Hybrid Classification and Symbolic-Like Manipulation Using Self-Regulatory Feedback Networks

Tsvi Achler[1] and Eyal Amir

Abstract. We propose a hybrid model based on self-regulatory feedback. It is a connectionist network, which can be composed in a modular fashion. It can be composed of simple constructs that can be combined together to interact logically without requiring a-priori declaration of 'interaction-type' connections. It is primarily a classifier but resolves binding and is pliable to certain symbolic manipulations. We show that 1) compositions can simply be combined to create a hybrid-recognition/symbolic network. 2) Demonstrate how these compositions perform binding logic. 3) Lastly, show how representations can be manipulated in a symbolic-like fashion. These properties are an integral part of intelligent inference and such networks provide a new direction for future research.

INTRODUCTION

Artificial Intelligence researchers continue to face huge challenges in their quest to develop truly intelligent systems. Neural-symbolic integration may bring an opportunity to integrate well-founded symbolic artificial intelligence. Hybrid models may inherit the best of both approaches.

We propose a hybrid model based on self-regulatory feedback. It is a connectionist network, but is modular. It can be composed of simple constructs that can be combined together to interact logically without requiring a-priori declaration of 'interaction-type' connections. It is primarily a classifier but resolves binding and is pliable to certain symbolic manipulations.

Self-Regulatory Feedback Networks (RFN) are unique because 1) they only require connections between inputs and outputs (simple connectivity that is resistant to a combinatorial explosion) and 2) they do not require conventional connection weights (maintain flexibility). They function in a recursive fashion [1-4].

This type of network is be better suited for symbolic manipulation than conventional connection-weight models and represents its own flavor of network symbolism.

Section 1 introduces the theory and equations of this network. Section 2 & 3 shows how compositions can simply be combined to create a hybrid-recognition/symbolic network. Section 3 shows how these compositions perform binding logic. Section 4 we show how representations can be manipulated in a symbolic-like fashion.

1. Model

RFNs use top-down regulatory feedback to modify input activation. The modified input activity is then re-distributed to the network. This is repeated iteratively to dynamically determine stimuli relevance. In this manner top-down regulatory feedback determines the relevance of inputs to an output node.

This model does not assume calculations are based on predetermined connection weights. All input features are connected equally to associated output nodes. Thus each representation is equally connected to all of its parts. Since connection weights are not relevant, only qualitative relations between feature membership to classes need to be determined during setup; e.g. $y_1 \epsilon \{x_1, x_3...\}$, $y_2 \epsilon \{x_2, x_3...\}$. These represent symbolic-like interconnections.

Regulatory feedback has been proposed as an alternate model of lateral inhibition [5]. Regulatory feedback was separately developed as a model for classification [1, 2], for binding and distributed processing [3], and as a multiclass classifier that can process multiple stimuli simultaneously [4]. This work focuses on the flexibility within this model to manipulate representations in a symbolic-like manner.

1.1 Model Function

Using our approach features weights are not assigned. Instead, a feature's information content is dynamically evaluated under different contexts. RFN inhibits *ambiguous* inputs during classification. An input is *ambiguous* if multiple candidate representations using that input are active. Thus, RFNs are more flexible towards combinations of priors since no weighted relations between features or nodes are defined a-priori. This distinction is crucial – traditional Neural Network (NN) notions of a feature being *uninformative* are associated statically between representation and its input feature(s). NN feature extraction methods will assign reduced weights to features that are generally uninformative as determined by the training set.

1.2 Model Schematic

RFN is unique due to the tight association between input features and outputs representations. This is implemented by a triad of interconnections between an input, the output it supports and feedback from that output (Figure 1). Every input has a corresponding feedback 'Q', which samples the output processes that the input cell activates. The feedback modulates the input.

