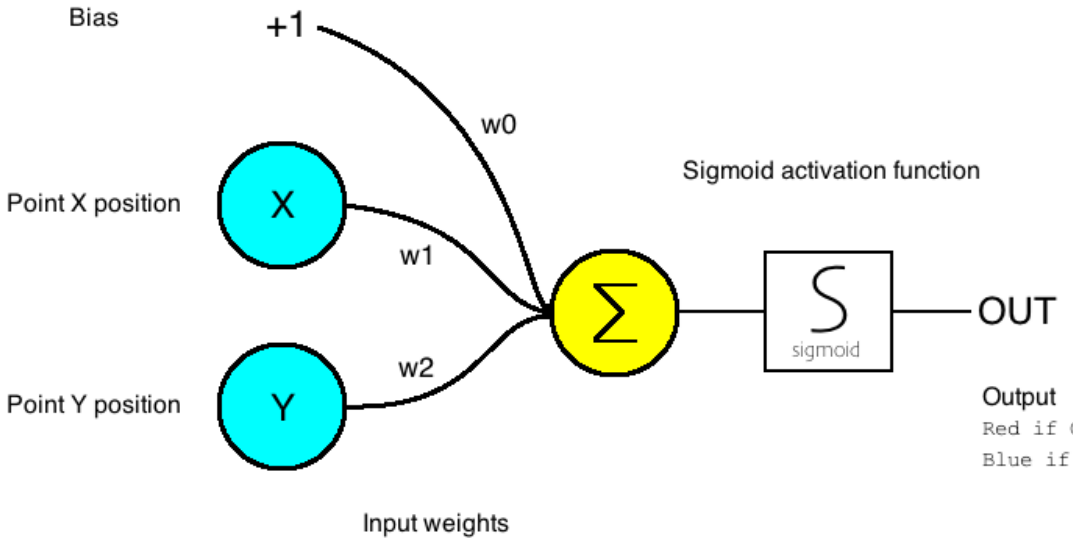
The Potential of Memory Augmented Neural Networks

Dalton Caron
Montana Technological University
November 15, 2019

Overview

- Review of Perceptron and Feed Forward Networks
- Recurrent Neural Networks
- Neural Turing Machines
- Differentiable Neural Computer

Basic Perceptron Review



$$z = \sum_{i=0}^{N-1} x_i w_i + b$$

$$y = f(z)$$

Gradient Descent on the Sigmoid Perceptron

- Goal: Compute error gradient with respect to weights
- Logit and Activation functions

$$z = \sum_k x_k w_k$$

$$y = \frac{1}{1 + e^{-z}}$$

Gradient Descent on the Sigmoid Perceptron

$$\begin{aligned}\frac{\partial y}{\partial z} &= \frac{e^{-z}}{(1+e^{-z})^2} \\ &= \frac{1}{1+e^{-z}}\frac{e^{-z}}{1+e^{-z}} \\ &= \frac{1}{1+e^{-z}}(1-\frac{1}{1+e^{-z}}) \\ &= y(1-y)\end{aligned}$$

$$\frac{\partial z}{\partial x_k} = x_k$$

$$\frac{\partial z}{\partial w_k} = w_k$$

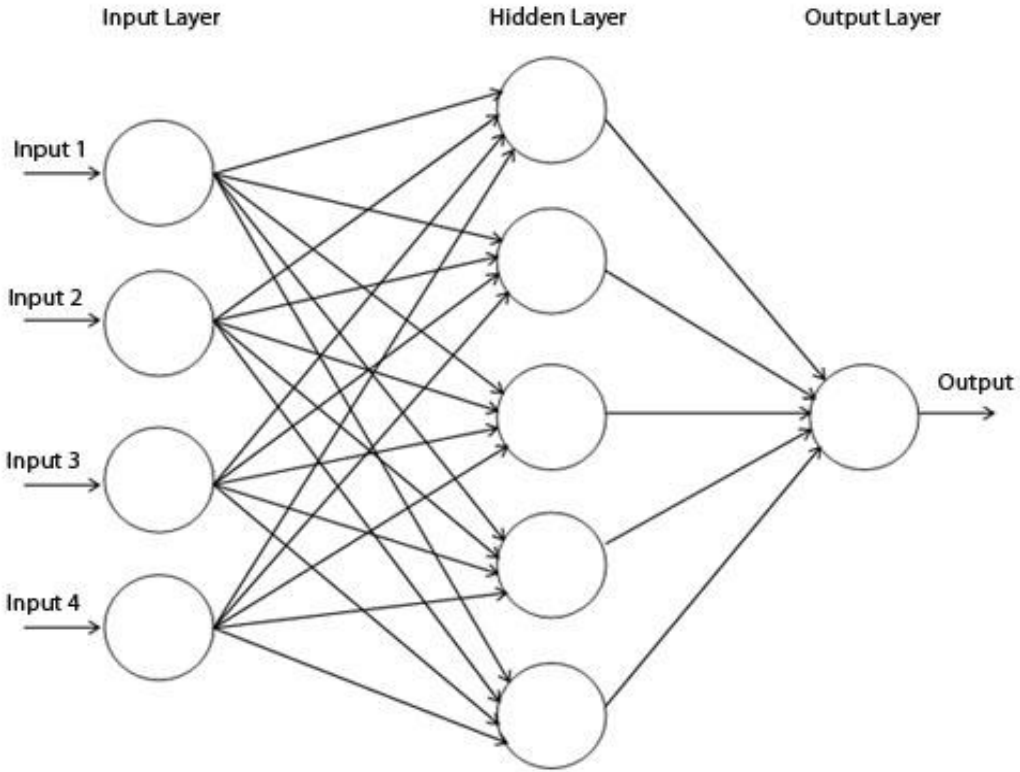
Gradient Descent on the Sigmoid Perceptron

