Mathematical Formalism and explanation for Dynamic Associative Networks

This document provides a mathematical formalism of Dynamic Associative Networks (DANs)

Ryan Cerauli

The recent popularity of massive models of AI has shown itself through its permeation in nearly all aspects of society, and while its successes are nothing short of astounding, nearly all of these models suffer from at least 1 of these 3 issues:

1. High resource expenditure during the training process
2. The opaqueness of what exactly is happening within the model (The "Black-Box" effect)
3. Phenomena such as chronic forgetting, hallucinations, etc.

Many methods of training these models have been utilized to help minimize these effects, however to many in the realm of neuromorphic computing this seems like attempting to remedy a self-inflicted problem. The models discussed in this paper (Dynamic Associative Networks, or DANs for short) are attempts at sidestepping these issues entirely by operating under a new paradigm entirely. Outlined below is the mathematical formalism for DANs and their subsequent interpretations.

DANs are composed of overlapping "clusters", where each cluster represents a data member in a dataset. The way that these clusters overlap follow a hebbian learning rule, where overlapping traits between clusters are connected (ie. "fire together, wire together"). A cluster can be represented as a 1 by 1 by n tensor made up of n binary vectors of varying lengths o_i, where n denotes the number of "characteristics" that a data member can have:

$$T^{1,1,n} = \left[\left[\left[v_1, v_2, \dots, v_n\right]\right]\right] \; with \; v_i \in \{0,1\}^{l_i}, \; l_i \in N$$

Where $T^{1,1,n}$ represents an individual tensor of length n with elements v_i being binary vectors. The space that these tensors reside in can be thought of as the "logic space", where all possible information that these tensors can hold can be unambiguously represented. The overlap happens when any number of tensors share a certain characteristic (with each characteristic being represented as its own binary vector), and as a result these vectors will hold the same value for that particular dimension. A "trained" DAN can be thought of as a bunch of tensors in this tensor space that overlap based on commonalities between the m data members in a training set, and can be thought of as a 1 by m by n tensor:

$$D^{1,m,n} = \left[T_{1,1,n}, T_{1,2,n}, \dots, T_{1,m,n}\right]$$

Where $T_{1,j,k}$ represents the kth binary vector of the jth tensor of the DAN $D^{1,m,n}$. Another way to visualize this is in the form of nested lists, where a DAN is a list of m lists, where each of those m nested lists contain n binary vectors. An important clarification that must be made for the sake of future calculations is that the binary vectors of a given cluster tensor o_i *are not necessarily the same length*, however the jth binary vector of *each* tensor in the DAN will be the same length. As such, this issue will turn out to be irrelevant, as these binary vectors are merely used for information storage, and future calculations will only be utilizing vectors of similar lengths within these tensors, allowing for meaningful vector calculations.

The way that novelty is extracted from these networks comes in the form of comparing novel data to all of the existing data in the DAN. This can be done in a variety of ways with an input tensor:

$$I^{1,1,n} = \left[\left[\left[w_1, w_2, \dots, w_n\right]\right]\right] \; with \; w_i \in \{0,1\}^{l_i}, \; l_i \in N$$

With $I^{1,1,n}$ representing a 1 by 1 by n tensor that stores the characteristics of the novel data in a similar way to that of the existing tensors in the DAN. An example of this can be seen below:

DAN $D^{1,4,3}$:

[[[0, 0, 1], [1, 0], [1, 0]],
[[1, 0, 0], [1, 0], [0, 1]],
[[0, 1, 0], [1, 0], [0, 1]],
[[1, 0, 0], [0, 1], [1, 0]]]

Input $I^{1,1,3}$:

[[[0, 1, 0], [0, 1], [0, 1]]]

Where $I^{1,1,3}$ is defined as a "column tensor" to allow for the appropriate multiplications. The first thing to note here is that each binary vector o_i of a given index i for all tensors within the DAN are the same length, which is an essential component of the DAN architecture. This can be thought of as the way each characteristic of each data member is represented as a binary vector. For example, the first binary vector of each tensor has three elements. Perhaps those elements could represent the color of each data member, maybe representing [1, 0, 0] as red, [0, 1, 0] as blue, and [0, 0, 1] as green. This would also mean that the 2nd and 4th data members (represented as the 2nd and 4th tensors in the DAN) are both red.

