A Penalized Maximum Likelihood Approach to the Adaptive Learning of the Spatial Pooler Permanence

Ernest Fokoue, Lakshmi Ravi, and Dhireesha Kudithipudi
NanoComputing Research Lab
Rochester Institute of Technology

***Abstract*—Hierarchical Temporal Memory is a machine learning algorithm for spatio-temporal information processing. One of the key functional units in this algorithm is the spatial pooler, which has been demonstrated to be efficient in classification, dimensionality reduction and for preprocessing non-spatial inputs. Formalization of the spatial pooler is proposed in recent literature. In this work, we present a principled theoretical formulation of the spatial poolers underlying learning scheme. Constraints from the active connected and emmeshed columns in a spatial pooler are included in the analysis. It has been observed that locally adaptive learning enhanced the performance of the spatial pooler as a feature selector.**

Keywords—Hierarchical temporal memory (HTM), Spatial Pooler, Bernoulli distribution, Likelihood function, Constrained Optimization, Penalty function, Regularization, Stochastic Gradient Ascent, Local Adaptive Update

I. Introduction and Motivation

Hierarchical Temporal Memory (HTM), proposed by Hawkins [1], is based on the ”memory-prediction” principle of the neocortex. Therefore, the algorithmic foundations of HTM are akin to the structural and information processing properties of the neocortex [1]. The advantage of such a system is that the hierarchical and homogeneous processing units are used for prediction on spatio-temporal signals, similar to the minicolumns in the cortex [2]. Developing models and architectures of hierarchical temporal memory is important in broadening the scope of machine intelligence. Several research groups have explored the algorithmic framework proposed by Numenta [3], and demonstrated applications in time-series data, predicting taxi passenger count in NewYork City, and anomaly detection [3], [4]. There are two critical functional components in HTM that enable continuous online learning, spatial pooler and temporal memory. These two substructures capture the spatial and temporal patterns of the input data. The spatial pooler is responsible for mapping an input data onto a sparse distributed representation and the temporal memory is responsible for learning sequences and making predictions [1], [5]. The focus of this research is on the spatial pooler, which maps encoded input into a higher-dimensional space. Output generated by a spatial pooler region is an N-dimensional sparse binary vector or a sparse distributed representation [6].

Previous studies have shown that spatial pooler can be used for classification, dimensionality reduction, and novelty detection among others. There is also observation that spatial pooler performs a type of unsupervised competitive learning [7] and uses a form of vector quantization [8] resembling self-organizing maps [9], [10]. In our earlier work, Mnatzaganian *et al.* [10] proposed a complete mathematical formalization of the spatial pooler for machine learning tasks. Prior to that, few researchers have shown certain aspects of the formalization with respect to the network initialization [11], vector quantization [12], and matrix notation without boosting [13]. Numenta, has also provided initial basis for the temporal memory formalization in its recent publication [14]. All these research efforts are important in formalizing the algorithm, however there is no specification of an objective function for a spatial pooler. Mnatzaganian's work *et al.* [10], introduces a plausible learning mechanism and lays a good foundation for mathematical optimization of the spatial pooler. In this work, we extend on our previous work in [10] and propose a principled theoretical formulation of the spatial pooler underlying learning scheme. This can be solved as Constrained Maximum likelihood estimation problem with locally adaptive update mechanisms. The proposed formalization is analyzed with MNIST dataset.

The rest of the paper is organized as follows: Sections II provides spatial pooler definition and terminology, section III describes the proposed specification of the spatial pooler, section IV presents the analysis of the objective function with two different datasets, followed by conclusions in section V.

II. Spatial Pooler Definition

Spatial Pooler (SP) is the critical learning component of HTM, under a single unifying framework [10]. In HTM, a region depicted in Fig. 1, consists of multiple columns. The columns pool the spatial information. Each column can consist of multiple cells. These cells connect the feedforward input, via proximal dendrites to synapses to a column. The distal segments connect cells within a region and a full system is formed by connecting regions hierarchically.