$$\frac{\partial y}{\partial w_k} = \frac{dy}{dz}\frac{\partial z}{\partial w_k} = x_k y(1-y)$$

$$\frac{\partial E}{\partial w_k} = \sum_i \frac{\partial E}{\partial y^{(i)}}\frac{\partial y^{(i)}}{\partial w_k} = -\sum_i x_k^{(i)} y^{(i)}(1-y^{(i)})(t^{(i)}-y^{(i)})$$

$$\Delta w_k = \sum_i \alpha x^{(i)} y^{(i)}(1-y^{(i)})(t^{(i)}-y^{(i)})$$

Backpropagation



- Induction problem?

Backpropagation Derivation

- Base case

$$E = \frac{1}{2} \sum_{j \in output} (t_j - y_j)^2 \rightarrow \frac{\partial E}{\partial y_j} = -(t_j - y_j)$$

- Now we must calculate error for the previous layers

$$\frac{\partial E}{\partial y_i} = \sum_j w_{ij} y_j (1 - y_j) \frac{\partial E}{\partial y_j}$$

Full derivation in appendix.

Backpropagation Algorithm

- The change in weights

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial z_j}{\partial w_{ij}} \frac{\partial E}{z_j} = y_i y_j (1 - y_j) \frac{\partial E}{\partial y_j}$$

- Summed across the entire model

$$\Delta w_{ij} = - \sum_{k \in dataset} \alpha y_i^{(k)} y_j^{(k)} (1 - y_j^{(k)}) \frac{\partial E^{(k)}}{\partial y_j^{(k)}}$$

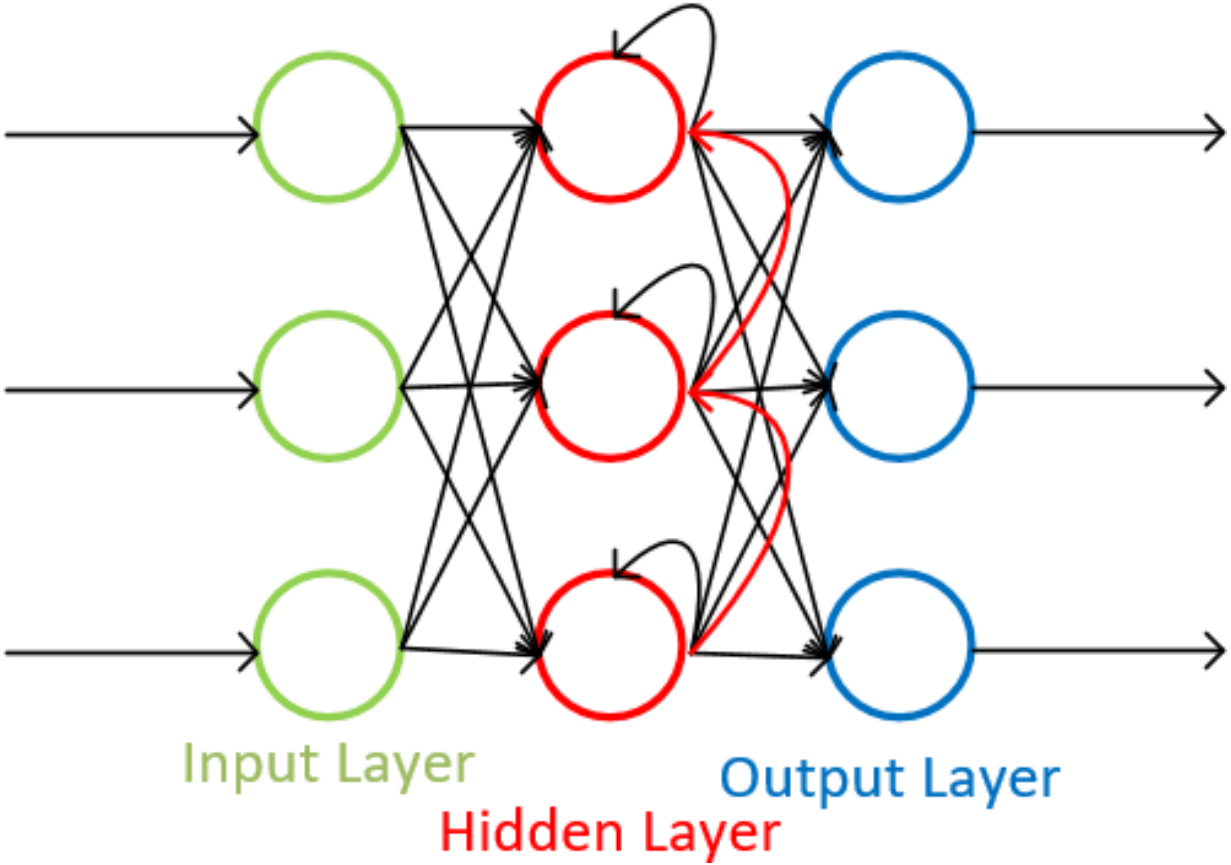
Note on Optimizers

- Improvements to the neural network will be made by modifying the network architecture rather than the optimizer
- Further discussion on optimizers is outside the scope of the presentation

