Exercícios Deep Learning
Aula 9

January 11, 2022

1 Modelos de Linguagem

1- Considere o seguinte exemplo:

Dados de treino:
$< s >$ I am Sam $< /s >$
$< s >$ Sam I am $< /s >$
$< s >$ Sam I like $< /s >$
$< s >$ Sam I do like $< /s >$
$< s >$ do I like Sam $< /s >$

Imagine que estamos treinando um modelo de linguagem bi-grama com os dados de treino mostrados acima.

a) Qual seria a próxima palavra mais provável de ser predita pelo modelo para cada uma das sequências de palavras abaixo?
(1) $< s >$ Sam ...
(2) $< s >$ Sam I do ...
(3) $< s >$ Sam I am Sam ...
(4) $< s >$ do I like ...

b) Observe essas três sentenças:
(5) $< s >$ Sam I do I like $< /s >$
(6) $< s >$ Sam I am $< /s >$
(7) $< s >$ I do like Sam I am $< /s >$.
Qual delas é mais provável de acodo com esse modelo?

2- Cite as principais desvantagens dos modelos n-grama e possíveis soluções para esses problemas.

3- Descreve brevemente como funcionam os modelos neurais de linguagem e o que são representações *one hot* e os *embeddings*.

2 RNN

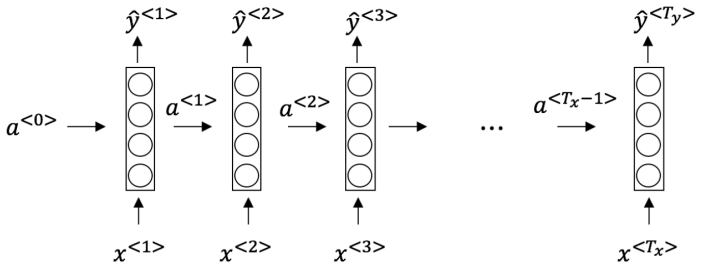
4- Qual a principal característica de uma Rede Neural Recorrente? Cite alguns exemplos de aplicações de RNNs.

5- Considere a tarefa de identificação de nomes próprios em uma frase. Projete uma arquitetura de RNN que, dado uma frase $x = (x^{<1>}, x^{<2>}, ..., x^{<T_x>})$ produza uma saída $\hat{y} = (\hat{y}^{<1>}, \hat{y}^{<2>}, ..., \hat{y}^{<T_y>})$ com a classificação de cada palavra $x^{<t>}$ como sendo nome próprio ou não.
(Observação: Basta esboçar a arquitetura! Não é necessário mostrar as funções de ativação.).

6- Considerando a arquitetura da rede projetada no exercício anterior, qual deve ser o tamanho da entrada (T_x) em relação ao tamanho da saída (T_y)? Cite outra aplicação que poderiam utilizar esse mesmo tipo de arquitetura.

7- Explique uma desvantagem ao utilizar a arquitetura da RNN tradicional na tarefa de identificação de nomes próprios em uma frase. Para isso, considere os exemplos: "He said, "Teddy Roosevelt was a great President!". " e "He said, "Teddy bears are on sale!". " A seguir, proponha uma nova arquitetura onde essa desvantagem possa ser solucionada.

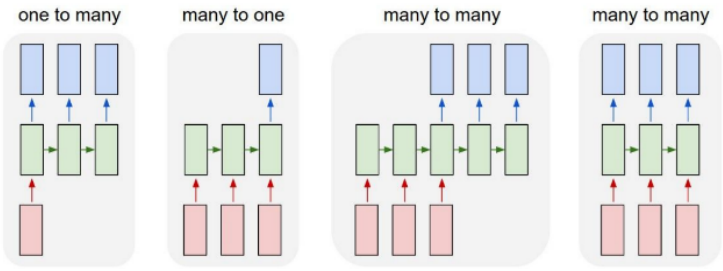
8- Considere a arquitetura de uma RNN *many-to-many* mostrada a seguir.



a) Especifique as funções que geram $a^{<t>}$ e $\hat{y}^{<t>}$. Lembre-se que $x^{<t>}$ é a t-ésima entrada, $\hat{y}^{<t>}$ é a t-ésima saída predita gerada a partir de $a^{<t>}$ que corresponde à t-ésima ativação que é repassada também para a célula $t + 1$.

b) Especifique a função de custo $L(\hat{y}, y)$ em termos das saídas preditas, $\hat{y}^{<t>}$.

9- Na figura a seguir, várias arquiteturas de RNN são apresentadas. Para cada uma delas, cite um exemplo de aplicação.

