RECURRENT NEURAL NETWORK OPTIMIZED DATA CLUSTERING IN TEXT DOCUMENT CLUSTERING ON CANCER DATASETS

M. Keerthana
Department of Computer Science and Engineering, Paavai Engineering College, India

Abstract

In this paper, we include an in-depth analysis of DL-based research clustering approaches. Cluster findings in the three separate use cases representing different data types reveal that DL-based clustering algorithms exceed these approaches. Overall assessments are however delayed because of the small number of marked data for GE-bio-imaging and clustering. Neural networks usually need a great deal of samples for generalization.

Keywords:

Deep Learning, Recurrent Neural Network, Clustering, Cancer

1. INTRODUCTION

The clustering of images, knowledge collection, data composition, identification patterns, word clusters and biologics is a basic, non-surveyed learning task widely used in the exploratory data mining field. [1]. The primary objective of the cluster is for data to be divided into clusters based on similarities, density, intervals or statistical distribution measures of the data space [1]-[3]. For example, the clustering of gene expressions (GE), in which genes of small distance share identical patterns of expression [4], will expose groups of functionally related genes. Such an investigation decides the genes under certain circumstances are turned on or off [4]-[6]. An example is also the clustering of genes or biomedical images through discovering from unlabelled data sets the secret patterns [7]. In addition, it can be astonishing to visualize, view, and analyze large-scale biological data in its unstructured integrity unless the data is grouped.

Biological entities such as chromosomes, pathogens, proteins, tracts and small molecules can be clustered based on input quantities, consistency and type [8] or samples (e.g. patients or distinct cells). While a large amount of biological data comes from various ubiquitous medical instruments, clustering implementations in genomics and gene clustering analytics are still limited [9] [10]. In addition, one-cell studies have become an emerging subject of bioinformatics science, which is critical to clustering [12].

However, as cluster analysis itself is not a particular algorithm, it is possible to use different methods to consider and discover effectively what makes a cluster. Various issues involve varying measurements of resemblance and separation [13] in practice. Furthermore, an efficient clustering algorithm is complex and can be conceived as a multifaceted optimisation problem [12] for a particular bioinformatics problem. Over the course of the years clustering analytics have been suggested in literature [1] such as hierarchical clustering [4], central center clustering [8], distribution-based clustering [9] [11], etc. Other methods include probabilistic clustering, map clustering, spectral clusters, and factoring of non-negative matrices [3]. Clusters with a default ordering include HC algorithms that combine lower-level clusters into even larger groups at higher levels, which gives clusters a hierarchy. Each data point is initially called an independent cluster in agglomerative clustering (AC). Similar clusters are then fusion with other clusters until in each iteration a cluster or *K* is created. HC algorithms have advantages in their simplicity and eyes, and depending on the required granularity, the hierarchy may be broken at the desired level in order to achieve an appropriate clustering process. Clustering accuracy (CA), however, is noise-sensitive [1], which complicates hierarchy perception. Furthermore, there is no possibility of reviewing the clustering [1] [7] and the data points are clustered with local judgments dependent on deterministic attributes.

CC algorithms by comparison, also achieve a higher level of precision in point density, conservation of topology and computing requirements. The non-convex clusters of CC algorithms are, however, unable to be found [12]. DC algorithms are based on distribution models that classify clusters as data points of the discovered distributions (s). DC methods generally yield dynamic models for clusters and can also identify correlations and dependencies between biological attributes. On the other hand, if the Gaussian distributions are built on a strong assumption of data, a concisely described model cannot be created. Moreover, if the model's sophistication is not limited, DC algorithms are inevitably affected by overfitting issues.

While these algorithms operate relatively well for medium-sized and low-dimensional results, their precision and efficiency are dramatically impaired by a large number of samples, primarily because of their dimensionality. In addition, the high computational difficulty in big data is commonly experienced by ML-based methods [2]. Recurrent Neural Networks (RNN) is widely used, along with clustering, to reduce computational complexity by mapping input data to a functional area in which the distinction is easier in the sense of the problem [13]. The explanation is that PCA is essentially restricted to linear embedding, which also loses critical characteristics [3]. High dimensional datasets therefore require non-linear and spectral DR, without losing essential features, for better clustering performance.

