University of Amsterdam
Amsterdam School of Economics

Machine Learning for Econometrics, 2022

Tutorial/Lecture 3: Backpropagation

Yi He

November 15, 2022

Week 2: Gradient Descent

Starting with an initial value $(w(0), b(0))$, the batch gradient descent algorithm updates the parameters for each step t:

$$\begin{bmatrix} w(t+1) \\ b(t+1) \end{bmatrix} = \begin{bmatrix} w(t) \\ b(t) \end{bmatrix} - \eta \cdot g(w(t), b(t)), \quad \eta > 0,$$

where g is the gradient function given by

$$\begin{aligned} & g(w, b) \\ = & \begin{bmatrix} \nabla_w (L_S(w, b) + \frac{\lambda}{2} \|w\|^2) \\ \nabla_b (L_S(w, b) + \frac{\lambda}{2} \|w\|^2) \end{bmatrix} \\ = & \frac{1}{n} \sum_{i=1}^{n} \begin{bmatrix} \nabla_w \ell(f(X_i; w, b), Y_i) \\ \nabla_b \ell(f(X_i; w, b), Y_i) \end{bmatrix} + \lambda \begin{bmatrix} w \\ \mathbf{0} \end{bmatrix}. \end{aligned}$$

How to compute $\nabla_w \ell(f(X_i; w, b), Y_i)$ and $\nabla_b \ell(f(X_i; w, b), Y_i)$?

Shallow Neural Network: $D = 2$

Consider the shallow neural networks

$$f(x) = \boldsymbol{\sigma}(a^{(2)}(x)),\ a^{(2)}(x) = W^{(2)} z^{(1)}(x) + b^{(2)},\ W^{(1)} \in \mathbb{R}^{K \times M}$$

$$z^{(1)}(x) = \sigma(a^{(1)}(x)), \quad a^{(1)}(x) = W^{(1)} \underbrace{z^{(0)}(x)}_{x} + b^{(1)}, \quad W^{(2)} \in \mathbb{R}^{M \times d},$$

where

- for classification tasks

$$\boldsymbol{\sigma}(a) = \text{Softmax}(a),$$

$$\ell(f(X_i), Y_i) = \ell_{\text{CE}}(f(X_i), Y_i^{\text{OH}}) = -\sum_{k=1}^{K} Y_i^{(k)} \log f_k(X_i)$$

- for regression tasks

$$\boldsymbol{\sigma}(a) = a, \quad \ell(f(X_i), Y_i) = \ell_2(f(X_i), Y_i) = \frac{1}{2} \sum_{k=1}^{K} (f_k(X_i) - Y_i^{(k)})^2$$

Unified Expression for Regression and Classification Tasks

Denote by $a_{k,i}^{(2)} = a_k^{(2)}(X_i)$ and $\boldsymbol{a}_i^{(2)} = (a_1^{(2)}(X_i), \ldots, a_K^{(2)}(X_i))^T$.

For classification tasks [Tute Q1]

$$\frac{\partial}{\partial a_{k,i}^{(2)}} \ell(f(X_i), Y_i) = \frac{\partial}{\partial a_{k,i}^{(2)}} \ell_{\text{CE}}(\text{Softmax}(\boldsymbol{a}_i^{(2)}), Y_i^{\text{OH}}) = f_k(X_i) - Y_i^{(k)}$$

For regression tasks

$$\frac{\partial}{\partial a_{k,i}^{(2)}} \ell(f(X_i), Y_i) = \frac{\partial}{\partial a_{k,i}^{(2)}} \ell_2(\boldsymbol{a}_i^{(2)}, Y_i) = f_k(X_i) - Y_i^{(k)}$$

Let $\delta_{k,i}^{(2)} = f_k(X_i) - Y_i^{(k)}$.