Problems with Feed Forward Networks

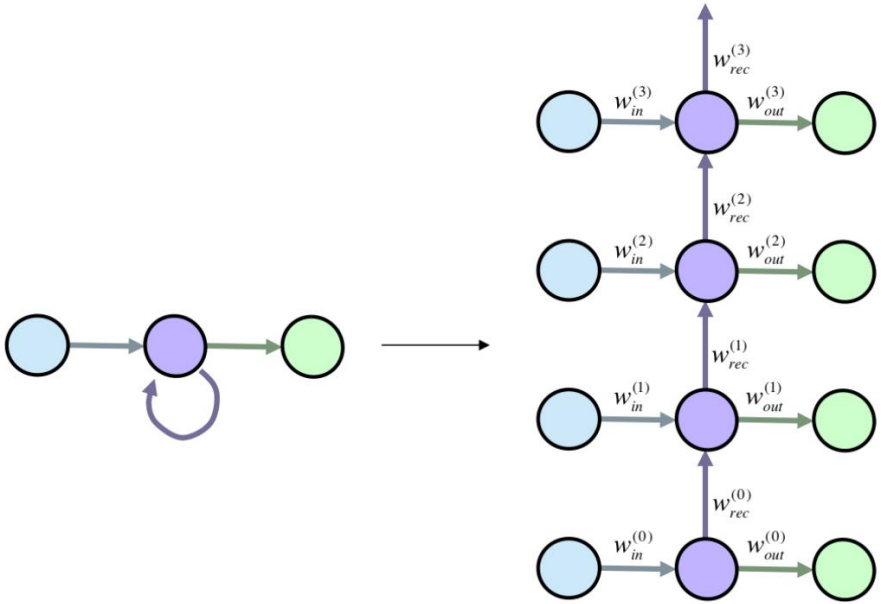
- Trouble with sequences of inputs
- No sense of state
- Unable to relate past input to present input

Recurrent Neural Network



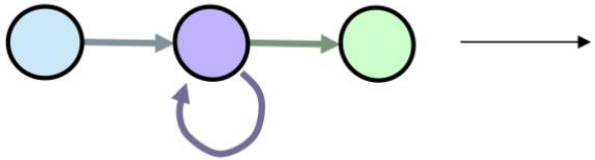
Training a RNN with Backpropagation

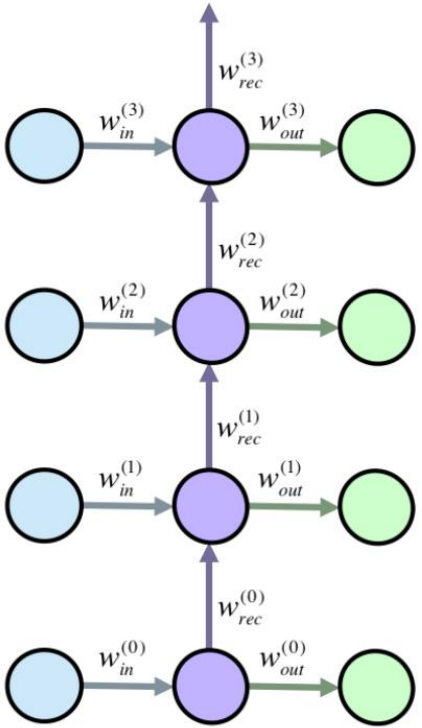
- Is there system differentiable?
- Yes. If unrolled over time steps



Vanishing and Exploding Gradients

$$|w_{rec}|^k \cdot w_{in}$$





$$h^{(t)} = f(w_{in}^{(t)} i^{(t)} + w_{rec}^{(t-1)} h^{t-1})$$

$$\frac{\partial h^{(t)}}{\partial i^{(t-k)}} = f'(w_{in}^{(t)} i^{(t)} + w^{(t-1)} h^{(t-1)}) \frac{\partial}{\partial i^{(t-k)}} (w_{in}^{(t)} i^{(t)} + w_{rec}^{(t-1)} h^{(t-1)})$$

$$\frac{\partial h^{(t)}}{\partial i^{(t-k)}} = f'(w_{in}^{(t)} i^{(t)} + w^{(t-1)} h^{(t-1)}) \frac{\partial}{\partial i^{(t-k)}} w_{rec}^{(t-1)} \frac{\partial h^{(t-1)}}{\partial i^{(t-k)}}$$

$$\left|\frac{\partial h^{(t)}}{\partial i^{(t-k)}}\right| \leq |w_{rec}^{(t-1)}| \cdot \left|\frac{\partial h^{(t-1)}}{\partial i^{(t-k)}}\right|$$

