Descrição dos mecanismos que atuam em um *Differentiable Neural Computer* (DNC)

Arnon Bruno Ventrilho dos Santos

asantos.quantum@gmail.com

1. Differentiable Neural Computer (DNC)

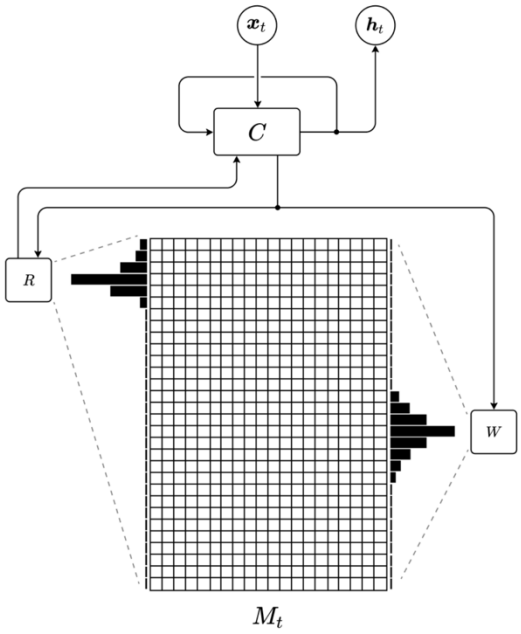

Figura 1. Representação gráfica do DNC

De acordo com [Graves et al., 2016], o controlador *C* pode ser qualquer sistema diferenciável, como uma Rede Neural Artificial (RNA), que seja capaz de calcular *N* para emitir os vetores $[v_t, \xi_t]$, onde v_t é o vetor de saída e ξ_to vetor de interface com a memória, conforme a Equação 1.

$$[v_t, \xi_t] = N(x_t, r_{t-1}^1, \dots, r_{t-1}^R) \tag{Eq.1}$$

Assim como em um computador tradicional, o DNC possui uma memória que é lida e (ou) escrita a cada instante t. Para efeitos de demonstração, suponha que tenhamos a seguinte memória M_t

$$M_t = \begin{bmatrix} -0.5 & 0.01 & 3.1 \\ 0.2 & 0.6 & 1.2 \\ 0 & 0 & 0 \\ -0.1 & -0.05 & 0 \end{bmatrix}$$

Diferentemente de um computador tradicional onde a memória é acessada com base em sua localização absoluta (endereço de memória), em um DNC a memória é acessada de forma "suave", ou seja, os endereços de memória são acessados de acordo com sua relevância w para um determinado contexto. Essa relevância não é mais um número discreto (como um índice de memória), mas uma distribuição contínua que indica o quanto o conteúdo daquela posição é importante no instante de treinamento t (esse comportamento é representado pelo histograma próximo aos cabeçotes R e W na Figura 1).

O fato de possuirmos índices contínuos viabiliza a diferenciação da memória e obtenção do gradiente em relação a algum objetivo. Como se pode obter o gradiente da memória, também é possível que o DNC armazene dados de um mesmo $t$em diferentes I (linhas de M_t), dessa forma a memória não é lida ou escrita em blocos contíguos.

O DNC realiza operações de leitura em M_ta partir do seu cabeçote de leitura R. Aqui a palavra cabeçote, além de propositalmente remeter á TM, também é uma outra forma de chamar uma função que realiza operações matriciais para determinar quais elementos de M_t precisam ser acessados e transmitidos de volta ao controlador C. Esse cabeçote de leitura realiza $M_t^T w_t^r$de forma que w_t^r representa a relevância de uma determinada posição no instante t para a operação de leitura. Para manter a brevidade, o valor w será definido mais a frente.

Para efeitos ilustrativos, suponha que na M_t descrita acima o controlador C indica que I_2 precisa ser lida integralmente, dessa forma a operação de leitura $M_t^T w_t^r$ nesta posição é realizada da seguinte forma:

$$\begin{bmatrix} -0.5 & 0.01 & 3.1 \\ 0.2 & 0.6 & 1.2 \\ 0 & 0 & 0 \\ -0.1 & -0.05 & 0 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5 & 0.2 & 0 & -0.1 \\ 0.01 & 0.6 & 0 & -0.05 \\ 3.1 & 1.2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.6 \\ 1.2 \end{bmatrix}$$

