B and D [4]. The initial tree was constructed with 20% of the data and the rest was clustered using Algorithm 1. The dendrogram was cut at the root node (shown in dotted lines) to obtain two clusters.

the weak assumption that the clusters published by these authors are perfect associations and quantify how close our approach gets to this solution. The second dataset is from a protein fold classification problem, constructed by Ding & Dubchak [3] on a subset of the Structural Classification of Proteins (SCOP) [12] database. This is essentially a classification problem, but we apply clustering to it (without using the class labels) and evaluate how well the resulting cluster membership matches the clusters returned by Single-Round-MC-UPGMA applied on the entire subset. We combined both training and testing sets provided in [3].

To quantify the quality of clusters, we use the F1 measure, widely used in information retrieval literature, and the results given in Table 1 are the average of the 10 runs where we randomised the initial subset and the order of presentation of the remaining data.

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

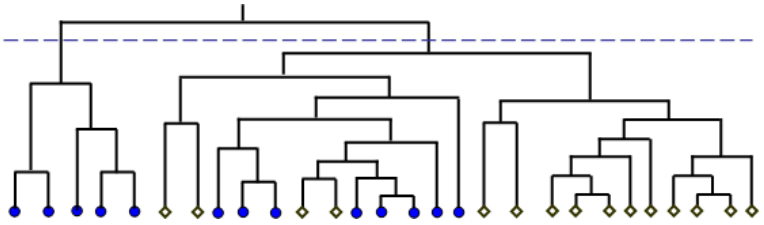

Fig. 7. Tree constructed by Single-Round-MC-UPGMA on the two selected folds Alpha: four-helical up-and-down bundle (depicted as filled circles) and Beta: ConA-like lectins (depicted as diamonds) of the SCOP data subset [3]. We used the whole 28 data points for the construction of this tree.

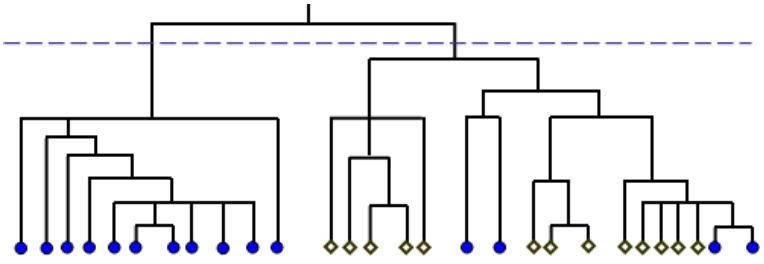

Fig. 8. Tree constructed by the proposed approach with the aid of an initial tree constructed by Single-Round-MC-UPGMA on the two selected folds Alpha: four-helical up-and-down bundle (depicted as filled circles) and Beta: ConA-like lectins (depicted as diamonds) of the SCOP data subset [3]. We used 20 out of 28 data for the construction of initial tree and used the rest in an online manner.

Table 1. Preliminary results of the hierarchical clustering performed on a subset of SCOP and Eisen's data. (1) Beta: ConA-like lectins, (2) Alpha: Four-helical up-and-down bundle, (3) Beta: Immunoglobulin-like beta-sandwich, and (4) A/B: beta/alpha (TIM)-barrel.

Dataset	No. of data	No. of data used or the initial tree	F1-measure
SCOP (1) & (2)	28	20	1.0
SCOP (3) & (4)	151	100	0.9921 ± 0.0086
Eisen (B & D)	20	4	1.0
Eisen (C & I)	104	26	1.0
Eisen (C, B & I)	113	25	1.0
Eisen (C, B, D & I)	124	30	0.9247 ± 0.0652

The threshold θ used in this paper was determined from the data sample used to construct the initial tree. θ was set as the sum of the pairwise Euclidean distances between the patterns in this data sample.

Fig. 9. Tree constructed by our approach on the selected clusters of Eisen's data (a) Clusters B, C, D and I; 30 out of 124 data were used for the construction of the initial tree. The tree was cut at the level indicated by the dotted line to yield four perfect clusters. (b) Clusters B, C, and I; 25 out of 113 data were used for the construction of the intial tree. The tree was cut at the level indicated by dotted line to yield three perfect clusters.

Fig. 5 shows the tree constructed by the Single-Round-MC-UPGMA scheme on the Eisen's clusters labelled as B and D in [4]. Fig. 6 shows the tree obtained by using 4 points for the initial tree, and the remaining 16 points inserted using our approach. Cutting at depth 1 to obtain 2 clusters shows that we get the exact same 2 clusters as in Fig. 5. However when we used our approach on the protein fold data of Ding & Dubchak [3] (see Fig. 8), we obtained a better separation than when the entire data was used with Single-Round-MC-UPGMA (see Fig. 7). Finally, Fig. 9 shows the trees obtained by using Eisen's data clusters (C,B,I and C,B,D, & I), and the cuts that return the desired clusters.

5 Dealing with Categorical or Symbolic Patterns

In the previous section, the illustration of the proposed algorithm was restricted to the Euclidean space. Here, we discuss the capability of our algorithm to handle categorical or symbolic patterns. Instead of updating the parent nodes using the arithmetic means of their leaf nodes, the most informative child node can be selected to act as a parent. However, for simplicity in this paper, we choose a random child to act as the parent node in Algorithm 1. Also, the measure of similarity used depends on the application of interest, and is not limited to numerical data. For example, if we are clustering protein sequences, then Smith-Waterman local alignment [16] or Needleman-Wunsch global alignment [13] measures can be used instead of Euclidean distances.