Backpropagation: The Second Hidden Layer

Let $W_k^{(2)}$ denote the k-th row of $W^{(2)}$. The chain rule gives that

$$\frac{\partial}{\partial W_k^{(2)}} \ell(f(X_i), Y_i) = \left(\underbrace{\frac{\partial}{\partial W_k^{(2)}} a_{k,i}^{(2)}}_{\text{no } \ell \text{ nor } \sigma} \right) \left(\frac{\partial}{\partial a_{k,i}^{(2)}} \ell(f(X_i), Y_i) \right).$$

where

$$a_{k,i}^{(2)} = W_k^{(2)} z_i^{(1)} + b_k^{(2)}, \quad z_i^{(1)} = z^{(1)}(X_i).$$

Then

$$\frac{\partial}{\partial W_k^{(2)}} \ell(f(X_i), Y_i) = \delta_{k,i}^{(2)} z_i^{(1)} \propto \underbrace{z_i^{(1)}}_{\text{input to the layer}}$$

Similarly, even easier

$$\frac{\partial}{\partial b_k^{(2)}} \ell(f(X_i), Y_i) = \delta_{k,i}^{(2)} \cdot 1$$

Backpropagation: The First Hidden Layer

Similarly, the chain rule gives that

$$\frac{\partial}{\partial W_m^{(1)}} \ell(f(X_i), Y_i) = \left(\underbrace{\frac{\partial}{\partial W_m^{(1)}} a_{m,i}^{(1)}}_{= z_i^{(0)} = X_i \text{ (input)}} \right) \left(\frac{\partial}{\partial a_{m,i}^{(1)}} \ell(f(X_i), Y_i) \right).$$

Recall the feedforward structure:

$$a_{m,i}^{(1)} \to \sigma(a_{m,i}^{(1)}) =: z_{m,i}^{(1)} \overset{W^{(2)}}{\to} \{a_{k,i}^{(2)} : k = 1, \ldots, K\} \to \ell(f(X_i), Y_i)$$

$$\begin{aligned} \delta_{m,i}^{(1)} := \frac{\partial}{\partial a_{m,i}^{(1)}} \ell(f(X_i), Y_i) &= \sum_{k=1}^{K} \frac{\partial \ell(f(X_i), Y_i)}{\partial a_{k,i}^{(2)}} \frac{\partial a_{k,i}^{(2)}}{\partial z_{m,i}^{(1)}} \frac{\partial z_{m,i}^{(1)}}{\partial a_{m,i}^{(1)}} \\ &= \sum_{k=1}^{K} \delta_{k,i}^{(2)} \cdot w_{k,m}^{(2)} \cdot \sigma'(a_{m,i}^{(1)}) \end{aligned}$$

Backpropagation Formula: Matrix Form

The last equation is known as the backpropagation formula

$$\boldsymbol{\delta}_i^{(1)} = \underbrace{\sigma'(\boldsymbol{a}_i^{(1)})}_{M\times 1} \circ \underbrace{(W^{(2)})^T \boldsymbol{\delta}_i^{(2)}}_{(M\times K)(K\times 1)=(M\times 1)}$$

where

$$\boldsymbol{\delta}_i^{(1)} = \begin{pmatrix} \delta_{1,i}^{(1)} \\ \vdots \\ \delta_{M,i}^{(1)} \end{pmatrix}, \ \boldsymbol{a}_i^{(1)} = \begin{pmatrix} a_{1,i}^{(1)} \\ \vdots \\ a_{M,i}^{(1)} \end{pmatrix}, \ \boldsymbol{\delta}_i^{(2)} = \begin{pmatrix} \delta_{1,i}^{(2)} \\ \vdots \\ \delta_{K,i}^{(2)}, \end{pmatrix}$$

and

- derivative function σ' is applied *element-wisely*
- $\circ$ denotes *element-wise* multiplication (Hadamard product)

Backpropagation Formula: More Hidden Layers

Start with $\boldsymbol{\delta}_i^{(D)} = f(X_i) - \widetilde{Y}_i$ where $\widetilde{Y}_i = Y_i^{\text{OH}}$ for classification and $\widetilde{Y}_i = Y_i$ for regression tasks.

