International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:5, 2015

A Probabilistic View of the Spatial Pooler in Hierarchical Temporal Memory

Mackenzie Leake, Liyu Xia, Kamil Rocki, Wayne Imaino

Abstract—In the Hierarchical Temporal Memory (HTM) paradigm the effect of overlap between inputs on the activation of columns in the spatial pooler is studied. Numerical results suggest that similar inputs are represented by similar sets of columns and dissimilar inputs are represented by dissimilar sets of columns. It is shown that the spatial pooler produces these results under certain conditions for the connectivity and proximal thresholds. Following the discussion of the initialization of parameters for the thresholds, corresponding qualitative arguments about the learning dynamics of the spatial pooler are discussed.

Keywords—Hierarchical Temporal Memory, HTM, Learning Algorithms, Machine Learning, Spatial Pooler.

I. Introduction

INSPIRED by the brain and based on the importance of time-dependent behavior and prediction, a machine learning paradigm, Hierarchical Temporal Memory (HTM) [3], enables real-time learning of sequences and demonstrates predictive capabilities. Prior works have explored the use of HTM for pattern recognition [5], speech-based learning tasks [9], land use recognition [7], content-based image retrieval [1], and stock trading [2]. One prior paper has compared the performance of the first phase of the HTM framework, the *spatial pooler*, on greyscale natural images with other coding schemes [8]. However, the mathematics of HTM has not yet been studied widely. Here we provide a mathematical and probabilistic framework to examine the behavior of the spatial pooler upon initialization before any inputs are introduced.

As outlined in the HTM white paper [3], the spatial pooling phase is the first step of the framework. It is a powerful online learning mechanism that demonstrates efficient dimensionality reduction and adaptation. The spatial pooler comprises a set of *columns*, analogous to biological cortical columns, the fundamental units of computation in the neocortex [6]. For clarity in our implementation of the spatial pooler, italicized *"column"* refers to a computational unit in the spatial pooler, while "column" refers to a column of a matrix. Each *column* has a distinct proximal dendrite segment, which is a set of synapses connected to specific entries of the input vectors. Each synapse has an associated permanence value, the magnitude of which decides whether the synapse is connected or not. Note that in our implementation of the spatial pooler each input to the spatial pooler is a binary vector of 0's and 1's.

The spatial pooler is responsible for converting each binary input vector into an internal representation called a sparse distributed representation (SDR), which is a set of *columns* that are activated by the input vector. These SDR's must preserve the similarities and differences between inputs. In order to group similar inputs and distinguish between differing inputs, the spatial pooler should exhibit the following behavior: similar inputs map to similar SDR's, and dissimilar inputs map to dissimilar SDR's. The quality of these SDR's is critical to the later predictive and temporal phases of the HTM paradigm, which potentially predict and recognize sequences of SDR's.

We consider inputs to be either overlapping or non-overlapping. We use the terms "overlapping inputs" to denote multiple inputs whose binary representations share at least one on-bit and "non-overlapping inputs" to denote inputs whose binary representations share no on-bits. In this work we study the specific initial conditions for the spatial pooler parameters in order to achieve the desired behavior of grouping overlapping inputs into similar sets of active *columns* and non-overlapping inputs into different sets of active *columns* before any learning occurs. Achieving this desired behavior contributes to a better understanding of the spatial pooler, which is an important step in laying a solid foundation for success in the temporal phase of HTM.

II. Formulation / Notation

In the following sections, we use the notation below to describe the status and behavior of the spatial pooler; this notation is more fully outlined in the Appendix.

The spatial pooler is responsible for providing SDR's for a set of inputs. The spatial pooler updates its information through changes in synaptic activity. In our representation this synaptic activity is stored in a permanence matrix, P. Each row of P is a vector of permanence values that is associated with a certain *column* in the spatial pooler. For example, the permanence value at P_{ij} denotes the strength of the connection between the i-th *column* and the j-th bit of the binary input vector. Another matrix, called the connectivity matrix, a binary version of P, is denoted by C and can be described by:

$$C = P > \tau_C \tag{1}$$

where τ_C is the synaptic threshold, and this notation indicates that entry C_{ij} is set to 1 if $P_{ij} > \tau_C$ and 0 otherwise. Permanence values that exceed the synaptic threshold result in the synapse being connected, and all lower permanence values result in the synapse not being connected. Binary values in the connectivity matrix reflect this behavior. We use C_j to denote

M. Leake, L. Xia, K. Rocki, and W. Imaino are with IBM Research Almaden, San Jose, CA 95120 USA (email: wayne1@us.ibm.com).

the connectivity of the j-th *column* in the spatial pooler, i.e. the j-th row of the connectivity matrix C.

If we consider the spatial pooler to be a function that maps input vectors to SDR's, the function can be formally characterized as the following:

$$\mathtt{ov} = Cx \tag{2}$$

$$c = \mathtt{ov} > \tau_o \tag{3}$$

where x is the binary input vector, $\mathtt{ov}$ is the vector of overlap scores between the input vector x and C, and τ_o is the proximal threshold. Here c is the SDR of x, and the proximal threshold τ_o determines whether there is sufficient overlap between an input x and a particular *column* C_j to activate C_j. For all calculations in this paper, boosting and inhibition as discussed in [3] are not considered in order to make the mathematical formulation more tractable. Boosting and inhibition are not critical to our work because our computations are performed upon initialization of the permanence and connectivity matrices.

