UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

LISTA 1 - REDES NEURAIS E DEEP LEARNING

JOSÉ AUGUSTO AGRIPINO DE OLIVEIRA: 20241030884

JOSÉ EDIVANDRO DE SOUSA JUNIOR: 20241026406

VINICIUS JOSÉ MIRANDA TOSCANO DE BRITO FILHO: 20241026676

Natal-RN
2025

1) As métricas de distancia entre vetores são aplicadas nos estudos de aprendizagem de máquina como medidas de similaridade/dissimilaridade entre vetores que representam padrões (atributos). Apresente um estudo sobre as seguintes distancias: Distancia Euclidiana, Distancia de Minkowski, Distancia City Block, Distancia de Mahalanobis, Coeficiente de Correlação de Pearson e Similaridade Cosseno. Apresente neste estudo aplicações onde cada tipo de métricas de distância é mais adequada.

Distância Euclidiana:

O artigo intitulado "Uso de Redes Neurais Profundas para o Aprendizado de Funções Heurísticas para Algoritmos de Busca de Caminhos", de Daniel Matheus Doebber, aborda a distância euclidiana como uma medida para avaliar a dissimilaridade entre dois pontos, sendo crucial para determinar a proximidade entre vetores de atributos em um espaço multidimensional.

Na aplicação discutida, a distância euclidiana se revelou útil em cenários nos quais é possível visualizar de maneira intuitiva agrupamentos em espaços com duas ou mais dimensões. Ela facilitou a identificação de regiões densas no espaço de atributos, separadas por áreas de baixa densidade, o que se alinha bem com a percepção dos clusters formados.

$$d(x,y) \rightarrow ||x-y||_2 = [\sum_{i}^{n}(x_i - y_i)^2]^{\frac{1}{2}} = \sqrt{\sum_{i}^{n}(x_i - y_i)^2} \tag{1}$$

Onde a Equação 1 é a expressão matemática da distância euclidiana.

City Block, Mahalanobis e Minkowski:

O artigo intitulado "Análise da Influência das Métricas de Distância no Algoritmo Semi-Supervisionado de Competição e Cooperação de Partículas", de Lucas Guerreiro e Fabricio Aparecido Breve, aborda o estudo das distâncias City Block, Mahalanobis e Minkowski.

A distância City Block, também conhecida como distância de Manhattan ou métrica do táxi, segue o princípio de que seu valor é sempre maior ou igual a zero. Quando o valor é zero, significa que os dados são idênticos. Embora seja semelhante à distância euclidiana, a diferença é que as diferenças entre os pontos não são elevadas ao quadrado. Essa métrica é útil quando se deseja atribuir maior importância às diferenças em características individuais dos dados.

$$d(x,y) \rightarrow ||x-y||_1 = \sum_{i}^{n}|x_i - y_i| \tag{2}$$

Onde a Equação 2 é a expressão matemática da distância City Block.

A distância de Minkowski é uma versão mais generalizada da distância Euclidiana, com um parâmetro ajustável denominado "p". Quando p=2, ela corresponde à distância Euclidiana, e quando p=1, torna-se a distância City Block. Essa métrica é útil quando se deseja ajustar a

medida de distância para se adequar a diferentes escalas das características dos dados.

$$d(x,y) \to ||x-y||_p = \left(\sum_i^n |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{3}$$

Onde a Equação 3 é a expressão matemática da distância Minkowski.

A distância do cosseno é uma medida de similaridade angular entre dois vetores, calculada a partir do produto escalar normalizado pelo produto das normas de cada vetor, ou seja, suas distâncias euclidianas em relação à origem. Essa métrica é amplamente utilizada em sistemas de recomendação baseados em conteúdo e no processamento de linguagem natural (PLN).

$$d(x,y) \to 1 - \frac{x \cdot y}{||x||_2 ||y||_2} = 1 - \frac{\sum_i^n x_i y_i}{\sqrt{\sum_i^n x_i^2} \sqrt{\sum_i^n y_i^2}} \tag{4}$$

Onde a Equação 4 é a expressão matemática da Similaridade do Cosseno.

O coeficiente de correlação de Pearson entre dois dados é uma medida da relação linear entre eles, sendo utilizado para identificar padrões correlacionados em mineração de dados e sistemas de recomendação

$$r_{xy} = \frac{\sum z_x z_y}{N} \tag{5}$$

Onde a Equação 5 é a expressão matemática do Coeficiente de Pearson.

