A Neural Architecture for Unsupervised Temporal Pattern Recognition

Shady El Damaty
Department of Neuroscience
Georgetown University
Washington, D.C. 20067
se394@georgetown.edu

James Maguire
Department of Linguistics
Georgetown University
Washington, D.C. 20067
jrm346@georgetown.edu

Abstract

In this paper we present a novel neuron architecture for unsupervised time dependent neural networks. We provide a new delay function, based on a mixture of Beta distributions, which addresses some of the problems with previous work in this area. In addition we also present a new method of thresholding the potential of a neuron based on input activations from both before and after the neuron spikes. Preliminary results over two simple patterns show extremely high performance on an unsupervised pattern segmentation task, learning in both clean and noisy environments.

Introduction

Noisy time series signals pose numerous challenges for pattern recognition algorithms. The absence of delimiters to signify the beginning and end of a pattern require either prior knowledge of the pattern's properties or a capability for identifying a pattern based on it's statistical properties within the input stream. Neural networks have shown great promise for temporal pattern recognition. A popular approach for detecting unique patterns embedded in a noisy signal have centered around representing the relative inputs of the constituent features that compose a pattern as a system of time delays relative to pattern onset (Hopfield 1996, Tank & Hopfield 1987). Supervised algorithms with time-delay neural networks have previously shown high performance rates ($> 98\%$) for classification of consonant phonemes (Waibel et al. 1989). The hidden layers within this model had learned to represent acoustic-phonetic features for classification of phonemes nested within a noisy signal. Current research has focused on developing unsupervised methods to learning acoustic-phonetic features in real time. Spike-Timing Dependent Plasticity (STDP) is an unsupervised learning rule derived from neurophysiological experiments (Izhikevich 2006). Neurons with an STDP learning rule potentiate their weights as a function of the relative time between input arrival and the neuron's firing time. Inputs that arrive before the neuron emits a spike are potentiated whereas those that arrive afterwards are depotentiated. A STDP learning rule drives the neuron to fire only for those inputs that actually contributed to its activity. Recent work has implemented STDP with time-delay learning for semi-unsupervised learning of weighted delays for spatio-temporal pattern recognition (Moisy et al. 2015). Others have implemented unsupervised STDP with time-delay learning by initializing a random system of time delays and retaining only those delays that represent the temporal structure of the input (Rekabdar 2015). In this paper, we build upon previous literature for fully-unsupervised learning of time-delays using a novel delay function based on the beta distribution.

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

Pattern Representation

Consider a noisy signal $g(\mathbf{f}, \mathbf{t})$ with spatial features $\mathbf{f} = [f_1 ... f_i]$ defined over a time interval $\mathbf{t} = [t_1 ... t_k]$. A unique first-order pattern within $g(\mathbf{f}, \mathbf{t})$ is defined as

$$\xi = \big((f_m, t_o)...(f_n, t_e)\big) \qquad n \geq m > 0 \tag{1}$$

where ξ is a list of elements chosen from $\mathbf{f}$ and appearing within $g(\mathbf{f}, \mathbf{t})$ between each pattern start, t_0, and end, t_e, for a pattern duration $c = t_e - t_o$. Each characteristic feature, f_x, on the interval $x \in [m...n]$ may appear more than once in ξ and is thus associated with a vector of times,

$$\boldsymbol{\tau}_{f_x} = [\tau_p ... \tau_j] \tag{2}$$

where j is the total number of times f_x is present within c. The spatio-temporal signature of ξ is the collected set of all orderings in time of every characteristic feature composing ξ,

$$\boldsymbol{T}_\xi = [\boldsymbol{\tau}_{f_m} ... \boldsymbol{\tau}_{f_n}] \tag{3}$$

The consistent appearance of ξ at times $\boldsymbol{T}_\xi$ constitutes a statistically reproducible pattern that may be learned by an unsupervised algorithm sensitive to the statistical properties of $g(\mathbf{f}, \mathbf{t})$. Pattern ξ is called a first-order pattern in time series $g(\mathbf{f}, \mathbf{t})$ if the signature $\mathbf{T}_\xi$ is a statistically regular sequence unique to ξ and not any other overlapping patterns in $g(\mathbf{f}, \mathbf{t})$.

