Leveraging Different Learning Rules in Hopfield Nets for Multiclass Classification

Pooja Agarwal, Abhijit J. Thophilus, Arti Arya, Suryaprasad Jayadevappa
PESIT Bangalore South Campus

***Abstract*—Retaining customer and finding the loyalty of an existing customer is an important aspect of today's business industry. In this paper the study of behavior of different machine learning rules on Hopfield Nets is conducted. This is a work in continuation w.r.t the classification of a real customer dataset into four different classes of Super Premium Loyal Customer (SPL), Premium Loyal Customer (PL), Valued Customer (VC), Normal Customer (NC). This model enhances the approach of finding the loyalty of customer using Hebbian learning and Storkey learning of Hopfield Neural Network (HNN). HNN is reported to give good accuracy with image datasets but with some data preprocessing on customer dataset, it is showing very reasonable accuracy of around 85%. The proposed framework is tested on Breast cancer dataset also and results are tabulated in the paper.**

***Keywords*—Customer loyalty; Hopfield Neural Networks; learning rate; momentum; Storkey learning; Hebbian learning; Softmax activation function**

I. Introduction

The successful operation of a commercial establishment primarily depends on the perspective of current and potential customers towards the establishment and the products or services it provides.

Customer Relationship Management, or CRM, is the systematic process of deriving solutions to meet customer demands, predict market trends and build marketing strategies by collecting, organizing and analyzing customer data [1]. The development of complex models that form highly accurate representations of data without being explicitly programmed to do so are what make the field of machine learning central to CRM. The techniques developed in this field may be applied either on a large scale, to provide real time predictions on large amounts of data, or on a small scale, identifying customer classes on transaction history.

For business the focus of customer retention is towards attracting prospective customers and ensuring existing customers continue to receive the quality of service they expect [2], [3]. This paper is an attempt towards modeling the records of customers visiting a grocery store to predict the loyalty bracket of a new customer. The data pertaining to a venture such as this is characterized by few attributes and a relatively small number of records collected through a online survey. The model analyzed in this paper is the Hopfield Network. Introduced in 1982, by J.J. Hopfield. The model explores the different aspects and learning in Hopfield Nets and gives the better results than the solution given in literature [4]. The simplicity of the model and ease of training make it a prospective candidate in the analysis of small data sets using minimal effort. Indeed, this is exactly what the results of the experiment outlined in this paper indicate. The Hopfield Network is able to successfully classify an arbitrarily distributed input data in spite of having a very simple and quick training technique [5]. The organization of paper is as follows: Section 2 highlights the work done in related area. Section 3 explains the proposed model, different learning rules used in HNN and softmax activation function. Sections 4 and 5 talks about the experimental setup and results and testing of the model and Section 6 is the conclusion of the work.

II. Related Work

To classify the customer in different categories of loyalty, number of factors need to be taken care. It is important for any organization to retain their existing customer than getting the new ones [1]. The cost of acquisition will always be much higher than the cost of retention.

The model considered in this paper is a Hopfield neural network, a recurrent neural network (RNN) that belongs to a broad family of computational models called Neural Networks (NNs) for multiclass classification. Most of the Neural network models use a significant amount of information for the training. Cabeza et al. [6] proposed a diagnostic system based on a Hopfield neural network to overcome the inconvenience due to different factors such as data losses in the data acquisition systems with successful performance.

Models of recurrent neural networks come in different flavors. Samuel et al. [7] distinguished between discrete and continuous values of the neurons and also between deterministic and stochastic updates representing the difference between binary and real values.

The activation function is necessary to scale the outputs of the neurons in neural networks, and to introduce a non-linear relationship between the input and output of the neuron. The sigmoid function, given by,

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

is the most frequently used activation function, although popular alternatives include the hyperbolic tan, piecewise linear, and the binary step functions [8].

However, due to the structure of the data set under consideration, both the sigmoid and hyperbolic tan functions cause the network to converge to a single spurious state regardless of the input pattern. Therefore, to overcome this, the Hopfield network has been adapted to use the softmax activation function (illustrated in Section III-D).The proposed framework uses Hopfield network with hebbian and storkey learning.