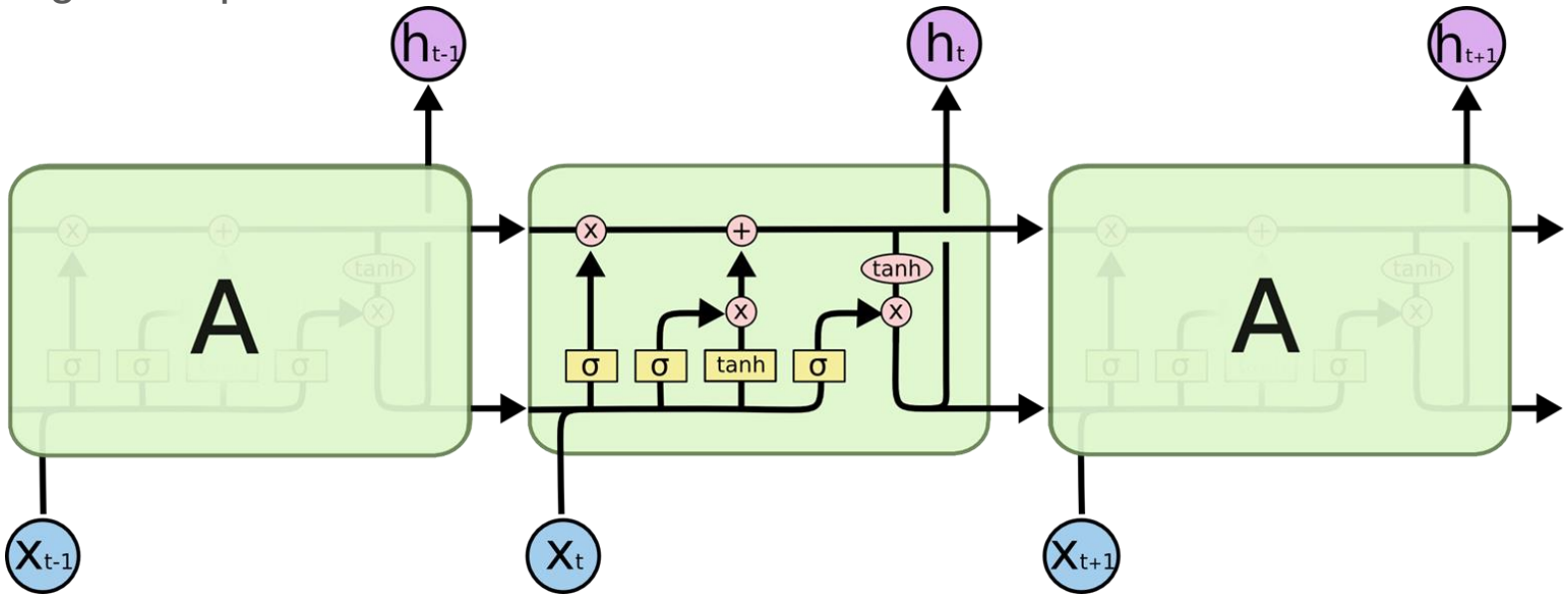
$$\left|\frac{\partial h^{(t)}}{\partial i^{(t-k)}}\right| \leq |w_{rec}^{(t-1)}| \cdot \ldots \cdot |w_{rec}^{(t-k)}| \cdot \left|\frac{\partial h^{(t-1)}}{\partial i^{(t-k)}}\right|$$

$$\left|\frac{\partial h^{(t)}}{\partial i^{(t-k)}}\right| \leq |w_{rec}^{(t-1)}| \cdot \ldots \cdot |w_{rec}^{(t-k)}| \cdot |w_{in}^{(t-k)}| = |w_{rec}|^k \cdot w_{in}$$

Vanishing Gradient Equation Derivation

Long Short-Term Memory Networks

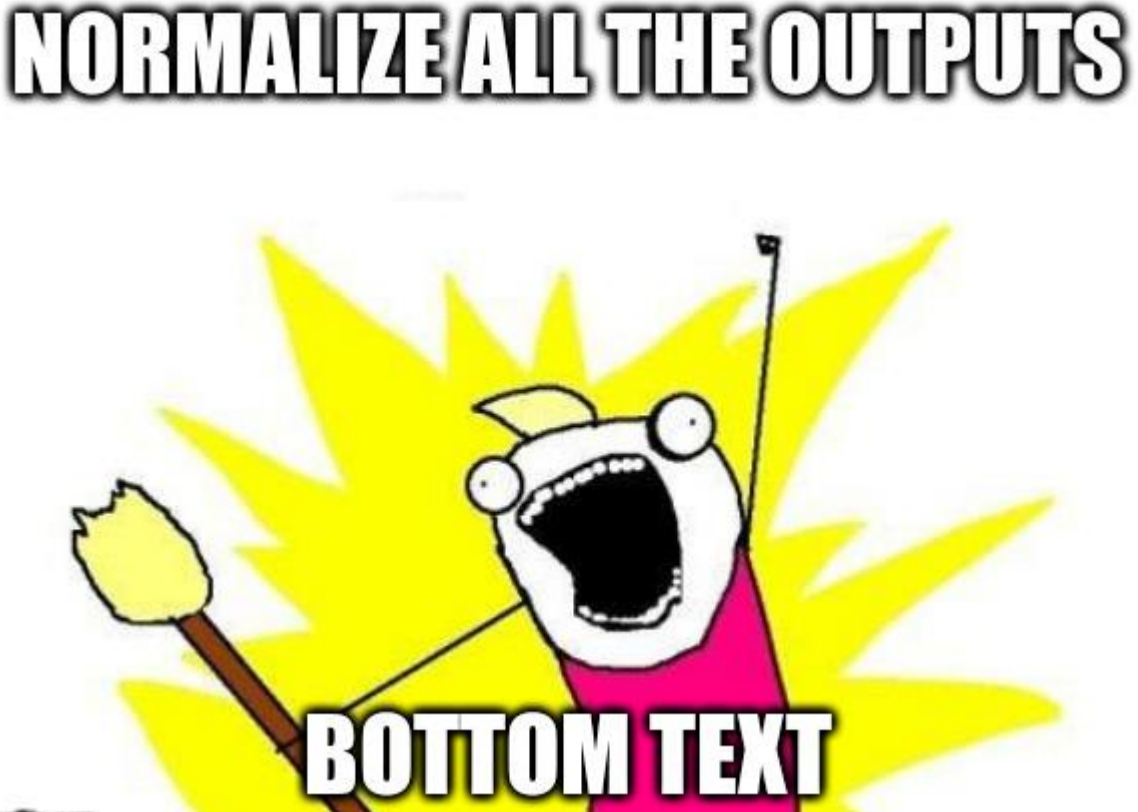
- How much information flows into the next state is regulated
- Sigmoid operations reduce information of w_{rec}