On the other hand, only deep learning (DL) bases are currently being actively used [8] in bioinformatics science, in particular for supervised learning tasks where data is handled separately and sequentially with DR and clustering dependent RL. But suppose, for example, how to partition them into K-groups with respect to inherently latent semantics from a vast collection of unlabeled images? Using an ML-based method, the function vectors should be first extracted according to domain specific information and (ii) the extracted features should be grouped using an all-inclusive algorithm [4]. DL-based methods, on the other hand, can be more efficient in the RL process and in the extraction of images that can be used to optimize clusters with an auxiliary goal distribution

extracted from the current allocation of a soft cluster and to boost clustering iteratively [2]. More nuanced and high-level characteristics can be integrated into the input data, and background information can be collected [5] using a deep neural network (DNN) architecture in particular. Lastly, understanding nonlinear mappings enables input data to be transformed into more cluster-friendly images that project data into a lower dimensional function area [2]. The cluster assignment can then be carried out using a basic clustering algorithm, and the clustering goal can be optimized iteratively [6].

2. RECURRENT NEURAL NETWORK

Machine learning approaches are neural networks. They are the same as the cortical biological networks. There are several neurons and neuronal links. A neural network example is Fig.1. Neurons are represented by white circles and associations between neurons are represented by arrows. Notice that we use arrows to reflect the relations guided.

First of all, what the neuron is, we need to remember. In genetics, stimuli, thresholds and outcomes are available to neurons. The neuron is triggered and the signal is sent to the output if the input voltage is greater than the threshold. Note that the neuron can have several inputs, but only one output signal is available. In neuroscience and machine learning, the neuron operation paradigm is very similar. The inputs and outputs are also available.

Despite the neuron's output is connected to many neurons in Fig.1, the value of the outputs are the same. Of course, there are some differences of them. Instead of the threshold, the "neuron" in machine learning use a function to transfer the inputs to the output. There are many choices of the activation function. We often choose it as the sigmoid function $\sigma(x)$.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{1}$$

The neural networks emerge as we bind several neurons. The colored rectangles are made up of a large number of neurons. One or more neurons in the layer are not linked to each other. The first layer is also called the input layer and the final layer is called the output layer. The cached layers are considered the layers between the input and the output layers. Each neuron in the previous layer is always connected to each neuron in the next layer.

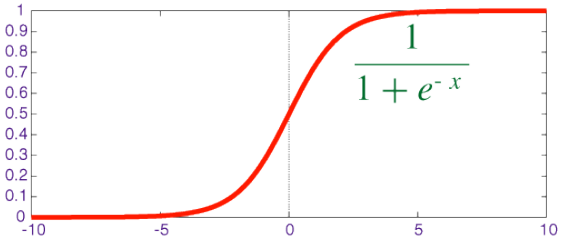

Fig.1. Sigmoid function

Another difference is the weight. The weights describe that how much each input affects the neuron. That is, we will not just put every inputs into the activation function. The value of activation function's input is the linear combination of the inputs. The mathematical representation is as follows:

$$\sigma(w_1x_1 + w_2x_2 + \cdots + w_Nx_N) \tag{2}$$

where N is the amount of the inputs, w_i are weights of x_i, and σ() is the activation function. However, there is a problem of it! We reduce the amount of inputs to 1 and change the weight to observe how weights influence on the output. The result is shown in Fig.3(a). One can see that 0 can be viewed as the threshold to determine whether the output is near to 0 nor near to 1. However, how do we modify the model if we want to change the threshold to a value other than 0? In this case, we add a bias θ to achieve that so that we can shift the sigmoid function. The result of the sigmoid function with bias is shown in Fig.3(b). So the new relation is revised as follows:

$$\sigma(\theta + w_1x_1 + w_2x_2 + \cdots + w_Nx_N) \tag{3}$$

The parameter θ is the bias and other notations are the same as above. And $\theta, w_1, ..., w_n$ are parameters that are needed to be learned. That is how neuron works.