According to Ue-Pyng et al. [9], Neural networks have been characterized in various categories according to many relevant features. HNN is a type of recurrent neural network (RNN) represented by a set of interconnected neurons, which asynchronously keep on updating their weights.

Hopfield Neural Network (HNN) [5] is a neural network with symmetrical connections between binary neural units. These networks can learn by imposing the patterns on the network and gets modified by Hebbian learning. It also tunes the network to minimize the energy function across the entire network and finally stabilize.

Amos Storkey [10] proposed a new learning rule and a heuristic calculation of the absolute capacity of learning algorithm that provides a good measure of capacity for finite network.

Xiao [11] summarized the Storkey Learning Rules for the Hopeld Model and evaluated performance relative to other learning rules. Hopeld Models are normally used for auto-association, and Storkey Learning Rules have been found to have good balance between local learning and capacity.

The proposed model improves the accuracy of the classifier over the model proposed in [4] With a combination of Hebbian learning and softmax activation function.

III. Proposed Model

A. Data set Description and Preprocessing

The dataset used in this paper is real customer dataset of a retail store. It is dataset collected through an online survey (https://goo.gl/p4LZjK). This dataset has a collection of more than 600 instances that are described by four attributes Total Expenditure (TE), Life time of customer (LT), Frequency of visit (FV), Mode of Payment (MP) [4].

A second data set, the Breast Cancer Wisconsin [12] (Diagnostic) Data Set hosted at the UCI Machine Learning Repository, has been considered as contrasting data to the customer data set for the proposed model. The data set has binary classes, where each observation corresponds to a digitized image of a fine needle aspirate (FNA) of a breast mass. With both of the data sets the proposed model gives promising results of around 85% accuracy. Hopfield networks inherently have a low recall capacity for their stored patterns. Although in a continuous network some extent of fault tolerance is permissible, there is an upper bound on the number of records that can exist in the state space of the network. Thus, instead of storing the entire data set in the network, a subset of four records is taken that consists of the median of records from each class, as

- these records are the most accurate representatives of the data members of its class, and
- these records have their corresponding state space vectors (The visible neurons in input layer) distant from each other,

is stored in the network. These representative records are hereby referred to as *prototypes*.

The method of extracting prototypes employed in this paper is through the application of Partitioning Around Medoids (PAM) variant of the K-Medoids algorithm. A high level description of the procedure used to select the prototypes is given by Algorithm 1.

The input data X consists of the training records from the data set. Prototypes are generated class-wise. The first step involves extracting all records belonging to a given class c into X_c. Depending on the number of prototypes desired, n_p, X_c is partitioned around n_p medoids. These medoids form the prototypes of their corresponding class. This is repeated over all classes, and the medoids for each class are extracted and stored in P, which is returned by the algorithm on completion.

Algorithm 1 Generating Prototypes

Input:
Input data X,
Number of prototypes n_p,
Function $pam(X, n)$ that partitions X around n medoids.
Output:
Set of prototypes P.

$P \leftarrow \{\}$
for all classes c **do**
  $X_c \leftarrow$ records in X belonging to class c
  $parts \leftarrow pam(X_c, n_p)$
  $meds \leftarrow$ retrieve n_p medoids from $parts$
  $P \leftarrow P \cup meds$
end for
return P

B. Hopfield Network

Complex interactions occur when the network is taken as a whole, and although the individual components seem simplistic, the emergent property of the network is universal approximation.

Conventional Neural Networks, called Feed-Forward Neural Networks are only connected to the immediately succeeding layer. Despite this, universality holds for these networks and time-series data cannot be easily handled by such feed-forward networks. In Recurrent Neural Networks, in contrast neurons may be interconnected in an arbitrary manner, thereby allowing directed cycles to form in the network.