Decline of RNNs

- Past applications: Siri, Cortana, Alexa, etc.
- Intensive to train due to network unrolling
- Being replaced by attention based networks

Recall: Softmax Layer



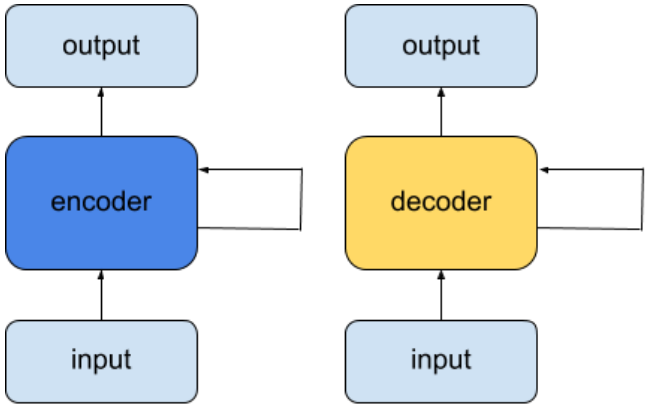
$$softmax(z) = \frac{e^z}{\sum_j e^{z_j}}$$

What is Attention?

- Focus on sections of input
- Usually in form of probability distribution

A Practical Example

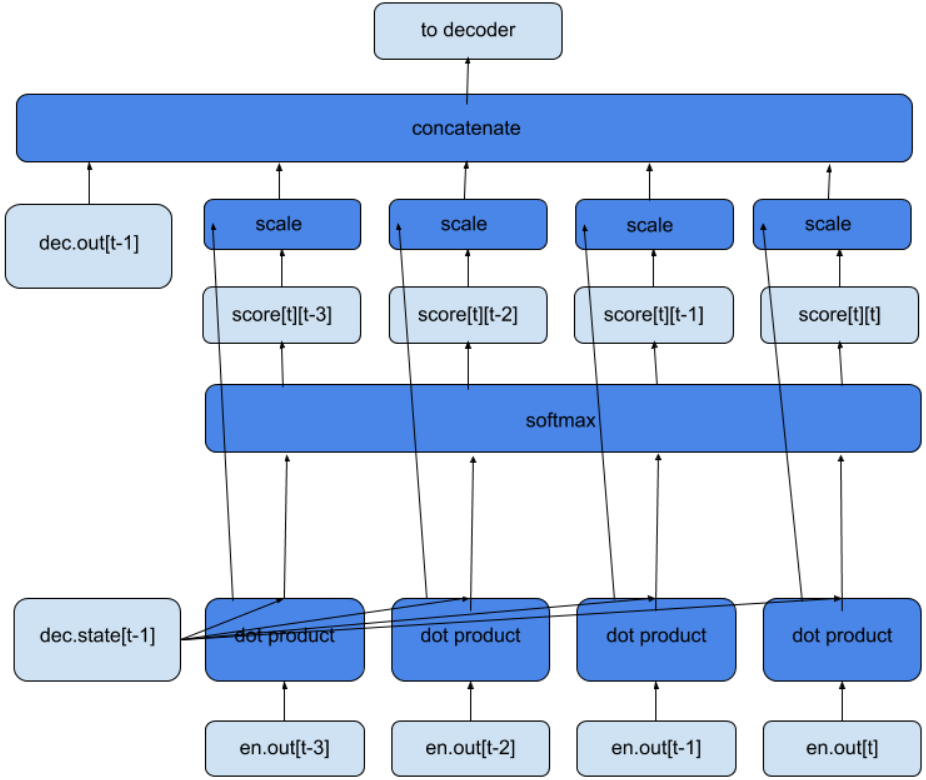
- Language translator network

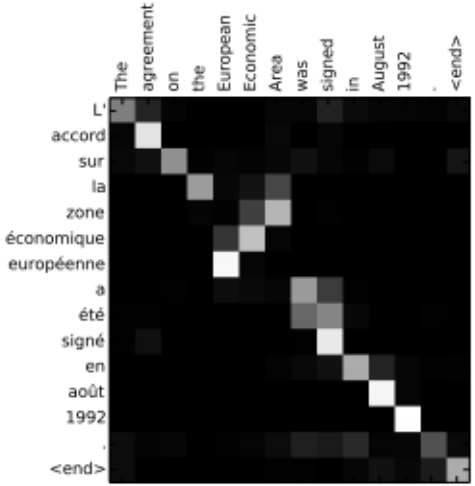


Problems and Solutions

- Human sentence inference
- Decoder only has access to state t-1 and t
- Decoder should see entire sentence
- But attention should only be given to input words