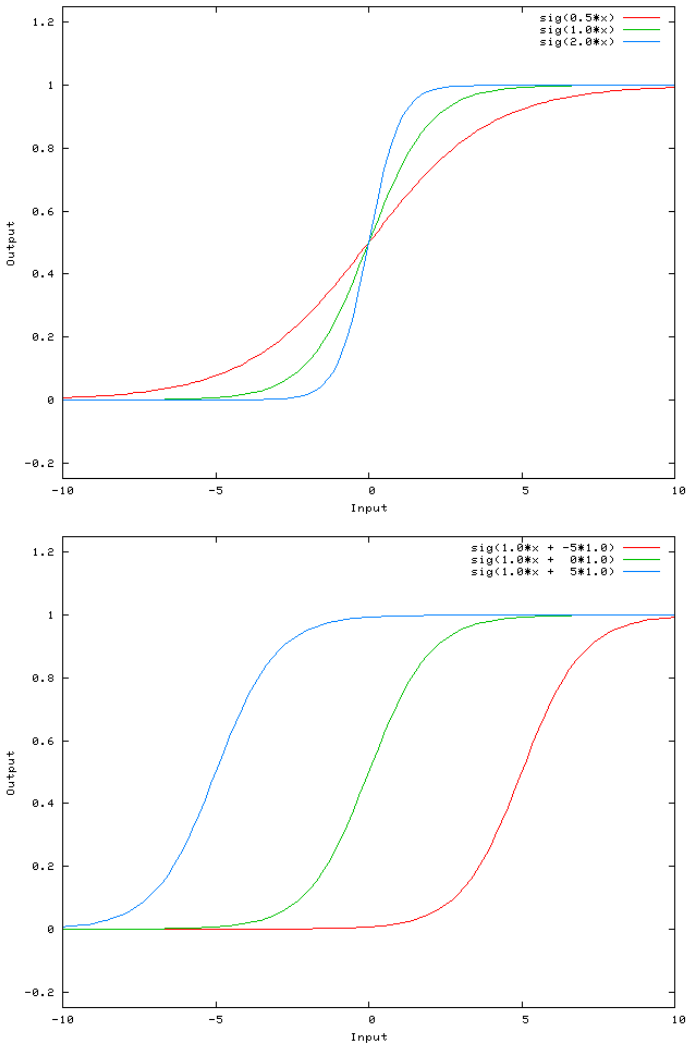

Fig.2. (a) Result of the sigmoid function with different weights of input but without bias (b) Result of the sigmoid function with different weights of bias

If we connect many neurons, the neural networks appear. The colored rectangles are consisted of many neurons and we call these rectangle layers. The layer contains one or many neurons and these neurons will not connect to each other. We often call

the first layer the input layer and we call the last layer the output layer. The layers between the input layer and output layer are called the hidden layers. We often connect every neuron in previous layer to every neurons in the next layer.

There are many neurons in the neural networks. Each neuron has many weights. Therefore, the goal is that we should find the proper weights to fit the data. That is training the networks so that the outputs are close to the desired outputs. In the next section, we will introduce the method of training – backpropagation.

3. NETWORK UPDATES AND TRAINING

Depending on the DNN architecture, various loss functions and training process, DL-based clustering algorithms can vary. However, since it would be difficult to cover both of them in depth in this comparative study, we are discussing network changes and pipeline preparation only in detail, with many of the potential measures outlined in other DL approaches. Two types of losses are configured in DL-based clustering:

- **Non-Clustering Loss**: these losses are independent of an algorithm of the clustering and normally impose a desired limit on the model you studied, which ensures valuable knowledge is preserved by the learned image, so that the original input can be reconstructed during decoding.
- **Clustering Loss**: This kind of loss depends on the classification system of the learning representations and on its clustering-friendliness.

4. RESULTS AND DISCUSSIONS

We concentrate on clustering genomic data, biomedical text mining and biomedical imagery using various methods to demonstrate the efficacy of NN-based clustering approaches.

The software stack included Keras and Scicit Learn with a TensorFlow backend was written in Python, and tested on a 32-center, 256GB RAM and Debian 9.9 OS machine.

Results based on the best hyperparameters generated by random search are recorded empirically and checked if the network converges with and increases K=2 slowly to the optimum number of clusters.

We also examined the way in which network training converged with other methods like ARI, NMI, ACC, completeness and homogeneity during cluster assignments and updates by using Elbow methods.

Table.1. Performance of Clustering Algorithm with Auto encoder in RNN

Clustering Algorithm	ACC	NMI	ARI	Homogeneity	Completeness
ANN	0.65	0.63	0.53	0.52	0.58
MLP	0.72	0.72	0.62	0.68	0.69
RBN	0.73	0.74	0.63	0.70	0.71
DPNN	0.78	0.75	0.69	0.67	0.73
RNN	0.80	0.83	0.84	0.72	0.75

Table.2. Performance of Clustering Algorithm with Convolutional Auto Encoder in RNN

Clustering Algorithm	ACC	NMI	ARI	Homogeneity	Completeness
ANN	0.73	0.7	0.65	0.67	0.68
MLP	0.85	0.83	0.84	0.75	0.77
RBN	0.82	0.81	0.80	0.73	0.74
DPNN	0.75	0.76	0.70	0.65	0.73
RNN	0.72	0.73	0.67	0.58	0.69

Table.3. Performance of Clustering Algorithm with Convolutional Auto Encoder in RNN