The overlap calculation controls *column* activity and hence gates synaptic changes. To understand the range of inputs that cause *column* activity, a reconstruction vector, r, is defined by:

$$r = C^T * c. \tag{4}$$

We note that the input will select a *column* whose reconstruction is closest in a least squares sense if the input and reconstruction vectors have a fixed number of 1's. The square error between the input and the reconstruction is given by:

$$error^2 = (x - r)^2 \tag{5}$$

$$= x^2 - 2xr + r^2 \tag{6}$$

$$= \lambda_1 - 2xr + \lambda_2 \tag{7}$$

where λ_1 and λ_2 correspond to the number of 1's in the input, x, and reconstruction, r, respectively, which are fixed. Certain encoding schemes ensure the number of 1's is fixed for the input. For classification problems where the candidates have been learned by the *columns*, the spatial pooler will minimize the error for the selection. The activation of a *column* with maximum overlap, which corresponds to xr, ensures the minimization of the error between the input and the reconstruction.

III. Non-overlapping inputs

A. Expectation Value of Overlap

In order to gain insight into the selection of initial τ_o and τ_C parameters to ensure two non-overlapping inputs activate different *columns*, the calculations discussed in this section can be performed. This insight is essential to the success of the spatial pooling phase by ensuring the spatial pooler does not pool dissimilar inputs together.

We can begin with two non-overlapping inputs, x^a and x^b. Let these inputs be encoded to be column vectors with length L and comprising d active bits, and assume that every possible input is equally likely. Note that there are certain encoding schemes that can ensure the number of on-bits in the inputs stays constant. Assume x^a activates the j-th *column*, C_j, that is, the inner product of x^a and C_j is greater than τ_o. These relationships can be described by:

$$\langle x^a, x^b \rangle = 0 \tag{8}$$

$$\langle x^a, C_j \rangle = \mathtt{ov_{ac}} \tag{9}$$

where $\mathtt{ov_{ac}}$ is the overlap score between input x^a and *column* C_j, which is a row of the connectivity matrix in our formulation. Beginning with a uniformly distributed permanence matrix with values that range from 0 to 1, a threshold, τ_C, can be used to convert the permanence matrix to a binary connectivity matrix. τ_C is in $[0, 1]$ so the density of the connectivity matrix can be described by:

$$100 * (1 - \tau_C)\% \text{ are } 1 \tag{10}$$

$$100 * \tau_C\% \text{ are } 0. \tag{11}$$

Note that the position of the d active bits in the input does not matter, since we assume every input is equally likely and the entries of C_j are independently and identically distributed. Thus, without loss of generality we assume $x^a = [1, \ldots, 1, 0, \ldots, 0]$, with d 1's at the beginning. To simplify notation let us use event Z to denote the conditions $\langle x^a, x^b \rangle = 0$, $\langle x^a, C_j \rangle = \mathtt{ov_{ac}}$, and $x^a = [1, \ldots, 1, 0, \ldots, 0]$. We use E_n as a dummy variable to denote this conditional expectation, $E_n = \mathbb{E}[\sum_{k=1}^{L} x_k^b C_{jk} | Z]$.

Using the linearity of expectation, we can compute the expected value for the overlap between x^b and C_j given Z. This gives:

$$E_n = \sum_{k=1}^{L} \mathbb{E}[x_k^b C_{jk} | Z]. \tag{12}$$

Since the entries of x^b and C_j are in $\{0, 1\}$, only when x^b and C_j are 1 will their values contribute to the expected value resulting in:

$$E_n = \sum_{k=1}^{L} \mathbb{P}[x_k^b = 1, C_{jk} = 1 | Z]. \tag{13}$$

Since x^a and x^b are non-overlapping, we know the first d entries of x^b are 0 so:

$$E_n = \sum_{k=d+1}^{L} \mathbb{P}[x_k^b = 1, C_{jk} = 1 | Z]. \tag{14}$$

Using Bayes' theorem we compute:

$$E_n = \sum_{k=d+1}^{L} \mathbb{P}[x_k^b = 1 | C_{jk} = 1, Z] * \mathbb{P}[C_{jk} = 1 | Z]. \tag{15}$$

Considering all possible x^b, the probability for each entry of x^b to be 1 is $\frac{d}{L-d}$ resulting in:

$$E_n = \sum_{k=d+1}^{L} \frac{d}{L-d} * \mathbb{P}[C_{jk} = 1 | Z] \tag{16}$$

$$= (L - d) * \frac{d}{L-d} * \mathbb{P}[C_{jk} = 1 | Z] \tag{17}$$

$$= d * \mathbb{P}[C_{jk} = 1 | Z]. \tag{18}$$

The threshold for connectivity is τ_C so the expected number of 1's within a row of the connectivity matrix is $L(1-\tau_C)$. The number of 1's in the first d entries of *column* C_j is equal to $\mathtt{ov_{ac}}$ because we assumed $x^a = [1,\ldots,1,0,\ldots,0]$. Therefore, the number of 1's left for the *column* to overlap with x^b is equal to $L(1-\tau_C) - \mathtt{ov_{ac}}$. This results in:

$$E_n = d * \frac{L(1-\tau_C) - \mathtt{ov_{ac}}}{L-d}. \tag{19}$$

Dissimilar inputs should have dissimilar representations so the desired behavior is for two non-overlapping inputs to activate different *columns*. In order to achieve this desired behavior, the proximal threshold τ_o can provide a loose upper bound for the expected overlap score between x^b and C_j. We can substitute $\mathtt{ov_{ac}}$ by τ_o, since x^b should not activate C_j, as long as x^a has already activated C_j by surpassing the minimum threshold τ_o. We then find:

$$d * \frac{L(1-\tau_C) - \tau_o}{L-d} \leq \tau_o \tag{20}$$

$$1 - \tau_C \leq \frac{1}{d}\tau_o. \tag{21}$$

This preceding relationship provides general guidelines for the initial selection of τ_C and τ_o values that on average will prohibit input x^b from activating *column* C_j, given that x^a already activates C_j.