An Attention Augmented Model

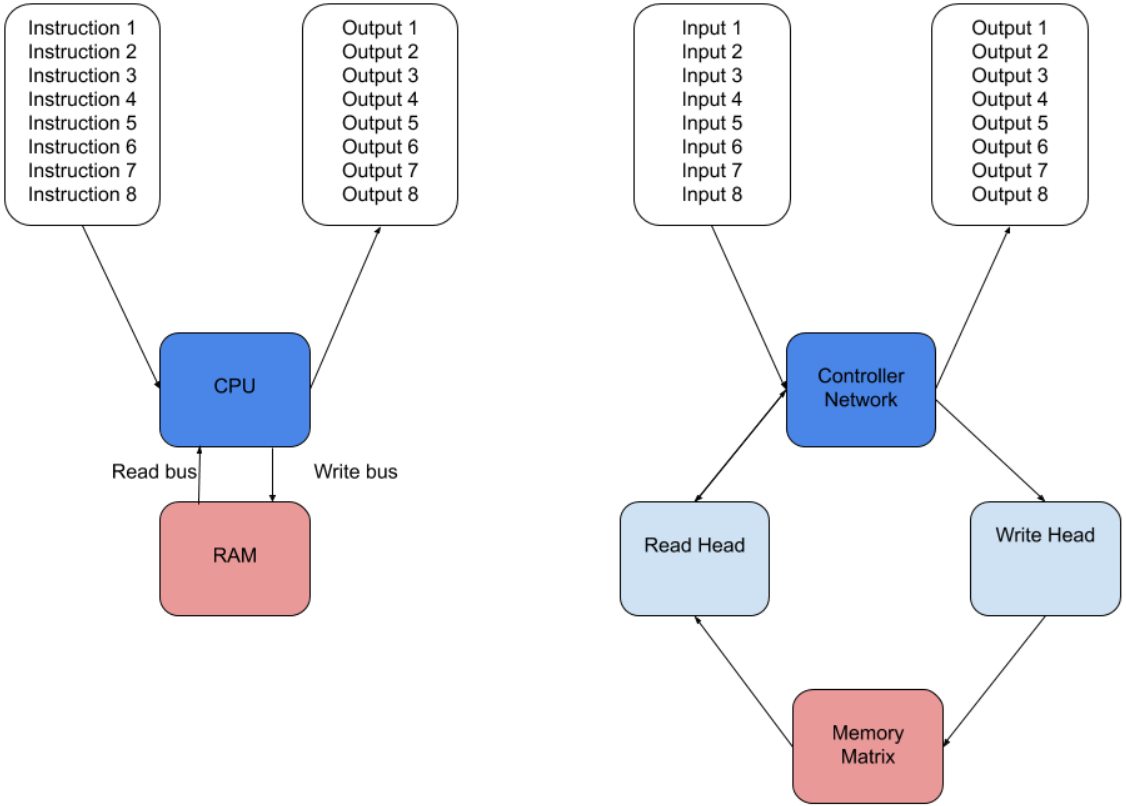




The Case for External Memory

- In order to solve problems, networks remember
  - Weight matrices
  - Recurrent state information
- A general problem solver requires a general memory

The Neural Turing Machine



Why is the NTM Trainable?

- The NTM is fully differentiable
- Memory is accessed continuously (attention)
- Each operation is differentiable

Normalization Condition

$$\sum_i w_t(i) = 1, 0 \leq w_t(i) \leq 1, \forall i$$

NTM Reading Memory

- Weight vector emitted by the read head.

$$r_t \leftarrow \sum_i w_t(i) M_t(i)$$

NTM Writing Memory

- Split into two operations: erase and add
- Add and erase vectors emitted from write head

$$\widetilde{M}_t = M_{t-1}(i) \odot [1 - w_t(i)e_t]$$

$$M_t \leftarrow \widetilde{M}_t(i) + w_t(i)a_t$$

NTM Addressing Mechanisms

- Read and write operations are defined
- Emissions from controller need to be defined
- NTM uses two kinds of memory addressing

Content-Based Addressing

- Let k_t be a key vector from the controller
- Let $K[\cdot, \cdot]$ be a similarity function
- Let β_t be a parameter that attenuates the focus

$$w_t^c(i) \leftarrow \frac{exp(\beta_t K[k_t, M_t(i)])}{\sum_j exp(\beta_t K[k_t, M_t(i)])}$$

Location-Based Addressing

- Focuses on shifting the current memory location
- Does so by rotational shift weighting
- Current memory location must be known

Location-Based Addressing

- Let w_{t-1} be the access weighting from the last time step
- Let g_t be the interpolation gate from the controller which contains values from (0,1)
- Let w_t^c be the content-based address weighting
- The gate weighting equation is given as follows

$$w_t^g \leftarrow g_t w_t^c + (1 - g_t) w_{t-1}$$

Location-Based Addressing

- Let s_t be a normalized probability distribution over all possible shifts
- For example, let all possible shifts be [-1, 0, 1], could be expressed as a probability distribution [0.33, 0.66, 0] s_t
- It is usually implemented as a softmax layer in the controller

