Advances in Engineering Software 88 (2015) 1–7

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Extending parallelization of the self-organizing map by combining data and network partitioned methods

Trevor Richardson[a,*], Eliot Winer[b]

[a] Department of Computer Engineering, Human Computer Interaction, Virtual Reality Applications Center, 1620 Howe Hall, Iowa State University, Ames, IA 50011, USA
[b] Department of Mechanical Engineering, Computer Engineering, Human Computer Interaction, Virtual Reality Applications Center, 1620 Howe Hall, Iowa State University, Ames, IA 50011, USA

ARTICLE INFO

Article history:
Received 16 February 2015
Received in revised form 16 April 2015
Accepted 5 May 2015
Available online 4 June 2015

Keywords:
GPU
Self-organizing map
Parallel computing
Neural network
Data visualization
High-dimensional data

ABSTRACT

High-dimensional data is pervasive in many fields such as engineering, geospatial, and medical. It is a constant challenge to build tools that help people in these fields understand the underlying complexities of their data. Many techniques perform dimensionality reduction or other “compression” to show views of data in either two or three dimensions, leaving the data analyst to infer relationships with remaining independent and dependent variables. Contextual self-organizing maps offer a way to represent and interact with all dimensions of a data set simultaneously. However, computational times needed to generate these representations limit their feasibility to realistic industry settings. Batch self-organizing maps provide a data-independent method that allows the training process to be parallelized and therefore sped up, saving time and money involved in processing data prior to analysis. This research parallelizes the batch self-organizing map by combining network partitioning and data partitioning methods with CUDA on the graphical processing unit to achieve significant training time reductions. Reductions in training times of up to twenty-five times were found while using map sizes where other implementations have shown weakness. The reduced training times open up the contextual self-organizing map as viable option for engineering data visualization.

1. Introduction

Many problems facing industry today are investigated using data acquired through a number of mediums (e.g., sensor recordings or simulation results). Data is becoming cheaper to acquire while at the same time becoming more accurate. Leveraging available data is increasingly important for remaining on industry’s cutting edge. One problem continuing to emerge is the shear amount and corresponding dimensional complexity of data being investigated. No longer can basic plotting methods (e.g., orthogonal plots, scatterplots, etc.) alone be relied upon to understand data characteristics. Researchers have realized this for a number of years and have developed many techniques in an effort to visualize growing levels of data complexity [1–4]. A number of current techniques map *n*-dimensional data sets into a two or three-dimensional representation that is directly interpretable by human visual perception. If, for example, a seven variable data set is to be visualized, it is common to set four or five of the variables to constant values and then plot the remaining two or three dimensions using a traditional graphing technique. This can be carried out multiple times using permutations of the dimensions held constant in an attempt to understand the complex relationship therein. This becomes impossible to understand as the number of permutations increase, leading to the necessity of new or further developed techniques.

Much research has been conducted on multidimensional data representations attempting to present pertinent data to a user [5]. Methods like parallel coordinates [6], graph morphing [7], and linking multiple visuals [1,8] lessen mental load on an investigator by limiting the visual complexity displayed in a single view. Each of the noted methods, however, remain difficult to interpret for high levels of dimensionality. Pixel-based [9] techniques approach the problem from a different direction by showing massive amounts of data while limiting focus only to overall data trends. In many cases a single choice from prior developed methods is not fully effective on its own. This has led to the development of tools that integrate multiple visualization techniques [3] into a suite of tools. Each of the methods noted, whether alone or combined, provide insights into the data under investigation but leave a large part of the complexity remaining for the investigator to make inferences from.

* Corresponding author. Tel.: +1 (515) 294 3092.
E-mail address: trevorr@iastate.edu (T. Richardson).

http://dx.doi.org/10.1016/j.advengsoft.2015.05.003
0965-9978/© 2015 Elsevier Ltd. All rights reserved.

Kohonen's Self-Organizing Map (SOM) [10] allows for low-dimensional visualization of a high-dimensional space while the data's topology is preserved. It has been used extensively to avoid many of the issues noted with modern data visualization methods [11–14]. The Contextual SOM (CSOM) supplements the standard SOM with the addition of a contextual label on the individual nodes of the resulting SOM. Prior work showed that the added contextual information allows an investigator to identify characteristics of the data set that are otherwise extremely difficult to find [15].

However, the original SOM method is a serial method and trains the map one data point at a time. It can take hours to days for a single 100,000 data set to train on a modern desktop computer. This limitation has led to the development of a batch processing method of training SOMs called batch-SOM [16]. The batch method allows for faster convergence with large data sets by allowing the independent processing of each data point in parallel. Prior literature [17–19] has shown training time reductions using the batch-SOM in combination with the two parallelization techniques commonly used to breakup the SOM training process: network partitioning and data partitioning. These methods alone are useful and can reduce training times, but are limited to certain cases as will be described in the remaining sections. A new implementation of the batch-SOM is developed in this work by combining network partitioning and data partitioning and further parallelizing the network partitioning method traditionally used.