We can support this relationship numerically using our Matlab implementation of the spatial pooler. In the following example, we begin with a uniformly random permanence matrix using 256 *columns* with $L = 1024$ and $d = 30$. In (Fig. 1) we see that by varying the permanence and connectivity thresholds, there is a clear dividing line where the number of shared active *columns* drastically decreases to zero. The linear boundary between the high and low parts of the surface corresponding to large and small numbers of shared active *columns* respectively coincides with the linear relationship we obtain through the calculation of the expected value of the overlap between x^b and C_j. From our results in equation (21), we would expect the slope of the boundary to be $-\frac{1}{d} = -\frac{1}{30}$. Our numerical example yields a steeper slope of approximately $-\frac{1}{18}$. Since the expected value calculation only provides a loose bound, this numerical example supports the linear relationship we obtain in (21). This indicates that using (21) can in practice provide guidelines to differentiate the *columns* activated by non-overlapping inputs.

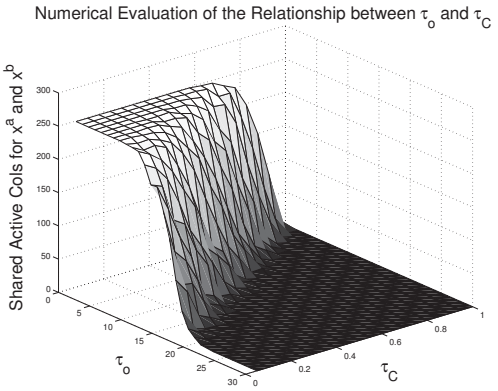


Fig. 1: Numerical example of the effect of the permanence and connectivity thresholds on the number of shared active bits for two inputs.

B. Variance

The computation of the variance for the expected overlap between x^b and C_j contributes to a better understanding of the overall distribution of the overlap between input x^b and *column* C_j as a random variable. We use V_n to denote the variance: $V_n = Var[\sum_{k=1}^{L} x_k^b C_{jk}|Z]$. Using the definition of variance we compute:

$$V_n = \mathbb{E}[(\sum_{k=1}^{L} x_k^b C_{jk})^2|Z] - E_n^2. \tag{22}$$

Since we already computed the expected value from the previous section, we must only compute the first half of the previous line resulting in:

$$\mathbb{E}[(\sum_{k=1}^{L} x_k^b C_{jk})^2|Z] = E_n + \sum_{m\neq n} \mathbb{E}[x_m^b C_{jm} x_n^b C_{jn}|Z]. \tag{23}$$

We use E_{mn} to denote the second half of the previous line. Note that only when x_m^b, C_{jm}, x_n^b, C_{jn} are all 1's will their values contribute to the expected value. Since x^a and x^b are non-overlapping, $m, n > d$, and we calculate:

$$E_{mn} = \sum_{m\neq n;\ m,n>d} \mathbb{P}[x_m^b = C_{jm} = x_n^b = C_{jn} = 1|Z] \tag{24}$$

$$= 2 * \sum_{m<n;\ m,n>d} \mathbb{P}[x_m^b = C_{jm} = x_n^b = C_{jn} = 1|Z] \tag{25}$$

$$= 2 * \sum_{m<n;\ m,n>d} \frac{d}{L-d}\frac{d-1}{L-d-1} \tag{26}$$

$$* \frac{L(1-\tau_C) - \mathtt{ov_{ac}}}{L-d} * \frac{L(1-\tau_C) - \mathtt{ov_{ac}} - 1}{L-d-1}. \tag{27}$$

We use α to denote $L(1-\tau_C) - \mathtt{ov_{ac}}$, so the expression for the expected value becomes $\frac{\alpha d}{L-d}$ accordingly resulting in:

$$E_{mn} = 2 * \binom{L-d}{2} \frac{d}{L-d} * \frac{d-1}{L-d-1} * \frac{\alpha}{L-d} * \frac{\alpha-1}{L-d-1} \tag{28}$$

$$= \frac{d(d-1)\alpha(\alpha-1)}{(L-d)(L-d-1)}. \tag{29}$$

Plugging E_{mn} into the previous computation of variance and using the result of the expected value from the previous section, we have:

$$V_n = \frac{d(d-1)\alpha(\alpha-1)}{(L-d)(L-d-1)} + \frac{\alpha d}{L-d} - (\frac{\alpha d}{L-d})^2. \tag{30}$$

Now that we have both the expression of the expected value and the variance of the overlap score between the input x^b and *column* C_j, we can use the normal distribution to approximate

the probability distribution of the overlap between x^b and C_j. For certain sets of parameters the normal approximation can provide a good estimate of the desired lower bound for τ_o. A detailed numerical example is provided later in this paper after the closed form formula for the probability distribution of $\mathtt{ov_{bc}}$ is introduced (Fig. 3).