The Hopfield network is a recurrent neural network where the connection weights are symmetric, and no neuron is connected to itself. The weights of the network are represented by an $N \times N$ matrix W, where N is the number of neurons in the network. W is symmetric, with the diagonal elements set to 0. The elements of W are initialized by a learning rule (Section III-C), an algorithm applied on the training data to yield appropriate connection strengths. The Hopfield network considered in this paper is the continuous variant of the model proposed by J.J. Hopfield [5]. The visible neurons in input layer of the Hopfield network is the set of vectors V, for which the components of each vector $\vec{v}$ are given by,

$$u_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} w_{ij} v_j \tag{2}$$

$$v_i = g(u_i) \tag{3}$$

where, u_i is the input to the i^{th} neuron, g is the activation function, v_i is the i^{th} component of $\vec{v}$ and, w_{ij} is the weight of the synapse between neurons i and j.

The state space of the Hopfield network is characterized by basins of attraction with attractors being state vectors corresponding to the minima of an energy function[13] defined by Hopfield to be,

$$E = -\frac{1}{2n}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}v_i v_j + \frac{1}{n}\sum_{i=1}^{n}\int_0^{v_i} g^{-1}(v)\mathrm{d}v \quad (4)$$

Consider an input pattern $\vec{\xi}$ with the corresponding input state vector $\vec{v} = g(\vec{\xi})$[14]. Hopfield showed that if $\vec{v}$ is within the basin of some attractor $\vec{\phi}$, $\vec{v}$ will converge to $\vec{\phi}$.

C. Learning Rules

In order to actually apply the Hopfield network to the problem of classification, the weights of the network must be initialized such that the prototypes discussed in Section III-A correspond to attractors in the network's state space. The equations that govern weight initialization form the learning rule for the model. Learning rules differ in the approach taken to ensure the minima of (4) are state vectors corresponding to training patterns. The capacity of a network is the maximum number of patterns that can be stored in the network. Network capacity varies among different learning rules.

1) Hebbian Learning Rule: Hebbian learning was the learning rule employed in the standard Hopfield network described by Hopfield. The learning rule is one shot, along with being incremental, immediate, and local in space and time. This ensures that a Hopfield network can be trained in finite time. If the number of patterns in the network is below its capacity, additional patterns may be stored making it suitable for time-series data. The learning rule is given by,

$$w_{ij} = \frac{1}{N}\sum_{p=1}^{n_p} v_i^p v_j^p \quad (5)$$

where, n_p is the number of patterns to be stored, $v_i^p = g(\xi_i^p)$ and $\vec{\xi^p}$ is the p^{th} pattern to be stored in the network.

For binary valued data, the Hebb learning rule has been shown to have a capacity of $0.14N$.

2) Storkey Learning Rule: Storkey proposed a learning rule that improved on the capacity of the original Hopfield rule. The Storkey rule, like Hebbian learning, is also local, incremental and immediate, thereby allowing patterns to be added and removed after the network has been trained. The improved capacity of the network, for binary values, was shown by Storkey to be five times greater than the Hebbian learning rule [15] at network sizes of a million neurons. The learning rule is given by the following recursive definition,

$$\begin{aligned} w_{ij}^0 &= 0 \\ w_{ij}^p &= w_{ij}^{p-1} + \frac{1}{N}\left(v_i^p v_j^p - v_i^p h_{ji}^p - h_{ij}^p v_j^p\right) \end{aligned} \quad (6)$$

where h_{ij}^p is a local field at neuron i given by,

$$h_{ij}^p = \sum_{\substack{k=1 \\ k\neq i,j}}^{N} w_{ij}^{p-1} v_k^p \quad (7)$$

and $1 \leq p \leq n_p$ corresponds to the next pattern to be stored.

D. Softmax Activation

The activation function defines how the weighted input to a neuron is transformed to produce its output. Although it may be a linear function, often the choice of a non-linear transformation is warranted to introduce non-linearity to the system. Traditionally, the sigmoid function has been used, especially in multilayer perceptron networks. However, due to the nature of the data, and the Hopfield network architecture, the sigmoid activation fails to cause the network to converge toward any desired state. For the network to converge to one of these desired states, it has been observed, that the number of neurons must be significantly higher than the number of patterns to be stored in the network. The number of neurons is dictated by the dimensions of the input data to the system. Further, the feature domains in the data set under consideration, and the weight matrix generated for the data by both learning rules is non negative. Therefore, for a sigmoid, or tanh, activation function, the input only comprises one half the range of its domain. Since both the sigmoid and tanh are monotonically increasing functions, the output of the neuron is restricted to red half image (Fig. 1) of the activation function.