Neuron Architecture

The set of features that characterize a given ξ nested within a noisy time series may be detected by a neuron, n, receiving feedforward input from a set of feature detector units, $\mathbf{s} = [s_1 ... s_i]$, in one-to-one correspondence with $\mathbf{f}$.

Each unit in $\mathbf{s}$ converts a binary value indicating the presence of an input at a time $\tau_s \in \mathbf{T_S}$ into a time-varying signal driving n over interval $\mathbf{t}$. Recognition of ξ occurs when the activity of n exceeds a suprathreshold value, γ, and emits a spike to signal the detection of the spatio-temporal signature, $\mathbf{T_S}$. The threshold must be set sufficiently high enough to filter out spurious activation and also such that n will only fire when all inputs arrive at the same time. Coincident arrival may only occur if all inputs activate at the same time or if the arrival time of the input from each unit in $\mathbf{s}$ is sufficiently delayed relative to the final detected feature. This constraint is satisfied if each input has a delay function $\mathbf{d} = [d_{s_1}(z) ... d_{s_i}(z)]$, where z is a positive real value and the neuron potential is written as the summation over the input activations within a window since the neuron last fired, ϕ,

$$h(t, \phi) = \sum_{\boldsymbol{\tau_s} \in \phi} d_s(t - \tau_\mathbf{s}) \tag{4}$$

such that t is the current time of the signal $g(\mathbf{f}, \mathbf{t})$. This architecture is illustrated in Figure 1a. The neuron fires a binary spike at time τ_n if $h(t, \phi) > \gamma$. However, if the temporal structure of activated inputs does not correspond to $\mathbf{T}_\xi$ then the arrival times are all out of phase with each other and the potential of the neuron never exceeds γ as shown in Figure 1b.

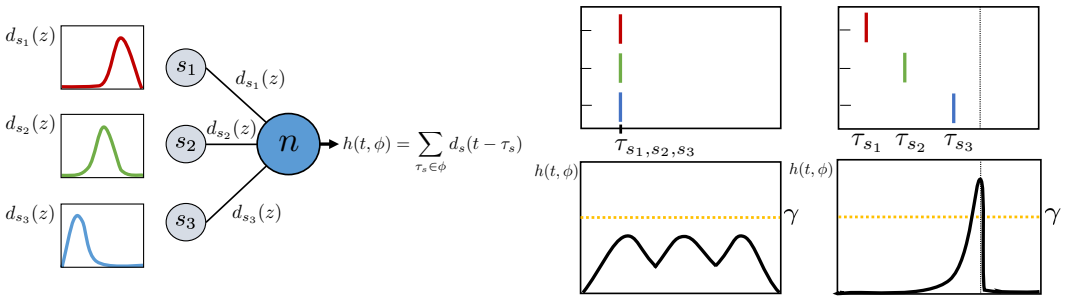


Figure 1: **Architecture of a Single Pattern Detecting Neuron**

Novel Delay Function

The choice of the delay function is crucial for achieving the previously mentioned constraints for n emitting a spike in response to a detected pattern ξ. A popular choice in the literature is the application of weighted fixed delay functions, where the value of the function is dependent solely on the time since input activation, $d_{fixed}(z)$. An example of a weighted fixed delay function appears in Figure 2. Learning with this kind of delay function consists of modulating the height of the delay function, $d_{fixed}(z)$, to push the pattern-detecting neuron over threshold. A major limitation of these models is that such a representation does not allow for the peak value of the delay function to be shifted in time according to the temporal statistics of the pattern to be learned. The beta distribution is ideal for

