NVEO
Natural Volatiles &
Essential Oils

Spike Modeling of Hebbian learning for Sensor Deployment

Kusuma S M[1], Veena K N[2], Vijaya Kumar B P[3]

[1]Research Scholar, School of Electronics and Communication Engineering, Reva University, Asst. Prof.,Dept. ETE, M S Ramaiah Institute of Technology, Bangalore, India

[2]Associate Professor, School of Electronics and Communication Engineering, Reva University, Bangalore 560064, India

[3] Professor, ISE Dept., MSRIT, Bangalore- 560054, Karnataka, India

Abstract

In the next generation wireless technologies like 5G/6G and beyond, there is a scope for intelligent and smart way of information transfer and services to the society, which is scalable. With such potential communication capacity, many smart activities are initiated to tackle more intrinsic and extrinsic services in the areas like smart city, agriculture, health, industry, automation etc. In all of these areas to become a smart environment, there is a requirement of embedded sensors, Internet of Things (IoT), edge computing, smart sensors to sense and tag the phenomenon of interest with digital decision support systems. Here, the work proposes a novel technique to identify the dynamics of the phenomenon using the gradient of different parameters. Spike based hebbian learning model is introduced with mathematical analysis to hand hold the dynamics of the phenomenon using the adaptable patterns for energy efficient sensor usage. Implementation results are compared with the analytics in a given simulation environment. System is trained and tested for stochastic gradient analysis to model the error.

Keywords: Hebbian learning, stochastic gradient descent, wireless sensor network, Sensor deployment

I. Introduction

Smart embedded sensors play a vital role in sensing, processing and communication of information from source to destination. In 5G and 6G generation wireless technologies there will be a lot of scope for optimal deployment of sensors in every sector. Machine learning algorithms are able to predict the proper deployment of sensors based on the observed historical data.

Normally sensors become a smart computing devices that capture the physical phenomenon of interest and communicate to the end devices effectively through the wireless network. Smart sensors are deployed in every sector namely health care, agriculture, weather monitoring, industrial

sector etc. Due to the increase in the large number of such smart devices in future 5G and 6G technologies, there will be constraints on the energy usage pattern of the wireless sensor network. To achieve the energy reduction in the network different machine learning algorithms have been introduced by many researchers. In the proposed work hebbian learning technique is adopted to map the sensor network with each single neuron being modeled as a sensor node to learn the behavior of the environment [4]. Later to compute the optimal number of sensor's deployment based on spatio-temporal distribution of parameter values to conserve the energy and cost of sensor deployment. Simulation of the proposed methodology is carried out to evaluate the performance. Here each sensor node data from the environment is modeled as a Gaussian distributed data and verified analytically.

Here the work is discussed in four major sections. section II discusses briefly on some of the related works in the area. Basic concepts of the sensor's energy model for the existing system are explained in section III, that involves neural networks, Hebbian learning and spike modelling. Proposed methodology and algorithm's design for the process involved with the use of hebbian spike model is described in section IV. Implementation and results with performance analysis for comparisons are narrated in section V. Conclusion remarks are given in section VI.

II. Related work

Mathematical formulations of hebbian learning based on activation potentiation of each neuron and the weight update with respect to synchronous firing of presynaptic and postsynaptic neuron to find the correlation among the neurons is discussed in [1]. Mathematical analysis on the structure and dynamics of hebbian learning for discrete time random recurrent neural networks is considered in [2]. Here a generic hebbian learning rule in. Support vector based machine learning to identify scales for neuronal activity and learning dynamics are node level energy saving based on Simulated Annealing [3] are some of the efficient techniques to reduce the energy consumption. Deployment of sensors in smart cities [5] with Triangulation-based Deployment for Smart Cities (EDTD-SC), which targets not only sensor distribution, but also sink placement is also one of the method to reduce energy consumption. In [6] automatic configuration of energy harvesting in sensor nodes with reduced node duty cycle by performing reinforcement learning, i.e., to maximize sensing quality of energy harvesting sensors for periodic and event driven sensing scenarios with available energy. In [7] a deterministic sensor deployment for target coverage is proposed to predict gas leakage using particle swarm optimization algorithm.to achieve the energy reduction in the network different machine learning algorithms have been introduced by many researchers. A support vector based machine learning classification method is used to detect the fault and anomalous sensors in wireless sensor networks [12]. Deterministic deployment of gas sensors using particle swarm optimization algorithm is proposed to monitor the coverage and its efficiency in [13]. A virtual force algorithm as a sensor deployment strategy and probabilistic target localization algorithm is proposed to improve the sensor's usage and coverage [14].