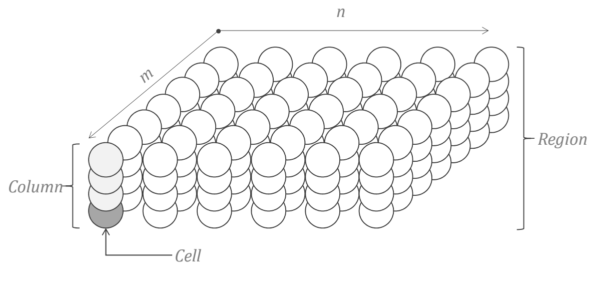

Fig. 1. HTM Structure showing a single region with multiple columns and cells within a column [5].

The learning component of the spatial Pooler is represented in the form of a matrix, which is known as a permanence matrix. It has fixed number of columns and the number of rows are determined from the number of the features in input space. The input to the spatial pooler is represented as a binary matrix.

TABLE I. Parameters in the spatial Pooler

Parameter	Description
ϕ	Permanence Matrix
M	Number of rows in Permanence Matrix
Q	Number of columns in Permanence Matrix
P_{inc}	Permanence increment amount
P_{dec}	Permanence increment amount
ρ_s	Connection threshold
ρ_d	Synaptic threshold

A. Initialization

The input is a binary feature vector V of size P, which is the representation of an image or digit or any object of interest. The p-dimensional input vector $\mathbf{V}$, should be converted in to matrix, with dimensions same as the permanence matrix Φ. It is done by the process of synaptic mapping.The synapticmap is a matrix, whose dimensions are same as the permanence matrix.
The synaptic map is constructed only once in the beginning of the process, as follows,
for every row, we randomly select values from $1 \cdots P$ with out replacement. This selection indicates randomly selecting M features from the binary feature vector. This is initialized only once in the beginning of the learning process. The input is then presented in matrix form, namely X, with $X_{mq} = V[\texttt{synapticmap}_{mq}]$

B. Learning Steps in Spatial Pooler

1) Overlap Phase: This phase determines the overlap between the input X, and the permanence matrix. At the end of this phase, we have a boolean active connected matrix. A cell X in this matrix is true if it is active and the corresponding permanence cell is connected.

2) Inhibition Phase: This phase computes the active connected quotient of every row in the active Connected matrix, which is the count values in every row that are true. The rows whose active connected quotient meets the required constraint(ACE Constraint) is selected for update in the next phase. ACE Constraint is discussed in detail in III

3) Update Phase: This phase updates the values of the Permanence matrix. For every row, selected in the inhibition phase, we update the value of the permanence cell by the following equation.

$$\phi_{mq} = \phi_{mq} + \texttt{P}_{\texttt{inc}} X_{mq} - \texttt{P}_{\texttt{dec}}(1 - X_{mq})$$

This update equation increments the permanence cell if the input is active, else decrements the cell by the increment and decrement factor respectively.

III. Specification of the Spatial Pooler Objective Function

The seminal algorithm underlying the spatial pooler operates essentially on matrix-valued inputs of the form

$$X = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1q} & \cdots & X_{1Q} \\ X_{21} & X_{22} & \cdots & X_{2q} & \cdots & X_{2Q} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ X_{m1} & X_{m2} & \cdots & X_{mq} & \cdots & X_{mQ} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ X_{M1} & X_{M2} & \cdots & X_{Mq} & \cdots & X_{MQ} \end{bmatrix}$$

whose entries $X_{mq} \in \{0, 1\}$ are binary, with each row, referred to here as X_m, represents a column vector modeling a minicolumn of the neocortex. One of the central goals of the HTM spatial pooler's learning algorithm is to sequentially (online) learn the values of the entries of a matrix known as the **permanence** matrix, which is isomorphic to the input matrix X, namely

$$\Phi = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1q} & \cdots & \phi_{1Q} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2q} & \cdots & \phi_{2Q} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \phi_{m1} & \phi_{m2} & \cdots & \phi_{mq} & \cdots & \phi_{mQ} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \phi_{M1} & \phi_{M2} & \cdots & \phi_{Mq} & \cdots & \phi_{MQ} \end{bmatrix}$$

