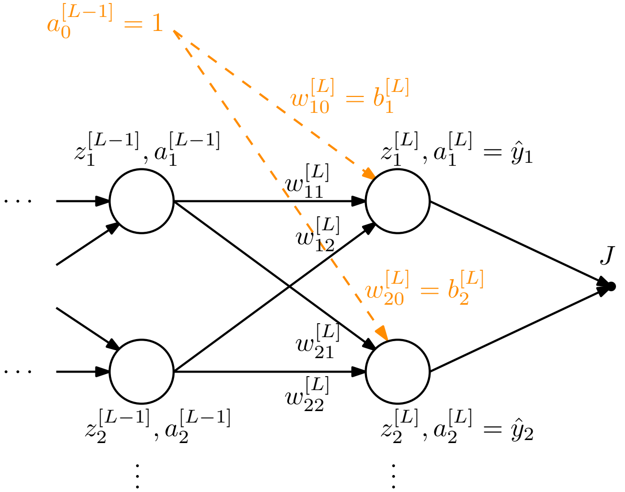
Redes Neurais II

Prof. Danilo Silva

EEL7514/EEL7513 - Tópico Avançado em Processamento de Sinais:
Introdução ao Aprendizado de Máquina

EEL / CTC / UFSC

Redes Neurais: Notação



Função Custo

- Função custo para uma única amostra $(\mathbf{x}, \mathbf{y})$, onde $\mathbf{x} = (x_1, \ldots, x_n)^T$ e $\mathbf{y} = (y_1, \ldots, y_K)^T$, como função dos parâmetros $\theta = \{w_{kj}^{[\ell]}\}$:

$$\begin{aligned}
J &= J(\theta) = J(\hat{y}_1, \ldots, \hat{y}_K, y_1, \ldots, y_K) \\
\hat{y}_k &= a_k^{[L]} = g_L(z_k^{[L]}) \\
z_k^{[\ell]} &= \sum_{j=0}^{n_{\ell-1}} w_{kj}^{[\ell]} a_j^{[\ell-1]} \quad (a_0^{[\ell-1]} = 1), \qquad \ell = L, \ldots, 1 \\
a_j^{[\ell]} &= g(z_j^{[\ell]}), \qquad \ell = L-1, \ldots, 1 \\
a_j^{[0]} &= x_j
\end{aligned}$$

Cálculo do Gradiente

- Cálculo do gradiente $\nabla J(\theta) = \left\{ \frac{\partial J}{\partial w_{kj}^{[\ell]}} \right\}$ via regra da cadeia:

$$\delta_k^{[L]} \triangleq \frac{\partial J}{\partial z_k^{[L]}} = \frac{\partial J}{\partial a_k^{[L]}} \cdot \frac{\partial a_k^{[L]}}{\partial z_k^{[L]}}$$

$$\frac{\partial J}{\partial w_{kj}^{[\ell]}} = \frac{\partial J}{\partial z_k^{[\ell]}} \cdot \frac{\partial z_k^{[\ell]}}{\partial w_{kj}^{[\ell]}} = \delta_k^{[\ell]} a_j^{[\ell-1]}, \qquad \ell = L, \ldots, 1$$

$$\begin{aligned}
\delta_j^{[\ell-1]} \triangleq \frac{\partial J}{\partial z_j^{[\ell-1]}} &= \sum_{k=1}^{n_\ell} \frac{\partial J}{\partial z_k[\ell]} \cdot \frac{\partial z_k^{[\ell]}}{\partial z_j^{[\ell-1]}} \\
&= \sum_{k=1}^{n_\ell} \frac{\partial J}{\partial z_k[\ell]} \cdot \frac{\partial z_k^{[\ell]}}{\partial a_j^{[\ell-1]}} \cdot \frac{\partial a_j^{[\ell-1]}}{\partial z_j^{[\ell-1]}} \\
&= \sum_{k=1}^{n_\ell} \delta_k^{[\ell]} w_{kj}^{[\ell]} g'(z_j^{[\ell-1]}), \qquad \ell = L, \ldots, 2
\end{aligned}$$

Cálculo do Gradiente: Casos Particulares

- Regressão: Ativação de saída linear com função custo de erro quadrático:

$$a_k^{[L]} = g_L(z_k^{[L]}) = z_k^{[L]}, \qquad J = \frac{1}{2}\sum_{k=1}^{K}(a_k^{[L]} - y_k)^2$$