From the related works and recent research in the areas, we have proposed a novel method to deploy the sensors based on environmental dynamics and its gradient using hebbian learning technique in the present work.

III Neural networks and Hebbian learning

A neural network is a collection of neurons and interconnected synapsis. An artificial neural network composed of artificial neurons are again connected to each other with synaptic weights. Thus a

neural network is either a biological neural network, made up of biological neurons to solve natural intelligence problems, whereas an artificial neural network is for solving artificial intelligence problems. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, like an image, sound, text or time series, must be translated. Typically an artificial neural network model consists of input layer, hidden layer and output layer, with synaptic weights. By showing the input patterns based on the computation done in each layer, the output is generated. Different learning algorithms are implemented to learn the system. Each neuron has its own potentiation for activation and are interconnected to make a artificial neural network. Such a system has been made to learn by applying supervised and/or unsupervised learning methods. Hebbian learning is one of the learning algorithm to learn the input patterns and take decision based on learning, where the synaptic weights gets updated based on the neuronal presynaptic and postsynaptic activation.

III.1 Spike modeling of Hebbian learning

From Hebb's postulate for hebbian neural network learning, the weight should increase if, during an experimental trial, both neurons are active together. Hebbian learning as a function of the activity of the pre and the postsynaptic neuron. Figure 1 shows a typical scenario of four neuron distributions and their synaptic weight values amongst them.

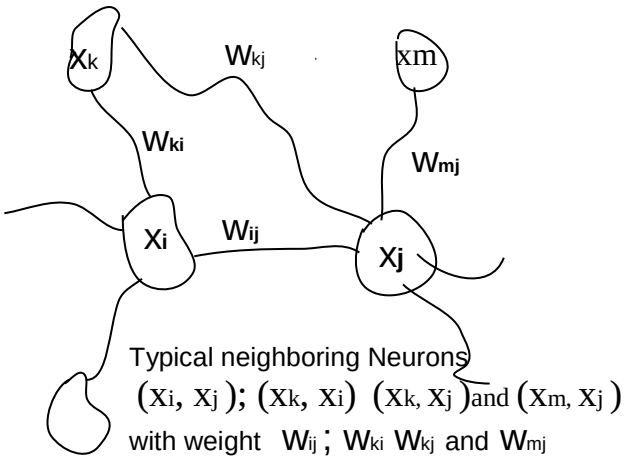

Figure 1: Typical Hebbian learning scenario with neighboring neurons

Considering the weight update using hebbian learning rule, for a given time step ***t,*** is shown in equation 1.

$$w_{ij}[t+1] = w_{ij}[t] + \alpha\, x_i[t] \,.\, x_j[t] \quad \ldots\ldots\ldots\ldots (1)$$

where $x_i[t]$ and $x_j[t]$ adopt the state of the adjacent neuron, distributed in a given sensing environment with sensor value having Gaussian distribution. **α** is the learning rate € (0, 1). Applying hebbian learning and performing statistical analysis to compute the correlation coefficient among

adjacent sensor nodes leads to the update of synaptic weight. Based on the synaptic weight matrix, redundancy among the sensor and identifies the correlation among the corresponding sensors. Here the spike means the time dependent plasticity, i.e., the learning rule for the synapse connecting neuron ***j*** to neuron ***i*** should depend only on the activity of *j* and *i* and not on the state of other neurons.