---

[1] University of Illinois at Urbana Champaign, Urbana, IL 61801 USA (217-244-7118; fax: 217-265-6591; e-mail: achler@uiuc.edu).
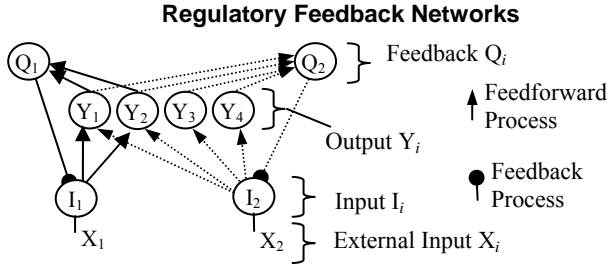
Regulatory Feedback Networks

Figure 1: Regulatory Feedback Schematic. Each external input X_i is modified by feedback Q_i to produce a feedback-updated input I_i. Each feedforward connection has an associated feedback connection. For example, if I_1 projects to Y_1 & Y_2, then Q_1 must receive projections from Y_1 & Y_2 and feedback to the input I_1. Similarly if I_2 projects to Y_1, Y_2, Y_3, & Y_4, then Q_2 receives projections from Y_1, Y_2, Y_3, & Y_4 and projects to I_2. Since the connections between I & Q are symmetric, thus can be drawn using bidirectional connections (fig 2).

Every output must project to the feedback Q processes that correspond to its inputs. For example if an output process receives inputs from I_1 and I_2 it must project to Q_1 and Q_2. If it receives inputs from I_1, it only needs to project to Q_1.

This creates a situation where an output cell can only receive full activation from an input if that input's Q is low. The Q is low if the sum of activation of the outputs that use that input is low. Thus, if representations that share the input are very active, no cell will receive full activation from that input. If outputs share inputs, they inhibit each other at their common inputs, forcing the outcome of competition to rely on other non-overlapping inputs. The more cells have overlapping inputs, the more competition exists between them. The less overlap between two output cells, the less competition, more independent from each other the output cells can be.

The networks dynamically test recognition of representations using 1) regulatory feedback to the individual inputs of representations 2) modifying the next input state based on the input's use 3) re-evaluating representations based on new activity. Steps 1-3 are continuously cycled through. This requires a tight association between inputs and outputs and feedback processes.

RFN is flexible because it doesn't a-priori define which input is ambiguous. Which input is ambiguous depends on which representation(s) are active which in turn depends on which stimuli and task are being evaluated.

1.3 Equations

This section introduces general nonlinear equations governing RFN. Borrowing nomenclature from engineering control theory, this type of inhibitory feedback is negative feedback, in other words stabilizing or regulatory feedback.

For any output cell Y denoted by index a, let N_a denote the input connections to cell Y_a. For any input cell I denoted by index b, let M_b denote the feedback connections to input cell I_b. The feedback to input I_b is defined as Q_b. Q_b, is a function of the sum of activity from all cells Y_j that receive activation from that input:

$$Q_b = \sum_{j \in M_b} Y_j(t) \qquad (1)$$

Input I_b is regulated based on Q_b, which is determined by the activity of all the cells that project to the input, and driven by X_b which is the raw input value.

$$I_b = \frac{X_b}{Q_b} \qquad (2)$$

The activity of Y_a is a product of its previous activity and the input cells that project to it. This property can arise biologically from NMDA channels that are found within neuron membranes. These channels are mediated by previous neuron activity. If a self-multiplicative (delay) term is not included in *equation 3*, the network can immediately change values and the feedback will not be a function of previous activity. Thus the equations are designed so the output cells are proportional to their input activity, inversely proportional to their Q feedback and also depend on their previous activity [1-4]. n_a represents the number of processes in set N_a.

$$Y_a(t + \Delta t) = \frac{Y_a(t)}{n_a} \sum_{i \in N_a} I_i \qquad (3)$$

$$= \frac{Y_a(t)}{n_a} \sum_{i \in N_a} \frac{X_i}{Q_i} = \frac{Y_a(t)}{n_a} \sum_{i \in N_a} \left(\frac{X_i}{\sum_{j \in M_i} Y_j(t)} \right) \qquad (4)$$