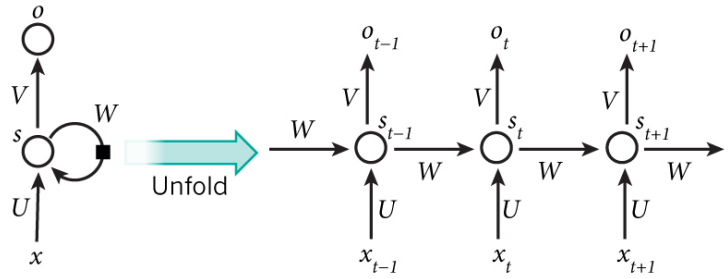


10- Considere uma arquitetura de RNN que, dado uma frase contendo duas palavras, $x = (x^{<1>}, x^{<2>})$, produza uma saída $\hat{y}$. Além disso, considere que as funções de ativação utilizadas são sigmóides e que o erro é dado pelo erro quadrático.

a) Esboce essa rede considerando todas as informações fornecidas no enunciado.

b) Faça o backpropagation nessa rede a partir do cálculo do erro $L(\hat{y}, y)$.

11- Considere a arquitetura da RNN abaixo:



Onde a entrada x é uma sequência de palavras e cada x_t é uma única palavra. A saída é uma distribuição de probabilidades sobre as palavras. Sabendo que o tamanho do dicionário C=8000 e o número de neurônios na camada escondida seja igual a H=100 e sendo as saídas s_t e o_t expressas com as seguintes funções de ativação:

$$s_t = \tanh\left(Ux_t + Ws_{t-1}\right)$$
$$o_t = \text{softmax}\left(Vs_t\right)$$

Escreva as dimensões das variáveis: x_t, o_t, s_t, U, V, W. Qual a quantidade total de parâmetros a serem aprendidos?

12- Como é possível utilizar RNNs em dados de vídeo?

Solução

1-

Probabilidade de cada bi-grama:
$P(Sam| < s >) = \frac{3}{5}$
$P(I| < s >) = \frac{1}{5}$
$P(I|Sam) = \frac{3}{5}$
$P(< /s > |Sam) = \frac{2}{5}$
$P(Sam|am) = \frac{1}{2}$
$P(< /s > |am) = \frac{1}{2}$
$P(am|I) = \frac{2}{5}$
$P(like|I) = \frac{2}{5}$
$P(do|I) = \frac{1}{5}$
$P(Sam|like) = \frac{1}{3}$
$P(< /s > |like) = \frac{2}{3}$
$P(like|do) = \frac{1}{2}$
$P(I|do) = \frac{1}{2}$

a)

(1) e (3): "I".
(2): "I" e "like" são igualmente prováveis.
(4): $< /s >$.

b)

(5): $\frac{3}{5} \times \frac{3}{5} \times \frac{1}{3} \times \frac{1}{2} \times \frac{2}{5} \times \frac{2}{3}$
(6): $\frac{3}{5} \times \frac{3}{5} \times \frac{2}{5} \times \frac{1}{2}$
(7): $\frac{1}{5} \times \frac{1}{5} \times \frac{1}{2} \times \frac{1}{3} \times \frac{3}{5} \times \frac{2}{5} \times \frac{1}{2}$
(6) é a sentença mais provável de acordo com o modelo.

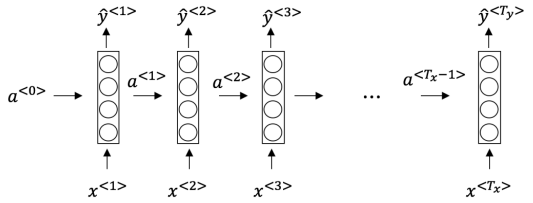
2- Problemas com modelos n-gramas: À medida que n aumenta, é necessário mais memória para armazenar os n-gramas e, além disso, as probabilidades tendem a ser cada vez menores dado o contexto cada vez mais específico. Esparsidade dos dados: a maioria dos n-gramas nunca aparece no corpus, mesmo quando são possíveis (e.g.: e se "estudantes abriram as mochilas" nunca apareceu nos dados?) dessa forma, a probabilidade seria zero.
Algumas maneiras de lidar com esparsidade dos dados são: Usar contexto menor (trade-off: modelo menos poderoso); Suavizar probabilidades (e.g., adicionando ocorrências imaginárias); Prevendo com um ensemble de modelos n-grama com n diferentes; ou Utilizar modelos distribuídos.