Neste exemplo, interpretamos que durante o treinamento C deseja ler integralmente o conteúdo do segundo espaço de memória (I_2), e não deseja ler os demais. No entanto, torna-se igualmente possível que por conta do processo de obtenção do gradiente da leitura da memória em relação ao objetivo, o controlador decida por ler a memória de maneira "suave", dividindo a relevância das posições da memória, conforme abaixo:

$$\begin{bmatrix} -0.5 & 0.01 & 3.1 \\ 0.2 & 0.6 & 1.2 \\ 0 & 0 & 0 \\ -0.1 & -0.05 & 0 \end{bmatrix}^{\top} \begin{bmatrix} 0 \\ 0.8 \\ 0.1 \\ 0.1 \end{bmatrix} = \begin{bmatrix} -0.5 & 0.2 & 0 & -0.1 \\ 0.01 & 0.6 & 0 & -0.05 \\ 3.1 & 1.2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0.8 \\ 0.1 \\ 0.1 \end{bmatrix} = \begin{bmatrix} 0.15 \\ 0.475 \\ 0.96 \end{bmatrix}$$

Notemos que o vetor resultante é muito parecido com o valor original em I_2, no entanto esse valor foi influenciado pelos demais I, conforme designado por C em w_t^r. Ou seja, durante o treinamento C optou por ler dados de M_t de maneira suave. Essa é uma maneira completamente diferente da que estamos habituados a pensar, onde endereços de memória são lidos conforme seus índices. Aqui, os endereços são lidos com intensidades diferentes de acordo com sua relevância w. Dessa forma [Graves et al. 2016] descrevem a operação de leitura contida em ξ_t como:

$$r_t = \sum_{i}^{n} w_i^r(i) M_t(i)$$

(Eq.2)

Outra operação realizada por C em M_té a operação de escrita a partir da interface ξ_t com o cabeçote W. Assim como em r_t, a operação de escrita também recebe o vetor w de C, que indica a relevância de uma posição ou conteúdo na tarefa de escrita. Esse vetor é descrito no formato w_t^w. Além desse vetor o cabeçote de escrita também recebe um vetor com o conteúdo que precisa ser escrito na memória no formato $v^t \in R^I$, e um vetor que orienta apagar posições na memória, no formato $e_t \in [0,1]^I$(lembrando que I representa linhas em M_t). Dessa forma, a equação que determina o novo conteúdo em $M_t$após a escrita de v^t é designada em termos do produto Hadamard do conteúdo da memória no instante anterior com os vetores previamente descritos:

$$M_t = M_{t-1} \circ (E - w_t^w e_t^T) + w_t^w v_t^T$$

(Eq.3)

Para ilustrar a operação de escrita na memória M_t, conforme enunciado na Equação 3, suponha os seguintes valores para a operação de escrita:

$$w_t^w = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad v_t = \begin{bmatrix} -1.5 \\ -1.3 \\ -1.1 \end{bmatrix} \qquad e_t = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Portanto, substituindo os valores na Equação 3, temos:

$$= \begin{bmatrix} -0.5 & 0.01 & 3.1 \\ 0.2 & 0.6 & 1.2 \\ 0 & 0 & 0 \\ -0.1 & -0.05 & 0 \end{bmatrix} \circ \left(\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \right)$$

$$+ \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} -1.5 & -1.3 & -1.1 \end{bmatrix}$$

$$= \begin{bmatrix} -0.5 & 0.01 & 3.1 \\ 0.2 & 0.6 & 1.2 \\ 0 & 0 & 0 \\ -0.1 & -0.05 & 0 \end{bmatrix} \circ \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ -1.5 & -1.3 & -1.1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -0.5 & 0.01 & 3.1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -0.1 & -0.05 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ -1.5 & -1.3 & -1.1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -0.5 & 0.01 & 3.1 \\ -1.5 & -1.3 & -1.1 \\ 0 & 0 & 0 \\ -0.1 & -0.05 & 0 \end{bmatrix}$$

Temos então que o vetor $I_2^{M_{t-1}}$ foi sobrescrito pelo valor contido em v^t. Aqui, novamente, *C* poderia ter optado por fazer uma gravação "suave" alterando w_t^w, mantendo partes do conteúdo anterior ou escrever trechos em diferentes *I*. Novamente, essa decisão cabe a *C* durante o treinamento e é incentivada pelo cálculo do gradiente desta função em relação a função custo, ou seja, *C* vai optar pela estratégia que melhor minimize o erro.