2) Considere a função $E(w)$ onde $w = [w_1, w_2, \ldots, w_n]^t$ é um vetor com múltiplas variáveis. Usando a expansão em série de Taylor a função pode ser expressa como

$$E(w(n+\Delta w)) = E(w(n)) + g^T(w(n)) \cdot \Delta w + \frac{1}{2} \Delta w^T H(w(n)) \Delta w + O(\|\Delta w\|^3) \tag{6}$$

onde $g(w(n))$ é o vetor gradiente local definido por

$$g(w) = \frac{\partial E(w)}{\partial w} \tag{7}$$

e $H(w)$ é matriz Hessiana, definida por

$$H(w) = \frac{\partial^2 E(w)}{\partial w^2} \tag{8}$$

Demonstre com base na expansão em série de Taylor:

a) Que o método do gradiente da descida mais íngreme é dado por $w(n+1) = w(n) - \eta g(w(n))$.

Considerando a expansão em série de Taylor fornecida:

$$E(w(n+\Delta w)) = E(w(n)) + g^T(w(n)) \cdot \Delta w + \frac{1}{2}\Delta w^T H(w(n))\Delta w + O(\|\Delta w\|^3) \tag{9}$$

Assumindo que Δw é pequeno para que a aproximação da expansão apresente um bom resultado apenas com o termo de 1ª ordem, podemos descartar os termos de ordem superior, ficando com:

$$E(w(n+\Delta w)) \approx E(w(n)) + g^T(w(n)) \cdot \Delta w \tag{10}$$

Agora, para minimizar $E(w(n+\Delta w))$, devemos escolher Δw para seguir na direção oposta ao gradiente, uma vez que o gradiente aponta para a direção de maior crescimento, sendo assim:

$$\Delta w = -\eta g(w) \tag{11}$$

Onde η é a taxa de aprendizado, já que apenas sabemos para onde andar, mas não sabemos o tamanho do passo.

Portanto, a atualização dos pesos é dada por:

$$w(n+1) = w(n) - \eta g(w(n)) \tag{12}$$

b) Que o método de Newton é dado por $w(n+1) = w(n) + H^{-1}w(n)g(w(n))$.

O método de Newton é derivada assumindo que queremos minimizar uma aproximação quadrática da função $E(w)$, assim, a expansão da série de Taylor da função é:

$$E(w(n+\Delta w)) \approx E(w(n)) + g^T(w(n)) \cdot \Delta w + \frac{1}{2}\Delta w^T H(w(n))\Delta w \tag{13}$$

Derivada em relação a Δw:

$$\frac{\partial E(w+\delta w)}{\partial(\Delta w)} = g(w) + H(w)\Delta w \tag{14}$$

Zeramos para minimizar:

$$g(w) + H(w)\Delta w = 0 \Rightarrow \Delta w = -H^{-1}(w(n))g(w) \tag{15}$$

Sendo assim, o método de Newton nos dá:

$$w(n+1) = w(n) - H^{-1}(w(n))g(w(n)) \tag{16}$$

c) Que o η^* ótimo é dado por $g(w) \cdot g^T(w)/(g^T(w) \cdot H(w) \cdot g(w))$ assumindo a matriz H definada.

Considerando os termos de 1ª e 2ª da expansão em série de Taylor da função $E(w)$:

$$E(w(n+\Delta w)) \approx E(w) + g^T(w) \cdot \Delta w + \frac{1}{2}\Delta w^T H(w)\Delta w \tag{17}$$

Assumindo que $\Delta w = -\eta g(w)$, como mostrado na alternativa "a", temos que:

$$\phi(\eta) = E(w - \eta g(w)) \approx E(w) - \eta g^T(w)g(w) + \frac{1}{2}\eta^2 g^T(w)H(w)g(w) \tag{18}$$

Agora derivando em relação a η:

$$\frac{\partial \phi(\eta)}{\partial \eta} = -g^T(w)g(w) + \eta g^T(w)H(w)g(w) \tag{19}$$

Para encontrar o ponto crítico da função parabólica $\phi(\eta)$ igualamos a zero:

$$-g^T(w)g(w) + \eta g^T(w)H(w)g(w) = 0 \tag{20}$$

Com isso, o η é dado por:

$$\eta^* = \frac{g^T(w)g(w)}{g^T(w)H(w)g(w)} \tag{21}$$

Isso nos garante que η é um ponto crítico, para garantir que é um ponto de mínimo, é preciso calcular a derivada segunda. Todavia, é dado que a Hessiana é maior que zero, logo a curvatura é positiva.

d) Qual será o valor de η sob as condições acima se o vetor gradiente estiver alinhado com um autovetor correspondente ao autovalor máximo (λ_{max}).