Location-Based Addressing

- The rotational shift applied to the gate weighting vector can now be given as a convolution operation

$$\tilde{w}_t(i) \leftarrow \sum_{j=0}^{N-1} w_t^g(j) s_t((i-j)\%N)$$

Location-Based Addressing

- Sharpening operation performed to make probabilities more extreme
- Let γ_t be a value emitted from a head where $\gamma_t \geq 1$
- The sharpened weighting is giving by the following equation

$$w_t(i) \leftarrow \frac{\tilde{w}_t(i)^{\gamma_t}}{\sum_j w_t(j)^{\gamma_t}}$$

Closing Discussion on NTM Addressing

- Given two addressing modes, three methods appear:
  - Content-based without memory matrix modification
  - Shifting for different addresses
  - Rotations allow for traversal of the memory
- All addressing mechanism are differentiable

NTM Controller

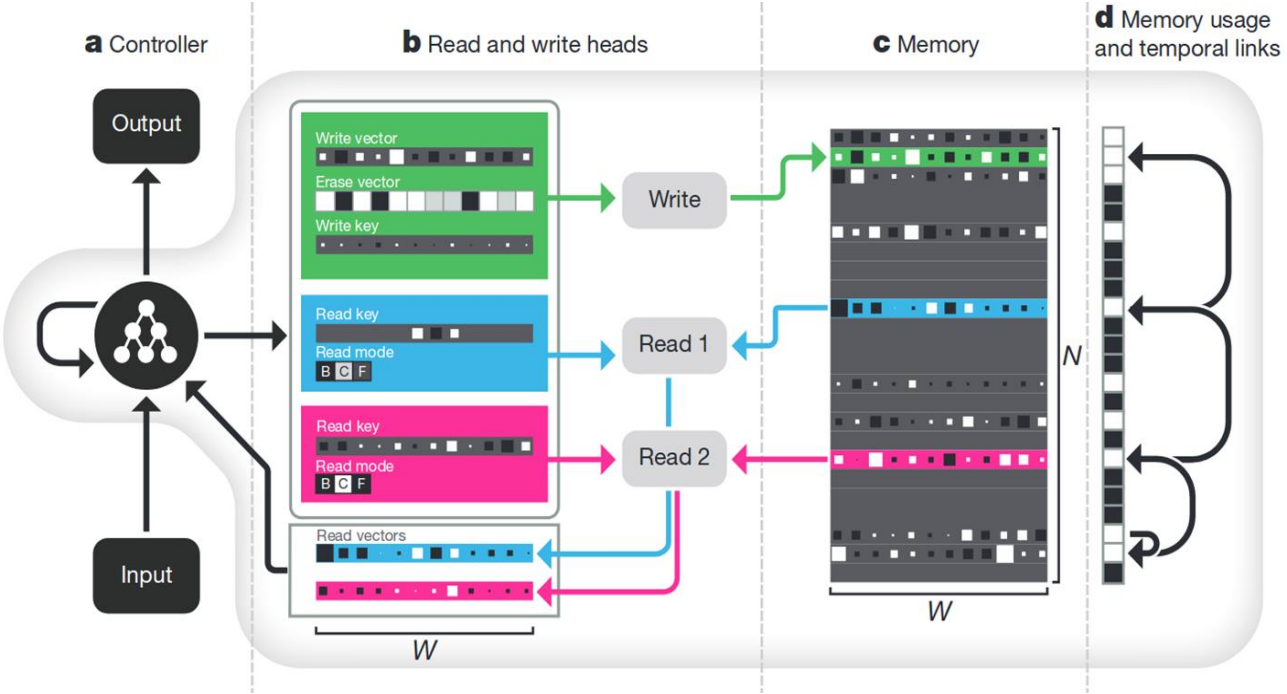
- Many parameters, such as size of memory and number of read write heads
- Independent neural network feeds on problem input and NTM read heads
- Long short-term memory network usually used for controller

NTM Limitations

- No mechanism preventing memory overwriting
- No way to reuse memory locations
- Cannot remember if memory chunks are contiguous

The Differentiable Neural Computer

- Developed to compensate for the NTMs issues



NTM Similarities and Notation Changes

- DNC has R weightings for read heads

$$w_t^r(0), w_t^r(1), ...w_t^r(R-1)$$

- Write operations are given as

$$M_t(i) = M_{t-1}(i) \odot [1 - w_t(i)e_t] + w_t^w(i)v_t$$

- Read operations are given as

$$r_t \leftarrow \sum_{i=0}^{R-1} M_t(i,\cdot)w_t^r(i)$$

Usage Vectors and the Free List

- Let u_t be a vector of size N that contains values in the interval [0,1] that represents how much the corresponding memory address is used at time t
- u_t Is initialize to all zeroes and is updated over time
- What memory is not being used?

