Redes Recorrentes

I) Qual é a principal vantagem das redes neurais recorrentes (RNNs) em relação a MLPs e redes convolucionais?

II) Descreva a estrutura de uma RNN básica.

III) Em frameworks de desenvolvimento de redes neurais (como o PyTorch) é comum que o bloco de rede recorrente conte apenas com a camada densa que computa o estado oculto (*hidden state*) e que não conte com a camada densa que computa a saída. Qual motivo disso?

IV) Uma rede recorrente tem um limite teórico para o tamanho de sequências que consegue processar? E do ponto de vista prático, existem limites?

V) Redes recorrentes tendem a sofrer mais dos problemas do gradiente que se dissipa e do gradiente explosivo. Por que isso acontece?

VI) *(Adaptado de CS224N Stanford 2017)* Considere que todos os parâmetros da rede recorrente ilustrada na figura abaixo são escalares ($\theta_{hh}, \theta_{ih}, b_h, \theta_{ho}$ e b_o). A função de ativação $f[.]$ é a função sinal definida abaixo:

$$f(z) = \begin{cases} z \geq 0, & 1, \\ z < 0, & 0 \end{cases}$$

Sua tarefa é descobrir valores para esses parâmetros que façam a rede recorrente ter como saída inicial o valor 0 e, assim que ela receba um valor 1 como entrada ela passe a ter como saída o valor 1 para toda e qualquer entrada futura. Por exemplo: Se a entrada for 00110 a saída deve ser 00111.

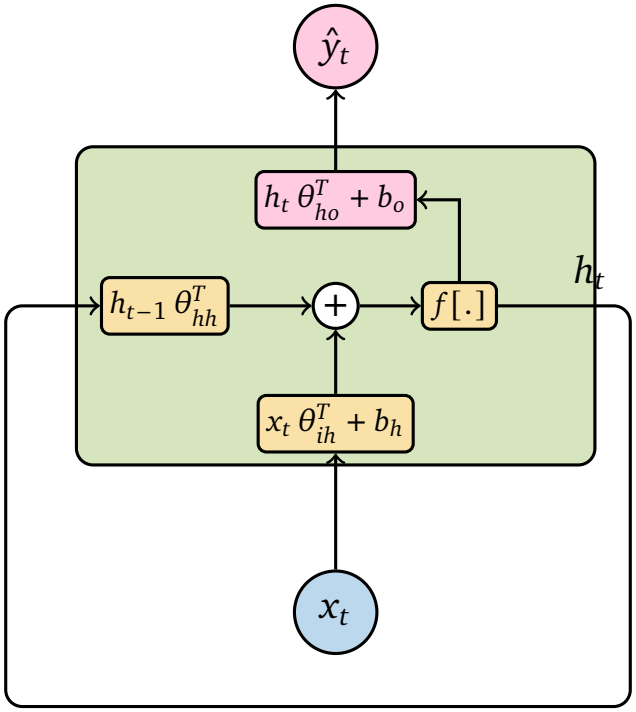

Figura 1: RNN para questão VI

VII) Considere uma rede híbrida que faz *Image Captioning* (gera legendas para imagens). A arquitetura é composta por um *encoder* (uma CNN pré-treinada, como a VGG-16) e um *decoder* (uma RNN simples). O encoder recebe como input uma imagem I e sua saída serve de entrada para o decoder, que então é responsável pela geração de legenda, uma palavra de cada vez $(y_1, y_2, ..., y_n...)$. Com base neste cenário, responda:

a) Caso utilizemos uma VGG-16, qual é o papel dela nesta tarefa? E qual é o papel da RNN? Por que não podemos usar apenas a VGG-16 para a tarefa inteira?

b) Este processo de geração de palavras é autorregressivo. Descreva o que serve como input (x_t) e o que serve como estado oculto anterior (h_{t-1}) para a RNN nos seguintes passos de tempo:

i.) Para gerar a primeira palavra (y_1) no tempo $t = 1$.

ii.) Para gerar a segunda palavra (y_2) no tempo $t = 2$.

c) A CNN (Encoder) processa a imagem uma única vez para gerar um vetor de contexto v. Qual é a origem e o formato (shape) mais comum deste vetor v? Ele é a saída da última camada convolucional (um tensor 3D), ou é a saída que foi "achatada"(flattened), ou é a saída já processada de uma camada totalmente conectada (FC)?

d) Digamos que não queremos perder a informação espacial de cada frame ao passá-la para a RNN. Ou seja, queremos que o *input* x_t e o *estado oculto* h_t sejam eles mesmos mapas de características 3D (ex: [Altura, Largura, Canais]), e não vetores 1D. É possível utilizar operações de convolução *dentro* da célula da RNN (para calcular h_t a partir de h_{t-1} e x_t) em vez das multiplicações de matrizes (camadas densas) tradicionais? Explique brevemente o conceito.