Note that $\phi_{mq} \in (0, 1)$. From a pure statistical perspective, the underlying estimation (learning) may seem to be of over-saturated, since we have as many entries to learn in Φ as we do have data in X. However, that is not the case, because many such binary matrices are sequentially fed to the learning machine to help update the permanence matrix Φ. In other words, from a statistical estimation perspective the sequential provision of data brings in all the information needed to make the learning of Φ a well-posed problem in Hadamard's sense. Central to the underlying HTM idea are the concepts of activation and connectivity, related to the input matrix X and the permanence matrix Φ respectively. Note that X and Φ are both $M \times Q$ matrices. Now, for a given HTM column, we define the Q-dimensional vector $Y_m = (Y_{m1}, Y_{m2}, \cdots, Y_{mQ})$ of indicators of cell connectivity, such that

$$Y_{mq} = 1_{\{\phi_{mq} > \rho_s\}} = \begin{cases} 1 & \text{if } \phi_{mq} > \rho_s \\ 0 & \text{otherwise} \end{cases}$$

Using the HTM column $X_m = (X_{m1}, X_{m2}, \cdots, X_{mQ})$, we define inner product

$$Z_m = X_m^\top Y_m = \sum_{q=1}^{Q} X_{mq} Y_{mq}.$$

Clearly, $Z_m \in \{0, 1, 2, \cdots, Q\}$ is a natural number representing the number of cells in the m^{th} column that are both active and connected. The quintessential idea underlying the updating of the permanence is based in the relative richness of a given column with respect to its neighbors. To further clarify, let $k < M$, and consider $\{Z_{(1)}, Z_{(2)}, \cdots, Z_{(k)}, \cdots, Z_{(M)}\}$, the ordered version of the sample $\mathbf{Z} = \{Z_1, \cdots, Z_M\}$ where $Z_{(1)} = \min\{Z_1, \cdots, Z_M\}$, $Z_{(M)} = \max\{Z_1, \cdots, Z_M\}$, and $Z_{(k)} = k^{th}$ largest value in $\mathbf{Z}$. An HTM column m is said to be **Active Connected and Emmeshed (ACE)** if its number of active and connected cells Z_m is beyond the preset threshold ρ_d, and Z_m is also one the k largest values. More succinctly, we define the set of HTM column indices

$$\texttt{ACE} = \{m : \; Z_m > \rho_d \text{ and } Z_m \geq Z_{(k)}\},$$

and we create $\mathbf{c} = (c_1, c_2, \cdots, c_m, \cdots, c_M)^\top$, the M-dimensional indicator vector such that

$$c_m = 1_{\{m \in \texttt{ACE}\}} = \begin{cases} 1 & \text{if } m \in \texttt{ACE} \\ 0 & \text{otherwise} \end{cases}$$

The update equation for the HTM spatial pooler is discrete in nature in the sense that the learning algorithm identifies those columns that pass the ACE test, and updates only the permanence value of a cell if the column it belongs to has passed the ACE test.

$$\phi_{mq} := \phi_{mq} + c_m \delta\phi_{mq} \tag{1}$$

where

$$\delta\phi_{mq} = \texttt{P}_{\texttt{inc}} X_{mq} - \texttt{P}_{\texttt{dec}}(1 - X_{mq}) \tag{2}$$

1) The update scheme is discrete, in the sense that the columns whose cells are eligible for updates are discretely selected. This kind of updating could be likened to a form of hard thresholding, in the sense that a cell either receives zero update or all the update, and nothing in between.
2) All the cells eligible for update receive the exact same amount of change in their permanence value, namely $\texttt{P}_{\texttt{inc}}$ for cell active at the update time $-\texttt{P}_{\texttt{dec}}$ for cells inactive at the update time. While there might be some merit in such a global and fixed updating scheme, one could rightly argue that its rigidity has the potential to cause limitations.
3) The determination of the indicator c_m of update eligibility has the potential to be computationally burdensome
4) There is no learning rate (step size) in the update equation as one would expect in a typical gradient descent/ascent algorithm
5) Crucially, there is no principled theoretical justification for the updating scheme, the lack of which makes it very difficult to formally and methodologically study the strengths and weaknesses/limitations of the HTM spatial pooler's learning scheme.
6) Many of the hyperparameters controlling global aspects of the algorithm are set without a refutable conceptual and/or methodological justification, which can lead to difficulties tuning the whole scheme optimally.