C. Probabilistic View

A more computationally difficult method to examine the behavior of non-overlapping inputs is to compute the entire probability distribution directly for the overlap between the input x^b and *column* C_j. This method provides more accurate guidelines than the expected value and variance calculations to select threshold values that ensure the probability of two non-overlapping inputs activating the same *column* is very small. With the same assumptions used for the expected value and variance calculations, the closed form formula for the probability of the overlap between x^b and C_j being $\mathtt{ov_{bc}}$ is:

$$\binom{d}{\mathtt{ov_{bc}}}\binom{L-2d}{\alpha-\mathtt{ov_{bc}}}/\binom{L-d}{\alpha}. \tag{31}$$

Since the first d positions of input x^a are 1's, the denominator is the total number of ways to arrange the 1's left for C_j in the remaining $L-d$ spots. In the numerator the first term is the number of possibilities to arrange the $\mathtt{ov_{bc}}$ 1's that C_j can overlap with x^b. The second term in the numerator is the number of arrangements for the 1's left for C_j after overlapping with both x^a and x^b. We can obtain the entire probability distribution enumerating $\mathtt{ov_{bc}}$ from 0 to d. With the entire distribution, we can get a lower bound for τ_o using the cumulative distribution for any set of parameters. For example, suppose $L = 1000$, $d = 30$, and $\alpha = 180$ ($\tau_C = 0.80, \mathtt{ov_{ac}} = 20$). If we would like the probability of x^b activating C_j to be smaller than $\epsilon = 10^{-3}$, $\tau_o = 14$ is sufficient (Table I).

In the non-overlapping case the binomial distribution $Bin(n, p)$ can serve as an even better approximation than the normal distribution (Fig. 3a). It is helpful to compute the binomial distribution because it is a discrete distribution and the probability distribution of $\mathtt{ov_{bc}}$ is also discrete. The first parameter of the binomial distribution, n, the number of trials, is d, since each trial can be considered as the overlap of one on-bit. The second parameter, p, the probability of success in each trial, can be determined by the fact that the expected value of the binomial distribution equals the product of the first and second parameters. Thus, we can divide d by the expected value $\frac{\alpha d}{L-d}$ computed previously to obtain this second parameter $p = \frac{\alpha}{L-d}$. Note that due to the Law of Large Numbers, as d, the number of on-bits, increases, both the normal approximation and $Bin(d, \frac{\alpha}{L-d})$ converge to the actual probability distribution given by the closed form formula in (31).

D. Numerical Experiments

We experiment with a variety of values for L, d, $\mathtt{ov_{ac}}$, τ_C, ϵ, τ_o to gain intuition about the interaction of these parameters (Table I). The parameters ϵ, d, and τ_C have a particularly significant impact on the lower bound of τ_o. However, L and $\mathtt{ov_{ac}}$ do not affect the lower bound of τ_o as significantly.

The ϵ value controls the likelihood that two non-overlapping inputs will activate the same *column*. As ϵ decreases and we desire a smaller probability of two inputs activating the same *column*, the proximal threshold increases to ensure this behavior.

The number of on-bits in the input vector is controlled by the scheme chosen to encode the input. The encoding scheme should be considered carefully because as d, the number of on-bits in the input vector, increases, the likelihood that the input will activate a certain *column* increases. This results in a higher chance of two non-overlapping inputs activating the same *column*. Therefore, the proximal threshold, τ_o, must have a larger value in order for the spatial pooler to distinguish between the non-overlapping inputs.

The connectivity threshold, τ_C controls the number of 1's in the connectivity matrix, C. As τ_C increases, the number of 1's in the connectivity matrix decreases. This results in there being a smaller likelihood that an input activates any particular *column*. Because an input is less likely to activate a certain *column*, a smaller proximal threshold suffices to distinguish non-overlapping inputs.

TABLE I:
EFFECTS OF PARAMETERS ON τ_o

L	d	$\mathtt{ov_{ac}}$	τ_C	ϵ	τ_o
1000	30	20	0.80	10^{-3}	14
950	30	20	0.80	10^{-3}	14
900	30	20	0.80	10^{-3}	14
850	30	20	0.80	10^{-3}	14
800	30	20	0.80	10^{-3}	13
800	33	20	0.80	10^{-3}	14
800	36	20	0.80	10^{-3}	15
800	39	20	0.80	10^{-3}	16
800	42	20	0.80	10^{-3}	17
800	42	23	0.80	10^{-3}	17
800	42	26	0.80	10^{-3}	17
800	42	29	0.80	10^{-3}	16
800	42	32	0.80	10^{-3}	16
800	42	32	0.82	10^{-3}	15
800	42	32	0.84	10^{-3}	14
800	42	32	0.86	10^{-3}	12
800	42	32	0.88	10^{-3}	11
800	42	32	0.88	10^{-4}	12
800	42	32	0.88	10^{-5}	14
800	42	32	0.88	10^{-6}	15

E. Qualitative Behavior

The calculations of expected overlap and variance between an input and a particular *column* in the previous sections provide insight into the selection of parameters for the initialization of the permanence matrix. These initial selections of parameters affect the learning dynamics of the spatial pooler as new inputs are introduced. Here the learning rule for the spatial pooler is introduced.

Once the overlap, ov, has been calculated and the active

and inactive *columns* for a particular input are computed, the learning rule described in [3] follows Hebbian learning [4]. Within this framework, as each new input is introduced, P is only updated for the active *columns* associated with the new input. This learning rule can be described mathematically, and for each active *column* C_i the entries of P are given by:

$$P_{ij} = \begin{cases} min(1, P_{ij} + \delta_P) : j \in j_{active} \\ max(0, P_{ij} - \delta_P) : j \notin j_{active} \end{cases}$$

where j_{active} is the set of on-bits in the binary input vector. The entries in the permanence matrix are incremented only when both the *column* is active and the corresponding entry of the input vector is 1. The entries in the permanence matrix are decremented only when the *column* is active and the corresponding entry of the input vector is 0. All other permanence values are left unchanged. The parameter δ_P controls how much the permanence matrix is updated with each input and thus also controls the rate at which a *column* can learn inputs.