O passo ótimo η foi derivado como:

$$\eta^* = \frac{g^T(w)g(w)}{g^T(w)H(w)g(w)} \tag{22}$$

Se o vetor gradiente $g(w)$ está alinhado com um autovetor correspondente a um autovalor máximo (λ_{max}), isso significa que é possível escrever:

$$g(w) = \alpha v_{max} \tag{23}$$

Onde v_{max} é o autovetor correspondente ao autovalor máximo λ_{max} e λ é uma constante escalar de proporcionalidade.

Substituindo $g(w) = \alpha v_{max}$ na fórmula para o passo ótimo:

$$\eta^* = \frac{(\alpha v_{max})^T (\alpha v_{max})}{(\alpha v_{max})^T H(w)(\alpha v_{max})} \tag{24}$$

Segundo a definição de autovalor e autovetor, $H(w)v_{max} = \lambda_{max} v_{max}$, então:

$$\eta^* = \frac{\alpha^2 v_{max}^T v_{max}}{\alpha^2 v_{max}^T \lambda_{max} v_{max}} \tag{25}$$

Aqui, $v_{max}^T v_{max} = 1$ (pois os autovetores são normalizados), e a constante α^2 aparece tanto no numerador quanto no denominador, podendo ser cancelada:

$$\eta^* = \frac{1}{\lambda_{max}} \tag{26}$$

e) Em um determinado ponto w o gradiente da função $E(w)$ é igual a zero, isto é, $g(w) = 0$. Isto não é suficiente para indicar que este ponto corresponde a um ponto de mínimo. Pode ser um ponto de máximo ou um ponto de sela. Com base nos autovalores da matriz Hessiana, definida como $H(w) = \frac{\partial^2 E(w)}{\partial w^2}$, apresente a condição para este ponto ser:

- Um ponto de mínimo

O ponto w é um mínimo local se todos os autovalores de $H(w)$ forem positivos. Isso indica que a função é côncava para cima em todas as direções.

$$\lambda_i > 0 \quad \forall i \tag{27}$$

- Um ponto de máximo

O ponto w é um máximo local se todos os autovalores de $H(w)$ forem negativos.

$$\lambda_i < 0 \quad \forall i \tag{28}$$

- Um ponto de sela da função

O ponto w é um ponto de sela se a Hessiana tiver autovalores de sinais diferentes.

$$\exists \lambda_i > 0 \quad \text{e} \quad \exists \lambda_i < 0 \tag{29}$$

3) Apresente um estudo comparando os seguintes algoritmos de otimização: Gradiente Estocástico (SGM), AdaGrad, RMSProp e Adam. Estes métodos ou otimizadores são utilizados no processo de aprendizagem de redes neurais/deep learning.

Link para apresentação do estudo realizado.

4) O modelo de neurônio artificial de Mc-Culloch-Pitts faz uso da função de ativação para resposta do neurônio artificial. A função sigmoíde (ou função logística) e a função tangente hiperbólica (ou tangsigmoíde) são normalmente utilizadas nas camadas ocultas das redes neurais perceptrons de múltiplas camadas tradicionais (uma ou duas camadas ocultas - shallow network). A função ReLu (retificador linear) é normalmente utilizadas nas camadas ocultas das redes Deep Learning.

a) (i) Faça uma análise comparativa de cada uma destas funções apresentando de forma gráfica a variação da função e da sua derivada com relação a v (potencial de ativação).

Link para o GitHub com a solução proposta.

(ii) Mostre que $\varphi'(v) = \frac{\mathrm{d}}{\mathrm{d}v}\varphi(v) = \alpha \cdot \varphi(v) \cdot [1 - \varphi(v)]$ para a função sigmoide.

