ISSN:2229-6107

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY

E-mail :
editor.ijpast@gmail.com
editor@ijpast.in

www.ijpast.in

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST

ISSN 2229-6107 www.ijpast.in
Vol 8,Issuse 1.Jan 2018

An method for classifying data with several dimensions

Dr. R. Rambabu, Mr. P S S K Sarma, Mrs. A. Josh Mary

Abstract
There is an unprecedented need for massively parallel machine learning due to the growth of large data and high-dimensional streaming data. Hardware deployment, rapid processing speed, dimensionality and volume scaling, learning from streaming data, and automated dimension reduction on high-dimensional data sets are all requirements for this machine learning. Large-scale machine learning problems of this nature are well suited for neural networks. This paper presents a fresh approach to large-scale high-dimensional data handling. This web-based method might manage enormous volumes of big data that are offline and in motion at the same time. Despite using a lot of Kohonen nets, we only retain a tiny portion of each net's neurons (or nodes) after training and delete all of the nets thereafter. We utilize Kohonen nets to choose features and build ensemble classifiers from individual Kohonen neurons. Using Kohonen net-based hardware that is optimized for enormous parallelism, the strategy should be simple to implement. This is where the computer lab's initial results were shown.

Keywords: high-dimensional data, online learning, Kohonen networks, feature selection

1 Introduction

2 The introduction of enormous and real-time data sets has led to considerable changes in the field of machine learning. Modern machine learning systems also face a number of other difficulties, including the requirement to incorporate new technologies, automate machine learning with little human involvement, and learn rapidly from large datasets. Artificial neural network-based classical algorithms are expected to play a significant role in the current revolutions because to their numerous advantages, especially when it comes to addressing the problems presented by massive data. Neural net methods have the ability to handle very large datasets concurrently since many of them rely on live, incremental learning.

Professor & HOD, Assistant Professor[1,2]
Department of Computer Science & Engineering,
Rajamahendri Institute of Engineering & Technology, Rajamahendravaram.

3 3.Instead of needing to laboriously sample from enormous datasets, we may solve a multitude of computing problems simultaneously by employing this learning method. Moreover, it makes neural net approaches highly scalable by allowing them to learn from all the data. As a result, they can handle massive data volumes without running into problems with running out of computer memory. The ability of learning (processing) time to scale up linearly with data volume is another advantage of incremental learning. Another advantage of neural network systems is their utilization of simple, easily parallelizable computations. Such algorithms are already being developed by Oh and Jung (2004) using technology that enables parallel computations, whereas Monroe (2014),

4 4Furber et al. (2013) and Poon and Zhou (2011) are developing hardware that is even more potent. In this era of large data and streaming data, neural network technologies appear to have provided the perfect basis for machine learning.

This article presents a unique method for training neural networks that satisfies a number of requirements: It utilizes an ensemble of classifiers learned from selected Kohonen neurons (nodes) from different Kohonen nets (Kohonen, 2001), addresses the issue of high-dimensional data, is readily implementable on hardware, can be parallelized at different levels of granularity, and so on. Using streaming data, we train several Kohonen nets concurrently to offer some data points for dimensionality reduction based on feature selection. Information that has been

2. Feature separation based on classes, dimensionality reduction, and feature separability index

5 5.Training machine learning models on high-dimensional data is a major difficulty. Novel approaches for online feature selection and feature extraction for high-dimensional streaming data have been presented in numerous recent studies. Yan et al. (2006), Hoi et al. (2012), Wu et al. (2010), and Law et al. (2006) are a few instances. However, none of them are designed to choose traits based on social classes. Using a subset of the original characteristics, Roy et al. (2013) provide a technique that follows his 1997 conference advocacy for class-specific classifiers. However, a method presented by Roy et al. (2013) is not applicable to streaming data. A goal of class-specific feature selection is to identify unique collections of attributes that may be applied to

It is easy to choose features that minimize the average distance between individual points within each class and maximize the average distance between the data points in each class while working with a dataset in offline mode. Computational research has demonstrated the efficacy of Roy et al. (2013)'s feature selection and ranking procedure. But since it doesn't really save any data, that strategy isn't appropriate for processing streaming data. The suggested approach is conceptually similar to feature selection, using streaming data to train multiple Kohonen nets. We overcome the problem of inadequate data points by training several Kohonen nets, which each give a tiny subset of representative data points for their respective classes. Following the assembly of a group of representative

1 forms clusters, with the nodes or neurons that are actively involved in the network serving as a sample of the streaming data. Our next step is to use these examples to classify character traits.