With the learning rule described, several observations and qualitative arguments about non-overlapping inputs can be made. First, the spatial pooler can be initialized in such a way that non-overlapping inputs are highly unlikely to activate the same *column*. Therefore, if all the inputs are non-overlapping, there must be a greater number of *columns* than the number of inputs for there to be any possibility that each non-overlapping input will activate a different *column*. Moreover, the learning rate, δ_P, merely controls how quickly the permanence matrix becomes saturated and reaches its minimum or maximum permanence values. As long as the spatial pooler continues to see the same inputs repeatedly and no novel inputs are introduced, the permanence values will tend toward the maximum and minimum values, and the inputs will become more differentiated over time until the corresponding entries of the permanence matrix reach these maximum or minimum values.

IV. Overlapping inputs

A. Statistics of the overlap score

Now we consider the case where x^a and x^b share a certain number of active bits. We denote this sharing of active bits by $\mathtt{ov_{ab}}$. Using the parameters given in the previous non-overlapping section, we can examine how the value of $\mathtt{ov_{ab}}$ affects the probability of the overlapping inputs activating the same *column*. The computation of the expected value and the variance in the overlapping case is a generalization of the computation in the non-overlapping case. Thus, we summarize the computations, and most of the computations employ strategies utilized in the non-overlapping case.

B. Expected value

We make the same assumptions as in the non-overlapping case, except that the overlapping case requires one additional parameter: the amount of overlap between inputs x^a and x^b, denoted by $\mathtt{ov_{ab}}$. To simplify notation let us use event Z' to denote the conditions $\langle x^a, x^b \rangle = \mathtt{ov_{ab}}$, $\langle x^a, C_j \rangle = \mathtt{ov_{ac}}$, and $x^a = [1, \ldots, 1, 0, \ldots, 0]$. Here we compute the expected value for the overlap between x^b and C_j given that x^a overlaps with x^b by $\mathtt{ov_{ab}}$ and x^a overlaps with C_j by $\mathtt{ov_{ac}}$. We use E_o to denote this expected value:

$$E_o = \mathbb{E}[\sum_{k=1}^{L} x_k^b C_{jk} | Z'] = \sum_{k=1}^{L} \mathbb{E}[x_k^b C_{jk} | Z'] \quad (32)$$

$$= \sum_{k=1}^{L} \mathbb{P}[x_k^b = C_{jk} = 1 | Z']. \quad (33)$$

Since x^a and x^b overlap by $\mathtt{ov_{ab}}$, we know x^b has $\mathtt{ov_{ab}}$ active bits in its first d bits. For each of these active bits, the probability that the corresponding entry in C_j is also 1 is $\frac{\mathtt{ov_{ac}}}{d}$. For the active bits of x^b that are not in the first d bits, the probability of these bits being 1 is $\frac{L(1-\tau_C)-\mathtt{ov_{ac}}}{L-d}$. Using these probabilities, our expected value calculation becomes:

$$E_o = \mathtt{ov_{ab}} * \frac{\mathtt{ov_{ac}}}{d} + (d - \mathtt{ov_{ab}}) * \frac{L(1-\tau_C) - \mathtt{ov_{ac}}}{L-d} \quad (34)$$

$$= \mathtt{ov_{ab}} * \frac{\mathtt{ov_{ac}}}{d} + (d - \mathtt{ov_{ab}}) * \frac{\alpha}{L-d}. \quad (35)$$

C. Variance

We can compute the variance for the overlapping case. We denote the variance with V_o:

$$V_o = Var[\sum_{k=1}^{L} x_k^b C_{jk} | Z'] = \mathbb{E}[(\sum_{k=1}^{L} x_k^b C_{jk})^2 | Z'] - E_o^2. \quad (36)$$

Since we already computed the expected value from the previous section, we only need to compute the first half of the previous line resulting in:

$$\mathbb{E}[(\sum_{k=1}^{L} x_k^b C_{jk})^2 | Z'] = E_o + \sum_{m \neq n} \mathbb{E}[x_m^b C_{jm} x_n^b C_{jn} | Z']. \quad (37)$$

We denote the second half of the previous line with E'_{mn}. Note that only when x_m^b, C_{jm}, x_n^b, C_{jn} are all 1's will their values contribute to the expected value. We calculate:

$$E'_{mn} = \sum_{m \neq n} \mathbb{P}[x_m^b = C_{jm} = x_n^b = C_{jn} = 1 | Z'] \quad (38)$$

$$= 2 * \sum_{m < n} \mathbb{P}[x_m^b = C_{jm} = x_n^b = C_{jn} = 1 | Z'] \quad (39)$$

$$= 2 * (\sum_{1 \leq m < n \leq d} \mathbb{P}[x_m^b = C_{jm} = x_n^b = C_{jn} = 1 | Z'] \quad (40)$$

$$+ \sum_{d+1 \leq m < n \leq L} \mathbb{P}[x_m^b = C_{jm} = x_n^b = C_{jn} = 1 | Z'] \quad (41)$$

$$+ \sum_{1 \leq m \leq d < n \leq L} \mathbb{P}[x_m^b = C_{jm} = x_n^b = C_{jn} = 1 | Z']). \quad (42)$$