Mathematically rate of change of weight update can be written as

$$d/dt\ f(w, x_i, x_j) \quad \ldots\ldots\ldots \quad (2)$$

where $\mathbf{x_i}$ is the presynaptic signal and **xj** is the post synaptic signal and $\mathbf{w_{ij}}$ is the connection link between $\mathbf{x_i}$ and $\mathbf{x_j}$. Time-dependent stimulation will be modeled in the context of the spike-based formulation. Consider the weight change $\mathbf{dw_{ij}}$ during one learning trial. Since the total weight change during a trial depends on the duration of the trial and is shown as rate of change of weight in equation (3)

$$\mathbf{dw_{ij} / dt = \nabla w_{ij}/T} \quad \ldots\ldots\ldots \quad \mathbf{(3)}$$

Mathematical formulations of hebbian learning based on activation potentiation of each neuron and the weight updates are discussed. Update with respect to synchronous firing of presynaptic and postsynaptic neuron to find the correlation among the neurons [1]. Mathematical analysis on the structure and dynamics of hebbian learning for discrete time random recurrent neural networks is considered in [2].

III.2 Wireless sensor networks and network energy model

Sensor node energy consumption includes sensing, processing and radio energy model described in this section, to understand the energy dissipation for sensing, transmitting and receiving in sensor node and in the WSN [11].

A Sensor Node Energy Model: Energy used by different components to perform the basic functions as a sensor node is explained [10].

Sensor sensing: Energy spent on sampling the physical parameter signal in to electrical signal and later analog to digital conversion, as $\mathbf{P_{sens}}$.

*Microcontroller Processing: E*nergy spent for processing and memorizing includes energy for electronics, P_{elec} and energy loss due to leakage current, $\mathbf{P_{current}}$

$$\mathbf{i.e.,\ P_{micro} = P_{elec} + P_{current}}$$

Radio Transmission: Energy spent for b bit transmit, in a distance $\boldsymbol{d_{ij}}$ i.e., distance between sensor node ***i*** and node ***j.***

$$\mathbf{i.e.,\ P_{tx}(b, d_{ij}) = b * P_{trans} + b * d_{ij}^{n} * P_{P\_amp}}$$

Where, P_{trans} is the energy dissipated to transmit electronics, P_{P_amp} is the energy dissipated by the power amplifier, 'n' is the distance based path loss exponent (Here n = 2 for free space, and n = 4 for air space).

Radio Reception: Energy spent in receiving b bit packet from the sensor node is given by $P_{rx}(b) = b * P_{recv}$

Where, P_{rx} is the total energy dissipation in receive electronics, where, P_{recv} is the energy dissipated in receive electronics. b is the number of bits in the packet.

Sensor node energy: The power consumption in the sensor node is given by

$P_{sn}(b)=P_{rx}(b)+P_{tx}(b, dij)+P_{micro}+P_{sens}(b)$, where P(b) is the b bit sensing energy of a node.

Network Energy model: Network energy consumption of embedded WSN is given below. N_s are the number of sensor nodes placed in a given region

$P_{clu} = n_c * P_{sn}(b) + P_{txsink}(b)$, P_{txsink} is the energy consumption in the cluster head to send the aggregated data to sink.

$P_{net} = k * P_{clu}$ Where, k is Number of clusters in a sensor network, n_c is number of nodes in a cluster and P_{net} is Energy consumed in the network.

Above model is considered in the simulation to create a realistic environment for performance evaluation.

IV Proposed model

The principle of spike based hebbian learning is considered with the principles discussed in section III. Further to the continuation of the previous work in [4], a mathematical analysis of the spike based hebbian learning approach has been proposed in this work. From our previous work done to achieve optimal energy and sensor nodes in a given scenario. Hebbian learning with sensor network layered architecture is used [4], to identify the correlation amongst the neuro-sensory node based on the presynaptic and postsynaptic firing of neuron working simultaneously. By considering hebbian learning weight calculation, here it is assumed that a wireless sensor network is deployed in an environment where a large number of nodes are distributed randomly to carry out sensing and gathering of data from the environment. When two sensors collect the same data from the environment, mapping to hebbian neurons and their activation, the data is correlated and the weight is increased. If they are collecting distinct data from the environment, the weights are decreased.