Think about all the kc courses together. We base our feature rating on the idea that a good feature should perform two things for each class k = 1...kc:(1) reduce the size of class k patterns and (2) effectively distinguish between class k and non-class k patterns. Roy et al. (2013) use a measure called the separability index to rank attributes for each class. It is based on these notions.It would be interesting to find the mean separation between feature n patterns that belong to class k and those that do not, and to compare them to the mean separation between patterns in class k with regard to feature n. While various distance metrics might be used, the one utilised by Roy et al. (2013) is the Euclidean distance.The separability index may be expressed as Rkn = d.in the absence ofBy using this separability index r, Roy et al. (2013) determine the relative relevance of class k traits; a bigger ratio denotes a higher rank. Taken together, this indicator is For the simple reason that features n with lower dins condense classes K and L, while features n with bigger douts make classes L more different from one other. Consequently, a bigger ratio rkn for feature n improves a feature's ability to differentiate class k from other classes.

1.1 Whyclass-basedfeatureselection?Anexample

GeneNumber	Separability Indices byClass	
	AML	ALL
AML Good Features		
758	82.53	2.49
1809	75.25	1.85
4680	39.73	2.82
ALL Good Features		
2288	0.85	114.75
760	0.93	98.76
6182	0.8	34.15

Table2.1–SeparabilityindicesforafewfeaturesintheAMLALLgeneexpressiondataset

To put this strategy to the test, we resolved many high-dimensional gene expression issues. Predicting AML or ALL from gene expression data is one such difficulty (Golub et al., 1999). The number of genes (features) is 7,129, and there are 72 samples of data. Some genes and the separability indices assigned to them are shown in Table 2.1, organized by class. Genes 75.25, 758, and 4680 all have high separability indices for the AML class, while genes 1809 and 4680 also serve as strong predictors of the AML class. But, for the identical set of genes in the ALL class, the corresponding separability indices are quite low: 2.49, 1.85, and 2.82, respectively. All things considered, these three genes do a poor job of predicting the ALL subtype. In Table 2.1, we can see that three genes—2288, 760, and 6182—have high separability indices for the ALL class—114.75, 98.76, and 34.15—and are therefore excellent predictors of the ALL class. Having separability scores of 0.85, 0.93, and 0.8, respectively, indicates that they are poor AML class predictors. Here we see how class-based feature selection works and how they may help us comprehend a phenomena.

Selecting features for classes in streaming data using a Kohonen network

At this point, we provide some notation. Assume an input pattern in the streaming data is represented by the N-dimensional vector x, where Xn is the nth element of the vector. For any integer q from 1 to FS, where FS is the total number of feature subsets, let FPq represent the qth feature subset. With q = 1…FS and g = 1…FG, where FG is the total number of distinct Kohonen net grid sizes, let KN g be the gthKohonen net of a specific grid size for the qth feature subset. The entire number of classes is represented as kc, where k represents a class. The effective and rapid computation of the separability indices for high-dimensional data may be achieved with the use of parallel distributed computing capabilities like Apache Spark (Franklin 2013). At its most fundamental level, this is computing parallelism. Having hardware that implementsKohonen nets allows for further parallelization of computing at a lower level.

Suppose that we employ 10 distinct grid sizes (FG = 10) and that we have the computer capacity to generate 500 Kohonen nets in parallel. Then FS would be fifty (500 divided by ten) and

Each Kohonen net is represented by a number from 1 to 10. It is also assumed that N= 1000 represents the number of features present in the data stream. There would therefore be fifty equal subsets of twenty characteristics each, for a total of one thousand features. To keep things simple, let's say that features X1–X20 are part of the first feature partition FP1, features X21–X40 are part of the second partition FP2, and so on. The features in the set FP would make up the input vector for the 1 Kohonen nets KN g, g = 1…10, the features in the set FP 1 would make up the input vector for the 1 Kohonen nets KN g, g = 1…10, and so on.

22

So, we'd train ten separate Kohonen nets with varying grid widths for each feature subset FPq. For classification problems with few classes, grid sizes such as 9x9, 8x8, 7x7, etc., should be enough. Grid sizes need to be greater if there are thousands of classes. For the sake of speed and efficiency, we are mostly using feature divisions. When trained in parallel, these smaller Kohonen nets outperform their larger counterparts that employ thousands of features. In order to get diverse representative samples for the purpose of computing the separability indices, it is necessary to use varied grid sizes for the same feature division.