Another important thing to notice is that each binary vector o_i has exactly 1 element filled out. This will be important for future calculations that will involve tensor/matrix multiplication. The way to construct a DAN in this manner would be to define the length of each binary vector for a given characteristic as being the number of potential unique qualities that a data member can have under that

characteristic, and inputting 1 for the relevant quality and 0 for the rest (ie. if a data member can be one of 3 colors, make a vector of length 3, denoting the first position as one color, the second as another color, and the third as the last color).

The first important output of a DAN is a comparison between the input tensor and all of the other tensors already in the DAN. This is done by a tensor multiplication between the DAN and the input tensor, and looks like this:

$$D^{1,m,n} \cdot I^{1,1,n} = \left[T_{1,1,n}\, T_{1,2,n}, \dots, T_{1,m,n}\right] \cdot I^{1,1,n}$$

Each multiplication between the input tensor and one of the DAN tensors will essentially boil down to a dot product, which will look like such:

$$T_{1,j,n} \cdot I^{1,1,n} = \left[\left[\left[v_1, v_2, \dots, v_n\right]\right]\right] \cdot \left[\left[\left[w_1, w_2, \dots, w_n\right]\right]\right] = \sum_{k=1}^{n} v_k \cdot w_k$$

Where $\sum_{k=1}^{n} v_k \cdot w_k$ will simply be a scalar. Doing this multiplication for all m data members will yield a 1 by m by 1 tensor filled with scalars $V^{1,m,1}$. Here we will also define a new vector u^m, which is essentially $V^{1,m,1}$ reduced to a column vector of m entries for simplicity.

Qualitatively, matrix-multiplying this input tensor with the DAN is essentially performing a dot product between the input tensor (which is our novel data that we are putting into the DAN) and each data member within the DAN. Recall that a characteristic of the DAN and its tensor clusters is that all of the binary vectors in a given index *j* across both the tensors in the DAN and the input tensor are of the same length, so when matrix multiplying the DAN by the input tensor, a dot product will take place between both the similar-length tensor elements *and* the similar-length binary vectors within each tensor. Ultimately, this operation will quantitatively determine how closely aligned the input tensor is with the other tensors in the DAN, which is the mathematical way of saying "how similar is the novel data to every other data member in the DAN". (Note: it is common practice to normalize the scalars in $V^{1,m,1}/u^m$ between 0 and 1)

Keeping with the example above, here is what this operation would look like:

$$D^{1,4,3} \cdot I^{1,1,3} = \begin{bmatrix} [\,[0,0,1],[1,0],[1,0]\,], \\ [\,[1,0,0],[1,0],[0,1]\,], \\ [\,[0,1,0],[1,0],[0,1]\,], \\ [\,[1,0,0],[0,1],[1,0]\,] \end{bmatrix} \cdot \begin{bmatrix} [\,[0,1,0], \\ [0,1], \\ [0,1]\,] \end{bmatrix}$$

$$= \begin{bmatrix} [\,[0]\,], \\ [\,[1]\,], \\ [\,[2]\,], \end{bmatrix}$$

$$[\,[1]\,]\;\;]$$

Or normalized to 3 and in vector notation ($V^{1,4,1} \Rightarrow u^{4}$):

$$[\,0, \tfrac{1}{3}, \tfrac{2}{3}, \tfrac{1}{3}\,]$$

This output tensor $V^{1,4,1}$ is essentially describing how similar the input tensor is with the other tensors in the DAN, showing no similarity to the first data member, $\frac{1}{3}$ similarity with the second, $\frac{2}{3}$ similarity with the third, and $\frac{1}{3}$ similarity with the last data member. You can also check this via inspection.