Here the hebbian learning is followed based on the data gathered by sensor nodes. When each presynaptic and postsynaptic value of the sensory neuron correlates with some threshold value, *a tuning parameter,* i.e, they synchronize or spike together, accordingly the synaptic weight gets updated. After the hebbian learning phase, based on the weight values, strong or weak connections is used for deciding and fixing the number of sensor nodes, i.e, to identify the redundant nodes that mimic the neuron's and their connection patterns distributed across the phenomenon.

After identifying the sensor nodes and their data correlation using hebbian learning, the regression learning is used to train a decision model based on neuro-sensory patterns generated as a training data set. Both stochastic and batch gradient descent algorithms are used to reduce the mean square errors as a performance measure for effective deployment of sensors.

By considering the similar sensing values (Statistical average of Gaussian distributed sensed data) of each sensor node at each and every instant with a certain threshold, the correlation is computed with statistical analysis over the sensed data. Based on the similarity of the correlation coefficient amongst multiple sensor nodes, the decision is taken to update the pattern of similarity and the number of sensor nodes to be deployed. The correlation coefficient (similarity index) at a given instant of sampling time and their respective pattern leading to number sensor nodes, are the training sample. Similarly for multiple sampling time to generate sufficient training data set to learn the pattern and their behavior. Here Learning algorithms are used to classify the given correlation coefficient values and output to identify the number of sensor nodes at a given instant of time to be enabled/disabled, or deploy/not in a long run. Here the training data set taken over for the values, that leads to the number of sensor nodes to be used, or enabled in a given sensing area.

Assumption is that each sensor node receives Gaussian distributed data. Here the sampling is done at regular intervals, which can be varied depending on the phenomenon or application type. This proposed model also considers a gradient descent algorithm to identify the number of sensor nodes to be deployed based on the correlation between the sensor node values from a given area. Training examples taken for each of the neighboring sensor nodes and their correlation. Based on the correlation coefficient value, the number of sensor nodes to be considered are taken as the expected number of sensor nodes. Using such labeled training data sets, stochastic gradient and mini batch gradient descent optimization algorithms for machine learning and deep learning can be performed to predict the number of sensor nodes to be deployed, and can be distributed across phenomena.

The performance of the gradient descent machine learning is evaluated and compared the overall performance of the optimal sensor deployment along with energy efficiency with and without spike modeling of hebbian learning. Batch gradient descent and stochastic gradient descent are two methods that have been implemented and tested for the performance of the model. Simulation has been carried out for different scenarios to evaluate the performance of the proposed model and is described in section V.

V Simulation and Results

V.1 Scenario for Hebbian learning and with Gradient descent

In a given sensor network deployment scenario, assuming a triangular distribution of sensor node, shown in figure 2 (here we can assume grid distribution or any 3 dimensional sensor distribution) with each sensor output values of the measured parameters having a Gaussian normal distribution. Here at each instant for one trial of experiment T, Compute the weight update of each sensor node depending on Gaussian input data spike together so that weight gets updated, but in the spatio-temporal domain correlation coefficient gets updated. Based on the weight update in each dual spiking, identify the enabling of sensor nodes and compute the energy, the residual and the total energy conserved. Performance measure is done with mean square error.