The goal of this paper, which follows from previous work by [15], is to provide a principled theoretical formulation of the spatial pooler underlying learning scheme, and thereby consequently address some or all of the issues mentioned above. We start by interpreting the permanence value $\phi_{mq} \in (0,1)$ as the local probability of activation of the given cell in its column with respect its neighbors. Based on such an interpretation and given the fact that $X_{mq} \in \{0,1\}$ and $\phi_{mq} \in (0,1)$, we hypothesize

$$\Pr[X_{mq} = 1 | \phi_{mq}] = \phi_{mq} \qquad \Pr[X_{mq} = 0 | \phi_{mq}] = 1 - \phi_{mq}$$

As a result, for each cell we have posit a Bernoulli distribution so that the local likelihood of ϕ_{mq} is given by

$$L(\phi_{mq}) = \phi_{mq}^{X_{mq}} (1 - \phi_{mq})^{1 - X_{mq}},$$

from which the corresponding loglikelihood for ϕ_{mq} is

$$\ell(\phi_{mq}) = X_{mq} \log(\phi_{mq}) + (1 - X_{mq}) \log(1 - \phi_{mq}). \tag{3}$$

Strictly speaking, since the goal is to estimate/learn the whole permanence matrix Φ, the corresponding loglikelihood is

$$\begin{aligned} \ell(\Phi) &= \sum_{m=1}^{M} \sum_{q=1}^{Q} \ell(\phi_{mq}) \\ &= \sum_{m=1}^{M} \sum_{q=1}^{Q} \{X_{mq} \log(\phi_{mq}) + (1 - X_{mq}) \log(1 - \phi_{mq})\} \end{aligned} \tag{4}$$

However, since the update is local at the cell level, it suffices to focus on the loglikelihood of ϕ_{mq} defined in Equation (3). The statistical estimation problem can be solved as Constrained Maximum likelihood estimation problem, namely

$$\widehat{\Phi}_{\texttt{MLE}} = \underset{\Phi}{\texttt{argmax}}\{\ell(\Phi)\} \quad \texttt{subject to} \quad \texttt{ACE constraints}$$

The above MLE for Φ is obtained by solving for Φ the equation

$$\frac{\partial \ell(\Phi)}{\partial \Phi} = 0 \tag{5}$$

subject to the constraints, or by using a stochastic gradient ascent iterative scheme of the form

$$\Phi^{(i)} = \Phi^{(i-1)} + \gamma^{(i)} \frac{\partial \ell(\Phi^{(i-1)}; X^{(i)})}{\partial \Phi} \tag{6}$$

where $\gamma^{(i)}$ is the learning rate (step size), and $X^{(i)}$ is the ith sequentially presented input. Thanks to the local nature of each ϕ_{mq}, it suffices to the partial derivative of the local log likelihood with respect to ϕ_{mq}, namely

$$\frac{\partial \ell(\phi_{mq})}{\partial \phi_{mq}} = \frac{1}{\phi_{mq}} X_{mq} - \frac{1}{1 - \phi_{mq}} (1 - X_{mq}). \tag{7}$$

The most obvious and immediate remark is that Equation (7) is same as Equation (2), clearly indicating that the intuition that led to (2) is potentially justified by the Bernoulli model underlying the basic cell in the HTM architecture, with the corresponding learning problem being the optimization of an objective function which in this case in the likelihood of the permanence matrix.