1.4 Simple Connectivity

Feedback networks do not require a vast number of connections; the number of connections required for competition is a function of the number of inputs the cell uses. Addition of a new cell to the network requires only that it forms symmetrical connections about its inputs and not directly connect with the other output cells. Thus the number of connections of a specific cell in feedback competition is independent of the size or composition of the classification network, allowing large and complex feedback networks to be combinatorially and biologically practical.

2. Simple Modular Composition
Networks can be created in a symbolic/modular fashion. Suppose a node to represent the patterns associated the letter P and another node represents the patterns associated the letter R. R shares some patterns with P. We can intuitively combine these nodes into one network, by ignoring this overlap and just connect the network. This defines a functioning network without formally learning how nodes R and P should interact.
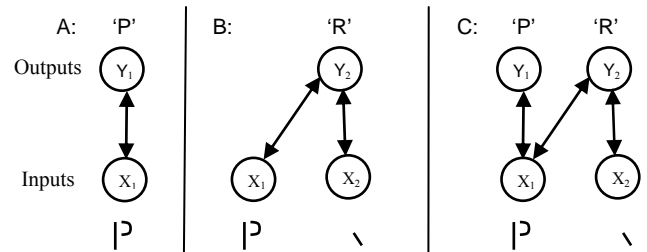

Figure 2 (A-C): Modular nodes Y_1 and Y_2 (A & B respectively) can be simply combined to form a combined network (C). Since I & Q are symmetric, the network can be drawn using bidirectional connections.

The two nodes are connected such that the inputs of Y_1 (input pattern 'P') completely overlaps with a larger Y_2. But node Y_2 also receives an independent input pattern '\'. The solutions are presented as (*input values*) → (*output vectors*) in the pattern $(X_1,X_2,) \rightarrow (Y_1,Y_2)$. The steady state solution for example 1 is $(X_1, X_2) \rightarrow (X_1 - X_2, X_2)$. Substituting our input values we get $(1,1) \rightarrow (0,1)$, $(1,0) \rightarrow (1,0)$. Given only input pattern X_1 ('P') the smaller node wins the competition for representation. Given both input patterns ('P' & '\') the larger node wins the competition for

representation. Though we used binary inputs, the solution is defined for any positive real X input values (Achler 2007). The mathematical equations and their derivation can be found in the Appendix.

Descriptively, Y_1 has one input connection, thus by definition its fixed connection weight is one. Its maximal activity is 1 when all of its inputs are 1. Y_2 has two input connections so its fixed input connection weights are one-half. When both inputs ('P' & '\') are 1 the activity of Y_2 sums to 1. Note these weights are predetermined by the network. Connections are permanent and never adjusted. Input I_1 projects to both Y_1 & Y_2, thus receives inhibitory feedback from both Y_1 & Y_2. Input I_2 projects only to Y_2 so it receives inhibitory feedback from Y_2. The most encompassing representation will predominate without any special mechanism to adjust the weighting scheme. Thus, if inputs 'P' & '\' are active Y_2 wins. This occurs because when both inputs are active, Y_1 must compete for all of its inputs with Y_2, however Y_2 only needs to compete for half of its inputs (the input shared with Y_1) and it gets the other half 'free'. This allows Y_2 to build up more activity and in doing so inhibit Y_1.

Thus smaller representation completely encompassed by a larger representation becomes inhibited when the inputs of the larger one are present. The smaller representation is unlikely given features specific only to the large representation. It demonstrates that RFN determines negative associations ('Y_1' *is unlikely given feature* 'X_B') even though they are not directly encoded. In order to encode such negative associations using conventional methods, they would have to be 'hard-wired' into the network. With NN, each possible set of stimuli combinations would have to be trained. With direct Competition each possible negative association would have to be explicitly connected.