For immediate comparison purposes with the previously shown cluster trees in section 3 (Capitals data and the selected subsets of Eisen's data), we used the Euclidean distances with the selective node approach. Our preliminary results gave us exactly the same clusters as in the Euclidean case, and hence the same F1-measure.

6 Conclusion and Future Work

In this paper we present an algorithm for on-line hierarchical clustering. The approach depends on the construction of an initial clustering stage using a random subset of the data. This establishes a scale structure of the input space. Subsequent data can be processed sequentially and the tree adapted constructively. We have shown that on small bioinformatics problems such an approach does not degrade the quality of the clusters obtained while saving computational cost.

The proposed technique could be significantly improved with an appropriate choice of the novelty threshold (θ). θ can be better estimated by taking into account the inter-cluster and/or intra-cluster information of the initial tree. This can be subsequently updated after the insertion of a newly arrived pattern. Another way of better estimating θ might be to use local thresholds associated with each parent or level of the tree, instead of a global threshold. The greatest benefit of the proposed technique lies in its application on very-large datasets such as UniRef90 proteins [18].

Acknowledgments. We thank the anonymous reviewers for their useful comments on this paper. AR is partially supported by a grant from the School of Electronics and Computer Science, University of Southampton, United Kingdom, and University of Jaffna, Sri Lanka, under the IRQUE Project funded by the World Bank.

References

1. Achtert, E., Bohm, C., Kriegel, H.-P., Kröger, P.: Online Hierarchical Clustering in a Data Warehouse Environment Data Mining. In: Proceedings of the Fifth IEEE International Conference on Data Mining, pp. 10–17 (2005)
2. Farran, B., Saunders, C.: Voted Spheres: An online Fast Approach to Large Scale Learning. In: IEEE International Symposium on Mining and Web (2009)
3. Ding, C.H., Dubchak, I.: Multi-class protein fold recognition using support vector machines and neural networks. Bioinformatics 17(4), 349–358 (2001)
4. Eisen, M.B., Spellman, P.T., Brown, P.O., Botstein, D.: Cluster analysis and display of genome-wide expression patterns. Proceedings of the National Academy of Sciences USA 95(25), 14863–14868 (1998)
5. Frey, B.J., Dueck, D.: Clustering by Passing Messages between Data Points. Science AAAS 315, 972–976 (2007)
6. Hasan, M., Jue, J.: Online Clustering for Hierarchical WDM Networks. In: IEEE/OSA Conference on Optical Fiber Communication, San Diego, CA, pp. 1–3 (2008)
7. Kadirkamanathan, V., Niranjan, M.: A Function Estimation Approach to Sequential Learning with Neural Networks. Neural Computation 5, 954–975 (1993)
8. Kaplan, N., Friedlich, M., Fromer, M., Linial, M.: A functional hierarchical organization of the protein sequence space. BMC Bioinformatics 5, 196 (2004)
9. Kull, M., Vilo, J.: Fast approximate hierarchical clustering using similarity heuristics. BioData Mining 1, 9 (2008)

10. Loewenstein, Y., Portugaly, E., Fromer, M., Linial, M.: Efficient algorithms for accurate hierarchical clustering of huge datasets: tackling the entire protein space. Bioinformatics 24(13), 41–49 (2008)
11. Molina, C., Niranjan, M.: Pruning with replacement on limited resource allocating networks by f-projections. Neural Computation 8(4), 855–868 (1996)
12. Lo Conte, L., Ailey, B., Hubbard, T.J., Brenner, S.E., Murzin, A.G., Chothia, C.: SCOP: a Structural Classification Of Proteins database. Nucleic Acids Research 28(1), 257–259 (2000)
13. Needleman, S.B., Wunsch, C.D.: A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of two Proteins. Journal of Molecular Biology 48(3), 443–453 (1970)
14. Platt, J.C.: A Resource-Allocating Network for Function Interpolation. Neural Computation 3, 213–225 (1991)
15. Ramanan, A., Niranjan, M.: Designing a Resource-Allocating Discriminant Codebook for Visual Object Recognition. Neural Computation (2009) (under review)
16. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. Journal of Molecular Biology 147, 195–197 (1981)
17. El-Sonbaty, Y., Ismail, M.A.: On-line hierarchical clustering. Pattern Recognition Letters 19, 1285–1291 (1998)
18. Wu, C.H., Apweiler, R., Bairoch, A., Natale, D.A., Barker, W.C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., et al.: The Universal Protein Resource (UniProt): an expanding universe of protein information. Nucleic Acids Research 34, D187–D191 (2006)
19. Zhao, Y., Karypis, G., Fayyad, U.: Hierarchical Clustering Algorithms for Document Datasets. Data Mining and Knowledge Discovery 10(2), 141–168 (2005)
20. Zhang, J., Marszalek, M., Lazebnik, S., Schmid, C.: Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. International Journal of Computer Vision 73, 213–238 (2007)