3- Prever a distribuição da próxima palavra dadas as K anteriores é apenas um problema de classificação (multi-classe). Nesse caso, usamos um 1-of-K (one-hot) encoding para palavras, a primeira camada pode ser vista como uma camada com pesos amarrados. *One-hot encoding* são representações esparsas (dimensão: $n_v \times 1$, onde n_v é igual ao tamanho do vocabulário) usadas para representar cada palavra do vocabulário. A matriz de pesos age como uma

lookup table (seleção de coluna) onde cada coluna é a representação de uma palavra, aka embedding, feature vector ou encoding. *Embeddings* são representações densas (dimensão: $n_e \times 1$, onde n_e é igual ao tamanho do embedding escolhido) que enfatizam a localização de uma palavra em um espaço de alta dimensão onde palavras próximas são mais similares semanticamente.

4- Uma rede neural recorrente (RNN) é uma classe de redes neurais artificiais onde as conexões entre nós formam um grafo direcionado ao longo de uma sequência temporal. As RNNs lembram e são influenciadas pelo passado, ou seja, coisas já aprendidas com entradas anteriores. Dentre os principais exemplos de aplicação de RNNs estão: reconhecimento de fala, geração de música, classificação de sentimento, tradução automática, reconhecimento de atividade em vídeo e reconhecimento de entidades em frases.

5-



6- $T_x = T_y$. Aplicações: Identificação de palavras específicas (entidades).

7- O problema ao usar uma RNN tradicional é que a predição até certo ponto considera somente as palavras que antecedem a palavra alvo, ou seja, a informação das palavras que ocorrem depois não são levadas em consideração na etapa de *forward*. Assim, nos exemplos apresentados, "Teddy" seria classificado como nome próprio em ambos os casos, já que as palavras que poderiam distinguir o nome próprio (ex: Roosevelt e bears), estão após a palavra alvo. Solução: RNN bidirecional (BRNN).

8-

a)

$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$
$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$

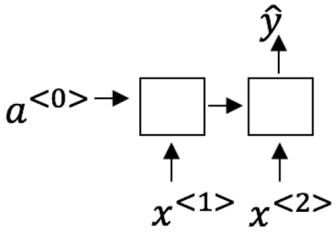
b) $L(\hat{y}, y) = \sum_{t=1}^{T_x} L^{<t>}(\hat{y}^{<t>}, y^{<t>})$

9-

one-to-many: Geração de texto.
many-to-one: Classificação de sentimentos.
many-to-many: Tradução automática.
many-to-many (pareada): Reconhecimento de entidades.

10-

a)



b) Considerando $z_y^{<t>} = W_{ya}a^{<t>} + b_y$ e $z_a^{<t>} = W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a$.

$$\frac{\partial L}{\partial W_{ya}} = \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial W_{ya}} = (\hat{y} - y)\sigma(z_y^{<2>})(1 - \sigma(z_y^{<2>}))a^{<2>}$$

$$\frac{\partial L}{\partial W_{aa}} = \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial a^{<2>}}\frac{\partial a^{<2>}}{\partial W_{aa}} =$$

$$(\hat{y} - y)\sigma(z_y^{<2>})(1 - \sigma(z_y^{<2>}))W_{ya}\sigma(z_a^{<2>})(1 - \sigma(z_a^{<2>}))(a^{<1>} + W_{aa}\frac{\partial a^{<1>}}{\partial W_{aa}})$$

$$\frac{\partial a^{<1>}}{\partial W_{aa}} = \sigma(z_a^{<1>})(1 - \sigma(z_a^{<1>}))a^{<0>}$$

$$\frac{\partial L}{\partial W_{ax}} = \frac{\partial L}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial a^{<2>}}\frac{\partial a^{<2>}}{\partial W_{ax}} =$$

$$(\hat{y} - y)\sigma(z_y^{<2>})(1 - \sigma(z_y^{<2>}))W_{ya}\sigma(z_a^{<2>})(1 - \sigma(z_a^{<2>}))(x^{<2>} + W_{aa}\frac{\partial a^{<1>}}{\partial W_{ax}})$$

$$\frac{\partial a^{<1>}}{\partial W_{ax}} = \sigma(z_a^{<1>})(1 - \sigma(z_a^{<1>}))x^{<1>}$$

11- $2 * H * C + H^2$ parâmetros.

$$x_t \in \mathbb{R}^{8000}$$
$$o_t \in \mathbb{R}^{8000}$$
$$s_t \in \mathbb{R}^{100}$$
$$U \in \mathbb{R}^{100\times 8000}$$
$$V \in \mathbb{R}^{8000\times 100}$$
$$W \in \mathbb{R}^{100\times 100}$$

12- Uma alternativa é primeiro usar uma rede convolucional e, em seguida, usar uma RNN ao invés de uma camada totalmente conectada. Dessa forma a rede mantém informação sobre a ordem temporal dos frames.