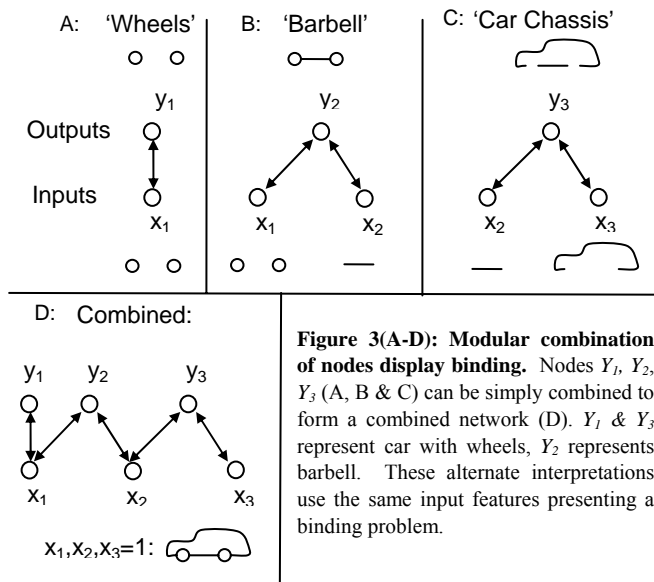## 3 Binding with Simple Modular Composition

A: 'Wheels' B: 'Barbell' C: 'Car Chassis'

D: Combined:

$x_1, x_2, x_3 = 1$:

Figure 3(A-D): Modular combination of nodes display binding. Nodes Y_1, Y_2, Y_3 (A, B & C) can be simply combined to form a combined network (D). Y_1 & Y_3 represent car with wheels, Y_2 represents barbell. These alternate interpretations use the same input features presenting a binding problem.

We simultaneously evaluate the criteria of three representations. In e.g. 2, expanded from e.g. 1, three cells partially overlap. As in e.g. 1, Y_1 competes for its single input with Y_2. However, now Y_2 competes for its other input with Y_3 and Y_3 competes for only one of its inputs.

The steady state solution is $(X_1, X_2, X_3) \rightarrow (Y_1 = X_1 - X_2 + X_3, Y_2 = X_2 - X_3, Y_3 = X_3)$.

If $X_2 \leq X_3$ then $Y_2=0$ and the equations become $(X_1, 0, \dfrac{X_2 + X_3}{2})$.

If $X_3 = 0$ the solution becomes that of e.g. 1: $(X_1, X_2, 0) \rightarrow (X_1 - X_2, X_2, 0)$.

The results are: $(1, 0, 0) \rightarrow (1, 0, 0)$;
$(1, 1, 0) \rightarrow (0, 1, 0)$;
$(1, 1, 1) \rightarrow (1, 0, 1)$.

Derivations can be found in the appendix.

Thus if input X_1 is active, Y_1 wins. If inputs X_1 and X_2 are active and Y_2 wins for the same reasons this occurs in e.g 1. However, if inputs X_1, X_2 and X_3 are active then Y_1 and Y_3 win. The network as a whole chooses the cell or cells that best represent the input pattern with the least amount of competitive overlap.

In e.g. 2, Y_2 must compete with all of its inputs: X_1 with Y_1, X_2 with Y_3. Y_3 only competes for half of its inputs (input X_2) getting input X_3 'free'. Since Y_2 is not getting its other input X_1 'free' it is at a competitive disadvantage to Y_3. Together Y_1 and Y_3, mutually benefit from each other and force Y_2 out of competition. Competitive information travels indirectly 'through' the representations. Given active inputs X_1 and X_2 =1, the activity state of Y_1 is determined by input X_3 through Y_3. If input X_3 is 0 then Y_1 becomes inactive. If input X_3 is 1, Y_1 becomes active. However, Y_3 does not even share input X_1 with Y_1.