Allocation Weighting

- Let ζ_t be a usage vector sorted in descending order
- The allocation weighting is then given as the following equation

$$a_t(\zeta_t(j)) = (1 - u_t(\zeta_t(j))) \prod_{i=0}^{j-2} u_t(\zeta_t(i))$$

Write Weighting

- Let g_t^w be defined as the write gate taking a value on the interval (0,1), emitted from the interface vector
- Let g_t^a be defined as the read gate taking a value on the interval (0,1), emitted from the interface vector
- Let w_t^c be the weighting from content-based addressing
- The final write weighting vector is given as

$$w_t^w = g_t^w [g_t^a a_t + (1 - g_t^a) w_t^c]$$

- What if $u_t = ?1$

Memory Reuse

- We must decide what memory is reused
- Let Ψ_t be defined as an N length vector that takes on values in the interval [0,1] known as the retention vector
- Let f_t^i be a value from the interface vector in the interval [0,1] known as the free gate
- Let w_{t-1}^r be a read vector weighting
- The retention vector is given as

$$\Psi_t = \prod_{i=0}^{R-1} (1 - f_t^i w_{t-1}^r(i))$$

Updating the Usage Vector

- Remember that u_t is the usage vector
- Remember that w_t^w is a write vector weighting
- The update to the usage vector is given as

$$u_t = (u_{t-1} + w_{t-1}^w - u_{t-1} \odot w_{t-1}^w) \odot \Psi_t$$

Precedence

- In order to memorize jumps in memory, the temporal link matrix is provided
- To update this matrix, the precedence vector is defined

$$p_t = (1 - \sum_{i=0}^{N-1} w_t^w(i))p_{t-1} + w_t^w$$

The Temporal Link Matrix

- Let L_t be an $N \times N$ matrix taking values on the interval [0,1] where $L(i,j)$ indicates how likely location i was written to before location j
- It is initialized to 0
- The update equation for the temporal link matrix is given as

$$L_t(i,j) = (1 - w_t^w(i) - w_t^w(j))L_{t-1}(i,j) + w_t^w(i)p_{t-1}(j)$$

DNC Read Head

- Recall function to generate w_t^c

$$w_t^c(i) \leftarrow \frac{exp(\beta_t K[k_t, M_t(i)])}{\sum_j exp(\beta_t K[k_t, M_t(i)])}$$

- Let K_t^r and β_t^r be emitted from the interface vector

DNC Read Head

- To achieve location-based addressing, a forward and backward weighting are generated

$$f_t(i) = L_t w^r_{t-1}(i)$$

$$b_t(i) = L^T_{t-1} w^r_{t-1}(i)$$

DNC Read Head

- At last, the final read weighting is given as

$$w_t^r(i) = \pi_t(i,1)b_t(i) + \pi_t(i,2)w_t^c(i) + \pi_t(i,3)f_t(i)$$

- π are known as the read modes (backward, lookup, forward) and are emitted from the interface vector

The Controller and Interface Vector

- Let η be the function computed by the controller
- Let χ_t be the controller input concatenated with the last read vectors
- Let the output of the controller be defined as $\eta(\chi_t)$
- The interface vector is a length-$R \times W + 3W + 5R + 3$ vector given by

$$\Phi_t = [k_t^r(0), ..., k_t^r(R-1), \beta_t^r(0), ..., \beta_t^r(R-1),$$
$$\beta_t^w, e_t, v_t, f_t(0), ..., f_t(R-1), g_t^a, g_t^w, \pi_t(0), ..., \pi_t(R-1)]$$

Interface Vector Transformations

- To ensure interface vector values sit within the required interval, a series of transformations are applied

$$e_t = \sigma(e_t), f_t(i) = \sigma(f_t(i)), g_t^a = \sigma(g_t^a), g_t^w = \sigma(g_t^w)$$

$$\beta_t^r(i) = oneplus(\beta_t^r(i)), \beta_t^w = oneplus(\beta_t^w)$$

$$\text{where } oneplus(z) = 1 + log(1 + e^z)$$

$$\pi_t(i) = softmax(\pi_t(i))$$

Final Controller Output

- Let W_y be a learnable weights matrix of size $|\eta| \times Y$
- Let $v_t = W_y \eta(\chi_t)$ be the pre output vector
- Let W_r be a learnable weights matrix of size $(R \times W) \times Y$
- The final controller output is given as

$$y_t = v_t + W_r[r_t(0), ..., r_t(R-1)]$$

- With this, the formal description of the DNC is complete

DNC Applications

- bAbi dataset
  - “John picks up a ball. John is at the playground. Where is the ball?”
  - DNC outperforms LSTM
- Trained on shortest path, traversal, inference labels
  - Given London subway and family tree
  - LSTM fails, DNC achieves 98.8% accuracy

A Conclusion of Sorts

- DNC outperforms NTM and LSTM
- Can there be a continuous computer architecture?
- Scalability?
- A general purpose artificial intelligence?

End

Appendix

- Complete derivation for error derivatives of layer i expressed in terms of the error derivatives of layer j

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial z_j}\frac{dz_j}{dy_i} = \sum_j w_{ij}\frac{\partial E}{\partial z_j}$$

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j}\frac{dy_j}{dz_j} = y_j(1-y_j)\frac{\partial E}{\partial y_j}$$

$$\frac{\partial E}{\partial y_i} = \sum_j w_{ij}y_j(1-y_j)\frac{\partial E}{\partial y_j}$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial z_j}{\partial w_{ij}}\frac{\partial E}{z_j} = y_i y_j(1-y_j)\frac{\partial E}{\partial y_j}$$