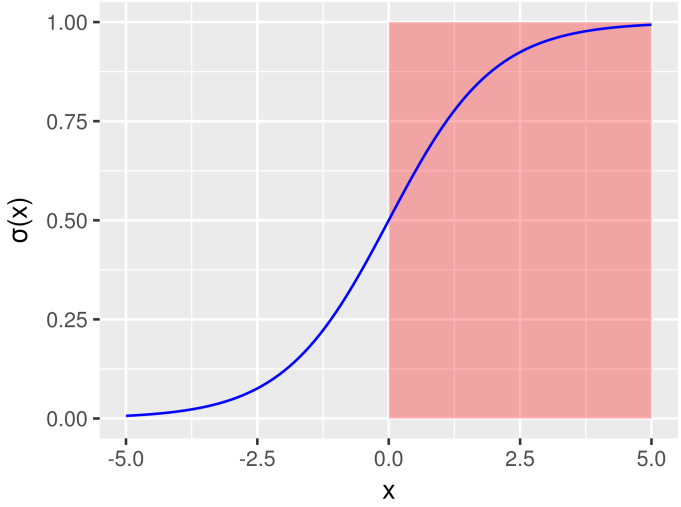


Fig. 1. Sigmoid Activation Function. The range of values within which the activation function operates is indicated by the red shaded region.

This problem can be solved by using the Softmax activation function,

$$S(u_i) = \frac{e^{-u_i}}{\sum_{j=1}^{d} e^{-u_j}} \quad (8)$$

The Softmax function balances itself across the network. The Softmax function, however, requires the original definition

of the Hopfield network to be tweaked. The domain of a Softmax function is a multidimensional vector, and therefore operates on the entire network state. Therefore, instead of applying the activation function on one neuron at a time, every time a neuron is updated, the Softmax function is applied on the current network state, with the value of the updated neuron set to its weighted input.

The final value of the output neuron does not depend on the range of inputs fed to the neuron, rather it depends on the extent to which the input differs from the activation values of the other neurons. As the entire image of Softmax activation function is utilized by the network for its output, causing the network to converge to one of the desired states, even with a few neurons.

E. Post-training

Once the prototypes of the network have been generated, they are used to initialize the weights of the network by one of the training rules i.e. Hebbian or Storkey learning. These initialized weights form the structure of the network and are used to predict the class labels of test data. Data is fed to the Hopfield network according to (2) and (3) and the final state of the network is taken as the output. However, the output of a test vector alone is not enough to decide its class label. In order to do so, a reference and a proximity metric is required. The outputs of the prototypes form the reference, and although any distance metric may be used, the Euclidean distance has been considered in this paper. Algorithm 2 illustrates the generation of the reference vectors.

Algorithm 2 Generate References

Input:
Set of prototypes P,
Weight matrix W,
Activation function g.
Output:
Set of stable states R corresponding to P.

1: $R \leftarrow \emptyset$
2: **for** ($\vec{p}$ in P) **do**
3: $\vec{u} \leftarrow \vec{p}$
4: $\vec{v} = g(\vec{u})$
5: **repeat**
6: equation (2)
7: equation (3)
8: **until** network is stable.
9: $\vec{\phi_p} \leftarrow \vec{v}$
10: $R \leftarrow R \cup \vec{\phi_p}$
11: **end for**

Each vector in P is fed to the network, until the values of $\vec{v}$ do not change, i.e., network has converged, after which the final state of the network is stored. The final states corresponding to all the prototypes are then compiled into a set returned by the Algorithm 2.

Once the references have been generated and the proximity metric decided, the next step is to predict the labels of test data. The process is similar to the generation of references. Each vector τ in the test set is fed to the network and the corresponding final state is generated. For each reference vector, the Euclidean distance from the final state is determined, and the reference vector corresponding to the minimum vector is taken to be the class representative of the test vector. The class label of this reference is assigned to the test vector.