1.2 Letting Kohonen neurons be labeled

It is possible to calculate the separability indices using a subset, but not a whole subset, of a Kohonen net's active nodes as training examples. The only neurons that are considered active are the ones that win. As soon as the Kohonen nets settle down, we run more streaming data through to

choose the classes that these nodes should be moved to. Currently, we do not modify the weights of the Kohonen nets; instead, we only record the number of times a certain neuron was activated by an input pattern that falls into a given class. Think about the following situation: We have two classes, A and B. Based on these two categories, we track the total number of activations that each active node has experienced as a result of input patterns. Think about a single neuron that receives stimulation from class B patterns 15 times and activation from class A patterns 85 times. At this node, the input patterns may be classified as class A 85% of the time and class B 15% of the time. Labeling is simple.

1.3 A feature ranking technique that calculates feature separability indices by class

Section 1: Features are divided into FS subgroups at random.
Second, for each feature partition, randomly initialize all Kohonen networks with varying grid sizes simultaneously.
Thirdly, divide the input vector based on the feature subsets assigned to each Kohonen net and train all of the nets simultaneously using streaming data. When every Kohonen net converges, training should end.
Fourth, to identify the active nodes (winning neurons) and their class counts, run more streaming data through the stabilized Kohonen nets without adjusting the weights.
The fifth step is to assign a class to every active node (neuron) if the proportion of that node's most active class is higher than a certain threshold. Activated neurons with percentage counts lower than the cutoff should be discarded.
Sixth Step: For each feature partition FPq, where q ranges from 1 to FS, compile a list of all active nodes sorted by class.
Step 7: Use the neurons in the active node list that are left as examples for the classes to compute the separability indices of the features in each feature partition FPq, where q = 1…FS.
Measure the features' average separability indices by iteratively repeating steps 1–7.
Nineth Step: Use Average Separability Indexes to Sort Features.

Onfeaturecombinationstoexploretobuildclassifiersand ontheconceptof bucketsoffeatures

After ranking the features, Roy et al. (2013) combine each feature one at a time, beginning with the feature that ranks highest. They then roughly fit a collection of hyperspheres to the data points to estimate the classification error rate for each feature combination. According to Roy et al. (2013), a hypersphere classifier's error rate is estimated using the feature that ranks highest.Then, it grabs the top two Features are rated, and the error rate is estimated; the top three are then and so on. After choosing a set of feature combinations with the lowest error rates, it uses those feature sets to construct hypersphere classifiers that are more accurate.We truly have the capacity to construct independent classifiers in parallel for various feature combinations with a parallel distributed computing system. Using this strategy, we may construct Kohonen nets with varying grid sizes for the top feature, top two features, top three features, and so forth—all of which can be completed concurrently. Following the parallel creation of these Kohonen nets, we are able to choose the feature combinations with the highest accuracy. Regarding the computational outcomes

2. Building a Kohonen neuron ensemble-based classification system

In this paper, we provide a method for building a Kohonen-based categorization system. The neurons used in these Kohonen nets come from a variety of networks that use varying grid sizes and feature spaces. Be careful to remember that at the conclusion of this stage, just a subset of the Kohonen neurons are kept and the rest of the Kohonen nets are eliminated.

1.2 How to use a majority to allocate neurons (active nodes) to classes

The last Kohonen nets are learned using streaming data, and they come in various feature spaces and grid sizes. The class count percentages at each active node are obtained in a manner similar to the first phase when these Kohonen nets converge and are stable. We continue processing streaming data without modifying the weights of the Kohonen nets. We repeat the first step of trimming the active nodes when the class count percentages at each node are steady. So, nodes that are active but have low total counts are removed, while nodes where a certain class has a strong majority (say, 70%) are kept. Where there is a distinct majority for a class, we choose neurons that are excellent at what they do.

1.3 Regarding the Kohonen neuron's radius

The idea of the radius of an active node (neuron) that governs roughly the border within which it is the winning neuron forms the basis of our categorization method. Because our method relies on keeping just the most active nodes in a Kohonen net and discarding the inactive ones, this idea is fundamental. We can't find the winning or optimal neuron for an input pattern after we remove the remaining nodes from the Kohonen net. So, the radius is an alternative metric for predicting which node would come out on top. After setting all active nodes' radii to zero, we examine further streaming data to update them and finally find the radius. We repeat this procedure until all active nodes' radii are stable. We revise the radius as follows. We find the distance between the active node and the streaming input pattern if their classes are similar. If the distance is more than the current radius, we update the node's radius. To remove the Kohonen nets, we first revise the radii of every node that is currently running.