3.1 Binding

Choosing Y_2 given $(1,1,1)$ is equivalent to choosing the irrelevant features for binding. If the inputs represent spatially invariant features where feature X_1 represents circles, X_3 represents the body shape and feature X_2 represents a horizontal bar. Y_1 is assigned to represent wheels and thus when it is active, feature X_1 is interpreted as wheels. Y_2 represents a barbell composed of a bar adjacent to two round weights (features X_1 and X_2). Note: even though Y_2 includes circles (feature X_1), they do not represent wheels (Y_1), they represent barbell weights. Thus if Y_2 is active feature X_1 is interpreted as part of the barbell. Y_3 represents a car body without wheels (features X_2 and X_3), where feature X_2 is interpreted as part of the chassis. Now given an image of a car with all features simultaneously (X_1, X_2 and X_3), choosing the barbell (Y_2) even though technically a correct representation, is equivalent to a binding error within the wrong context in light of all of the inputs. Most classifiers if not trained otherwise are as likely to choose barbell or car chassis (see figure 4). In that case the complete picture is not analyzed in terms of the best fit given all of the information present. Similar to case 1, the most encompassing representations mutually predominate without any special mechanism to adjust the weighting scheme. Thus the networks are able to evaluate and bind representations in a sensible manner for these triple cell combinations.
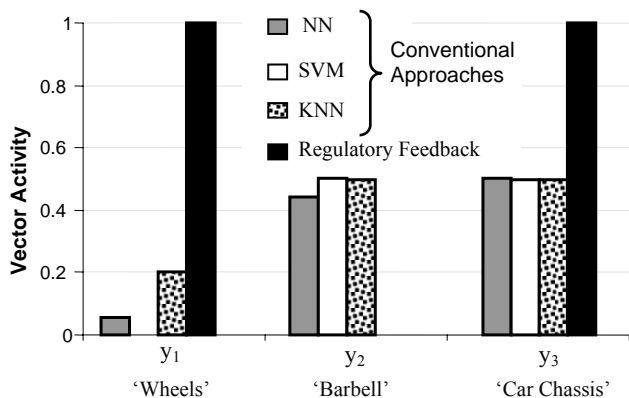
Figure 4: Presentation of this binding problem to conventional classifier approaches. The Neural Networks (NN) and Support Vector Machines (SVM) were trained on y_1, y_2, y_3 based on x_1 x_2 and x_3, using the WEKA classifier tool. K-nearest neighbors was determined by the closest neighbor, 1-N.

Optimized NN and SVM learning is performed using the most recent version of the *Waikato Environment for Knowledge Analysis* (WEKA) package currently available [6].

4. Symbolic-like Manipulation

The behavior of the network can be changed by forcing the values of the output nodes. The value of a node can be artificially increased or decreased. For example, forcing a representation to have a zero value is equivalent to eliminating it from the network. Artificially activating a representation gives it priority over other nodes and can forces its representation to override inherent binding processes.

We repeat example 2 but can ask the question: can a barbell shape be found in any form? We introduce a small bias to Y_2 (representing barbell) according to the equation modified from example 2:

$$Y_2(t+dt) = \frac{Y_2(t)}{2}\left(\frac{X_1}{Y_1(t)+Y_2(t)} + \frac{X_2}{Y_2(t)+Y_3(t)}\right) + b$$

Choosing a bias b of 0.2 and activating all inputs, such as car, the network results are: $(1, 1, 1) \rightarrow (0.02, 0.98, 0.71)$. The network now overrides its inherent properties and responds to whether inputs matching Y_2 are present. Each representation can be manipulated in a similar manner. This is a form of symbolic manipulation closely tied to recognition. This demonstrates how a symbolic-like manipulation can manipulate a classification scenario.