Clustering Algorithm	ACC	NMI	ARI	Homogeneity	Completeness
ANN	0.67	0.69	0.56	0.57	0.65
MLP	0.72	0.72	0.62	0.68	0.69
RBN	0.71	0.72	0.66	0.69	0.70
DPNN	0.74	0.73	0.68	0.66	0.71
RNN	0.83	0.81	0.82	0.70	0.73

Table.4. Performance of Clustering Algorithm with Variation Auto Encoder in RNN

Clustering Algorithm	ACC	NMI	ARI	Homogeneity	Completeness
ANN	0.67	0.69	0.56	0.57	0.65
MLP	0.72	0.72	0.62	0.68	0.69
RBN	0.71	0.72	0.66	0.69	0.70
DPNN	0.74	0.73	0.68	0.66	0.71
RNN	0.83	0.81	0.82	0.70	0.73

Table.5. Performance of Clustering Algorithm with LSTM in RNN

Clustering Algorithm	ACC	NMI	ARI	Homogeneity	Completeness
ANN	0.70	0.69	0.67	0.68	0.69
MLP	0.80	0.82	0.81	0.74	0.75
RBN	0.81	0.79	0.81	0.75	0.76
DPNN	0.80	0.81	0.80	0.79	0.80
RNN	0.83	0.84	0.85	0.81	0.82

5. CONCLUSION

In this article, we include an in-depth analysis of DL-based research clustering approaches. Cluster findings in the three separate use cases representing different data types reveal that DL-based clustering algorithms exceed these approaches. Overall assessments are however delayed because of the small number of marked data for GE-bio-imaging and clustering. Neural networks usually need a great deal of samples for generalization.

REFERENCES

[1] Xin-She Yang, "Firefly algorithm, stochastic test functions and design optimization", *International Journal of Bioinspired Computation*, Vol. 2, No. 2, pp. 78-84, 2010.

[2] Ujjwal Malik and Sanghamitra Bandyopadhyay, "Genetic Algorithm-based Clustering Technique", *Pattern Recognition*, Vol. 33, No. 9, pp. 1455-1465, 2000.

[3] Xin-She Yang, "*Nature-Inspired Metaheuristic Algorithms*", Luniver Press, 2010.

[4] D.M. Van and A.P. Engelbrecht, "Data Clustering using Particle Swarm Optimization", *Proceedings of International Congress on Evolutionary Computation*, pp. 215-220, 2003.

[5] Xin-She Yang and Suash Deb, "Engineering Optimization by Cuckoo Search", *International Journal of Mathematical Modelling and Numerical Optimization*, Vol. 1, No. 4, pp. 330-343, 2010.

[6] Ajith Abraham, Ravi Jain, Johnson Thomas and Sang Yong Han, "D-SCIDS: Distributed Soft Computing Intrusion Detection System", *Journal of Network and Computer Applications*, Vol. 30, No. 1, pp. 81-98, 2007.

[7] T. Amalraj Victoire and M. Sakthivel, "A Refined Differential Evolution Algorithm Based Fuzzy Classifier for Intrusion Detection", *European Journal of Scientific Research*, Vol. 65, No. 2, pp. 246-259, 2011.

[8] Mostaque Morshedur Hassan, "Current Studies on Intrusion Detection System, Genetic Algorithm and Fuzzy Logic", *International Journal of Distributed and Parallel Systems*, Vol. 4, No. 2, pp. 35-47, 2013.

[9] J. Senthilnath, S.N. Omkar and V. Mani, "Clustering using Firefly Algorithm: Performance Study", *Swarm and Evolutionary Computation*, Vol. 1, No. 3, pp. 164-171, 2011.

[10] Miao Wan, Lixiang Li, Jinghua Xiao, Cong Wang and Yixian Yang, "Data Clustering using Bacterial Foraging Optimization", *Journal of Intelligent Information Systems*, Vol. 38, No. 2, pp. 321-341, 2012.

[11] Tunchan Cura, "A Particle Swarm Optimization Approach to Clustering", *Expert Systems with Applications*, Vol. 39, No. 1, pp. 1582-1588, 2012.

[12] Sherif M. Badr, "Implementation of Intelligent Multi-Layer Intrusion Detection Systems (IMLIDS)", *International Journal of Computer Applications*, Vol. 61, No. 4, pp. 41-49, 2013.

[13] Daniela Zaharie, "A Comparative Analysis of Crossover Variants in Differential Evolution", *Proceedings of the International Multiconference on Computer Science and Information Technology*, pp. 171-181, 2007.