Algorithm 3 Classifying a Test Sample

Input:
Set of prototypes P,
Weight matrix W,
Activation function g,
Reference set R for P as generated by Algorithm 2,
Test sample τ,
Proximity measure $prox$.
Output:
Class label of test sample τ.

1: $\vec{u} \leftarrow \vec{\tau}$
2: $\vec{v} \leftarrow g(\vec{u})$
3: **repeat**
4: equation (2)
5: equation (3)
6: **until** network is stable.
7: $\vec{\phi_\tau} \leftarrow \vec{v}$
8: $min\_dist \leftarrow min\{prox(\vec{\phi_p}, \vec{\phi_\tau}) \ \forall \ \vec{\phi_p} \in R\}$
9: $\vec{p_c} \leftarrow \vec{p}$ such that, $prox(\vec{\phi_p}, \vec{\phi_\tau}) = min\_dist$
10: **return** class label represented by p_c.

From Algorithm 3, lines 8, 9 and 10 form the decision rule for the model.

IV. Experiment Setup

The aforementioned algorithms have been implemented and were executed in the R programming environment. Both data sets have been normalized by feature scaling. If X is the input data set, let $\vec{x_i}$ be the i^{th} column vector in X, then the normalized values of $\vec{x_i}$ are given by,

$$\vec{x_i}' = \frac{\vec{x_i} - min(\vec{x_i})}{max(\vec{x_i}) - min(\vec{x_i})} \tag{9}$$

The data sets were split into training and test samples at a $70:30$ ratio[16]. The prototypes corresponding to each class label were extracted from the training data set by partitioning around medoids according to Algorithm 1. These prototypes were then used to initialize the weight matrix to store their corresponding state vectors using either hebbian or storkey training rules. Their corresponding state vectors, or references, were then generated by Algorithm 2. Once the model was constructed, each observation in the test data set was classified by Algorithm 3. These predicted labels were then matched with their true labels and an accuracy score was generated. This experiment is illustrated in Algorithm 4.

As an artifact of the implementation, in order to gauge the stability of the network after a given iteration, a threshold was introduced. For a change in the state of the network to be considered significant, the sum of squared differences of the current state from the previous state must be greater than or equal to the threshold. If not, the network reverts to the previous state. Further, the number of significant figures in the threshold was reflected onto the state vector of the network, i.e., the state vector was rounded to the number of significant

Algorithm 4 Experiment Procedure

Input:
Input data set X,
Number of prototypes per class n_p,
Learning rule $lrule$,
Activation function g.
Output:
Classification Accuracy.

shuffle X
split X into a training set $train$ and a test set $tests$
$P \leftarrow$ extract prototypes from $train$ using Algorithm 1
if $lrule$ is "Hebbian" **then**
  initialize the weight matrix using P by equation (5)
else if $lrule$ is "Storkey" **then**
  initialize the weight matrix using P by equation (6)
end if
$R \leftarrow$ generate the references for P using Algorithm 2
$corrects \leftarrow 0$
$total \leftarrow 0$
for all obs in $tests$ **do**
  $ypred \leftarrow$ classify obs by Algorithm 3
  $ytrue \leftarrow$ the actual label of obs
  if $ypred = ytrue$ **then**
    $corrects \leftarrow corrects + 1$
  end if
  $total \leftarrow total + 1$
end for
$accuracy \leftarrow \dfrac{corrects}{total}$
return $accuracy$

digits of the threshold. The threshold values, along with the number of prototypes per class, for each data set on both the learning rules, were set through heuristic fine tuning performed over several iterations of the experiment. These values are listed in Table I and explained in Section V.

The experiment illustrated in Algorithm 4 was repeated for 100 iterations and the accuracy for each iteration was recorded.

TABLE I. HYPER-PARAMETERS FOR THE IMPLEMENTATION

	Customer		Cancer	
	Hebb	**Storkey**	**Hebb**	**Storkey**
Threshold	0.001	0.1	0.0001	0.0001
Prototypes	7	3	4	4

V. RESULTS

The results of the experiment described in Section IV for Hebbian learning and Storkey learning, for both data sets, has been aggregated and layered out in Table II.

