

TODAM: Pseudocode

Constants

D = 100 ! Vector dimensionality
N = 5 ! Number of cue-response pairs we'll learn

Data Structures

Memory[D]
Retrieved[D]
Cue[N,D]
Response[N,D]

Some Tools

Generate_Vectors(Input[N,D]) ! Returns random Gaussian vectors for items
Convolve(Vector1, Vector2) ! Return association for two input items
Correlate(Vector1, Vector2) ! Retrieve response, given cue and memory
Cosine(Vector1, Vector2) ! Similarity $-1 < x < 1$

Creating Memory

Let's create memory, adding in each item and their association as we read the list:

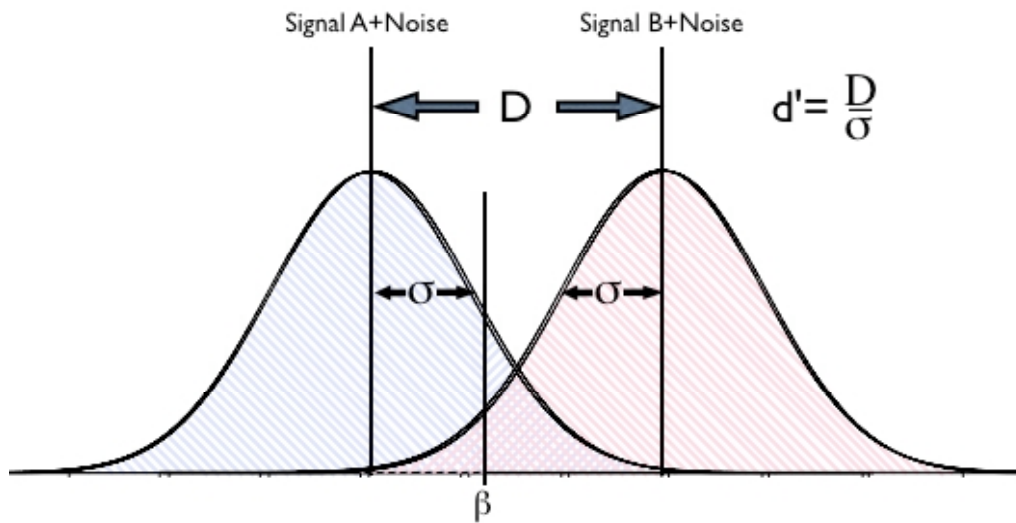
$$M_t = M_{t-1} + c + r + (c \otimes r)$$

Generate_Vectors(Cue)
Generate_Vectors(Response)

for i = 1..N **do**
 Memory = Memory + Cue[i,:] + Response[i,:] + Convolve(Cue[i:], Response[i:])
end

Now our memory contains all the item and associative information for the words learned....we can determine if a word was learned (recognition memory):

Generate_Vectors(New) ! Creating new items not learned
for i = 1..N **do**
 Cosine(Cue[i:], Memory)
 Cosine(New[i:], Memory)
end



Then, we can give a cue and ask subjects to produce a response (retrieval):

Correlate(Cue[1,:], Memory) will retrieve a facsimile of Response[1,:]

Or, we can do a more complicated task, and present subjects with a list of words and have them learn the words in order to perform a later serial recall task:

$$M_t = \alpha M_{t-1} + \gamma w_t + \omega(w_t \otimes w_{t-1})$$

Generate_Vectors(Item) ! Producing N new items to learn in order

for i = 1..N do

Memory = (Alpha*Memory) + (Gamma*Item[i,:]) + (Omega*Convolve(Item[I,:], Item[i-1]))

end

and then we cue with Item 1, and see what comes out. Even if what comes back cannot be interpreted (although we hope it matches item 2), it can still be a useful cue for what is in position 3...this is how TODAM bypasses the “broken chain” problem....retrieving something ambiguous at time T doesn’t mean that it’s not useful as a probe to retrieve something interpretable at T+1

Cue = Item[1,:]

for i = 2..N do

Retrieved = Correlate(Cue, Memory)

{insert function to determine which candidate item the retrieved item best matches}

Cue = Retrieved ! Whether it’s useful or not, use retrieved vector as cue for next

end