AULA 5
INICIALIZAÇÃO DOS PARÂMETROS DA RNA

1. Objetivos

- Apresentar algumas formas de inicializar os parâmetros de uma RNA.
- Apresentar alguns problemas causados por inicialização não adequada dos parâmetros.
- Mostrar o que ocorre se os parâmetros forem inicializados com uma constante.

1. Introdução

- Como o método do Gradiente descendente é um método iterativo é necessário inicializar os parâmetros da RNA de alguma forma.
- Existem diversos problemas que podem aparecer durante o treinamento da RNA se os parâmetros forem inicializados de forma inadequada $\Rightarrow$ cuidado deve ser tomado para inicializar os parâmetros de uma RNA.
- Os parâmetros devem ser sempre inicializados com números aleatórios pequenos.
- **Importante:**
  - Nunca se deve inicializar os parâmetros com zeros, ou com outra constante qualquer $\Rightarrow$ a inicialização de todos os parâmetros com um único valor transforma a RNA em uma função linear $\Rightarrow$ uma demonstração desse fato se encontra na seção 4 dessa aula.
  - Se inicializarmos todos os parâmetros de uma RNA com zero ou alguma outra constante, no treinamento eles passam a ter todos o mesmo valor e a RNA se comporta como se as camadas tivessem um único neurônio.
- Ressalta-se que inicializar os vieses com zeros não gera nenhum problema, porque basta que os pesos das ligações sejam diferentes para quebrar a simetria entre os neurônios.

2. Problemas causados por inicialização inadequada dos parâmetros

- Inicialização inadequada de parâmetros pode causar problemas de saturação das funções de ativação que resulta em incapacidade da rede ser treinada.

- Dois problemas comuns podem ocorrer em decorrência de inicialização errada dos parâmetros:
  - Gradientes tendendo a zero;
  - Gradientes tendendo a infinito.
- O valor médio dos parâmetros de cada camada deve ser pequeno para evitar saturação das funções de ativação.
- Observa-se, contudo, que, mesmo números aleatórios pequenos podem gerar problemas de saturação em uma RNA profunda.
- O problema de saturação das funções de ativação pode ser visualizado na Figura 1.
- A saturação das funções de ativação gera problemas de "vanishing gradients" e "exploding gradients" em RNA profundas.
- Problemas de "vanishing gradients" e "exploding gradients" dificultam o treinamento da RNA.
- No caso de ocorrer saturação não é possível sair dessa condição se forem usadas funções de ativação tipo sigmoide ou tangente hiperbólica ⇒ função de ativação tipo ReLu é mais robusta a problemas de saturação.

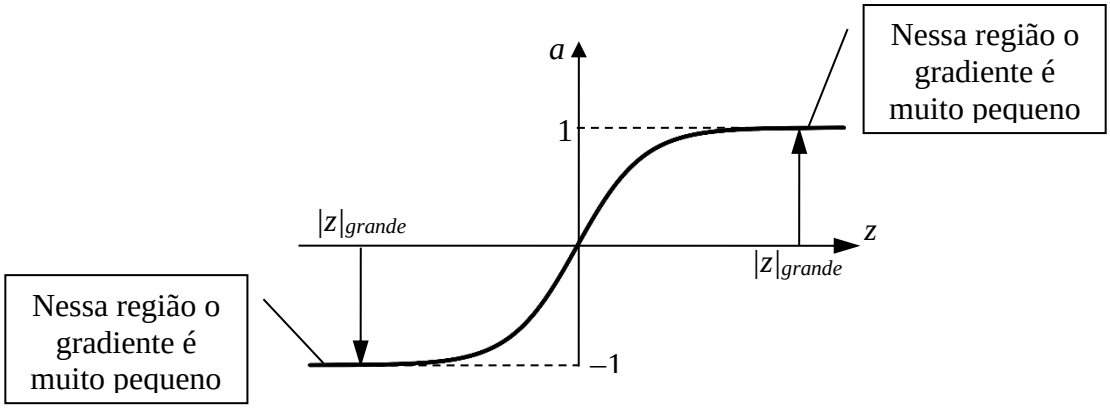

Figura 1. Problema de saturação da função de ativação tangente hiperbólica para valores grandes do estado do neurônio.

- **"Vanishing Gradients"**
  - Para RNAs profundas, em razão dos parâmetros terem valores pequenos, o valor absoluto dos gradientes se torna cada vez menor na medida em que avançamos para as camadas iniciais na retro-propagação ⇒ com isso, as camadas iniciais das RNAs são as mais lentas para serem treinadas.
  - A atualização dos pesos nas camadas iniciais é pequena e resulta em uma convergência lenta fazendo com que a otimização da função de custo seja demorada ⇒ no pior caso, os gradientes tendem a zero e o treinamento é impossível de ser realizado.

- No caso das funções de ativação serem sigmoide, ou tangente hiperbólica, e se os pesos tiverem valor absoluto grande, os gradientes irão tender a zero impedindo o treinamento da RNA.