Now let's say we input an *incomplete* tensor with a missing characteristic, say height (which can be represented by the second binary vector, with the first place representing tall and the second place representing short), and we want to know, based off of the other data members in the DAN, what is likely the height of this input tensor. Holding with the example above, we can simply input our input above, but with [0, 0] as the height vector. This will look like such:

$$D^{1,4,3} \cdot I^{1,1,3} = \begin{bmatrix} [\,[0,0,1],[1,0],[1,0]\,], \\ [\,[1,0,0],[1,0],[0,1]\,], \\ [\,[0,1,0],[1,0],[0,1]\,], \\ [\,[1,0,0],[0,1],[1,0]\,]\;\;] \end{bmatrix} \cdot \begin{bmatrix} [\,[0,1,0], \\ [0,0], \\ [0,1]\,]\;] \end{bmatrix}$$

$$= \begin{bmatrix} [\,[\varepsilon]\,], \\ [\,[1]\,], \\ [\,[2]\,], \\ [\,[\varepsilon]\,]\;\;] \end{bmatrix}$$

Or normalized to 3 and in vector notation ($V^{1,4,1} \Rightarrow u^{4}$):

$$[\,\varepsilon, \tfrac{1}{3}, \tfrac{2}{3}, \varepsilon\,]$$

With $\varepsilon \approx 0$ (This will be important later). What we can do now is construct a new tensor $M^{1,m,n} = [U_{1,1,n}, U_{1,2,n}, \ldots, U_{1,m,n}]$ that is identical to the original DAN $D^{1,m,n}$, except the 1's will be replaced with the associated scalar from the output vector u^{m}. We call this the Updated DAN Tensor, which will look as such:

$$M^{1,4,3} = \begin{bmatrix} [\,[0, 0, \varepsilon], [\varepsilon, 0], [\varepsilon, 0]\,], \\ [\,[\frac{1}{3}, 0, 0], [\frac{1}{3}, 0], [0, \frac{1}{3}]\,], \\ [\,[0, \frac{2}{3}, 0], [\frac{1}{3}, 0], [0, \frac{1}{3}]\,], \\ [\,[\varepsilon, 0, 0], [0, \varepsilon], [\varepsilon, 0]\,] \end{bmatrix}$$

Formally, the calculation for any specific binary vector in a given tensor in $M^{1,m,n}$ will look like such:

$$M_{1,j,k} = D_{1,j,k} \cdot u_j$$

Where $D_{1,j,k} \cdot u_j$ is a scalar multiplication between tensor $D_{1,j,k}$ and scalar u_j. The final element that will be needed for this calculation is another tensor $O^{1,1,n}$. This tensor can be thought of as an "output tensor" and will be defined as a "row tensor" made up of n vectors. The elements of these vectors, however, will be the maximum value of the given index p_i of all the binary vectors among a given index k of all the tensors in $M^{1,m,n}$. Formally, a given input within a given binary vector within $O^{1,1,n}$ will be derived by:

$$O_{1,1,k_{p_i}} = max(S),\ where\ S = \left\{ U_{1,1,k_{p_i}},\ U_{1,2,k_{p_i}},\ \dots,\ U_{1,m,k_{p_i}} \right\}$$

Where $O_{1,1,k_{p_i}}$ is the p_ith element of the kth vector of the tensor $O_{1,1,k_{p_i}}$, and $U_{1,j,k_{p_i}}$ is the p_ith element of the kth vector of the jth tensor of the tensor $M^{1,m,n}$. $O^{1,1,n}$ is known as the Max Value Tensor, with each individual element containing the maximum value of a given index among all vectors of a given index among all tensors in $M^{1,m,n}$. Taking this maximum value $O_{1,1,k_p}$ basically returns, for the p_ith quality of the kth characteristic k_{p_i} of the data members, the value of the elements of a cluster tensor(s) $U_{1,j,k}$ in $M^{1,m,n}$, with this cluster tensor(s) aligning best with the input tensor $I^{1,1,n}$ *relative to* all of the other cluster tensors that have the p_ith quality of the kth characteristic k_{p_i} of the data members. For our example above with $M^{1,4,3}$, this is what the Max Value Tensor would look like:

$$O^{1,1,3} = [\,[\tfrac{1}{3}, \tfrac{2}{3}, \varepsilon], [\tfrac{2}{3}, \varepsilon], [\varepsilon, \tfrac{2}{3}]\,]$$