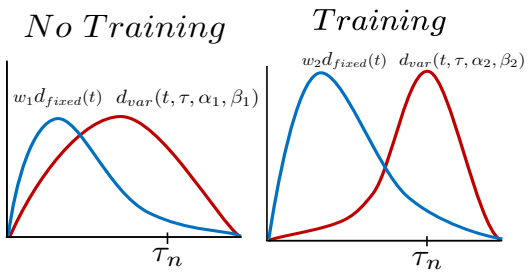


Figure 2: **Fixed vs. Variable Delay Functions**

representing temporal delays due to its adjustable hyperparameters, α, β. These hyperparameters can be adjusted to move the peak value of the distribution in any direction (Figure 2). This means that the peak values of the input functions can be adjusted to arrive coincidentally thus creating a spike in the potential of the neuron as observed in Figure 1b. Taking inspiration from spike-timing dependent plasticity (STDP) models, the peak of the delay function is skewed based on the relative difference between input activation, τ_s, and neural firing time, τ_n. Thus the arrival time of the peak of the delay function is learned to ensure coincidental arrival across inputs at firing time. The beta distribution can be defined as function of the gamma distribution, Γ, where $z \in [0, 1]$ and $\alpha, \beta \geq 0$

$$\beta(z, \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \cdot z^{\alpha-1}(1 - z)^{\beta-1} \tag{5}$$

The maximal duration of the delay function is a hyperparameter, c, which should be selected based on the expected length of the input pattern. The pattern duration must be appropriately scaled since the beta distribution is only defined on the interval $z \in [0, 1]$. In order to calculate the value of the beta delay function at time t with an input activation time τ_{s_i}, z is defined as:

$$z = \frac{(t - \tau_{s_i})}{c} \tag{6}$$

The variable delay function d_{var} with hyperparameters α and β updated during learning, can be written as:

$$d_{var}(t, \tau, \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \cdot \left(\frac{t - \tau_{s_i}}{c}\right)^{\alpha-1} \left(1 - \frac{t - \tau_{s_i}}{c}\right)^{\beta-1} \tag{7}$$

Beta Mixtures

While the definition of d_{var} with only a single beta distribution allows for learning the delay time of an input which is activated only once within the pattern window, c, it fails to approximate the appropriate delay for when an input activates multiple times within c. In order to resolve this problem, d_{var} is defined as a mixture of beta distributions as follows,

$$d_{var}(t, \tau, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{\alpha,\beta \,\in\, \boldsymbol{\alpha},\boldsymbol{\beta}} \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \cdot \left(\frac{t - \tau_{s_i}}{c}\right)^{\alpha-1} \left(1 - \frac{t - \tau_{s_i}}{c}\right)^{\beta-1} \tag{8}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are vectors of α and β values respectively. If f_{var} is activated in the correct sequence, the multiple peaks corresponding to different input activation times will all arrive in phase. Coincident input will drive the neuron potential, now defined as:

$$h(t, \phi) = \sum_{\tau_s \in \phi} d_{var}(t, \tau, \boldsymbol{\alpha}, \boldsymbol{\beta}) \tag{9}$$

to spike and exceed γ. The difference between a mixture and a single component distribution is exemplified in figure 3.

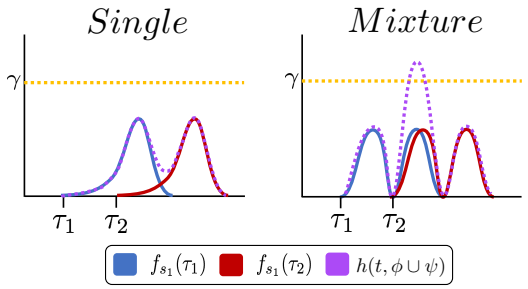


Figure 3: **Representing Multiple Inputs with a Beta Mixture**

Setting the Threshold

Integrate and fire neurons generally possess either a fixed or variable threshold which is dependent on the potential of the neuron at firing. The problem with a fixed threshold is that a static γ causes the firing time, τ_n of the neuron to be prior to the end of the pattern, generally causing τ_n to move to the beginning of the pattern. This is problematic as the neuron only detects the onset and not the complete duration of the pattern. Some groups have addressed this by assembling multiple neurons to represent a single pattern. A static threshold can also be easily overwhelmed by increased noise in the signal. This issue can be addressed with a single neuron that fires at the end of ξ by defining,

$$\phi = \{\tau \mid (t - \tau_s) \leq c \quad and \quad \tau_n < \tau_s\} \tag{10}$$
$$\psi = \{\tau \mid (t - \tau_s) \leq c \quad and \quad \tau_n \geq \tau_s\} \tag{11}$$