1.1. Traditional self-organizing map

In 1982, Tuevo Kohonen modeled the human brain's learning processes in the cerebral cortex using an artificial neural network [10]. The SOM uses an unsupervised learning strategy to train a lattice of neurons. Determining the structure of the original neuron lattice is done in a heuristic fashion by the investigator based on data set size and dimensionality, for example. Each neuron i in the lattice has its own weight vector w as shown in Eq. (1) with the same dimensionality as each data point x as shown in Eq. (2). This structure allows the SOM to scale to any dimensionality k as shown in Eqs. (1) and (2).

$$w_i = \langle w_{i1}, w_{i2}, \ldots, w_{ik} \rangle \tag{1}$$

$$x = \langle x_1, x_2, \ldots, x_k \rangle \tag{2}$$

SOM training involves two phases, ordering and convergence, each containing many iterations through the data. The number of iterations for the method to run is decided upon by the investigator. To begin an iteration, a data point is randomly selected from the data set and is compared against each node in the map using the Euclidean distance metric shown in Eq. (3). The neuron found with the smallest Euclidean distance is determined to be the winner or "activated" neuron.

$$Distance = \sqrt{(x_1 - w_{i1})^2 + (x_2 - w_{i2})^2 + \ldots + (x_k - w_{ik})^2)} \tag{3}$$

With the winning node decided, the weight vector w of each node in the surrounding neighborhood h of the map is "influenced" using Eq. (4) to have its values become more like the current data point x. The influence's magnitude depends on the neuron's position, r_i, relative to the winning node's position, r_j, in the map as well as the current number of training iterations n that have elapsed. η is a time-varying learning rate and defines the amount of the influence on neighboring nodes. The neighborhood influence h is a Gaussian-based influence factor that effects nodes closer to the winner more than those farther away. Learning rate and neighborhood influence are shown in Eqs. (5)–(7) respectively.

$$w_i(n+1) = w_i(n) + \eta(n)h_{j,i}(n)(x(n) - w_i(n)) \tag{4}$$

$$\eta(n) = \eta_0 * exp^{-n/\lambda} \tag{5}$$

$$\sigma(n) = \sigma_0 * exp^{-n/\lambda} \tag{6}$$

$$h_{j,i}(n) = \exp\left(\frac{-\|r_i - r_j\|^2}{\sigma(n)^2}\right) \tag{7}$$

Applying a label to each node of the map is referred to as the Contextual Self-organizing Map (CSOM). When using the CSOM, data points are passed into the SOM a final time to determine the closest node, again using Euclidean distance. In the contextual phase, the data point's label is added to the activated neuron instead of updating the neighborhood. Fig. 1 [20] shows an example from Haykin that trained a CSOM using animal attributes. The animal's attributes make up the training data and the contextual label is the animal's name. The trained map shows groups of animals with similar attributes. Zebras, horses, and cows were grouped together by the training. This group is noted by Haykin as peaceful, four-legged large mammals.

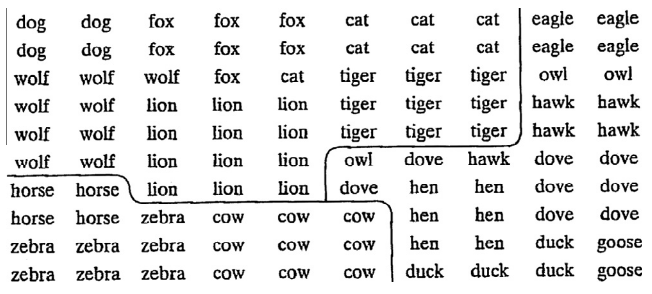
1.2. Batch-SOM

Using the traditional SOM as described above, the map node weights are updated after every data point. With the batch-SOM, however, all data points are first evaluated (for their winner) before updating the map [16]. Two types of parallelization become possible using the batch formulation. First, because node updates only occur once per iteration (as opposed to per data point), all winning node calculations can be performed in parallel. Secondly, the map itself can be parallelized because each node is only required to perform a summation of the influence of all other nodes. The formulation can be thought of as similar to a weighted average being performed across the map. Eq. (8) shows the batch-SOM equation for updating the weight vectors of each node following a single map iteration where t_0 and t_f represent the start and finish of the present iteration, respectively.

$$w_i(t_f) = \frac{\sum_{t'=t_0}^{t'=t_f} h_{ji}(t')x(t')}{\sum_{t'=t_0}^{t'=t_f} h_{ji}(t')} \tag{8}$$

1.3. Multi-core processing

The standard personal computer today often has between two and eight cores that make up the CPU. The drive behind an increasing number of cores has been due to the hardware venders beginning to reach the limits of how much performance can be achieved from each core. Multi-core processing has thus become the



dog	dog	fox	fox	fox	cat	cat	cat	eagle	eagle
dog	dog	fox	fox	fox	cat	cat	cat	eagle	eagle
wolf	wolf	wolf	fox	cat	tiger	tiger	tiger	owl	owl
wolf	wolf	lion	lion	lion	tiger	tiger	tiger	hawk	hawk
wolf	wolf	lion	lion	lion	tiger	tiger	tiger	hawk	hawk
wolf	wolf	lion	lion	lion	owl	dove	hawk	dove	dove
horse	horse	lion	lion	lion	dove	hen	hen	dove	dove
horse	horse	zebra	cow	cow	cow	hen	hen	dove	dove
zebra	zebra	zebra	cow	cow	cow	hen	hen	duck	goose
zebra	zebra	zebra	cow	cow	cow	duck	duck	duck	goose

Fig. 1. Contextual SOM showing separation of hunters, peaceful species, and birds [20].