An interesting thing to note here is that, if you recall, $M^{1,4,3}$ was created via the scalar multiplication between the tensors of $D^{1,4,3}$ and the elements of u^4, where u^4 was constructed via the

tensor $V^{1,4,1}$, which was derived via matrix multiplying the original DAN $D^{1,4,3}$ and the input tensor $I^{1,1,3}$. Remember, however, that our input tensor for this specific $O^{1,1,3}$ looked like this:

$$[\ [\ [0,\ 1,\ 0],\ [0,\ 0],\ [0,\ 1]\ ]\ ]$$

Which notably has no input filled out for its second binary vector. Despite this, our Max Value Tensor $O^{1,1,3}$ does have a filled out second binary vector. So what does this mean and why is this observation significant?

By inputting an incomplete input tensor, that is equivalent to us saying that we have a data member where we know some of its qualities but not others. For this example, perhaps we know the input's color and shape but not its height. When we input this tensor into the DAN, we essentially compare the input tensor to all of the other tensors in the DAN, which represent the data members that the DAN was trained on, *to see which data member in the DAN aligns closest with the given input tensor*. What then happens is we determine, for each quality of each characteristic of the data members, what the maximum value of all of the cluster tensors that have the given quality of the given characteristic (this value being the alignment of each cluster tensor with the input tensor). This results in $O^{1,1,3}$.

If you look closely at $O^{1,1,3}$, you'll notice that for each vector, the index of the greatest value of the vector corresponds with the index of the vector with a 1 of the cluster tensor with the greatest alignment to the input tensor. For our example, this basically means that the indices with $\frac{2}{3}$ as their value in $O^{1,1,3}$ (since $\frac{2}{3}$ is the largest number in $O^{1,1,3}$) match the indices with 1 of the most aligned cluster tensor. Here is a side-by-side comparison of $O^{1,1,3}$ and the cluster tensor in the original DAN $D^{1,4,3}$ that aligned most with the Input tensor $I^{1,1,3}$:

$$[\ [\tfrac{1}{3}\ ,\ \tfrac{2}{3}\ ,\ \varepsilon],\ [\tfrac{2}{3}\ ,\ \varepsilon],\ [\varepsilon,\ \tfrac{2}{3}\ ]\ ]$$
$$[\ [0,\ 1,\ 0],\ [1,\ 0],\ [0,\ 1]\ ]$$

If there is a single cluster tensor that is the "winner", then this phenomena will always be true. For situations with $q > 1$ winners, the number of occurrences r of the max value within each vector of $O^{1,1,n}$ will be $1 \leq r \leq q$, with the exact value depending on how much overlap exists between the "winners" for a given characteristic.

Another way to interpret $O^{1,1,n}$ is to say that it "picks out" the winning cluster(s) among the rest, which emerges from the max values that lie in each vector of $O^{1,1,n}$. This observation can help us answer the question of why, despite our input tensor not holding an input for the second binary vector, $O^{1,1,3}$ returns a value for the second vector. This is because $O^{1,1,3}$ essentially returns the winning cluster(s) that arise between a comparison of the input tensor and the DAN cluster tensors, so regardless of what the input is, $O^{1,1,3}$ will always return the winning cluster(s) in the form of max values at certain indices.

This is also where some of the novelty of DANs come into play. We inputted an incomplete tensor into this DAN, and it outputted a complete tensor. Depending on the way the DAN is structured, DANs can "predict" what the missing data of our input tensor might be based on the other data in the

DAN, which can lead to a variety of avenues including simple database lookup, future prediction, agent based modelling, unsupervised learning, and much more. On top of that, DANs are never fully "trained" in the same sense as current ANNs are, as the addition of data is as easy as adding a new tensor to the DAN and proceeding calculations with this updated DAN.