- Classificação: Ativação de saída logística com função custo de entropia cruzada:

$$a_k^{[L]} = g_L(z_k^{[L]}) = \sigma(z_k^{[L]}), \quad J = -\sum_{k=1}^{K} y_k \log a_k^{[L]} + (1 - y_k)\log(1 - a_k^{[L]})$$

- Em ambos os casos, temos:

$$\delta_k^{[L]} = \frac{\partial J}{\partial z_k^{[L]}} = a_k^{[L]} - y_k$$

Algoritmo *Backpropagation*

- Entrada: $\{w_{kj}^{[\ell]}\}, \{z_j^{[\ell]}\}, \{a_j^{[\ell]}\}, \{y_j\}$
- Saída: $\left\{\frac{\partial J}{\partial w_{kj}^{[\ell]}}\right\}$
- Para $\ell = L, \ldots, 1$:

$$\delta_j^{[\ell]} = \begin{cases} a_j^{[L]} - y_j, & \ell = L \\ \left(\sum_{k=1}^{n_{\ell+1}} \delta_k^{[\ell+1]} w_{kj}^{[\ell+1]}\right) g'(z_j^{[\ell]}), & \ell < L \end{cases}$$

$$\frac{\partial J}{\partial w_{kj}^{[\ell]}} = \delta_k^{[\ell]} a_j^{[\ell-1]}$$

Algoritmo *Backpropagation*: Notação Vetorial

- Entrada: $\{\mathbf{w}_k^{[\ell]}\}, \{b_k^{[\ell]}\}, \{\mathbf{z}^{[\ell]}\}, \{\mathbf{a}^{[\ell]}\}, \{\mathbf{y}\}$
- Saída: $\left\{\frac{\partial J}{\partial \mathbf{w}_k^{[\ell]T}}\right\}, \left\{\frac{\partial J}{\partial b_k^{[\ell]}}\right\}$
- Para $\ell = L, \ldots, 1$:

$$\boldsymbol{\delta}^{[\ell]} = \begin{cases} \mathbf{a}^{[L]} - \mathbf{y}, & \ell = L \\ \left(\sum_{k=1}^{n_{\ell+1}} \delta_k^{[\ell+1]} \mathbf{w}_k^{[\ell+1]}\right) \odot g'(\mathbf{z}^{[\ell]}), & \ell < L \end{cases}$$

$$\frac{\partial J}{\partial \mathbf{w}_k^{[\ell]T}} = \delta_k^{[\ell]} \mathbf{a}^{[\ell-1]T}, \qquad \frac{\partial J}{\partial b_k^{[\ell]}} = \delta_k^{[\ell]}$$

- Obs: $\odot$ = multiplicação elemento a elemento,

$$\mathbf{w}_k^{[\ell]} \triangleq (w_{k,1}^{[\ell]}, \ldots, w_{k,n_{\ell-1}}^{[\ell]})^T, \qquad b_k^{[\ell]} \triangleq w_{k,0}^{[\ell]}$$

Algoritmo *Backpropagation*: Notação Matricial

- Entrada: $\{\mathbf{W}^{[\ell]}\}, \{\mathbf{b}^{[\ell]}\}, \{\mathbf{z}^{[\ell]}\}, \{\mathbf{a}^{[\ell]}\}, \{\mathbf{y}\}$
- Saída: $\left\{\frac{\partial J}{\partial \mathbf{W}^{[\ell]}}\right\}, \left\{\frac{\partial J}{\partial \mathbf{b}^{[\ell]}}\right\}$
- Para $\ell = L, \dots, 1$:

$$\boldsymbol{\delta}^{[\ell]} = \begin{cases} \mathbf{a}^{[L]} - \mathbf{y}, & \ell = L \\ \left(\mathbf{W}^{[\ell+1]^T} \boldsymbol{\delta}^{[\ell+1]}\right) \odot g'(\mathbf{z}^{[\ell]}), & \ell < L \end{cases}$$

$$\frac{\partial J}{\partial \mathbf{W}^{[\ell]}} = \boldsymbol{\delta}^{[\ell]} \mathbf{a}^{[\ell-1]^T}, \qquad \frac{\partial J}{\partial \mathbf{b}^{[\ell]}} = \boldsymbol{\delta}^{[\ell]}$$

- Obs: $\odot$ = multiplicação elemento a elemento,

$$\mathbf{W}^{[\ell]} \triangleq \begin{bmatrix} \mathbf{w}_1^{[\ell]^T} \\ \vdots \\ \mathbf{w}_{n_\ell}^{[\ell]^T} \end{bmatrix}, \qquad \mathbf{b}^{[\ell]} \triangleq \begin{bmatrix} b_1^{[\ell]} \\ \vdots \\ b_{n_\ell}^{[\ell]} \end{bmatrix}$$