where ϕ is the set of input activations within the window, $(t - \tau_s) \leq c$, since the neuron last fired, $\tau_n < \tau_s$, and ψ is the set of activations within the window prior to the neuron's last firing $\tau_n \geq \tau_s$. In contrast to previous work, where the thresholds are dependent on the potential of the network at firing, we define γ to consider the potential of activations for both before and after firing:

$$\gamma = f(h(t, \phi), \gamma) \tag{12}$$

$$\gamma = f(h(t, \phi), h(t, \psi), \gamma) \tag{13}$$

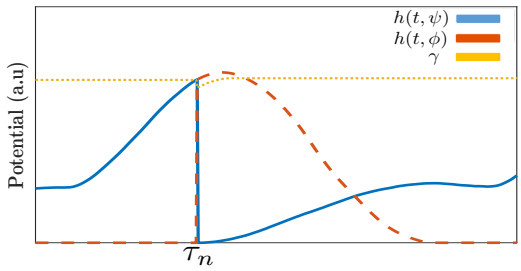


Figure 4: **Contribution of Potential to Threshold**

Learning Rules

In the described neural architecture, there are two sets of parameters that must be learned, the vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to approximate the delay function for each input given $\mathbf{T}_\xi$ and the threshold γ of the neuron.

Updating the Delay Function

$\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are initialized to $[2], [2]$ for all inputs. The rate of learning is adjustable based on a hyperparameter η which determines the rate of change in the skew of an input activation time brought about by redistributing values in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

Weight Redistribution. Redistribution of α and β values amongst a mixture of betas representing some input feature f is important for winnowing out those component delay functions that do not significantly contribute to the neuron's potential while increasing the amplitude of the components with the largest peaks. Thus unused beta functions decay to 0, leaving only the input-specific components in the mixture. Weight redistribution is accomplished with two functions, `Subtract` and `Add Weights`. For each input, a variable w is defined to represent the total sum of α and β that must be redistributed amongst the mixtures composing that input. A factor of η is taken from all α, β pairs as shown in `Subtract Weight` (Algorithm 1). The subtracted value w is then added only to those pairs that contribute the most to $h(x)$, incrementing them in proportion to their total contribution (Algorithm 2). The α and β pairs that contribute most to the neuron's potential are identified by ordering all pairs from greatest to least using a function `ORDERED`. The contingency of additional inputs being encountered is addressed by function, `INCREASE`, in the add weights function. This function evaluates whether a given input is activated more times than the number of α, β pairs for that input and if so adds additional pairs as required.

Algorithm 1 Weight Subtraction

procedure SUBTRACT WEIGHT
  $w \leftarrow 0$
  for $s \in S$ **do**
    for $\alpha_j, \beta_j, \in \boldsymbol{\alpha}_{\boldsymbol{s_i}}, \boldsymbol{\beta}_{\boldsymbol{s_i}}$ **do**
      $w \leftarrow w + \eta\alpha_j + \eta\beta_j$
      $\alpha_j \leftarrow \alpha_j - \eta\alpha_j$
      $\beta_j \leftarrow \beta_j - \eta\beta_j$
  $w \leftarrow w/|\phi|$
  return w

Algorithm 2 Weight Addition

procedure ADD WEIGHT(w)
  for $s \in \{s \mid \tau_s \in \phi\}$ **do**
    if $\mid \tau_{s_i} \mid > |\mathbf{A}_{\mathbf{s_i}}|$ **then** ▷ # of activations > than # of distributions
      INCREASE($\boldsymbol{\alpha}_{\boldsymbol{s_i}}$, $\mid \tau_{s_i} \mid$)
      INCREASE($\boldsymbol{\beta}_{\boldsymbol{s_i}}$, $\mid \tau_{s_i} \mid$)
    $count \leftarrow 0$
    while $count <$ $\mid \tau_{s_i} \mid$ **do**
      $\alpha,\ \beta \leftarrow$ORDERED($\boldsymbol{\alpha}_{\boldsymbol{s_i}}$, $\boldsymbol{\beta}_{\boldsymbol{s_i}}$)$[count]$
      $ratio_\alpha \leftarrow \alpha/(\alpha + \beta)$
      $ratio_\beta \leftarrow \beta/(\alpha + \beta)$
      $\alpha \leftarrow \alpha + w *$ ratio$_\alpha$
      $\beta \leftarrow \beta + w *$ ratio$_\beta$
      $count \leftarrow count + 1$