$$f(u) = \frac{1}{20} \frac{u}{1+u} + \frac{1}{200} \tag{8}$$

Notice that by Equation (7), for a column eligible for update, we have the following: If the cell is active, ie $X_{mq} = 1$, then the value of the update is $\frac{\partial \ell(\phi_{mq})}{\partial \phi_{mq}} = \frac{1}{\phi_{mq}}$. Similarly, if the cell is inactive, ie $X_{mq} = 0$, then the value of the update is $\frac{\partial \ell(\phi_{mq})}{\partial \phi_{mq}} = -\frac{1}{1-\phi_{mq}}$

$$g(u) = \frac{1}{20} \frac{1-u}{2-u} + \frac{1}{40} \tag{9}$$

Instead of using the raw updates of $1/\phi_{mq}$ and $-1/(1-\phi_{mq})$ suggested by Equation (7), we thought it ideal for numerical and statistical reasons, to consider instead functions of ϕmq

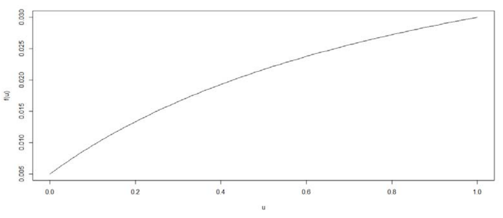

Fig. 2. Weighting Function for the MLE increase

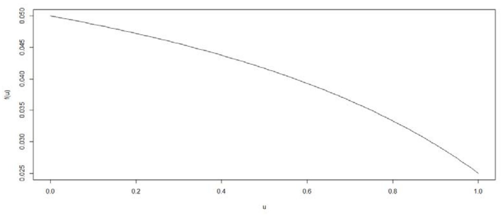

Fig. 3. Weighting Function for the MLE decrease

suitably chosen to be stable and inherently learning rate-like. The functions $f(u)$ and $g(u)$ were used, so that in place of an increment of $1/\phi_{mq}$, we had an increment of $f(\phi_{mq})$. We also use a decrement of $g(\phi_{mq})$ instead of $-1/(1-\phi_{mq})$. It should be noted that the functions $f(u)$ and $g(u)$ are just examples we deemed suitable[1] Instead of fixed and global updates, the likelihood function now provides a flexible mechanism by which updates are local and adaptive as one's intuition would suggest. It is our view/opinion that such local and adaptive updates have the potential to help learning more complex underlying structures, thereby making our principled theoretical formulation of the spatial pooler beneficial and more powerful in representing underlying hypothesis spaces. From a pure modeling perspective, it also makes ample sense that the updating of a given parameter would depend in some sense on its previous value when an iterative scheme is used. Although the vector $\mathbf{c}$ in the discrete global and fixed update did depend on the previous value of the permanence, we deem a direct dependence more readily controllable for better learning.

If one were to solve (5) without constraints, the resulting solution would simply be a estimated permanence matrix whose entries would then be the empirical proportion of 1's in that cell. Such a result could be obtained computationally via an online (sequential) estimation procedure of the form

$$\phi_{mq}^{(i)} = \phi_{mq}^{(i-1)} + \frac{1}{i}(X_{mq}^{(i)} - \phi_{mq}^{(i-1)}) \tag{10}$$

We refer to the updating scheme in (10) as brute force Bernoulli because it is obtained without any additional sophisticated modeling, and all the cells are updated. In keeping with the terminology used in the original HTM spatial pooler the increment with brute force is $(1/i)(1-\phi_{mq}^{(i-1)})$, while the decrement is $(1/i)(-\phi_{mq}^{(i-1)})$. It is interesting to notice that this rather simple scheme yields relatively good results with and without the ACE constraints. It was suggested and presented here merely as a baseline for comparison against the more elaborate approaches. See results section for details.

[1]In their development of the original spatial pooler, the authors suggested using $\mathtt{P_{inc}} = 0.03$ and $\mathtt{P_{dec}} = 0.05$. Many other functions with similar properties can be considered. Indeed, one of the richness of our proposed approach lies in the fact that it provides a rich array of possible ways to control the learning of the entries of the permanence matrix.