Seja a função sigmoide:

$$\varphi = \frac{1}{1 + e^{-\alpha v}} \tag{30}$$

Derivando a sigmoide:

$$\varphi'(v) = \frac{\mathrm{d}}{\mathrm{d}v}\left(\frac{1}{1 + e^{-\alpha v}}\right) \tag{31}$$

Utilizando a regra da cadeia:

$$\varphi'(v) = \frac{\mathrm{d}}{\mathrm{d}v}\left(1 + e^{-\alpha v}\right)^{-1} = -1 \cdot \left(1 + e^{-\alpha v}\right)^{-2} \cdot \frac{\mathrm{d}}{\mathrm{d}v}\left(1 + e^{-\alpha v}\right) \tag{32}$$

$$\varphi'(v) = -\frac{1}{\left(1 + e^{-\alpha v}\right)^2} \cdot \left(-\alpha e^{-\alpha v}\right) = \frac{\alpha e^{-\alpha v}}{\left(1 + e^{-\alpha v}\right)^2} \tag{33}$$

Sabemos que:

$$\varphi(v) = \frac{1}{1 + e^{-\alpha v}} \Rightarrow 1 - \varphi(v) = \frac{e^{-\alpha v}}{1 + e^{-\alpha v}} \tag{34}$$

Então:

$$\varphi \cdot (1 - \varphi(v)) = \left(\frac{1}{1 + e^{-\alpha v}}\right) \cdot \left(\frac{e^{-\alpha v}}{1 + e^{-\alpha v}}\right) = \frac{e^{-\alpha v}}{(1 + e^{-\alpha v})^2} \tag{35}$$

Logo:

$$\varphi'(v) = \alpha \cdot \frac{e^{-\alpha v}}{(1 + e^{-\alpha v})^2} = \alpha \cdot \varphi(v) \cdot (1 - \varphi(v)) \tag{36}$$

Portanto:

$$\varphi'(v) = \alpha \cdot \varphi(v) \cdot (1 - \varphi(v)) \tag{37}$$

(iii) Mostre que $\varphi'(v) = \frac{\mathrm{d}}{\mathrm{d}v}\varphi(v) = \frac{\alpha}{2}[1 - \varphi^2(v)]$ para a função tangsigmoide.

É sabido que:

$$\frac{\mathrm{d}}{\mathrm{d}v}\tanh(\alpha v) = \alpha \cdot (1 - \tanh^2(\alpha v)) \tag{38}$$

Como:

$$\varphi(v) = \tanh\left(\frac{\alpha v}{2}\right) \tag{39}$$

Então:

$$\varphi'(v) = \frac{\mathrm{d}}{\mathrm{d}v}\tanh\left(\frac{\alpha v}{2}\right) = \frac{\alpha}{2} \cdot \left(1 - \tanh^2\left(\frac{\alpha v}{2}\right)\right) = \frac{\alpha}{2} \cdot (1 - \varphi^2(v)) \tag{40}$$

Desse modo:

$$\varphi'(v) = \frac{\alpha}{2} \cdot (1 - \varphi^2(v)) \tag{41}$$

b) As funções de saída das redes neurais dependem do modelo probabilístico que se busca gerar com a rede. Faça uma análise sobre a escolhas destas funções considerando os seguintes problemas:

(i) Classificação de padrões com duas classes.

Função de saída:

$$\varphi(v) = \frac{1}{1 + e^{-v}} \tag{42}$$

A saída da função sigmoide está no intervalo (0, 1), sendo possível interpretar como a probabilidade da classe positiva.

(ii) Classificação de padrões com múltiplas classes.

Função de saída:

$$\varphi_i(v) = \frac{e^{v_i}}{\sum_j e^{v_j}} \tag{43}$$

Essa função transforma o vetor de saída da rede neural em uma distribuição de probabilidade sobre as classes.

(iii) Problema de regressão (aproximação de funções).

Função de saída:

$$\varphi_i(v) = v \tag{44}$$

Permite que a saída assuma qualquer valor real, o que é necessário para estimar variáveis contínuas.

c) Para cada uma das condições do item (b) apresente a função custo a ser considerada no processo de treinamento de uma rede neural com múltiplas camadas em um processo de aprendizagem supervisionada.