VIII) Considere uma vanilla RNN escalar com pesos $\theta_{hh} = 0.3$, $\theta_{ih} = 0.8$, $\theta_{ho} = 1.0$, *bias* $b_h = b_o = 0$, ativação tanh e estado inicial $h_0 = 0$. A célula é definida por

$$h_t = \tanh(x_t\,\theta_{ih} + h_{t-1}\,\theta_{hh}), \qquad \hat{y}_t = h_t\,\theta_{ho}.$$

A rede recebe a sequência $x = [1.0, -0.5]$ e deve prever, ao final do segundo passo, o alvo $y = 1.0$. A *loss* é o MSE, $J = \frac{1}{2}(\hat{y}_2 - y)^2$.

a) Compute o *forward pass*: z_1, h_1, z_2, h_2, $\hat{y}_2$ e J.

b) Compute $\partial J/\partial\theta_{ho}$.

c) Compute $\partial J/\partial\theta_{hh}$ via *backpropagation through time*, mostrando os gradientes intermediários $\partial J/\partial h_2$, $\partial J/\partial z_2$, $\partial J/\partial h_1$ e $\partial J/\partial z_1$.

IX) **Perplexidade**: um *language model* atribui, a uma sequência de $m = 4$ *tokens*, as probabilidades condicionais

$$p(y_t \mid y_{<t}) = [\,0.25,\ 0.5,\ 0.125,\ 0.5\,].$$

Calcule a perplexidade do modelo nessa sequência (usando a definição Perplexidade $= 2^{-L/m}$ com $L = \sum_t \log_2 p(y_t \mid y_{<t})$) e interprete o resultado em termos do "número médio efetivo de *tokens*" entre os quais o modelo está em dúvida.

Gabarito

I) As RNNs podem capturar e analisar informações sequenciais, tornando-as adequadas para tarefas que envolvem séries temporais, processamento de linguagem natural e dados sequenciais. Elas atingem esse feito usando uma recorrência, ou seja, um estado oculto da rede é computado e serve como entrada para o processamento do próximo item da sequência.

II) Uma RNN consiste em uma camada de entrada, uma camada oculta com conexões recorrentes e uma camada de saída. A camada oculta mantém um estado oculto que é atualizado a cada passo temporal, e é usado como entrada para os passos subsequentes, permitindo que a rede memorize informações de entradas anteriores.

III) As tarefas que envolvem dados sequenciais são muito variadas. Podemos fazer análise de sentimentos (classificar uma sequência), *image captioning* no qual temos uma entrada e uma sequência como saída ou ainda sequence-to-sequence onde temos sequências na entrada e na saída. Dada essa grande variedade de problemas, seria um desperdício manter parâmetros de saída para cada unidade recorrente. Mantendo a unidade recorrente sem saída por padrão, os *frameworks* transferem a responsabilidade de definir quando a saída deve ser computada (e onde os parâmetros devem ser aprendidos) para o desenvolvedor.

IV) Do ponto de vista teórico uma RNN consegue processar entradas de qualquer tamanho, um item por vez. Contudo, na prática vemos que RNN não trabalham bem com dependências muito longas. Isso ocorre pois em RNN estamos sempre atualizando o estado oculto em cada uma das iterações ao longo do tempo.

V) Redes recorrentes são mais sensíveis a esse tipo de problema pois além de serem profundas no número de camadas elas são profundas também na dimensão tempo. Dessa forma, durante o *backpropagation through time* o gradiente sofre mais multiplicações e é acumulado em múltiplas iterações potencializando ambos problemas.

VI) Existem muitas soluções para essa questão. A solução abaixo é apenas uma das possibilidades:

$\theta_{hh} = 1$

$\theta_{ih} = 1$

$b_h = -1$

$\theta_{ho} = 1$

$b_o = 0$

VII) a) Sua função é processar a imagem, analisando e extraindo as características visuais importantes, comprimindo toda a informação em um único vetor (v). Usando uma VGG-16 de encoder (como a VGG) não é possível, sozinha, gerar uma saída sequencial. A tarefa de decoder sequencial (gerar uma frase) requer uma arquitetura diferente, como uma RNN ou um Transformer.

b) O processo é chamado de autorregressivo porque a saída de um passo de tempo é usada como entrada para o próximo passo.

i. **Para gerar y_1 (Tempo $t = 1$):**

- **Estado Oculto Anterior (h_{t-1}):** É o h_0, que é o vetor da imagem v fornecido pela CNN.