A partir dessas definições cabe descrever como o valor *w* é obtido. [Graves et al. 2016] nomeia esse valor como um "mecanismo de atenção" que pode ser dividido em **a)** atenção baseada em similaridade de conteúdo, **b)** atenção de conexão temporal e **c)** alocação dinâmica de memória. O mecanismo **(a)** é combinado com o mecanismo **(c)** em *W*, ou com **(b)** em *R*.

De forma resumida, podemos enunciar o mecanismo **(a)** como uma estratégia que compara uma chave (vetor) *k* emitida por *C*, utilizando a distância de cosseno *D* entre essa chave e um determinado conteúdo em M_t , de forma que o conteúdo com maior similaridade será utilizado por *R* para leitura ou por *W* para gravação. Dessa forma:

$$\boldsymbol{C(M, k, \beta)[i]} = \frac{\boldsymbol{exp(D(k, M[i, \cdot]))^{\beta}}}{\sum_{j=1}^{N} \boldsymbol{exp(D(k, M[j, \cdot]))^{\beta}}}$$

(Eq.4)

Onde o expoente βe exp funcionam como fatores de maximização entre as distâncias dos valores contidos em *k.* Por fim, podemos definir os valores de $w_t^{r,i}$ e w_t^wpara **(a)** como:

$$w_t^{r,i} = \boldsymbol{C}\left(\boldsymbol{M}_t, k_t^{r,i}, \beta_t^{r,i}\right) \quad i = 1, \ldots, \boldsymbol{R}$$
$$w_t^w = \boldsymbol{C}\left(\boldsymbol{M}_t, k_t^w, \beta_t^w\right)$$

(Eq.5)

O mecanismo **(b)** pode ser descrito em termos de uma matriz que funciona como "histórico" de como as posições em M_t foram escritas. Suponha que *C* precise gravar nessa "matriz histórico" que a posição 4 foi escrita depois da posição 2. Nesse sentido a matriz histórico seria representada da seguinte forma:

$$\boldsymbol{L}_t = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

O controlador poderia se movimentar ao longo desse histórico para frente, emitindo um vetor $f_t = L_t w_t$, e para trás emitindo um vetor $b_t = L_t^T w_t$(de maneira similar a fita numa TM). Por meio da interpolação desses valores com o valor obtido em **(a)**, o controlador pode resolver w_t^w. Essa interpolação é descrita por [Graves et al., 2016] como:

$$w_t^{r,i} = \pi_t^i[1] b_t^i + \pi_t^i[2] c_t^{r,i} + \pi_t^i[3] f_t^i$$

(Eq.6)

Por fim, o mecanismo **(c)** funciona como orientação de *C* para o próximo instante t, indicando o quanto um dado *I* em M_t pode ser alocável. Em outras palavras, esse mecanismo indica o quão fortemente *C* deseja manter aquela informação para as próximas etapas e é definido pelo vetor a_t. Supondo $a_t = [0,1,0,0]$, indica que a segunda posição é completamente alocável enquanto as demais devem permanecer imutáveis durante aquele t. A decisão de qual mecanismo será utilizado para gravação é dada a partir de:

$$w_t^w = g_t^w \left[g_t^a \, a_t + \left(1 - g_t^a\right) c_t^w\right]$$

(Eq.7)

onde a_té o mecanismo **(c)**, c_t^wé o mecanismo **(a)** e gé um valor binário emitido por *C*. Se $g_t^w = 0$ nenhuma escrita será feita naquele instante, se $g_t^w = 1$e $g_t^a = 1$, o controlador *C*

escreverá exclusivamente baseado em **(c)** e se $g_t^w = 1$e $g_t^a = 0$ apenas o mecanismo de escrita por similaridade será utilizado.

A existência desses mecanismos conforme disposto na Equação 1, permite que todas as componentes em *N* sejam diferenciáveis. Portanto, a obtenção de N', em tese, viabilizaria um sistema de aprendizado de máquina capaz de executar instruções contidas em memória.

Referências

Graves, Alex, Greg Wayne, and Ivo Danihelka. "Neural turing machines." arXiv preprint arXiv:1410.5401 (2014).

Graves, Alex, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo et al. "Hybrid computing using a neural network with dynamic external memory." Nature 538, no. 7626 (2016): 471-476.