Custo Médio sobre um Conjunto de Treinamento

- Conjunto de treinamento: $\mathcal{D} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m}$
- Função custo:

$$J(\theta) = J_{\text{train}}(\theta) = \frac{1}{m} \sum_{i=1}^{m} J^{(i)}(\theta) = \frac{1}{m} \sum_{i=1}^{m} J(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$$

- Cálculo do gradiente:

$$\frac{\partial J}{\partial \mathbf{W}^{[\ell]}} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{\delta}^{[\ell](i)} \mathbf{a}^{[\ell-1](i)^T}, \qquad \frac{\partial J}{\partial \mathbf{b}^{[\ell]}} = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{\delta}^{[\ell](i)}$$

Vetorizando ao longo das amostras de treinamento

- Concatenando horizontalmente vetores-coluna:

$$\mathbf{A}^{[0]} = \mathbf{X}^T = \begin{bmatrix} \mathbf{x}^{(1)} & \cdots & \mathbf{x}^{(m)} \end{bmatrix} \qquad \mathbf{Y}^T = \begin{bmatrix} \mathbf{y}^{(1)} & \cdots & \mathbf{y}^{(m)} \end{bmatrix}$$

$$\mathbf{A}^{[\ell]} = \begin{bmatrix} \mathbf{a}^{[\ell](1)} & \cdots & \mathbf{a}^{[\ell](m)} \end{bmatrix} \quad \mathbf{Z}^{[\ell]} = \begin{bmatrix} \mathbf{z}^{[\ell](1)} & \cdots & \mathbf{z}^{[\ell](m)} \end{bmatrix}$$

$$\boldsymbol{\Delta}^{[\ell]} = \begin{bmatrix} \boldsymbol{\delta}^{[\ell](1)} & \cdots & \boldsymbol{\delta}^{[\ell](m)} \end{bmatrix}$$

- Propagação direta:

$$\mathbf{Z}^{[\ell]} = \mathbf{W}^{[\ell]}\mathbf{A}^{[\ell-1]} + \mathbf{b}^{[\ell]}, \qquad \mathbf{A}^{[\ell]} = g(\mathbf{Z}^{[\ell]})$$

- Propagação reversa:

$$\boldsymbol{\Delta}^{[\ell]} = \begin{cases} \mathbf{A}^{[L]} - \mathbf{Y}^T, & \ell = L \\ \left(\mathbf{W}^{[\ell+1]^T}\boldsymbol{\Delta}^{[\ell+1]}\right) \odot g'(\mathbf{Z}^{[\ell]}), & \ell < L \end{cases}$$

$$\frac{\partial J}{\partial \mathbf{W}^{[\ell]}} = \frac{1}{m}\boldsymbol{\Delta}^{[\ell]}\mathbf{A}^{[\ell-1]^T}, \qquad \frac{\partial J}{\partial \mathbf{b}^{[\ell]}} = \frac{1}{m}\boldsymbol{\Delta}^{[\ell]}\mathbf{1}_{(m\times 1)}$$

Regularização

- Ajuda a prevenir overfitting, o que é essencial à medida que aumentamos a capacidade do modelo (número de parâmetros)
- Mesmas equações da regressão linear e logística, uma vez que apenas a função custo é modificada
- Função custo (regularização ℓ_2):

$$J(\theta) = J_{\text{train}}(\theta) + \frac{\lambda}{2m} \sum_{\ell=1}^{L} \sum_{k=1}^{n_\ell} \sum_{j=1}^{n_{\ell-1}} (w_{kj}^{[\ell]})^2$$

- Cálculo do gradiente (regularização ℓ_2):

$$\frac{\partial J}{\partial \mathbf{W}^{[\ell]}} = \frac{1}{m} \mathbf{\Delta}^{[\ell]} \mathbf{A}^{[\ell-1]^T} + \frac{\lambda}{m} \mathbf{W}^{[\ell]}, \qquad \frac{\partial J}{\partial \mathbf{b}^{[\ell]}} = \frac{1}{m} \mathbf{\Delta}^{[\ell]} \mathbf{1}_{(m \times 1)}$$