1.4 A method for using streaming data to train the last batch of Kohonen nets for categorization

First, set the value of bucket j to zero.
Next, build the jth bucket by adding a few additional top-ranked features by class k to the (j-1)th bucket and incrementing the number of buckets by one (j = j + 1).
Third, for each pair of features for classes k and j, randomly initialize final Kohonen nets with varying grid sizes in parallel using a distributed computing system. Return to Step 2 to establish additional Kohonen nets for the remaining feature buckets if their indices are more than 1. Continue to step 4 if not.
Step 4: Train all Kohonen nets simultaneously using streaming data and input pattern parts chosen with each net's feature subset in mind. When every Kohonen net converges, training should end.
The fifth step is to run more streaming data through the stabilized Kohonen nets without adjusting the weights. This will reveal the collection of neurons that are active for each class k and bucket j. Make sure to collect the class counts of all active nodes as well. Once the class percentages for all nodes are constant, you may stop collecting them.
Step6: If the majority class's class percentage and absolute class count are more than minimal criteria, then assign an active node to that class.
Step7: Calculate the radius of each active node by processing additional streaming data. Once the widths or radii have stabilized, stop.
Step 8: Remove all non-eligible nodes from each Kohonen net and keep just the active nodes that meet the criteria.

1.5 The classifier is not a network of Kohonen nets but rather an ensemble of dangling neurons.

We use an ensemble of Kohonen neurons trained in several feature spaces to do classification at the end. Combining numerous classifiers may often increase overall performance on a topic, according to studies. The review and taxonomy of ensemble learning techniques provided by Rokach (2009) are of high quality. Many different combinations of base classifiers are possible for the final prediction. At the moment, these metrics are used to decide how a test case is ultimately classified.
a. Maximum Probability—Assign the class of the ensemble neuron with the greatest confidence (or probability) to the test case.
b. Minimum Distance—Assign the class of the neuron that is geographically nearest to the test example.
c. Majority voting—In this method, we find out what a test example's class is by tallying up the votes of neurons in each feature space, taking into account both maximum probability and minimum distance neurons.

2 Outcomes from the Computer

There has been a lot of focus lately on issues with gene expression. High dimensionality (sometimes including thousands of features or genes) and sparse training samples define these types of issues. To evaluate the efficacy of our approach, we used seven prominent gene expression datasets. The key features of these datasets are summarized in Table 5.1. From the available data, we created two sets: one for training and one for testing. We used a random selection process to choose 90% of the data for training and the remaining 20% for testing. The outcomes of this random allocation were averaged across 50 runs, and they are shown below. By reading just one input pattern at a time, our system mimicked online learning.

	No.ofgenes	No.ofclasses	No.ofexamples
Leukemia(AML-ALL) (Golubetal.1999)	7129	2	72
CentralNervousSystem (Pomeroyetal.2002)	7129	2	60
RBCT (Khanetal.2001)	2308	4	63
rostrate (Singhetal.2002)	6033	2	102
Brain (Pomeroyetal.2002)	5597	5	42

Table5.1 –Characteristicsofthegeneexpressionproblems

1.2 Configuring Parameters

For every issue that this method resolved, no settings were adjusted. Although we are currently working on an implementation using Apache Spark, we did not use a parallel distributed computing platform for these studies. Desktops and laptops handled all issues. In order to solve these issues, we utilized the following parameter values. Grid sizes of 9x9, 8x8, 7x7, 6x6, 5x5, 4x4, and 3x3 were used for Kohonen nets, with FG equal to 7. Each feature subset had ten features for feature selection. The class percentage has to be at least 70% to be considered.

1.2 Findings from the research - Selection of features

The typical amount of characteristics used by this approach to address gene expression issues is shown in Table 5.2. It is clear that the suggested approach performs adequately when narrowing down the thousands of genes (features) to a manageable number. Therefore, this approach is effective for dimensionality reduction via feature selection.

	Total no. of attributes	**Average No. of featuresused**	**% of features used**
Leukemia (AML-ALL)	7129	20	0.28%
Central Nervous System	7129	10	0.14%
ColonTumor	2000	31	1.55%
SRBCT	2308	54	2.34%
Lymphoma	4026	24	0.60%
Prostrate	6033	74	1.23%
Brain	5597	28	0.50%

Table5.2–AveragenumberoffeaturesusedbytheKohonenneuronensemblemethodforthegeneexpressionproblems.