As indicated earlier, the discrete nature of the updating scheme can easily lead to a computational bottleneck and render the whole scheme not scalable. We herein proposed a continuous updating scheme by which the discrete selection of the columns eligible for update is replaced a function whose domain is the natural numbers $\{1, 2, \cdots, Q\}$. Recall that Z_m represents the number of active and connected cells in the m^{th} HTM column. All the columns in $c_m = 1$ receive updates, and those with $c_m = 0$ do not receive any update. As we said earlier, this is hard thresholding. Rather than this all or nothing approach, we could imagine a function on the Z_m that performs updates in a continuum, with the largest amount (unit) allocated to the column with largest Z_m. Let $h(Z_m)$ be such a function, then the general update is

$$\phi_{mq} = \phi_{mq} + h(Z_m)\big[f(\phi_{mq})X_{mq} - g(\phi_{mq})(1 - X_{mq})\big] \tag{11}$$

Besides $h(Z_m)$ can be specified based on either the value of Z_m or the rank of Z_m. Equation (11) offers a very flexible updating scheme for learning the values of the permanences. It even takes away the need to specifically address the constraints, since the function $h(Z_m)$ captures the relative importance of each HTM column and apportions the amount of update accordingly.

Finally, the formulation we have proposed here lends itself to a crucial aspect of learning machines, namely the tuning of the hyperparameters. The theoretical formulation proposed here along with the corresponding learning algorithm now allows us to tune the hyperparameter with approaches such as cross validation, thereby making it possible to control the level of sparsity in a more principled way. This will also be explored in our subsequent work.

IV. Experimental Analysis

The preliminary set of results obtained using the above mentioned with MNIST dataset and interesting observations have been made. The spatial pooler's efficiency has been measured by four metrics discussed below. Initially for reference, we used all the features of the input (image), for classification. We obtained an accuracy of 77.61%, using Support Vector Machine with Linear Kernel, implemented using Scikit-learn utility.
Four different update mechanisms, discussed above have been implemented and there is no change in execution time observed with respect to the existing method. The table below shows the execution time. The time taken to train a classifier using 800 MNIST samples and test them on 200 MNIST data samples is tabulated below.

TABLE II. Execution Time - Different update mechanisms

Method	Time taken in seconds
Existing Method	1.24
MLE (Maximum Likelihood Estimation)	1.23
MLE without ACE condition	1.26
Brute Force Bernoulli	1.24
Brute Force Bernoulli without ACE	1.29

The other metrics computed using the features with spatial pooler are as follows,

A. *Column Accuracy*

After the learning phase of the spatial pooler,
Obtain the active connected quotient vector $Z_{1..Q}[1..Q]$ for a given input and binarize the vector Z,

$$ColumnVector[i] = Z[i] > \rho_s \text{ and } z_m > z_k$$

The classifier is trained and tested on the column vector of inputs. With support vector machine, shown in Fig. 4, the new update methods MLE and brute force Bernoulli were not as effective as the existing method. This can be attributed to the fact that the parameter ρ_d defines the selection of the column, but is not used in the learning process by MLE update and the brute force Bernoulli.

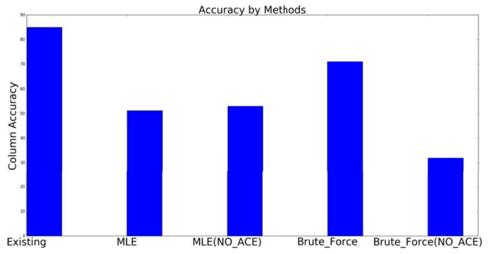

Fig. 4. Spatial Pooler Column accuracy for different update mechanisms

B. *Probabilistic Accuracy*

Probabilistic accuracy is the classification rate obtained by using a modified feature vector using the spatial pooler. Obtain the maximum probability vector, which is the measure of the maximum probability of a corresponding feature value being active, based on the permanence matrix values. This is obtained from the synapticmap.

$$MPV[i] = max(\phi[m, q], \forall m, q, synaptic_{map}[m, q] == i) \quad (12)$$

The new feature vector is constructed by using the maximum probability vector value inplace of the active features.

$$V_{new}[i] = V[i] * MPV[i] \quad (13)$$

The classifier (SVM with linear kernel) is trained and tested using the new features V_{new} and the accuracy plot is shown in Fig. 5. The update by MLE approach gives better probabilistic accuracy than the existing method and the brute force Bernoulli.