Substituting α for $L(1-t) - \mathtt{ov_{ac}}$ gives:

$$E'_{mn} = 2[\binom{d}{2}\frac{\mathtt{ov_{ab}}}{d}\frac{\mathtt{ov_{ab}}-1}{d-1}\frac{\mathtt{ov_{ac}}}{d}\frac{\mathtt{ov_{ac}}-1}{d-1} \tag{43}$$

$$+\binom{L-d}{2}\frac{d-\mathtt{ov_{ab}}}{L-d}\frac{d-\mathtt{ov_{ab}}-1}{L-d-1}\frac{\alpha}{L-d}\frac{\alpha-1}{L-d-1} \tag{44}$$

$$+ d(L-d)\frac{\mathtt{ov_{ab}}}{d}\frac{d-\mathtt{ov_{ab}}}{L-d}\frac{\mathtt{ov_{ac}}}{d}\frac{\alpha}{L-d}] \tag{45}$$

$$\tag{46}$$

$$E'_{mn} = \frac{\mathtt{ov_{ab}}(\mathtt{ov_{ab}}-1)\mathtt{ov_{ac}}(\mathtt{ov_{ac}}-1)}{d(d-1)} \tag{47}$$

$$+\frac{(d-\mathtt{ov_{ab}})(d-\mathtt{ov_{ab}}-1)\alpha(\alpha-1)}{(L-d)(L-d-1)} \tag{48}$$

$$+\frac{2\mathtt{ov_{ab}}(d-\mathtt{ov_{ab}})\mathtt{ov_{ac}}\alpha}{d(L-d)}. \tag{49}$$

We can plug the expression above into the expression for the variance resulting in:

$$V_o = \mathbb{E}[(\sum_{k=1}^{L} x_k^b C_{jk})^2 | Z'] - E_o^2 \tag{50}$$

$$V_o = \frac{\mathtt{ov_{ab}}(\mathtt{ov_{ab}}-1)\mathtt{ov_{ac}}(\mathtt{ov_{ac}}-1)}{d(d-1)} \tag{51}$$

$$+\frac{(d-\mathtt{ov_{ab}})(d-\mathtt{ov_{ab}}-1)\alpha(\alpha-1)}{(L-d)(L-d-1)} \tag{52}$$

$$+\frac{2\mathtt{ov_{ab}}(d-\mathtt{ov_{ab}})\mathtt{ov_{ac}}\alpha}{d(L-d)} + E_o - E_o^2. \tag{53}$$

For the overlapping case the normal approximation given by the expected value and variance can provide a good approximation for the probability distribution (Fig. 3b). However, we can still provide a closed form formula for the probability density of $\mathtt{ov_{bc}}$.

D. Probabilistic View

Just as in the non-overlapping case, we can compute the probability distribution for the overlapping case. In addition to the assumptions made in the non-overlapping calculations, we consider an additional assumption: the overlap between inputs x^a and x^b is $\mathtt{ov_{ab}}$. The probability that the overlap score between input x^b and C_j is $\mathtt{ov_{bc}}$ is:

$$\sum_{k=0}^{\mathtt{ov_{bc}}} \mathbb{1}[\text{every combinatorial term is well defined}]$$

$$* \frac{\binom{\mathtt{ov_{ab}}}{k}\binom{d-\mathtt{ov_{ab}}}{\mathtt{ov_{ac}}-k}}{\binom{d}{\mathtt{ov_{ac}}}} * \frac{\binom{d-\mathtt{ov_{ab}}}{\mathtt{ov_{bc}}-k}\binom{L-2d+\mathtt{ov_{ab}}}{\alpha-\mathtt{ov_{bc}}+k}}{\binom{L-d}{\alpha}}. \tag{54}$$

Note that the terms in combinatorial expressions must be non-negative integers. Thus, we need to include the indicator that every combinatorial number is well defined in the first place.

Assume that $x^a = [1, \ldots, 1, 0, \ldots, 0]$, so the number of active bits in the first d bits of x^b must be exactly $\mathtt{ov_{ab}}$. Therefore, the number of active bits in the rest of x^b is $d - \mathtt{ov_{ab}}$. We can separate a given $\mathtt{ov_{bc}}$ into the first d bits and the rest $L-d$ bits. Thus, we can use k to increment from 0 to $\mathtt{ov_{bc}}$ to enumerate all the possible cases of separation. For each case we have independent parts of possibility. The first combinatorial term of (54) is the probability that the overlap between x^b and C_j in the first d bits is k, and the second combinatorial term of (54) is the probability that the overlap in the rest $L-d$ bits is $\mathtt{ov_{bc}} - k$ bits.

Note that the probabilistic computation for the overlapping case is much more costly compared to the approximation given by the expected value and variance since combinatorial numbers grow extremely fast. The binomial approximation is not as accurate as the normal approximation in the overlapping case, but it does correlate strongly with the actual probability distribution (Fig. 3b). However, for small parameters we can still compute the entire probability distribution for the sake of high accuracy. Now that we know how to compute the probability distribution for $\mathtt{ov_{bc}}$, we can use the guidelines for the selection of τ_o computed from the non-overlapping case (21) to calculate for each $\mathtt{ov_{ab}}$ value the probability that x^b will activate C_j, i.e. the probability that $\mathtt{ov_{bc}} \geq \tau_o$. This computation allows us to examine further how the probability of x^b activating C_j varies with respect to the change in $\mathtt{ov_{ab}}$, the overlap between the two inputs.

V. Discussion of Overlapping and Non-overlapping Calculations

We can examine the relationships between the overlaps for the two inputs and a single *column*. Given that x^a activates C_j, if there is a significant amount of overlap between the on-bits for two inputs x^a and x^b, there is a high probability that input x^b will also activate *column* C_j. However, if x^a and x^b barely overlap, it is very unlikely for x^b to activate C_j (Fig. 2).