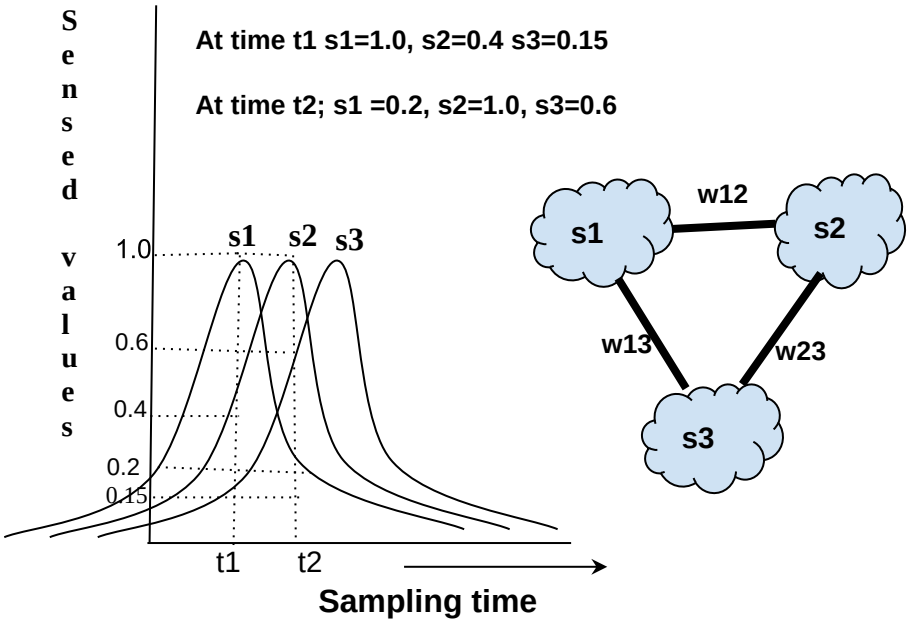

Figure 2: Three Sensor node's measured value with Gaussian distributed data values at time t1 and t2

Gaussian distribution: Most commonly used distribution over real numbers is normal distribution also known as the Gaussian distribution, defined by the equation 6.

$$\mathcal{N}(x; \mu, \sigma^2) = \sqrt{1/2\pi\,\sigma^2}\ \ \exp\ \left(-\frac{1}{2\sigma^2}(x-\mu)^2\right) \qquad \ldots\ldots\ldots\ldots (6)$$

The two parameters $\mu \in R$ and $\sigma \in (0, \infty)$ are control the normal distribution, that μ defines

the coordinate of the center peak and mean of the distribution, and standard deviation is the given by σ and the variance is given by σ^2 .

In the region of interest the number of sensor nodes deployed are ***s1, s2, s3*** upto ***si***. Consider the distribution of sensor nodes in triangular form, with ***s1, s2, s3*** as vertices of a triangle shown in figure 2. *Similarly the modelling or actual distribution can be considered for grid topology, hexagonal or any polygon with three dimensional field deployed sensor patterns.* Here with three sensor nodes ***s1, s2, s3***, it is assumed that the sensor node values are Gaussian distributed as shown in figure 2 again. Statistical correlation among sensor data with each time instant t1, also named as sensor data correlation (SDC), i.e., measured as an average value at the end of a certain time span fixed for the given application. (here the time span can be any duration like 1 minute, or 30 minutes or 1 hour, etc which is application dependent). Here the measured values are taken as the correlation values taken randomly at different time instant for the simulation (in figure 2, it shows two instant of time t1 and t2), and accordingly the updation of hebbian weights (with some random values for t1 and t2) that follows the equation 1 or 8.

The table content shows for the time duration T, observed for 5 samples in 5 time instances, and assumed a threshold value of 0.7 (can be varied) to correlate the average values of sensor data as correlation value. Similarly, the hebbian weight updates are shown with some random initial values and the updates are made based on hebb's rule. Here the computation is to optimize the number of similar sensor data with gradient of their values for the Gaussian distribution of the sensed data taken over a time period T. Here the measurements are correlated with the sensor nodes having some threshold value with spatial distribution.