C. *Reduction Accuracy and Reduction Factor*

This metric is used to measure the effectiveness of spatial pooler as a feature selector. Reduction accuracy is the classification rate obtained by reducing the input features with the help of spatial pooler. Reduction factor is the ratio to which it has been reduced. Hence, in this case, the best update method is the one which has accuracy and also high reduction factor. The features are reduced as follows,

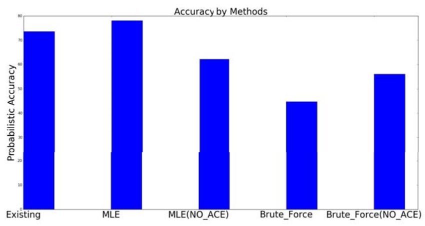

Fig. 5. Probabilistic accuracy with Spatial Pooler for varying update mechanisms

After training the spatial pooler, obtain the maximum probability vector as mentioned in 12 and filter the features for which the maximum probability vector value is above the connection threshold ρ_s and use truncation. The best update method offers high accuracy with fewer features, as shown in Fig. 7 The plot

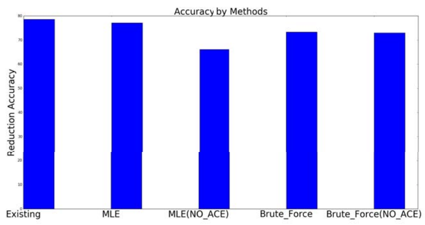

Fig. 6. Reduction accuracy with Spatial Pooler, for varying update mechanisms

of the reduction factor can be seen here. This is the factor of reduction, higher the value is higher features are reduced.

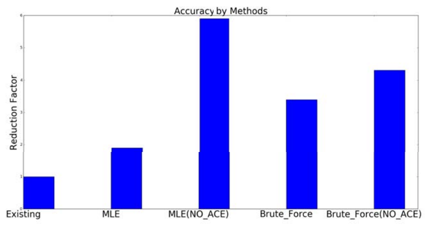

Fig. 7. Reduction Factor with Spatial Pooler, for varying update mechanisms

Even though the reduction accuracy for previous methods looks higher, the reduction factor is equal to 1, this indicates that none of the features were reduced. But we can observe that MLE update, is able to reduce features to almost one-sixth and achieved almost equal accuracy.

V. Conclusion

In this paper, we have proposed and substantially developed a rigorous statistical formulation of the spatial pooler, specifically showing that the underlying algorithm is essentially solving a penalized maximum likelihood estimation problem on a binary input space with constraints (penalties) driven by the core ideas behind the functioning of the HTM column. The formulation we have proposed has, as one of its central benefits, the fact that it lends itself to a crucial aspect of learning machines, namely the tuning of the hyperparameters and thereby an ability to control the level of sparsity and the predictive performance in a more principled way. It is particularly of great interest to establish the benefits/gains (if any) of using adaptive updates in place of fixed global ones, and to investigate the effects of all the thresholds used on the HTM paradigm. To this end, we intend to fully implement the extension herein developed, namely with different scenarios of artificial data on the one hand, and a vast exploration of real life data (benchmark or otherwise) of the other.

Acknowledgments

The authors would like to acknowledge James Mnatzaganian and Luke Boudreau from Nanocomputing research lab who have initiated and accelerated this work through several discussions on the formalization of spatial pooler. Authors would like to thank Subutai Ahmad from Numenta and Kevin Gomez from Seagate for supporting this effort.

References

[1] J. Hawkins, S. Ahmad, and D. Dubinsky, "Hierarchical temporal memory including htm cortical learning algorithms," Available at http://numenta.com/assets/pdf/whitepapers/hierarchical-temporal-memory-cortical-learning-algorithm-0.2.1-en.pdf, 2011, accessed on 2014-11-02.

[2] V. B. Mountcastle, *Perceptual Neuroscience: The Cerebral Cortex*. Harvard Univ. Press, 1998.