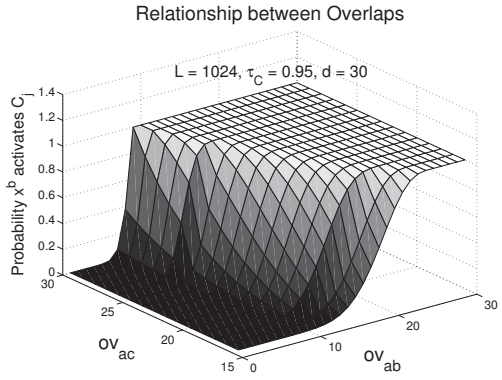
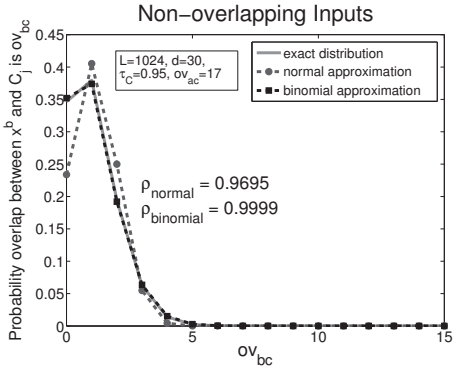


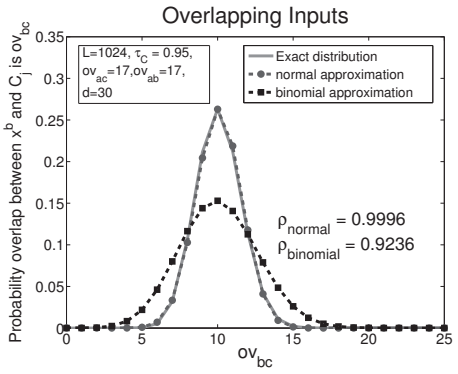
Fig. 2: Relationship between overlap values for two inputs and a single *column*.

In previous sections we use expected value and variance calculations to examine the behavior of non-overlapping and overlapping inputs upon initialization of the spatial pooler. In (31) and (54) we also provide closed form formulas for the overlap between input x^b and *column* C_j given that x^a

activates C_j. We can compute a numerical example to show how good of an approximation the normal and binomial distributions are for both the non-overlapping and overlapping cases using the following selection of parameters: $L = 1024$, $\tau_C = 0.95$, and $d = 30$ (Fig. 3). The probability distribution of $\mathtt{ov_{bc}}$ is computed using the closed form formulas given in (31) and (54) in the probabilistic view sections. The variable ρ_{normal} denotes the correlation between the actual distribution and the normal approximation, and $\rho_{binomial}$ denotes the correlation between the actual distribution and the binomial distribution. As is shown in (Fig. 3), for the selected L and d values, $L = 1024$ and $d = 30$, the binomial approximation is better than the normal approximation in the non-overlapping case ($\rho_{binomial} > \rho_{normal}$). However, in the overlapping case the normal distribution is very close to the actual probability distribution and serves as a better approximation than the binomial approximation ($\rho_{normal} > \rho_{binomial}$). For small parameter values, we can still compute the distribution using the closed form formulas for probability given in (31) and (54) to obtain full accuracy.



(a) Non-overlapping



(b) Overlapping

Fig. 3: Comparison of exact, normal, and binomial distributions for indicated parameters.

VI. Incremental learning

Once the permanence and connectivity matrices are initialized, the learning phase can begin as inputs are introduced to the spatial pooler. As the spatial pooler is repeatedly exposed to inputs, synapses form or break and the permanences of the *columns* start to move away from the threshold because of the Hebbian learning rule [4]. Moving beyond our examination of the interaction between non-overlapping inputs and the spatial pooler upon initialization, we can make observations about this interaction during the learning phase. Characterizing the behavior of the spatial pooler for any arbitrary input is exceptionally challenging. Making a number of assumptions about the inputs, we describe the behavior for two very specific examples, one non-overlapping and one overlapping.

If upon initialization the set of parameters is carefully chosen such that each non-overlapping input activates a different *column*, this behavior will continue during the learning phase. Suppose input x^a activates *column* C_i and no other input activates C_i upon initialization. Once the learning starts, if x^a is presented to the spatial pooler, it will readily activate C_i, and the overlap between x^a and C_i will increase incrementally. However, if another non-overlapping input x^b is presented to the spatial pooler, x^b will not activate C_i. Thus, the incremental learning process only increases the overlap between x^a and C_i while all the other inputs will not have any effect on C_i. Therefore, x^a will continue to activate C_i during the learning phase. We can extend this reasoning to all other non-overlapping inputs and *columns* to ensure non-overlapping inputs continue to activate different *columns*.

We can also make qualitative arguments about the behavior of overlapping inputs during the learning phase. Let us examine a specific example for two overlapping inputs. Suppose we have two inputs, x^a and x^b, which have sufficient overlap to activate the same *column* C_i, and suppose the permanence values corresponding to this *column* are far enough away from the connectivity threshold, τ_C, that one update will not result in a change of the connectivity matrix. From our experiments we find the order and frequency with which inputs are introduced to the spatial pooler affects the final state of the permanence matrix. Therefore, in this specific example we control the order and frequency of inputs. If x^a and x^b are the only inputs and they are presented to the spatial pooler repeatedly the same number of times in an alternating fashion, C_i will reinforce the common parts of x^a and x^b. If x^a is presented to the spatial pooler during the learning phase C_i will be activated. The entries in the permanence matrix that are associated with C_i and correspond to the on-bits in x^a will be incremented, while the entries of P that are associated with C_i and are unique to x^b will be decremented. Similarly, when x^b is presented, C_i will be activated. The entries of P that are associated with C_i and correspond to the on-bits in x^b will be incremented, while the entries that are associated with C_i and are unique to x^a will be decremented.