TABLE II. CLASSIFICATION ACCURACY

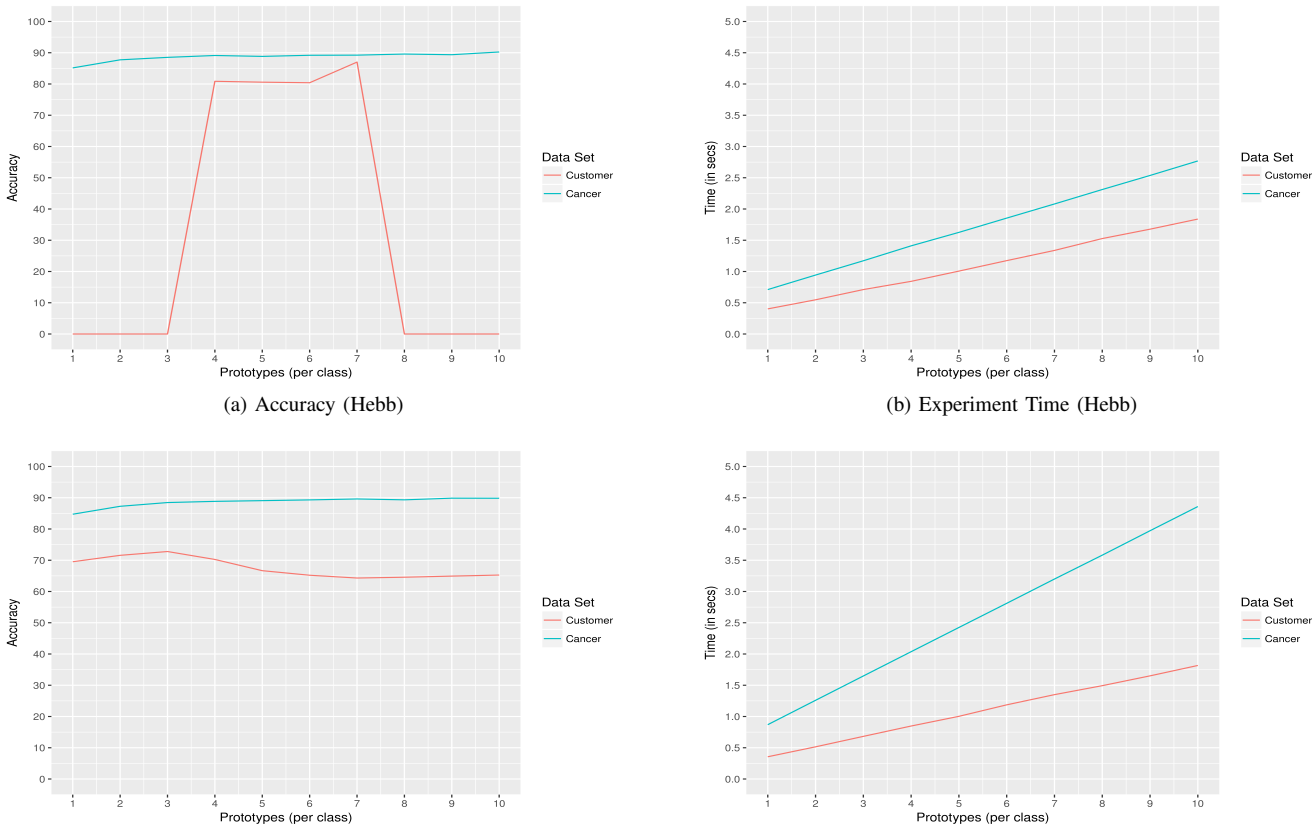
	Customer		Cancer	
	Hebb	**Storkey**	**Hebb**	**Storkey**
Max.	93.7%	87.4%	94.1%	95.3%
Mean	87.2%	72.3%	89.2%	87.5%
Min.	77%	56%	82.5%	77.8%

These results indicate the network successfully converges close to one of the references given a test sample. Due to the higher dimensionality of the cancer data set, accuracy is marginally better. However, the simplicity of the model makes it equally successful for customer loyalty classification.

