ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

An algorithm for fast and reliable ESOM learning

Mario Nöcker, Fabian Mörchen, Alfred Ultsch

Data Bionics Research Group
Phillips-University Marburg
35037 Marburg, Germany
noeckerm,fabian,ultsch@informatik.uni-marburg.de

Abstract. The training of Emergent Self-organizing Maps (*ESOM*) with large datasets can be a computationally demanding task. Batch learning may be used to speed up training. It is demonstrated here, however, that the representation of clusters in the data space on maps trained with batch learning is poor compared to sequential training. This effect occurs even for very clear cluster structures. The k-batch learning algorithm is preferrable, because it creates the same quality of representation as sequential learning but maintains important properties of batch learning that can be exploited for speedup.

1 Introduction

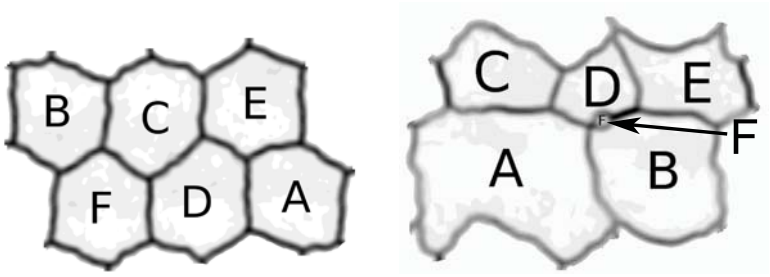
Emergent Self-organizing Maps (ESOM) with U-Matrix visualizations [1, 2] offer a high-resolution view of the distance structures in complex datasets. High-dimensional data is projected in a self-organizing process onto a low dimensional grid of neurons. For ESOM, groups of neurons in valleys on the U-Matrix correspond to clusters while mountain ridges point to cluster boundaries.

There are two well-known learning algorithms for SOM [3]. In sequential learning, the best matching unit (BMU) for a data point is searched and the map is adapted immediately. In batch learning the updates are deferred to the end of a leaning epoch (i.e. the presentation of the whole dataset). For all neurons with more than one data point assigned, the mean vector is used for training. Batch learning is used in particular for large datasets to speed up computing time [4] as less updates of the map are performed and other optimizations can be applied [5, 6].

In this work we demonstrate, however, that the results of these learning procedures are not the same (Section 3). The representation of clusters on batch trained ESOM can be very poor compared to the sequential trained ESOM even for simple examples. To achieve both good speedup capabilities and a faithful representation of the data a combination of both training methods is presented (Section 4) and analyzed (Section 5).

2 Experiments

We used two datasets in our experiments. The first dataset is called *hexa* and consists of 6 clearly separated Gaussian clusters in 3 dimensions with the same



(a) Sequential learning (b) Batch learning

Fig. 1: Example U-maps for hexa dataset.

variance and 1000 points per cluster. The dataset *skating* is obtained from experiments in sports medicine. It contains 29,900 points in 6 dimensions with three overlapping clusters.

The Databionics ESOM Tools [2][1] were used for training and visualization of the maps. We used rectangular toroid maps of sizes 110×70 to avoid topology errors [7]. The learning radius and rate were linearly cooled down from 24 to 1 and 1.0 to 0.1, respectively. The map was randomly initialized with a normal distribution of same mean and variance as the data. The maps were trained for 25 epochs on an Athlon 3000+ processor with 512 MB RAM. The final maps are visualized with U-Maps [8].

To analyze each map we define the quality measure *map space*. The map space of one class, is the number of neurons within the voronoi cell of a class. It represents the space on the map occupied by a class.

3 Batch vs. sequential learning

Figure 1 shows U-maps for the hexa dataset trained with each algorithm. On the sequentially trained map in Figure 1(a) every class of the dataset occupies about the same space on the map, corresponding to the cluster structure of this dataset. In Figure 1(b), however, the space occupied by the six classes differs significantly. Class F in particular is hardly visible in Figure 1(b) while class A is much larger, even though they are of the same size and density. The known structure of the dataset is not well represented. The boxplots in Figure 2 summarize the distribution of map space values of all classes at the end of 25 training runs. While sequential learning shows a very low variation around the expected value of $\frac{1}{6} = 16.7\%$, the map spaces differ widely in the results of batch training.