- **Input (x_1):** Não temos uma palavra anterior. Portanto, usamos um *token* especial de início de sentença (ex: `<START>`).
- **Cálculo:** $h_1 = f(x_1\,\theta_{ih}^T + v\,\theta_{hh}^T + b_h)$ e $y_1 = g(h_1\,\theta_{ho}^T + b_o)$.

ii. **Para gerar y_2 (Tempo $t = 2$):**

- **Estado Oculto Anterior (h_{t-1}):** É o h_1, o estado oculto que acabou de ser calculado no passo anterior.
- **Input (x_2):** É a palavra que foi gerada no passo anterior, ou seja, y_1.
- **Cálculo** $h_2 = f(y_1\,\theta_{ih}^T + h_1\,\theta_{hh}^T + b_h)$ e $y_2 = g(h_2\,\theta_{ho}^T + b_o)$.

c) A abordagem mais comum e direta **é** usar a saída 'achatada' (flattened) da última camada convolucional.

i. Processo: A imagem passa pelas camadas convolucionais (ex: VGG-16), resultando em um *feature map* 3D (ex: $7 \times 7 \times 512$). Achatamos (flattening) para um vetor 1D (ex: 25088×1). A saída é usada como o vetor v.

ii. Formato (Shape): O vetor v é um vetor 1D (ex: '(25088,)').

iii. Uso: Este vetor 1D representa a imagem inteira, é usado para inicializar o primeiro estado oculto (h_0) da RNN, 'preparando' o decoder com o que ele deve descrever.

d) Sim, é possível, e esta arquitetura é conhecida como **Convolutional RNN** (ou ConvLSTM/ConvGRU).

i. Conceito: Em uma RNN padrão, o cálculo do novo estado oculto envolve multiplicações de matrizes (camadas densas):

$$h_t = \tanh(x_t\,\theta_{ih}^T + h_{t-1}\,\theta_{hh}^T + b_h)$$

Exigindo que h_t, h_{t-1} e x_t sejam vetores 1D.

ii. Em uma ConvRNN: As multiplicações de matrizes por θ_{ih} e θ_{hh} são substituídas por operações de convolução. Neste caso, o *input* x_t é o próprio mapa de características 3D da CNN (ex: $7 \times 7 \times 512$). É necessário que sejam preservadas as dimensões espaciais, permitindo que a RNN "lembre"não apenas o que viu nos frames anteriores, mas onde (espacialmente) viu.

VIII) *Forward* e BPTT na vanilla RNN escalar.

a) *Forward pass*:

$$\begin{aligned}
z_1 &= x_1\,\theta_{ih} + h_0\,\theta_{hh} = 1.0 \cdot 0.8 + 0 \cdot 0.3 = 0.8 \\
h_1 &= \tanh(0.8) \approx 0.6640 \\
z_2 &= x_2\,\theta_{ih} + h_1\,\theta_{hh} = (-0.5) \cdot 0.8 + 0.6640 \cdot 0.3 \approx -0.2008 \\
h_2 &= \tanh(-0.2008) \approx -0.1982 \\
\hat{y}_2 &= h_2\,\theta_{ho} \approx -0.1982 \\
J &= \tfrac{1}{2}(\hat{y}_2 - y)^2 \approx \tfrac{1}{2}(-1.1982)^2 \approx 0.7178
\end{aligned}$$

b) Derivada em θ_{ho}:

$$\frac{\partial J}{\partial \theta_{ho}} = (\hat{y}_2 - y)\,h_2 \approx -1.1982 \cdot (-0.1982) \approx 0.2374.$$

c) BPTT para θ_{hh}. Gradientes intermediários:

$$\frac{\partial J}{\partial h_2} = (\hat{y}_2 - y)\,\theta_{ho} \approx -1.1982$$
$$\frac{\partial J}{\partial z_2} = \frac{\partial J}{\partial h_2}\,(1 - h_2^2) \approx -1.1982 \cdot (1 - 0.0393) \approx -1.1511$$
$$\frac{\partial J}{\partial h_1} = \frac{\partial J}{\partial z_2}\,\theta_{hh} \approx -1.1511 \cdot 0.3 \approx -0.3453$$
$$\frac{\partial J}{\partial z_1} = \frac{\partial J}{\partial h_1}\,(1 - h_1^2) \approx -0.3453 \cdot (1 - 0.4410) \approx -0.1931$$

A contribuição direta de θ_{hh} em $t = 2$ tem fator h_1; em $t = 1$ tem fator $h_0 = 0$, anulando o termo. Logo,

$$\frac{\partial J}{\partial \theta_{hh}} = \frac{\partial J}{\partial z_2}\,h_1 + \frac{\partial J}{\partial z_1}\,h_0 \approx -1.1511 \cdot 0.6640 + 0 \approx -0.7644.$$

IX) Perplexidade. Calculando $\log_2$ de cada probabilidade:

$$\log_2 p = [-2,\ -1,\ -3,\ -1] \quad \Rightarrow \quad L = \sum_t \log_2 p_t = -7.$$

Expoente da perplexidade: $-L/m = 7/4 = 1.75$. Portanto,

$$\text{Perplexidade} = 2^{1.75} \approx 3.364.$$

Interpretação: em média, o modelo se comporta, em cada passo, como se estivesse em dúvida entre cerca de 3.4 *tokens* igualmente prováveis.