In order to fix the number of prototypes, the experiment was extended to compare the effect of employing an increasing number of prototypes per class. Although the training time increased, it was observed, that this did not necessarily correspond to an increase in the performance of the network. The mean accuracy of the Cancer data set, for the Hebbian learning rule (Fig. 2a) and the Storkey learning rule (Fig. 2c) grows very slowly over the number of prototypes and stagnates close to 90% accuracy. The prototypes were fixed at 4 to provide a convenient speed-accuracy tradeoff.

For the Customer data set, however, the results drastically vary among the learning rules. It was observed, that when the network was trained for the Hebb learning rule at a threshold of 0.001, up until 4 prototypes, the references (as outlined in Section III-E) were identical for every input vector fed to the network. Therefore, no classification was possible. This occurs when trained by the Storkey learning rule as well at a 0.001 threshold, although this effect persists up to 7 prototypes. After 4 prototypes, for the Hebb rule, the references corresponding to different prototypes are distinct across classes and therefore, the network can function as a classification model with a peak accuracy at 7 prototypes. Beyond 7, the network exceeds its capacity and all input states once again converge to the same state. For the Storkey learning rule, although the network classifies at more than 7 prototypes, the classification accuracy is poor. Therefore, the threshold was set to 0.1 and the number of prototypes determined from the graph, i.e. 3. For this threshold value and 3 prototypes, the performance of the Hebb learning was observed to be the same as that of the Storkey rule.

The comparison of Hebbian and Storkey learning rules have been illustrated in Fig. 3. Here, the accuracies per prototype has been displayed as a bar graph for a clear distinction in the performance variation among the learning rules.



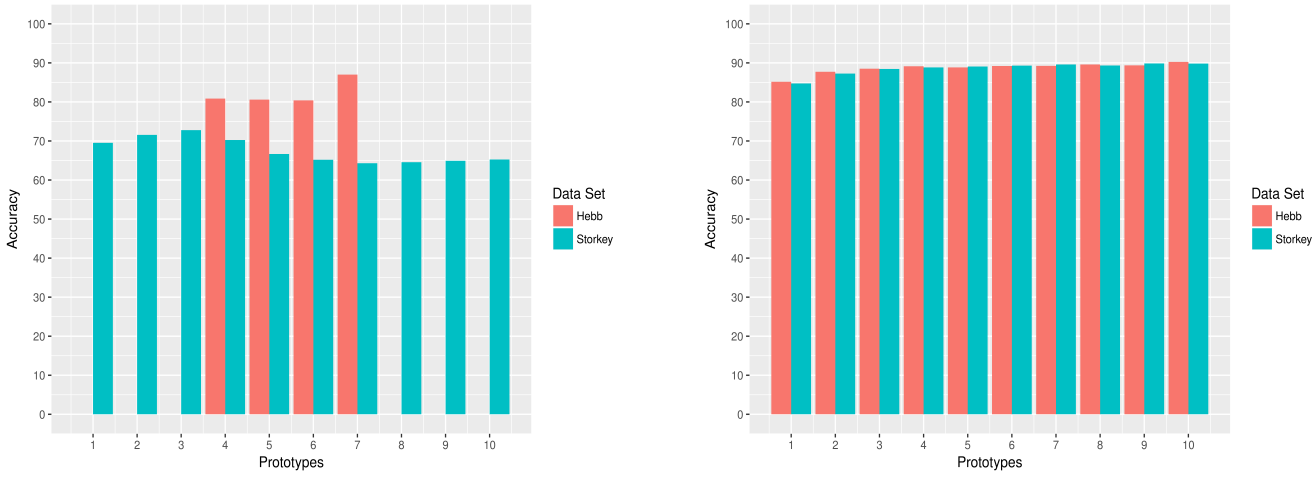
(a) Accuracy (Hebb)

(b) Experiment Time (Hebb)

(c) Accuracy (Storkey)

(d) Experiment Time (Storkey)

Fig. 2. Model Performance with Prototypes. Graphs (a) and (c) indicate the change in mean accuracy as the number of prototypes are increased for both learning rules. Graphs (b) and (d) indicate the change in execution time of the complete experiment as the number of prototypes per class is increased.