- No caso das funções de ativação serem ReLu, a ocorrência de problemas de "vanishing gradients" é menor, pois somente ocorreria se os estados de todos os neurônios de uma mesma camada forem negativos, o que provocaria gradientes iguais a zero em todos os neurônios dessa camada impedindo, assim, o treinamento da RNA.

- **"Exploding Gradients"**

  - Esse caso é o oposto ao problema de "vanishing gradients".

  - Se os pesos das ligações forem valores grandes, os estados dos neurônios serão também valores grandes e consequentemente as ativações serão próximas de um, ou zero, em todos os neurônios, se forem utilizadas funções de ativação sigmoide ou tangente hiperbólica, e valores muito grandes se forem utilizadas funções de ativação ReLu $\Rightarrow$ quanto maior o número de neurônios nas camadas pior será o problema.

  - Na medida em que a propagação para frente é realizada os estados podem aumentar cada vez mais, gerando valores de saída muito grandes e, assim, erros entre as saídas desejadas e as saídas calculadas muito grandes.

  - Erros grandes geram gradientes também muito grandes, provocando grandes alterações dos parâmetros em cada iteração $\Rightarrow$ isso resulta em oscilação da função de custo em torno do mínimo e a convergência do treinamento pode não ocorrer.

  - Outro impacto de valores grandes de gradientes é poder provocar problemas numéricos de overflow.

- **Uma boa prática para evitar esses problemas de saturação, "vanishing gradients" em RNAs profundas é utilizar funções de ativação tipo ReLu ou LeRelu nas camadas intermediárias. Porém, funções de ativação tipo ReLu são mais sensíveis a problemas de "exploding gradientes" e também a problemas de "Dead ReLu".**

3. Métodos práticos de inicialização dos parâmetros

- Existem métodos para inicializar os parâmetros de uma RNA de forma a evitar os problemas de saturação, "vanishing gradients" e "exploding gradients".

- Em todos esses métodos, os pesos das ligações de uma RNA são inicializados com **números aleatórios** com **distribuição uniforme** e com **variância inversamente proporcional ao número de neurônios da camada**.

- O método de inicialização mais utilizado é o **Xavier**, que é também chamado de **Glorot Normal**. Nesse método a inicialização dos pesos da *l*-ésima camada é feita da seguinte forma:

$$\mathbf{W}^{[l]} = \text{random}(n^{[l]}, n^{[l-1]})\sqrt{\frac{1}{n^{[l-1]}}} \tag{1}$$

- Outras formas de inicializar parâmetros podem ser usadas. Para funções de ativação ReLu a inicialização dos pesos da *l*-ésima camada pode ser feita por:

$$\mathbf{W}^{[l]} = \text{random}(n^{[l]}, n^{[l-1]})\sqrt{\frac{2}{n^{[l-1]}}} \tag{2}$$

onde random($n^{[l]}$, $n^{[l-1]}$) é uma função que gera uma matriz de números aleatórios com distribuição uniforme de dimensão ($n^{[l]}$, $n^{[l-1]}$).

- Ainda outro método comumente usado é inicializar os pesos da *l*-ésima camada por:

$$\mathbf{W}^{[l]} = \text{random}(n^{[l]}, n^{[l-1]})\sqrt{\frac{2}{n^{[l]} + n^{[l-1]}}} \tag{3}$$

- No Keras a inicialização dos parâmetros é realizada na configuração da RNA ao adicionar as camadas.

- Abaixo seguem alguns exemplos de inicialização dos parâmetros usando o Keras.

```
from tensorflow.keras.layers import Dense
from tensorflow.keras import initializers

# Camada inicializada com zeros nos pesos das sinapses e vieses
rna.add(Dense(units=64, activation='relu', kernel_initializer='zeros',
              bias_initializer='zeros', input_dim=12288))

# Camada inicializada com números aleatórios de distribuição normal,
com média zero e desvio padrão 0,05 nos pesos e bias
rna.add(Dense(units=64, activation='sigmoid',
    kernel_initializer=initializers.RandomNormal(mean=0.0,stddev=0.05),
    bias_initializer=initializers.RandomNormal(mean=0.0,stddev=0.05)))

# Camada inicializada com números aleatórios de distribuição uniforme
com valores entre -0.05 e 0.05 para os pesos das sinapses.
rna.add(Dense(units=64, activation='sigmoid',
              kernel_initializer=initializers.RandomUniform(minval=
              -0.05, maxval=0.05)))

# Camada inicializada com o método de Xavier
rna.add(Dense(units=32, activation='relu',
             kernel_initializer=initializers.glorot_normal(seed=None)))
```

- Observa-se que para inicializar os parâmetros usando outra forma que não seja a padrão, é preciso antes importar do Keras a classe de inicializadores.

➢ O padrão de inicialização de parâmetros no Keras é o método de Xavier para os pesos das ligações e zeros para os vieses ⇒ como essa é a forma mais eficiente de inicialização de parâmetros sugere-se não alterar esse padrão.

4. Inicialização de parâmetros com uma constante

➢ O que aconteceria se todos os parâmetros da RNA forem inicializados com zero?

➢ Considere uma RNA simples, como a mostrada na Figura 2.