CONCLUSION

Further symbolic manipulation may be possible by changing internal node properties. The equations currently sum inputs (I's) in a linear fashion. However they can be summed using a sigmoid or other function. The activation function can be designed so that when any of a node's inputs are matched, the node becomes active. Conversely it can be designed so that when only one input is active that node is active. Initial results show that a sigmoid function gives the node an 'AND'-like function. The location of the sigmoid slope can be manipulated determining node behavior between 'AND'-like nodes to more 'OR'-like nodes. Since RFNs

are nonlinear, at this point, we can simulate general cases, but not provide analytical axioms. This is left for future work.

Using a very simple set-up process, RFN allows modular combination of nodes. It is robust in multiple scenarios and computationally economical because it does not require many variables. It only requires simple combinatorialy-plausible connectivity between inputs and outputs. RFN offers a flexible and dynamic approach to intelligent applications. The network can also be manipulated to perform search tasks by biasing output representations. These properties demonstrate that such networks can be an integral part of intelligent inference and provide a new direction for future research.

APPENDIX

Example 1. RFN equations are:

$$y_1(t+dt) = \frac{y_1(t)x_1}{y_1(t)+y_2(t)} \quad , \quad y_2(t+dt) = \frac{y_2(t)}{2}\left(\frac{x_1}{y_1(t)+y_2(t)} + \frac{x_2}{y_2(t)}\right)\cdot$$

The network solution at steady state is derived by setting $y_1(t+dt)=y_1(t)$ and $y_2(t+dt)=y_2(t)$ and solving these equations. The solutions are $y_1 = x_1 - x_2$ and $y_2 = x_2$. If $x_1 \leq x_2$ then $y_1 = 0$ and the equation for y_2 becomes: $y_2 = \frac{x_1 + x_2}{2}$.

Example 2. Equation y_1 (t+dt) remains the same as example 1. y_2 (t+dt) and y_3 (t+dt) become:

$$y_2(t+dt) = \frac{y_2(t)}{2}\left(\frac{x_1}{y_1(t)+y_2(t)} + \frac{x_2}{y_2(t)+y_3(t)}\right) \quad ,$$

$$y_3(t+dt) = \frac{y_3(t)}{2}\left(\frac{x_2}{y_2(t)+y_3(t)} + \frac{x_3}{y_3(t)}\right)$$

Solving for steady state by setting $y_1(t+dt)=y_1(t)$, $y_2(t+dt)=y_2(t)$, and $y_3(t+dt)=y_3(t)$, we get $y_1=x_1-x_2+x_3$, $y_2=x_2-x_3$, $y_3=x_3$. If $x_3=0$ the solution becomes that of e.g. 1: $y_1=x_1-x_2$ and $y_2=x_2$. If $x_2 \leq x_3$ then $y_2=0$ and the equations become $y_1 = x_1$ and $y_3 = \frac{x_2 + x_3}{2}$.

REFERENCES

[1] Achler T, (2002) "Input Shunt Networks, Neurocomputing", 44-46c: 249-255.
[2] Achler T, (2007) "Object classification with recurrent feedback neural networks", Proc. SPIE, Evolutionary and Bio-inspired Computation: Theory and Applications, Vol. 6563.
[3] Achler, T., Amir, E. (2008) "Input Feedback Networks: Classification and Inference Based on Network Structure", Artificial General Intelligence Proceedings V1: 15-26.
[4] Achler, T., C. Omar, Amir, E. (2008). "Shedding Weights: More With Less." Neural Networks IJCNN Proceedings, In Press.
[5] Reggia, J. A., C. L. Dautrechy, et al. (1992). "A Competitive Distribution-Theory of Neocortical Dynamics." Neural Computation 4(3): 287-317.
[6] Ian H. Witten and Eibe Frank (2005) "Data Mining: Practical machine learning tools and techniques", 2nd Edition, Morgan Kaufmann, San Francisco, 2005. Software available at http://www.cs.waikato.ac.nz/ml/weka/ (version 3.5.6)