Table I: Training data set based on correlation coefficient and weight update status

Measured average values at time	**correlation coefficient between s1 and s2 r1**	**correlation coefficient between s1 and s3 r2**	**correlation coefficient between s2 and s3 r3**	**weight update (synaptic link)between s1 and s2 w12**	**weight update between s1 and s3 w13**	**Weight update between s2 and s3 w23**
at t1	**0.9**	**0.4**	**0.1**	**0.849**	**0.379**	**0.272**
at t2	**0.9**	**0.4**	**0.7**	**0.955**	**0.256**	**0.467**
at t3	**0.9**	**0.3**	**0.4**	**0.343**	**0.247**	**0.871**
at t4	**0.8**	**0.4**	**0.6**	**0.124**	**0.673**	**0.387**
at t5	**0.7**	**0.8**	**0.7**	**0.825**	**0.756**	**0.988**

Table II: Training data set based on correlation coefficient and number of sensor nodes

Measured average values at time	**correlation coefficient between s1 and s2 r1**	**correlation coefficient between s1 and s3 r2**	**correlation coefficient between s2 and s3 r3**	**Number of sensor nodes OUTPUT Y**
at t1	**0.5**	**0.4**	**0.1**	**3 (all with less r_i)**
at t2	**0.9**	**0.4**	**0.7**	**2 r1 is hgh**
at t3	**0.1**	**0.3**	**0.4**	**3 (all with less ri)**
at t4	**0.8**	**0.4**	**0.6**	**2 (r1 ≅ r3 and high)**
at t5	**0.7**	**0.8**	**0.7**	**1 (all r_i's are high)**

Adapting Karl pearson's methods product moment for finding the correlation [9], the values found are shown in table II as r1, r2, and r3.

r1= cov(s1, s2) / (σs1 * σs2) r2= cov(s2, s2) / (σs1 * σs2) r1= cov(s1, s2) / (σs1 * σs2) …… (7)

σ_{si} **- deviation for gaussian distributed data for S_i^{th} sensor.**

cov (s1, s2) - the covariance between ***s1*** and ***s2.***

Based on the correlation the Hebbian learning will update the corresponding weights according to Hebb's rule [4] shown in the equation (8).

w12(new)=w12(old) +α s1.s2; w13(new)=w12(old) + α s1 . s3; w23(new)=w23(old) + α s2 . s3; …….. (8)

Based on the above weight values, the number of sensor nodes are defined as the output based on the redundancy and correlation between the sensor data.

The stochastic gradient descent and its variants are the most optimization algorithms which are used to obtain an unbiased estimate of the gradient by taking the average gradient on a mini-batch of n examples with i.i.d from the data generating distribution. The same distribution is shown in table II.

This minimizes the generalization error. Generally, the cost function for the stochastic gradient descent can be written as an average over the training set.

Cost function J(W)= $\mathbb{E}_{(r, y) \in \text{Hebbian data set}}$ and **Error function E = (f (s,w), y) …….. (9)**

The above equation defines an objective function with respect to the training set to minimize the objective function the expectation is taken across the data-generating distribution of data. **E** is the per example error function ***f (s,w)*** is the predicted output when input is ***s***, and data is the empirical distribution taken from the Hebbian learning patterns, ***y*** is the target output in the supervised learning.

V.2 PSEUDO CODE

Stochastic Gradient Descent (SGD) algorithm

Begin

{

initialize learning rate alpha

initialize the parameter w

while stopping criterion not met do

Sample an examples from the training set **{ r1, ... r_i }** with

corresponding targets **yi (table II)**

compute gradient estimate

} **end while**

V.3 Results and Discussions

Figure 3 shows convergence of the learning algorithm for about 500 epochs. Cost function decays as number epochs increase. Figure 4 below shows stochastic gradient descent the convergence of error for Gaussian data distribution of sensor data. Here the cost function converges after 10000 learning epochs. To adjust the weights, batch gradient will use all training samples whereas stochastic gradient will use randomly selected training samples.