1.2 Testing the Kohonen Neuron Ensemble Classifier System in the Real World

Here we provide the experimental results of the two-stage Kohonen ensemble method, which involves choosing unique features for each class and then training a cluster of Kohonen neurons to use these features for classification. We test the algorithm against various categorization methods using Apache Spark's MLlib(2015) machine learning module. Additionally, fifty iterations of Spark testing were conducted after randomly splitting the data into a training set and a test set. Several algorithms in Spark's MLlib have their standard deviations and average error rates shown in Table 5.3. Both SVMwithSGD and LogRegWithSGD employ stochastic gradient descent (SGD) as a classifier training technique. All of these algorithms were executed with their default parameters without initial feature selection. While SVMwithSGD and LogRegWithSGD work well for two-class problems, they do not provide findings for multiclass problems in their tables. SRBC, Brain, and Lymphoma.

	Kohonen ensemble	**SVMwithSGD**	**NaiveBayes**	**LogRegWithSGD**	**RandomForest**
Leukemia(AML-ALL)	1.14(3.96)	4.4(7.0)	10.26(8.0)	10.29(12)	12.8(10)
Central Nervous System	28(16.47)	33.25(14)	42.84(11)	36.25(13)	41.26(11)
ColonTumor	10.67(9.48)	17.56(11)	8.33(12)	12(12)	18.67(12)
SRBCT	0.67(3.33)	-	7.33(12)	-	21.33(18)
Lymphoma	0.67(3.33)	-	1.9(3)	-	5.67(10)
Prostrate	5.6(7.12)	13.64 (5)	36.97(12)	13.4(10)	19.6(13)
Brain	20.0(11.18)	-	18(19)	-	47.5(25)

Table5.3–
AveragetesterrorratesandstandarddeviationsforvariousclassificationalgorithmsofApacheSparkMLlib(2015).

2. Final Thoughts

We provide a new method for learning from high-dimensional data that is both static and dynamic in this article. Because it is an internet solution, data storage would need to be streamed. There is no need for online methods to have access to all of the training data at once, and the learning (processing) time scales up linearly with data volume, making them very scalable. On top of that, they don't have to choose extract data in order to get insight from all of it. This method has the potential to use several levels of parallelization. It is easily parallelizable on distributed computing systems like Apache Spark and can even be implemented on the neural hardware's level of massively parallel computing. Internet of Things (IoT) applications that prioritize learning and reacting to high-speed streaming data would greatly benefit from a neural hardware implementation. Moreover, data saved may be processed rapidly via the use of brain circuitry. Our experiments also show that the method works wonders when it comes to reducing the dimensionality of a high-dimensional problem.

List of Sources

In 2004, Oh and Jung published a research. building neural networks using GPUs. This paper may be found on pages 1311–1314 of the journal Pattern Recognition, published in June 2013. Danny Monroe in the year 2014.The era of neuromorphic computing is going to explode. The article may be found in Communications of the ACM journal, volume 57, number 6, pages 13–15. 2011 saw the writing of Poon and Zhou.Large-scale neural networks with neuromorphic silicon neurons: benefits and drawbacks. Frontiers in neurological research: 5, 5. The study team includes Furber, Steve; Lester, David; Plana, Luis; Jim Garside; Steve Temple; Andrew Brown; and Eustace Painkras. A overview of the SpiNNaker framework.The piece appeared in December 2013 on pages 2454–2467 in IEEE

Transactions on Computers, volume 62, number 12. Takko wrote this in 2001. Automata-based mapping (Volume 30). The publishing house is Springer.
The work was published in 2013 by Roy, A., Mackin, P., and Mukhopadhyay, S. Class-Specific Feature Selection Strategies, Automated Learning, and Pattern Selection Methods (Neural Networks, 41, 113–129).

Yan et al. published an essay in 2006. They were joined by Yan, Zhang, Liu, Yang, and Cheng. using a supervised, scalable method to minimize dimensionality in streaming data... 2042–2065 is the publication date; volume: 176, issue: 14. Hoi, S. C., Jin, R., Huang, J., and Zhao, P. (2012).using online searches to extract attributes from large datasets.The first international conference on big data, streams, and heterogeneous source mining, articles 93–100: algorithms,Law and Jain released a book in 2006. reducing nonlinear dimensionality progressively by the use of manifold learning. The IEEE Transactions on Pattern Analysis and Machine Intelligence, volume 28, number 3, pages 377–391, published this work. Li (2006), Jiang T., Zhang K., and Li X. R. Determining the most reliable and effective margin criterion for feature extraction... IEEE Transactions on Neural Networks, Volume 17, Number 1, Pages 157–165.(Oct. 2013) Franklin, M. From now on, here is where the Berkeley Data Analytics Stack will go. The relevant article may be found on pages two and three of the 2013 IEEE International Conference on Big Data. IEEE.
The work by Golub, Slonim, Tamayo, and associates was published in 1999. Molecular