  - Lembre-se que o termo de bias não é regularizado

Dica de implementação: Checagem de gradiente

- Uma maneira de confirmar se sua implementação do cálculo do gradiente está correta é comparar o gradiente calculado com uma aproximação numérica baseada na função custo
- Aproximação numérica do gradiente (por diferenças finitas):

$$\frac{\partial J(\theta)}{\partial \theta_j} \approx \frac{J(\theta + \epsilon \mathbf{1}_j) - J(\theta - \epsilon \mathbf{1}_j)}{2\epsilon}$$

  onde θ_j representa qualquer parâmetro $w_{kj}^{[\ell]}$ ou $b_k^{[\ell]}$, e $\mathbf{1}_j$ denota um vetor com valor 1 na j-ésima posição e 0 nas demais
- Vale a pena testar para uma rede pequena
- Após a confirmação a checagem deve ser desabilitada pois torna a implementação significativamente mais lenta

Eficiência Computacional

- Seja N a dimensão do vetor θ (i.e., o número total de parâmetros)
- Descrever a superfície de custo (por exemplo, numa aproximação quadrática) requer pelo menos $O(N^2)$ valores
- Sem usar propagação reversa, seriam necessárias $O(N^3)$ operações para encontrar o mínimo da função:
  - Cada avaliação da função custo requer $O(N)$ operações e retorna apenas 1 valor
  - O gradiente fornece N valores, mas aproximá-lo por diferenças finitas requer $O(N^2)$ operações ($O(N)$ para cada parâmetro)
  - Calcular o gradiente analiticamente, mas separadamente para cada parâmetro, também exigiria $O(N^2)$ operações
- Usando propagação reversa, o gradiente é calculado com $O(N)$ operações, resultando em um total de $O(N^2)$ operações para encontrar o mínimo da função

Inicialização de pesos

- Embora os pesos da camada de saída e todos os termos de bias possam ser inicializados com zeros, os demais pesos devem ser inicializados com valores distintos para quebrar a simetria existente entre as unidades ocultas
  - Caso contrário, as unidades aprenderão os mesmos pesos, tornando-se idênticas
  - Recomenda-se utilizar uma inicialização aleatória, como uniforme ou gaussiana
- Por outro lado, os valores não podem ser muito altos, caso contrário tenderão a causar uma saturação da função de ativação (logística), o que por sua vez resulta em um aprendizado muito lento
  - Pequenas variações nos parâmetros daquela unidade praticamente não terão impacto no custo final
- Múltiplas reinicializações podem ser necessárias, uma vez que a função custo é não-convexa

Inicialização de pesos

- Recomendações para inicialização de $w_{kj}^{[\ell]}$:
  - Distribuição gaussiana $\mathcal{N}(0, \sigma^2)$, onde $\sigma^2 = 1/n_{\ell-1}$
  - Distribuição uniforme entre $[-\epsilon, \epsilon]$, onde $\epsilon = \sqrt{1/n_{\ell-1}}$
  - [Para ativação tanh] Distribuição uniforme entre $[-\epsilon, \epsilon]$, onde

$$\epsilon = \sqrt{6/(n_{\ell-1} + n_\ell)}$$

  - [Para ativação logística] Distribuição uniforme entre $[-\epsilon, \epsilon]$, onde

$$\epsilon = 4\sqrt{6/(n_{\ell-1} + n_\ell)}$$

- Os biases $b_k^{[\ell]}$ podem ser inicializados com zeros ou como $\mathcal{N}(0, 1)$
- Obs: valores diferentes podem ser necessários dependendo do problema

Otimização em mini-batches

- Cada iteração do método de otimização (cálculo da função custo e gradiente) é realizada usando um subconjunto (*mini-batch*) de $B < m$ amostras de treinamento
  - Função custo efetivamente muda a cada iteração
  - Amostragem sem reposição, i.e., total de m/B *mini-batches*
- Cada passagem por todo o conjunto de treinamento (m/B iterações) é chamada de época
- Para $B > 1$, generaliza o método do gradiente estocástico (SGD)
- Vantagens:
  - Menor custo computacional (em particular, insensível a redundância nos dados)
  - Maior capacidade de escapar de mínimos locais

Função de ativação Softmax

- Função softmax:

$$g(z_k) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

  - Generaliza a função logística para $K > 2$
- Tipicamente usada em conjunto com a função custo de entropia cruzada categórica (K-ária):

$$J(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{k=1}^{K} y_k \log \hat{y}_k$$

- Nesse caso, temos também:

$$\delta_k^{[L]} = \frac{\partial J}{\partial z_k^{[L]}} = a_k^{[L]} - y_k$$

  - Mesmas equações para cálculo do gradiente

Outras funções de ativação

- Tanh
- ReLU