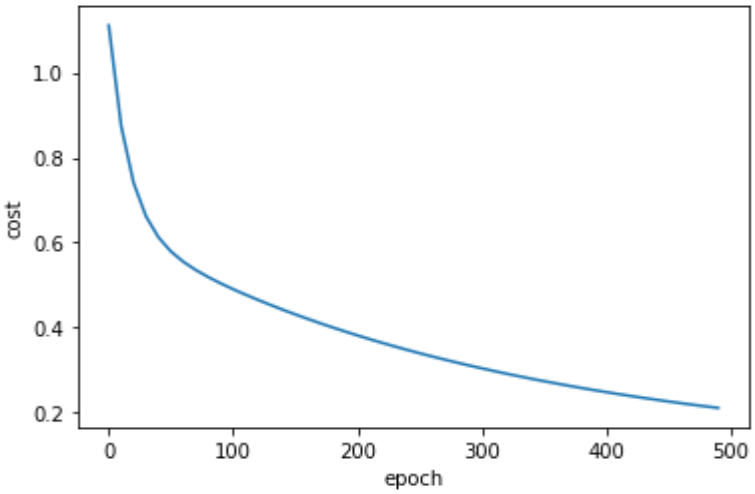

Figure 3: Normalized Error cost for batch training with respect to epochs.

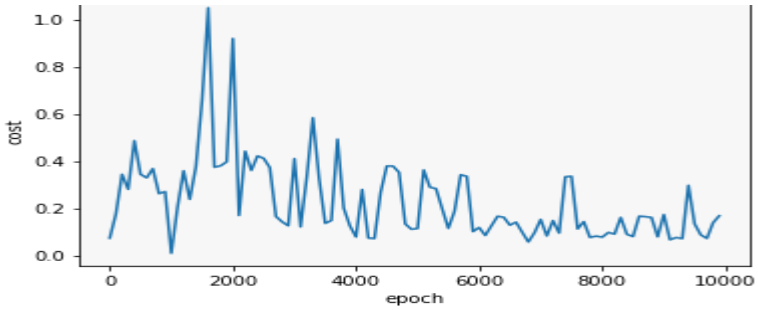

Figure 4: Normalized Error cost for stochastic gradient training with respect to epochs

Energy efficiency of the network with hebbian and without the hebbian is considered by considering the normalized values with respect to total lifetime of the WSN with maximum epochs as 100% life time and total energy in the network is the summation of residual energy in each sensor nodes. This graph in figure 5 depicts that the total residual energy depletes slowly with the hebbian model since the distribution of the energy and sensor usage is considered based on the gradient of environmental parameters to be measured. Also the stochastic gradient model increased the accuracy of the system.