(a) Customer Loyalty

(b) Breast Cancer Wisconsin (Diagnostic)

Fig. 3. Learning Rule Comparison. Graph (a) shows the comparison chart for the Customer Loyalty data set, and graph (b) shows the same for the Breast Cancer data set. The red bars indicate the accuracy for the Hebb learning rule, while the green indicate Storkey. As seen, there is not much difference in the performance of either across the number of prototypes.

VI. Conclusion

Implementation of proposed framework provides evidence of improved accuracy of multi class classification of customer dataset and binary classification of cancer dataset. For customer dataset, Hopfield Nets with Hebbian and Storkey learning along with Softmax function provides an accuracy of average 87% and 72% and maximum 93.7% and 87.4%. For cancer dataset it represents the accuracy of average 89% and 87% and maximum 94% and 95.3%. The accuracy achieved in literature[8] for the customer problem was 87%. In this paper, for both the cases of customer and cancer datasets it is improved. With this kind of accuracy, we are further investigating the role of Restricted Boltzmann machines and

Deep nets towards customer dataset.

REFERENCES

[1] Shaw, Colin, and Ryan Hamilton. "Imperative 7: Realize the Only Way to Build Customer Loyalty Is through Customer Memories." The Intuitive Customer. Palgrave Macmillan UK, 2016. 141-159.

[2] Arya, Arti, and Agarwal Pooja. "Fuzzy Decision Tree based Automatic Classifier for Customer Loyalty." In proc. of Intl. COnf. on Data Management, 2010.

[3] Arya, Arti, Agarwal, Pooja et al. "Automatic Fuzzy Classification tool for Customer Loyalty using Gaussian Membership Function." Data Mining and Knowledge Engineering 2.7 (2010): 168-173.

[4] P. Agarwal, J Surya Prasad and A Arya. Article: A Naïve Hopfield Neural Network based Approach for Multiclass Classification of Customer Loyalty. Communications on Applied Electronics 2(5):36-43, July 2015. Published by Foundation of Computer Science (FCS), NY, USA

[5] Hopfield, John J. "Neural networks and physical systems with emergent collective computational abilities." Proceedings of the national academy of sciences 79.8 (1982): 2554-2558.

[6] Cabeza, Raquelita Torres, et al. "Fault Diagnosis with Missing Data Based on Hopfield Neural Networks." Mathematical Modeling and Computational Intelligence in Engineering Applications. Springer International Publishing, 2016. 37-46.

[7] Muscinelli, Samuel P., Wulfram Gerstner, and Johanni Brea. "Exponentially long orbits in Hopfield neural networks." Neural computation (2017).

[8] Oda, Tetsuya, et al. "A Neural Network Based User Identification for Tor Networks: Comparison Analysis of Different Activation Functions Using Friedman Test." Network-Based Information Systems (NBiS), 2016 19th International Conference on. IEEE, 2016.

[9] Wen, Ue-Pyng, Kuen-Ming Lan, and Hsu-Shih Shih. "A review of Hopfield neural networks for solving mathematical programming problems." European Journal of Operational Research 198.3 (2009): 675-687.

[10] Storkey, Amos. "Increasing the capacity of a Hopfield network without sacrificing functionality." Artificial Neural Networks—ICANN'97 (1997): 451-456.

[11] Hu, Xiao. "Storkey Learning Rules for Hopfield Networks." (2013).

[12] https://archive.ics.uci.edu/ml/datasets/breast+cancer UCI Machine Learning Repository, Breast Cancer data set. This breast cancer domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing the data.

[13] Storkey, Amos J., and Romain Valabregue. "The basins of attraction of a new Hopfield learning rule." Neural Networks 12.6 (1999): 869-876.

[14] Hopfield, John J. "Neural networks and physical systems with emergent collective computational abilities." Proceedings of the national academy of sciences 79.8 (1982): 2554-2558.

[15] Swingler, Kevin. "On the Capacity of Hopfield Neural Networks as EDAs for Solving Combinatorial Optimisation Problems." IJCCI. 2012.

[16] Stranieri, Andrew, and John Zeleznikow. Knowledge discovery from legal databases. Vol. 69. Springer Science & Business Media, 2006.