(i) Classificação binária

Função custo $\mathcal{L}$: Binary Cross-Entropy

$$\mathcal{L} = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})] \tag{45}$$

(ii) Classificação multiclasse

Função custo: Categorical Cross-Entropy

$$\mathcal{L} = -\sum_{i=1}^{K} y_i \cdot \log(\hat{y}_i) \tag{46}$$

(iii) Regressão

Função custo: Mean Squared Error

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \tag{47}$$

4) O algoritmo backpropagation é o algoritmo base no processo de treinamento de redes neurais do tipo: rede perceptron de múltiplas camadas (MLP), rede convolucional (CNN), rede recorrente (ex: LSTM).

(i) Apresente a dedução do algoritmo para uma rede perceptron de múltiplas camadas com múltiplas variáveis de entradas e múltiplas variáveis de saída.

4 - (i)

A motivação para o algoritmo de backpropagation é a atualização dos pesos nas camadas ocultas. Até então outras estratégias eram empregadas. A principal intuição é a de retropropação do erro da camada de saída em direção às primeiras camadas.

O núcleo do treinamento de uma rede neural MLP é minimizar a função custo $\mathcal{E}(n)$. Também é sabido que uma MLP é uma grande função composta, em que a entrada de uma camada é a saída de outra. Então, como é possível minimizar a função custo? Regra da cadeia! Ela é utilizada na atualização dos pesos dos neurônios, como é possível ver na dedução a seguir. É importante ressaltar que "j" representa a camada ou neurônio sobre os quais as operações estão sendo realizadas, "i" é o passado e "k" é o futuro. Ex.: $y_i(n)$ é a saída do neurônio da camada passada.

* A função custo, neste caso a MSE, é dada por:

$$\mathcal{E}(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$$

* Seja e_j o erro do neurônio j na camada de saída da rede dado por:

$$e_j(n) = d_j(n) - y_j(n)$$

* Função de ativação interna de um neurônio j:

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n)\, y_i(n)$$

* Saída do neurônio

$$y_j(n) = \varphi_j(v_j(n))$$

O objetivo é minimizar $\mathcal{E}(n)$ em função dos ganhos sinápticos (pesos) $w_{ji}(n)$.

Para isso, utiliza-se o método do gradiente:

* $w(n+1) = w(n) + \Delta w_{ji}(n) =$

* $w(n+1) = w(n) + \dfrac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)}$

Segundo a regra da cadeia:

* $\dfrac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \dfrac{\partial \mathcal{E}(n)}{\partial e_j(n)} \dfrac{\partial e_j(n)}{\partial y_j(n)} \dfrac{\partial y_j(n)}{\partial v_j(n)} \dfrac{\partial v_j(n)}{\partial w_{ji}(n)}$

* $\dfrac{\partial \mathcal{E}(n)}{\partial e_j(n)} = \dfrac{\partial}{\partial e_j(n)} \left[\dfrac{1}{2} \sum_{j \in C} e_j^2(n) \right] = e_j(n)$

* $\dfrac{\partial e_j(n)}{\partial y_j(n)} = \dfrac{\partial}{\partial y_j(n)} \left[d_j(n) - y_j(n) \right] = -1$

* $\dfrac{\partial y_j(n)}{\partial v_j(n)} = \dfrac{\partial \varphi_j(v_j(n))}{\partial v_j(n)} = \varphi_j'(v(n))$

* $\dfrac{\partial v_j(n)}{\partial w_{ji}(n)} = \dfrac{\partial}{\partial w_{ji}(n)} \left[\sum_{i=0}^{p} w_{ji}(n)\, y_i(n) \right] = y_i(n)$

Dessa modo,

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = -e_j(n) \varphi_j'(v_j(n)) y_i(n)$$

Temos que,

$$\Delta w_{ji}(n) = -\eta(n)\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \eta(n)\gamma_j(n)y_i(n)$$

$$\gamma_j(n) = -\frac{\partial \mathcal{E}(n)}{\partial v_j(n)} = -\frac{\partial \mathcal{E}(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)}$$

$\gamma_j(n)$ é definido como o gradiente local.