Skewing the Beta to Represent Structure in Time. The skew in time of each component distribution is evaluated after the α, β weights have been redistributed towards the most contributing functions. The process for skewing the beta functions comprising the mixture for each input is

outlined in Algorithm 3. Only the most contributing α, β pairs are skewed. The skew is computed as the difference between the scaled firing time z and the peak time of a given α, β pair. A given component distribution is moved to the right/left if it occurs before/after firing time, τ_n, resulting in entrainment of τ_n to occur at the end of pattern ξ.

Algorithm 3 Adjusting Skew of Delay

procedure SKEW
  for $s \in \{s \mid \tau_s \in \phi\}$ **do**
    $counter \leftarrow 0$
    while $counter < |\{\tau_{s_i} \in \phi\}|$ **do**
      $\alpha,\ \beta = ORDERED(\boldsymbol{\alpha_{s_i}}, \boldsymbol{\beta_{s_i}})[count]$
      skew$= x - ((\alpha - 1)/(\alpha + \beta - 2))$
      if skew< 0 **then** ▷ Peak of the distribution is to the right of τ_n
        redist$\leftarrow \alpha\eta * -$skew ▷ Move peak to the left
        $\alpha \leftarrow \alpha - redist$
        $\alpha \leftarrow \beta + redist$
      else ▷ Peak of the distribution is to the left of τ_n
        $redist \leftarrow \eta\beta * skew$ ▷ Move peak to the right
        $\beta \leftarrow \beta - redist$
        $\alpha \leftarrow \alpha + redist$

Adaptive Thresholding

Recognition of a pattern ξ embedded in a noisy time series occurs when n exceeds a threshold y and emits a spike. The value of γ must be set high enough to filter out noisy inputs but not too high so as to prevent n from ever firing. An adaptive threshold dependent on the total potential before and after firing is detailed in Algorithm 4. The threshold is incremented as a function of the total potential windowed around τ_n and multiplied by learning rate η.

Algorithm 4 Adaptive Threshold

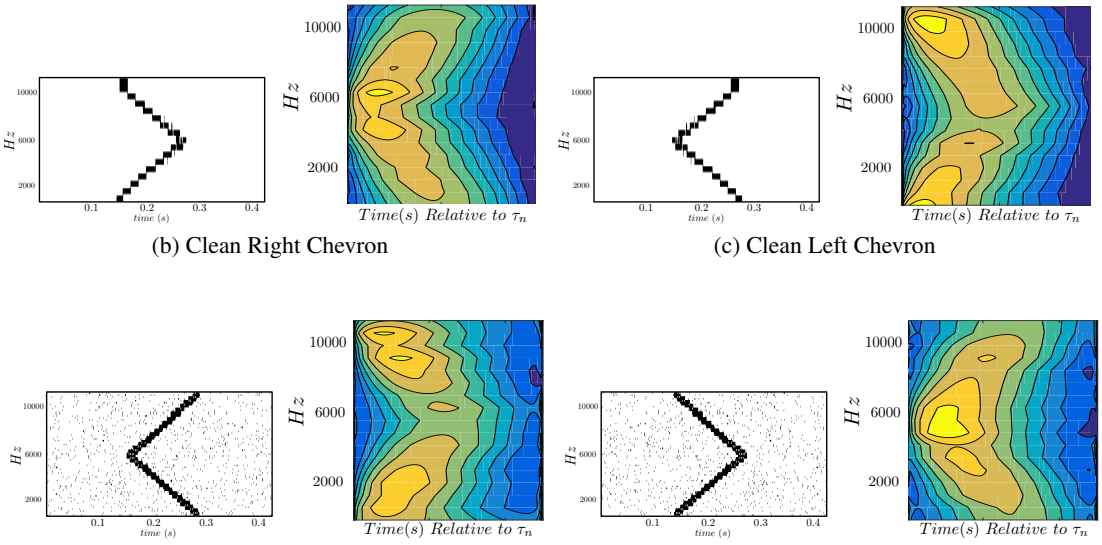
procedure UPDATE THRESHOLD
  if $\eta * (h(t, \phi) + h(t, \psi)) > \gamma$ **then**
    $\gamma \leftarrow \eta * (h(t, \phi) + h(t, \psi))$
  if $h(t, \phi) > \gamma$ **then**
    $\gamma \leftarrow \eta * h(t, \phi)$