- Numero de entradas: $n_x = 2$;
- Número de saídas: $n_y = 1$;
- Numero de camadas: $L = 2$;
- Numero de neurônios na camada intermediária: $n^{[1]} = 2$.

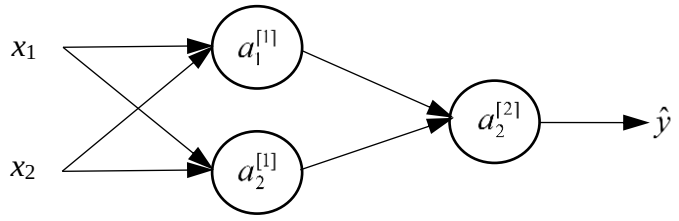

Figura 2. RNA rasa de duas entradas, uma saída e uma camada intermediária de dois neurônios.

➢ A propagação para frente nessa RNA é calculada por:

$$a_1^{[1]} = g^{[1]}(w_{1,1}^{[1]} x_1 + w_{1,2}^{[1]} x_2 + b_1^{[1]}) = g^{[1]}(z_1^{[1]}) \tag{4}$$

$$a_2^{[1]} = g^{[1]}(w_{2,1}^{[1]} x_1 + w_{2,2}^{[1]} x_2 + b_2^{[1]}) = g^{[1]}(z_2^{[1]}) \tag{5}$$

$$\hat{y} = a_2^{[2]} = g^{[2]}(w_1^{[2]} a_1^{[1]} + w_2^{[2]} a_2^{[1]} + b^{[2]}) = g^{[2]}(z^{[2]}) \tag{6}$$

Como todos os parâmetros são iguais a zeros, então:

$$z_1^{[1]} = z_2^{[1]} = 0 \quad \text{e} \quad a_1^{[1]} = a_2^{[1]} \tag{7}$$

➢ A retro-propagação é calculada usando as equações a seguir.

Derivada parcial da função de erro em relação aos estados da segunda camada:

$$dEdz^{[2]} = \left[\frac{\partial E(a^{[2]}, y)}{\partial a^{[2]}}\right]\left[\frac{dg^{[2]}(z^{[2]})}{dz^{[2]}}\right] \tag{8}$$

As derivadas parciais dos pesos das ligações de 2ª camada são dadas por:

$$dEd\mathbf{W}^{[2]} = dEdz^{[2]}\mathbf{a}^{[1]T} \tag{9}$$

Como todas as ativações dos neurônios da primeira camada são iguais ($a_1^{[1]} = a_2^{[1]}$), então:

$$dEdw_1^{[2]} = dEdw_2^{[2]} \tag{10}$$

Ou seja, as derivadas parciais da função de custo em relação aos pesos da 2ª camada serão todas iguais. Assim, após a atualização dos pesos da 2ª camada eles serão diferentes de zero ⇒ mas serão todos iguais.

Como os pesos da segunda camada são todos iguais a zero ⇒ as derivadas parciais da função de erro em relação às ativações dos neurônios da 1ª camada são todos iguais a zero:

$$dEd\mathbf{a}^{[1]} = dEdz^{[2]}\mathbf{W}^{[2]T} = \begin{cases} 0 \\ 0 \end{cases} \tag{11}$$

Como as derivadas parciais $dEd\mathbf{a}^{[1]}$ são iguais para todos os neurônios da 1ª camada e os seus estados são todos iguais a zero, então, as derivadas parciais $dEd\mathbf{z}^{[1]}$ também serão iguais a zero, ou seja:

$$dEd\mathbf{z}^{[1]} = dEd\mathbf{a}^{[1]} * \frac{dg^{[1]}(\mathbf{z}^{[1]})}{d\mathbf{z}^{[1]}} = \begin{cases} 0 \\ 0 \end{cases} \tag{12}$$

Como as derivadas parciais $dEd\mathbf{z}^{[1]}$ são todas iguais a zero ⇒ temos que as derivadas parciais da função de erro em relação aos parâmetros da 1ª camada serão todas iguais:

$$dEd\mathbf{W}^{[1]} = dEd\mathbf{z}^{[1]}\mathbf{x}^{)T} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{13}$$

Assim, após a atualização dos parâmetros da 1ª camada eles serão diferentes de zero ⇒ mas serão todos iguais.

- Quando inicializamos todos os parâmetros com zero, ou com o mesmo valor, as derivadas da função de custo serão as mesmas para todos os parâmetros da RNA de uma mesma camada ⇒ isso faz com que todos os neurônios de uma mesma camada intermediária sejam iguais e continuam assim durante todo o treinamento.

- Ressalta-se que inicializar os vieses com zeros não gera nenhum problema, porque basta que os pesos das ligações sejam diferentes para quebrar a simetria entre os neurônios.

- **Conclusões:**
  - **Se inicializarmos todos os parâmetros de uma RNA com zero, no treinamento eles passam a ter todos o mesmo valor e a RNA se comporta como se as camadas tivessem um único neurônio;**
  - **O mesmo problema acontece se todos os parâmetros fossem inicializados com uma constante diferente de zero;**
  - **A inicialização de todos os parâmetros com um único valor transforma a RNA em uma função linear.**