[3] Y. Cui, C. Surpur, S. Ahmad, and J. Hawkins, "Continuous online sequence learning with an unsupervised neural network model," *arXiv preprint arXiv:1512.05463*, 2015.

[4] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2015, pp. 38–44.

[5] L. Streat, D. Kudithipudi, and K. Gomez, "Non-volatile hierarchical temporal memory: Hardware for spatial pooling," *arXiv preprint arXiv:1611.02792*, 2016.

[6] S. Ahmad and J. Hawkins, "Properties of sparse distributed representations and their application to hierarchical temporal memory," 3 2015.

[7] D. E. Rumelhart and D. Zipser, "Feature discovery by competitive learning," *Cognitive science*, vol. 9, no. 1, pp. 75–112, 1985.

[8] A. Gersho and R. M. Gray, *Vector quantization and signal compression*. Springer Science & Business Media, 2012, vol. 159.

[9] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[10] J. Mnatzaganian, E. Fokoué, and D. Kudithipudi, "A mathematical formalization of hierarchical temporal memory cortical learning algorithm's spatial pooler," *arXiv preprint arXiv:1601.06116*, 2016.

[11] M. Leake, L. Xia, K. Rocki, and W. Imaino, "A probabilistic view of the spatial pooler in hierarchical temporal memory," *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 9, no. 5, pp. 1111–1118, 2015.

[12] S. Lattner, "Hierarchical temporal memory-investigations, ideas, and experiments," Master's thesis, Johannes Kepler Universität, 2014.

[13] F. Byrne, "Encoding reality: Prediction-assisted cortical learning algorithm in hierarchical temporal memory," *arXiv preprint arXiv:1509.08255*, 2015.

[14] J. Hawkins and S. Ahmad, "Why neurons have thousands of synapses, a theory of sequence memory in neocortex," *Frontiers in Neural Circuits*, vol. 10, no. 23, Mar. 2016.

[15] J. Mnatzaganian, E. Fokoué, and D. Kudithipudi, "A Mathematical Formalization of Hierarchical Temporal Memory's Spatial Pooler," *ArXiv e-prints*, Jan. 2016.

Appendix
Algorithms - Permanence Update

Algorithm 1 Main Algorithm- Update Permanence Matrix for given Inputs

```
1: for all $X \in [X_1 \cdots X_n]$ do
2:     $Z \leftarrow$ Find_Active_Connected_Quotient($\phi, X$)
3:
4:     $Mag_z \leftarrow$ Find_Magnitude_Active_Connected_Quotient($Z$)
5:
6:     $Inc\_Factor, Dec\_Factor \leftarrow$ Find_Factors($\phi$)
7:
8:     for all $row_i \in \phi$ do
9:         for all $col_j \in row_i$ do
10:            if $X[i][j]$ is Active then
11:                $\phi[i][j] \leftarrow \phi[i][j] + Inc\_Factor[i][j]$
12:            else
13:                $\phi[i][j] \leftarrow \phi[i][j] - Dec\_Factor[i][j]$
```

Algorithm 2 Compute Active-Connected Quotient, Vector Z

```
1: for all $row_i \in \phi$ do
2:     for all $col_j \in row$ do
3:         $Y[i][j] \leftarrow$ isConnected($\phi[i][j]$)
4: for all $row_i \in \phi$ do
5:     $Ac[i] \leftarrow$ dotProduct($X[i], Y[i]$)
6: for all $row_i \in AC$ do
7:     $Z[i] \leftarrow$ CountTruevalues($Ac[i]$)
```

Algorithm 3 Find Magnitude of active connected quotient vector

```
1: for all $z_k \in Z$ do
2:     $magnitude\_z[k] \leftarrow 1/$ percentile($z[K]$)
```

Algorithm 4 Find Increment and Decrement factor of Permanence Matrix

```
1: for all $row \in \phi$ do
2:     for all $column \in row$ do
3:         $Factor_{Inc}[row][column] \leftarrow \frac{\phi[row][column]}{1+\phi[row][column]}$
4:
5:         $Factor_{Dec}[row][column] \leftarrow \frac{1-\phi[row][column]}{2-\phi[row][column]}$
```