To further analyze this phenomenon we measured the map space of each class during the training. Figure 3 shows the development of the map space per class. For sequential learning the number of neurons per class quickly stabilized

[1] http://databionic-esom.sf.net

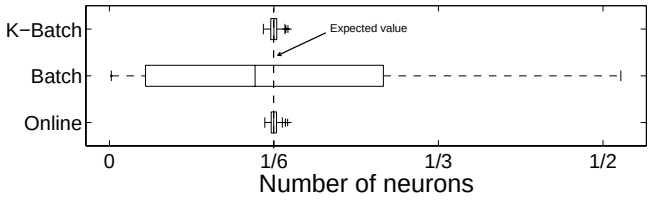
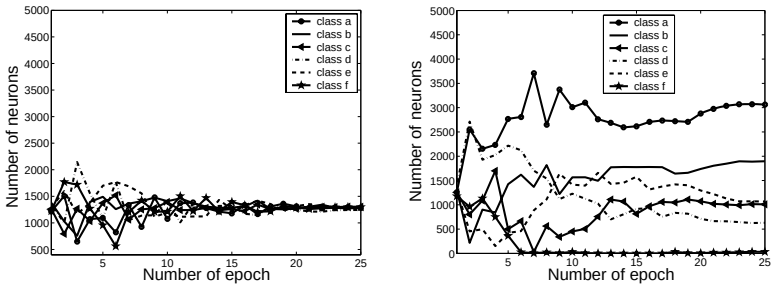


Fig. 2: Boxplots of neurons per class for hexa dataset from 25 training runs each.



(a) Sequential learning (b) Batch learning

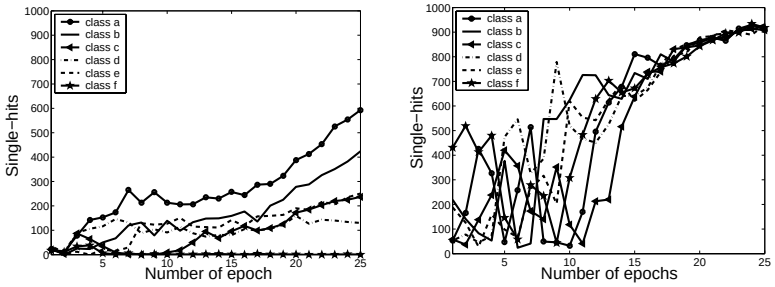
Fig. 3: Number of neurons per class during training of hexa dataset.

after a few epochs to about the same percentage of the map. For batch learning all classes start out with about the same map space, but after one epoch there are already large differences. Classes that have more space in early epochs tend to grow, while classes with fewer map space are represented with decreasing amounts of neurons.

There are two differences between the algorithms - the late updating and the averaging of data vectors. Experiments activating only one of these changes indicated that the effect is caused by the averaging of data points prior to an update. We further measured the number of data points, that do not have to share their BMU with others during batch learning. These so-called *single-hits* are shown in Figure 4(a) for the same batch training run shown in Figure 3(b). A clear correspondence between the development of single hits and map space per class is visible. This makes sense, because a class that occupies more map space will have more single hits. While this effect is initially caused by the random initialization, it is magnified by the batch training over several training epochs.

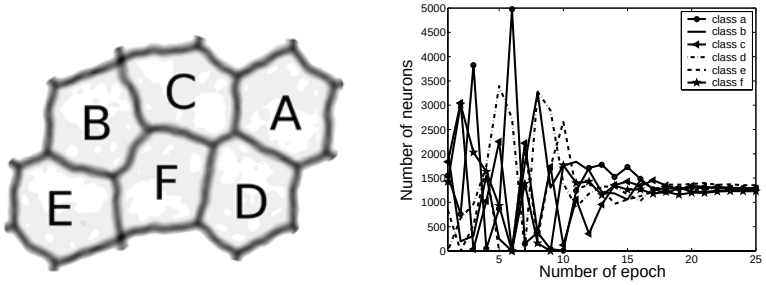
4 k-batch learning algorithm

To achieve both the good representation of a sequentially trained ESOM and the speedup by batch learning we combined the approaches. The k-batch training



(a) Batch learning (b) k-batch learning

Fig. 4: Single-Hits during one training of hexa dataset.



(a) U-Map (b) Number of neurons per class during one training.

Fig. 5: k-Batch training with hexa dataset.

algorithm includes delayed and averaged updates. In contrast to batch learning an update is applied after processing $k \ll n$ of n data points instead of only once per epoch. This actually corresponds to the original batch definition [3], but in practice almost always $k = n$ is chosen leading to the negative effects demonstrated above. We propose to make k much smaller. Our experiments show that in large datasets 15% of all training samples is a good choice for k.