Simple-Patterns

Here we present preliminary results using the proposed learning algorithm on simple patterns. Sounds composed of simple frequency sweeps in time with 200ms duration were generated and Fourier transformed to generate our $g(\mathbf{f}, \mathbf{t})$ which is a time-frequency spectrogram where $\mathbf{f}$ is defined over the set of frequencies that sum linearly to reconstruct the original time-domain signal. The algorithm was initialized with $\eta = 0.1$ for both the threshold update and weight redistribution rule with 10 distributions composing the initial beta mixture. The window was set to 1.5x the length of the pattern. The network was trained and tested on left and right chevron shaped frequency sweeps in time both with and without 1:1 signal to noise ratio. Training on clean images resulted in 100% classification accuracy after only 15 presentations. The network also exhibited perfect performance noisy images resulted in 100% classification accuracy but only after 50 presentations (Figure 5).

Discussion

Here we propose a new model for unsupervised recognition of patterns nested within a noisy time series signal. The model was able to learn different simple images embedded in noise with 100% classification rate after only 50 training presentations. The internal representation of the time delays



(b) Clean Right Chevron (c) Clean Left Chevron

(e) Noisy Right Chevron (f) Noisy Left Chevron

Figure 5: **Model Network Performance.** The model was trained on simple patterns with and without noise as shown in column 1. The internal representation of the trained pattern is shown in column 2. Note the internal representation is a mirror image of the input pattern since earlier inputs must be delayed greater features at the end of the pattern.

in the form of the beta mixture was sharpened with training to reflect the arrival times of each of the feature inputs relative to neuron firing time. The internal representation sharpens with increased training presentations and greater number of initial mixture components. Our results demonstrate the choice of a beta distribution for representing time delays of an input with adaptive thresholding is a promising method for development of unsupervised learning temporal pattern algorithms. Future work is required to evaluate the performance of this algorithm on various temporal pattern structures and with different hyper parameter settings.

References

[1] Hopfield, J.J (1995) Pattern Recognition Computation Using Action Potential Timing for Stimulus Representation. *Nature* **376**(6):33-36.

[2] Hopfield, J.J (1996) Transforming Neural Computations and Representing Time. *Proc. Natl. Acad. Sci* **93**:15440-15444

[3] Izhikevich, E.M (2005) Polychronization: Computation with Spikes. *Neural Computation* **18**:245-282.

[3] Rekabdar, B., Nicolescu, M., Kelley R., & Nicolescu, M. (2015) An Unsupervised Approach to Learning and Early Detection of Spatio-Temporal Patterns Using Spiking Neural Networks. *Intell Robot Sys* **80**:83-97

[4] Paugam-Moisy, H., Martinez, R., & Bengio, S. (2006) A Supervised Learning Approach Based on STDP And Polychronization in Spiking Neuron Networks. *Proc. Natl. Acad. Sci* **84**:1896-1900

[5] Tank, D.W. & Hopfield, J.J (1987) Neural Computation by Concentrating Information in Time. *Proc. Natl. Acad. Sci* **84**:1896-1900

[6] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K.L (1989) Phoneme Recognition Using Time-Delay Neural Networks. *IEEE Transactions on Acoustics, Speech & Signal Processing* **37**(3):328-338.

[7] Vries, B. & Principe, J.C (1992) The Gamma Model - A New Neural Model for Temporal Processing. *Neural Networks* **5**:565-576