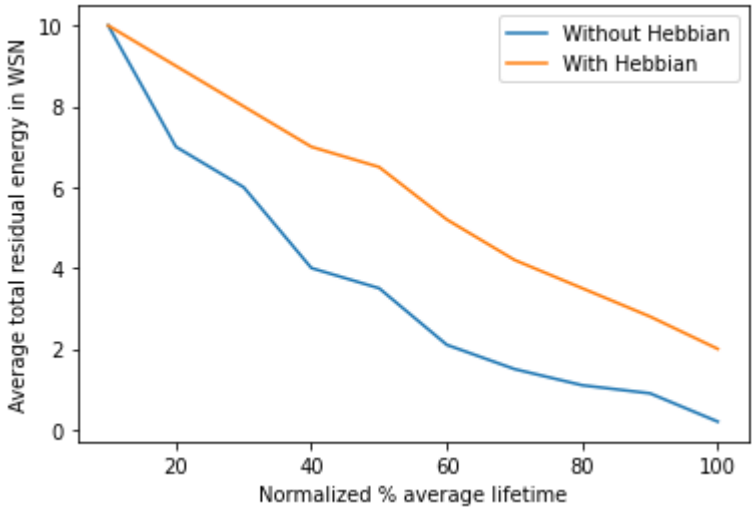

Figure 5: Energy conservation is WSN with and without Hebbian learning

VI. Conclusion

Sensor deployment investigation study is made to understand the challenges and functional requirements in the areas of WSN. Analytical modeling relating to the hebbian learning and its mapping to the sensor nodes are proposed to update the relationships between the sensor nodes based on the sensed values and their gradient to fix the number of sensors to be deployed in a given area to conserve energy and cost effectiveness. Analytical and mathematical models are considered to justify the deployment strategy using hebbian learning rules. Simulation experiments are carried out to evaluate the performance of the system with respect to energy saving and cost effective deployment of sensors. Overall performance is studied to improve the life time of the sensor networks and effective deployment of wireless sensor networks.

References

1. Wulfram Gerstner, Mathematical formulations of Hebbian learning. *Biological Cybernetics.* 87, 404–415, 2002.
2. Benoıt Siri, Hugues Berry, Bruno Cessac, Bruno Delord, and Mathias Quoy, et al. A mathematical analysis of the effects of Hebbian learning rules on the dynamics and structure of discrete-time random recurrent neural networks, *https://arxiv.org/abs/0705.3690*, 2008.

3. Kang et al., "Machine Learning-Based Energy-Saving Framework for Environmental States-Adaptive WSN*", IEEE Access* VOLUME 8, 2020.
4. Kusuma S M, Veena K N and Aparna N., "Effective deployment of sensors in wireless sensor networks using Hebian machine learning technique" *IEEE international conference on computing communication and intelligent systems ICCCIS-2021*, February 202.
5. Ibtihal Alablani, and Mohammed Alenazi, "EDTD-SC: An IoT Sensor Deployment Strategy for Smart Citie", *Sensors* 2020, 20, 7191; doi:10.3390/s20247191.
6. Francesco fraternal et.al, "ACES: Automatic configuration of energy Harvesting Sensors with Reinforcement Learning", *ACM Transactions on Sensor Networks*, Vol.16, No.4, July 2020.
7. Tajudeen O. Olasupo and Carlos E. Otero, "A Framework for Optimizing the Deployment of Wireless Sensor Networks", *IEEE Transactions on Network and Service Management,* Volume 15, Issue 3 , Sept. 2018, pp.1105 - 1118.
8. Dali Wei, Yichao Jin, Vural S, Moessner K, Tafazolli, R. ,'An Energy-Efficient Clustering Solution for Wireless Sensor Networks", *IEEE Transactions on Wireless Communications,*, Volume: 10 , Issue: 11 ,Page(s): 3973 - 3983, November 2011.
9. https://www.toppr.com/guides/business-mathematics-and-statistics/correlation-and-regression/karl-pearsonscoefficientcorrelation/?__cf_chl_managed_tk__=3b7PfYCITM_vn6dopVkn8g4.BOdPXAeKsKuQGEyCNpJs-1641807556-0-gaNycGzNCFE.
10. Gao, Q., Blow, K.J., Holding, D.J., Marshall, I. and Peng, X.H., 2006. Radio range adjustment for energy efficient wireless sensor networks", *Ad-Hoc Networks*, 4(1), pp.75–82, 2006.
11. Nihar Ranjan and Roy Pravin Chandra, "Energy dissipation model for wireless sensor networks: A survey", *International Journal of Information Technology,* 12(2), pp. 1-11, 2019.
12. Salah Zidi, Tarek Moulahi, and Bechir Alaya, "Fault detection in wireless sensor networks through SVM classifier", *IEEE sensors journal*, vol. 18, no. 1, January 1, pp 340-347, 2018.
13. Ye Jiang, Shuyan Xiao, Jian Liu, Bo Chen, Jiang,et al, "A Deterministic Sensor Deployment Method for Target Coverage "Journal of sensors", Article ID 2343891, 14 pages, 2018.
14. Yi zou and Krishnendu C, "Sensor deployment and target localization in distributed sensor networks", *ACM transactions on Embedded computing systems*, Vol. 3, no. 1, Feb 2004. pp. 61-91.