Since we assume x^a and x^b are presented an equal number of times in an alternating manner, we consider one iteration to be the presentation of x^a and x^b once. For each iteration we analyze the individual bits of the inputs separately and enumerate all the possibilities for each entry of the inputs and *column*:

TABLE II:
EFFECT OF LEARNING ON P AND C

x_j^a	x_j^b	C_{ij} before	C_{ij} after	P_{ij}
1	1	0	1	↑
1	0	0	0	—
0	1	0	0	—
0	0	0	0	↓
1	1	1	1	↑
1	0	1	1	—
0	1	1	1	—
0	0	1	0	↓

where x_j^a and x_j^b denote the j-th position of the input vectors x^a and x^b respectively, C_{ij} *before* is the connectivity between C_i and the j-th bit of the input vectors before any learning has occurred, and C_{ij} *after* is the same connectivity after sufficient learning. Here *sufficient* indicates the connectivity of the *column* no longer changes as inputs x^a and x^b continue to be presented to the spatial pooler. For example, if the j-th bit of both inputs is 1, regardless of the initial value of C_{ij}, C_{ij} will become 1 eventually. This is because P_{ij} will be incremented every time when x^a or x^b activates C_i. If the j-th bit of only one of the inputs is 1, C_{ij} will stay the same, because P_{ij} will be incremented for one input and decremented for the other input. Finally, if the j-th bit of both inputs is 0, P_{ij} will be decremented for every input and C_{ij} will become 0. We use "↑" to indicate that the interaction between x^a, x^b, and C_{ij} *before* results in P_{ij} being incremented by δ_P, "↓" to indicate that the interaction between x^a, x^b, and C_{ij} *before* results in P_{ij} being decremented by δ_P, and "—" to indicate P_{ij} does not change due to the the interaction between x^a, x^b, and C_{ij}.

We can regard (Table II) as a truth table for our specific example, and the corresponding Boolean expression is:

$$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma) \vee (\beta \wedge \gamma), \tag{55}$$

where α denotes x_j^a, β denotes x_j^b, and γ denotes C_{ij} *before*. The symmetric structure of the Boolean expression implies that both the inputs and the *column* have the same influence on determining the final state of C_{ij} after learning. This truth table (Table II) shows the behavior of the connectivity matrix after learning for this specific example having made a number of underlying assumptions. Future work is needed to develop a deeper understanding of how the order and frequency of inputs affects how the spatial pooler learns inputs.

VII. CONCLUSION

We show that with the careful selection of parameters the spatial pooler maps distinct inputs into distinct SDR's as indicated by the *column* activity. We also provide a variety of probabilistic mathematical tools to explore the interaction of overlapping and non-overlapping inputs with the spatial pooler. Finally, assuming the careful selection of parameters at initialization, we can make observations and arguments about the learning dynamics for both non-overlapping inputs and overlapping inputs.

APPENDIX

A. Matrix/scalar operations

If A is a matrix and β is a value, then $A > \beta$ denotes the matrix of conditions $A_{ij} > \beta$. Also $\max\{\beta, A\}$ denotes the matrix B with entries $B_{ij} \leftarrow \max\{\beta, A_{ij}\}$, and $A \leftarrow \beta$ denotes the assignment operation $A_{ij} \leftarrow \beta$.

B. Indicators

For condition e, let

$$[\![e]\!] \leftarrow \begin{cases} 1 & \text{if } e \\ 0 & \text{otherwise} \end{cases}$$

If e is a matrix of conditions, then $[\![e]\!]$ is the corresponding 0-1 matrix.

C. Variables

TABLE III:
EXPLANATION OF VARIABLES

math	type	initialization	purpose
x	n-vec		input signal
n	integer		number of inputs
m	integer		number of *columns*
τ_C	real	0-1	convert P to C
τ_o	real		proximal threshold
δ_P	real		increment for P based on c and x
P	$m \times n$	$U[0,1]$	*permanence* values
C	$m \times n$	$[\![P \geq \tau_C]\!]$	large permanence values
ov	m-vec		col activity, stimulated by input x via C
c	m-vec		ov entries larger than τ_o

ACKNOWLEDGMENT

The authors would like to thank K. Clarkson, R. Fagin, and G. Burr for their advice and support for this work.

REFERENCES

[1] B. Bobier and M. Wirth, "Content-based image retrieval using hierarchical temporal memory," in *Proc. 16th ACM Int. Conf. on Multimedia*, 2008, pp. 925-928.
[2] P. Gabrielsson, R. Konig, and U. Johansson, "Evolving hierarchical temporal memory-based trading models," in *EvoApplications 2013-Applications of Evolutionary Computing*, Vienna, April 3-5, 2013.
[3] J. Hawkins, S. Ahmad, and D. Dubinsky, "Hierarchical temporal memory including HTM cortical learning algorithms," Numenta, Redwood City, CA, Tech. Rep. ver. 0.2.1, 2011.
[4] D.O. Hebb, "The first stage of perception: growth of the assembly," in *The Organization of Behavior*, New York, Wiley, 1949, intro. and ch. 4, pp. xi-xix, 60-78.
[5] D. Maltoni, "Pattern recognition by hierarchical temporal memory," DEIS Univ. Bologna, Tech. Rep., pp. 1-46, Apr. 13, 2011.
[6] V. Mountcastle, "The columnar organization of the neocortex," *Brain*, vol. 120(4), pp. 701-722, 1997.
[7] A.J. Perea, J.E. Merono, and M.J. Aguilera, "Application of Numenta hierarchical temporal memory for land-use classification," *S. Afr. J. Sci.*, vol. 105, pp. 370-375, Sept./Oct. 2009.
[8] J. Thornton and A. Srbic, "Spatial pooling for greyscale images," *Int. J. Mach. Learn. & Cyber.*, vol. 4, pp. 207-216, 2013.
[9] J. van Doremalen and L. Boves, "Spoken digit recognition using a hierarchical temporal memory," Interspeech, pp. 2566-2569, Brisbane, Sept. 22-26, 2008.