The next important piece of information pertinent to DANs is what's known as a Max/Sub Count Tensor. This tensor can be thought of as a means of storing information related to the number of occurrences that certain aspects of the data appear in the dataset. More concretely, it can be said like this: For a given quality of a given characteristic k_{p_i}, how many clusters $U_{1,j,k_{p_i}}$ that contain k_{p_i} will have a given quantitative alignment x with the input tensor $I^{1,1,n}$. As an example, you could say that for a dataset that consists of elements color, shape, and height; how many clusters in the dataset have the component red and have a ⅔ alignment with the input tensor?

The Max/Sub Count Tensor is a 1 by 1 by n tensor $S^{1,1,n}$ with each element being a matrix with dimensions $n + 1$ by o_i, with o_i being the number of qualities in the kth characteristic and n, as usual, being the number of characteristics that a data member can have. To construct this tensor, we must first start my defining a new vector q^{n+1} with elements:

$$q^{n+1} = [\varepsilon, \frac{1}{n}, \frac{2}{n}, \dots, \frac{n}{n}]$$

This vector will hold all of the possible values that any given index in $M^{1,m,n}$ could hold. Formally, any entry of a given matrix inside is derived by such:

$$S_{1,1,k_{l,p_i}} = \sum_{j=1}^{m} \delta_{M_{1,j,k_{p_i}},q_l}$$

$$where\ \delta_{M_{1,j,k_{p_i}},q_l} \equiv \begin{cases} 0\ for\ M_{1,j,k_{p_i}} \neq q_l \\ 1\ for\ M_{1,j,k_{p_i}} = q_l \end{cases}$$

Where l indexes from 1 to $n + 1$, k indexes the categories, p_i indexes the qualities, and j indexes the tensors/data members in the DAN. This was also the reason that we needed to define ε, as these tensors are filled with 0s for null values, there needs to be a way to discriminate between values within a cluster that have no alignment to the input tensor and null values in the tensor. For interpretation sake, ε can be assumed to be 0.

A lot of interesting interpretations can be made about these Max/Sub Count Tensors. The kth matrix in the tensor can be thought of as holding all of the possible information regarding the clusters in

the DAN and their overlap with respect to the kth characteristic, and the p_ith row in this kth characteristic basically says that, for all of the clusters that have this p_ith quality in the DAN, how much does each of these clusters align with the given input tensor, with each entry in the row representing a different amount of "completedness" (more rigorously, the lth entry of the p_ith row of the kth matrix contains the number of clusters with the p_ith quality of the kth characteristic that have a q_lth alignment with the input tensor). Also, If we assume that, for the kth characteristic, every cluster only has 1 quality filled out, then the sum of all of the elements of the kth matrix will equal m, and if every cluster has exactly 1 element filled out in each of its binary vectors, then the elements in every matrix in the Max/Sub Count Tensor $S^{1,1,n}$ will sum to m. It is also important to note that q^{n+1} is constructed the way it is because it represents all of the possible alignments that a given cluster can have with the input tensor, so all clusters with a given p_i will be represented somewhere in the corresponding Max/Sub Count Tensor. Here is an interesting calculations that incorporates both the Max/Sub Count Tensor and the Max Value Tensor:

To start, we will be reutilizing the output tensor $O^{1,1,n}$ to hold all of the outputs we receive. This calculation will look like such:

$$O_{1,1,k_{p_i}} = \sum_{l=1}^{n+1} S_{1,\,1,\,k_{l,p_i}} \cdot q_l$$

This can be thought of as, for all of the clusters with the p_ith quality, multiplying the number of clusters with a given alignment q_l to the input tensor by q_l for all q_l and adding up the results. This is one way where meaningful calculations can be made to make AI predictions, however further testing is needed to see where this calculation thrives. An important feature of this calculation that is lost in previous calculations is that this calculation takes into consideration not only the alignment of each cluster with the input tensor, but also the *quantity* of clusters that have said given alignment. This leads to the interesting possibility for application in unsupervised learning, where repeated exposures to a phenomena can have an impact in decision making.