Figure 5(a) shows the map of k-batch learning on the hexa dataset with $k = 500$, i.e. 12 map updates per training epoch with the desired equal representation of classes. The results from 25 training runs summarized in the rightmost boxplot of Figure 2 are almost as consistent as for sequential learning. The map space (Figure 5(b)) and single-hits (Figure 4(b)) per class during k-batch training show large variations for the first 15 epochs. In contrast to batch learning, the effects are not magnified, however. Classes that have many single-hits in one epoch can have few in the next. After about 15 training epochs the results stabilized to the same number of single-hits and the same map space per class.

5 Speedup

The main benefit of running batch learning instead of sequential learning should be faster learning. The introduced k-batch learning algorithm avoids the problems of batch learning at the cost of a higher computational complexity, as map updates are performed more frequently. We have measured the speed of both traditional and the new algorithm on the two datasets. The mean values and standard deviations of 25 runs are shown in Table 1.

	training time				**updates**	
	hexa		**skating**		**hexa**	**skating**
sequential	497.78±3.8	100.0%	2768.0±8.5	100.0%	150000	747500
k-batch	486.71±2.9	97.7%	2665.6±12.7	96.3%	300	1500
batch	466.82±6.0	93.7%	2578.3±76.5	93.1%	25	25

Table 1: Training time and number of updates for all algorithms and datasets.

We can see that k-batch achieves about half of the acceleration of the batch algorithm on these datasets. The large standard deviations of batch show its higher dependency on the number of single-hits over several training runs. It should be noted, however, that no algorithm was able to significantly outperform sequential learning of the large ESOM. This indicates that the search for BMU and not the updates of the map dominates the cost during training. This search could be speeded up using search trees [5] or other indexing methods [9]. The number of updates should be minimized in this case, as every update of the map also involves an update of the search tree. Few updates are also of advantage for fast local search methods [6]. We therefore also compared the number of updates performed by each algorithms shown in Table 1. The k-batch algorithm is a promising candidate for applying such advanced speedup techniques, because it has much fewer updates than sequential learning without showing the negative effects of batch learning.

6 Discussion

Using the batch learning for ESOM we have made two important observations. First, batch learning qualitatively differs from sequential learning and produces undesired results. Second, for large ESOM and datasets the speedup for batch vs. sequential learning is less that 10%. We observed the cost of BMU search, that is identical for all analyzed algorithms, to be dominating taking up about 70% of the complete training time. The small absolute benefits of (k-)batch can further be explained by the high resolution of the emergent maps where only few dataitems are projected on the same neuron. We are not aware of better speedup results for the batch algorithm. In [4] a speedup factor of 10 is reported, but many other optimizations techniques were additionally applied to achieve this result. Distortion of class sizes during batch training has also been observed in [10] but was neither quantified nor compensated.

7 Summary

We have demonstrated the batch learning algorithm to show undesired effects for ESOM even on very simple datasets. Clusters of equal size and density are shown severely distorted on the map. Whenever speed is not an issue, sequential learning should be used in order to avoid this mis-representation. The k-batch algorithm is proposed to avoid the disadvantages of batch learning while preserving some of the speedup and the applicability of accelerated best match search.

Acknowledgement: We thank Ingo Löhken for suggesting the k-batch algorithm.

References

[1] A. Ultsch. Self-Organizing Neural Networks for Visualization and Classification. In *Proc. GfKl, Dortmund, Germany*, pages 307–313, 1992.

[2] A. Ultsch and F. Mörchen. ESOM-Maps: tools for clustering, visualization, and classification with Emergent SOM. Technical Report 46, Dept. CS, University of Marburg, Germany, 2005.

[3] T. Kohonen. *Self-Organizing Maps*. Springer, 1995.

[4] T. Kohonen, S. Kaski, K. Lagus, J. Salojrvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. *IEEE Trans. on Neural Networks*, 11(3):574–585, 2000.

[5] E. Cuadros-Vargas, R.A.F. Romero, and K. Obermayer. Speeding up algorithms of SOM family for large and high dimensional databases. In *Proc. WSOM*, pages 167–172, 2003.

[6] M. Kinouchi and K. Yoshihiro. Much faster Learning algorithm for Batch-Learning SOM and its application to Bioinformatics. In *Proc. WSOM*, pages 107–111, 2003.

[7] A. Ultsch and L. Herrmann. The architecture of emergent self-organizing maps to reduce projection errors. In *Proc. ESANN*, pages 1–6, 2005.

[8] A. Ultsch. Maps for the Visualization of high dimensional Data Spaces. In *Proc. WSOM*, pages 225–230, 2003.

[9] T. Bozkaya and M. Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. In *Proc. SIGMOD*, pages 357–368, 1997.

[10] J.-C. Fort, P. Letremy, and M. Cottrell. Advantages and drawbacks of the batch kohonen algorithm. In *Proc. ESANN*, 2002.