* Considerando agora que o neurônio j não está na camada de saída, ou seja não é sabido o valor esperado $d_j(n)$, sendo assim, o não é possível calcular diretamente o erro $e_j(n)$. Assim, o gradiente local é definido como:

$$\gamma_j(n) = -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial \mathcal{E}(n)}{\partial y_j(n)}\varphi'(v_j(n)) \quad (1)$$

Sendo,

$$\mathcal{E}(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n)$$

Com e_k sendo o erro do neurônio da camada seguinte, por isso o nome da abordagem: Backpropagation.

Logo,

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = \sum_{k\in C} e_k(n)\frac{\partial e_k(n)}{\partial y_j(n)} = \sum_{k\in C} e_k(n)\frac{\partial e_k(n)}{\partial v_k(n)}\frac{\partial v_k(n)}{\partial y_j(n)} \quad (2)$$

Onde,

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n))$$

Assim,

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v(n)) \quad (3)$$

Temos que,

$$\frac{\partial v_k(n)}{\partial y_j(n)} = \frac{\partial}{\partial y_j(n)}\left[\sum_{j=0}^{q} w_{kj}(n)\, y_j(n)\right] = w_{kj} \quad (4)$$

Substituindo (3) e (4) em (2):

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = -\sum_{k \in C} e_k(n)\, \varphi_k'(v(n))\, w_{kj}$$

Com,

$$\gamma_k(n) = e_k(n)\, \varphi_k'(v_k(n))$$

Então,

$$\frac{\partial \mathcal{E}(n)}{\partial y_j(n)} = -\sum_{k \in C} \gamma_k(n)\, w_{kj} \quad (5)$$

Sendo assim, substituindo (5) em (1), temos que,

$$\gamma_j(n) = \sum_{k \in C} \gamma_k(n)\, w_{kj}\, \varphi_j'(v_j(n))$$

Portanto,

$$\gamma_j(n) = \begin{cases} e_j(n)\, \varphi_j'(v_j(n)) \; ; \; j \in C \\ \sum_{k \in C} \gamma_k(n)\, w_{kj}\, \varphi_j'(v_j(n)) \; ; \; j \notin C \end{cases}$$

O ajuste dos pesos sinápticos é dado por:

$$w_{ji}(n+1) = w_{ji}(n) + \eta(n)\, \gamma_j(n)\, y_i(n), \text{ com } \eta(n) \text{ pequeno.}$$

(ii) Apresente o pseudo código do algoritmo

A seguir está o pseudocódigo apresentado pelo Goodfellow:

Após a computação direta (forward pass), calcule o gradiente na camada de saída:

$g \leftarrow \Delta_{\hat{y}} L(\hat{y}, y)$

for $k = l, l-1, \ldots, 1$ do:

- Convert the gradient on the layer's output into a gradient into the prenonlinearity activation (element-wise multiplication if f is element-wise):
- $g \leftarrow g \odot f'(a^{(k)})$
- Compute gradients on weights and biases (including the regularization term, where needed):
- $\Delta_{b^{(k)}} J = g + \lambda \Delta_{b^{(k)}} \Omega(\theta)$
- $\Delta_{W^{(k)}} J = g h^{(k-1)T} + \lambda \Delta_{W^{(k)}} \Omega(\theta)$
- Propagate the gradients w.r.t. the next lower-level hidden layer's activations:
- $g \leftarrow W^{(k)T} g$

end for

Pseudocódigo alternativo com notação do gradiente local:

```
Após a computação direta (forward pass), calcule o erro no neurônio de saída:

Para cada neurônio j na camada de saída:
    _j(n) ← e_j(n) · '_j(v_j(n))

Para cada camada oculta m = l1, l2, ..., 1:
    Para cada neurônio j na camada m:
        _j(n) ← _k _k(n) · w_kj(n) · '_j(v_j(n))

    Para cada peso w_ji(n) conectando o neurônio i da camada anterior ao neurônio j:
        w_ji(n+1) ← w_ji(n) + (n) · _j(n) · y_i(n)
```

(iii) Pesquise apresente um estudo da implementação computacional do algoritmo backpropagation para deep learning fazendo uso de tensores e computação gráfica.

Link para o GitHub com a solução proposta.

Resposta para a questão 5: Link para o GitHub com as soluções prospostas.

Trabalho abordando a aplicação de redes neurais do tipo perceptron de múltiplas camadas nos Transformers para a área de linguagem natural.

Link para o GitHub com o estudo de caso.

Link para o GitHub com a apresentação do estudo de caso.