Backpropagate

$$\boldsymbol{\delta}_i^{(D)} \to \boldsymbol{\delta}_i^{(D-1)} \to \ldots \to \boldsymbol{\delta}_i^{(1)}$$

using the formula

$$\boldsymbol{\delta}_i^{(h-1)} = \sigma'(\boldsymbol{a}_i^{(h-1)}) \circ (W^{(h)})^T \boldsymbol{\delta}_i^{(h)}, \quad h = D, D-1, \ldots, 2$$

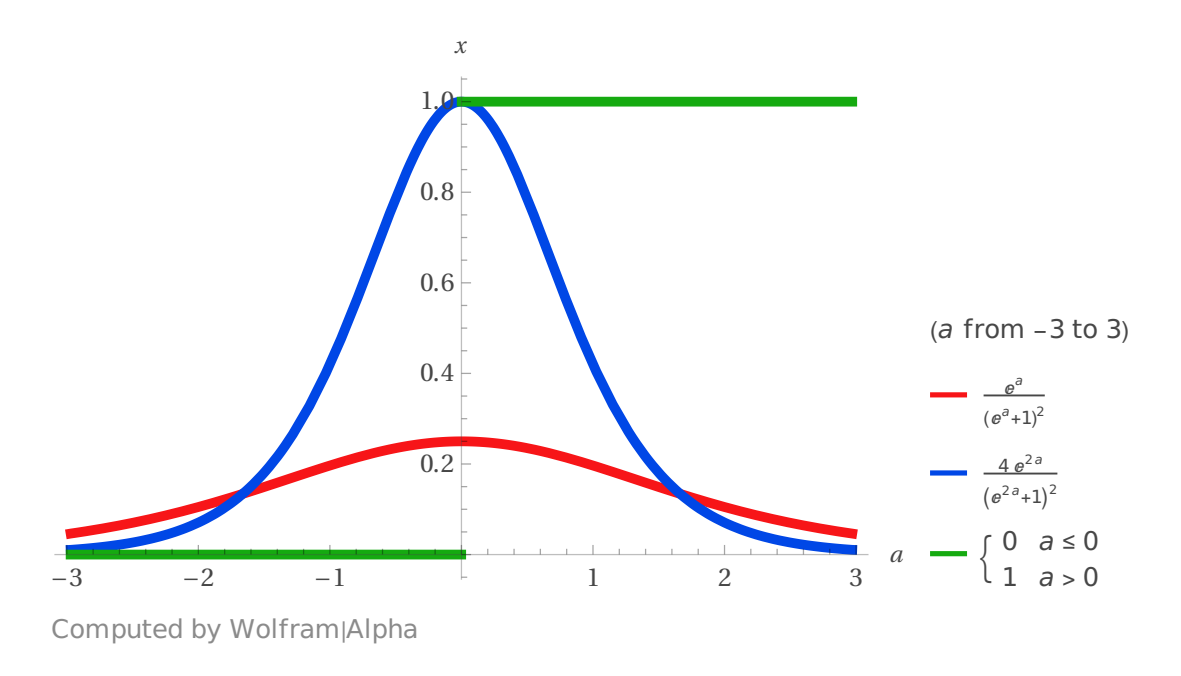
Then by the chain rule,

$$\frac{\partial}{\partial W_j^{(h)}} \ell(f(X_i), Y_i) = \left(\frac{\partial}{\partial W_j^{(h)}} a_{j,i}^{(h)} \right) \left(\frac{\partial}{\partial a_{j,i}^{(h)}} \ell(f(X_i), Y_i) \right) = \boldsymbol{z}_i^{(h-1)} \delta_{j,i}^{(h)}$$

$$\frac{\partial}{\partial b_j^{(h)}} \ell(f(X_i), Y_i) = \left(\frac{\partial}{\partial b_j^{(h)}} a_{j,i}^{(h)} \right) \left(\frac{\partial}{\partial a_{j,i}^{(h)}} \ell(f(X_i), Y_i) \right) = \delta_{j,i}^{(h)}$$

On The Derivative of Activation Function

- $\sigma(a)$ involves in the gradient via its derivative $\sigma'(a)$
- ReLU$'(0)$ is undefined: use the weak derivative 0
- ReLU$'$ is a binary function: saves computational costs+ 'shuts down' some units in backpropagation



Red=Sigmoid, Blue=tanh, Green=ReLU

Guest Lecturers From AethiQs

- Friday 13-15, REC C0.01

Mark Verschuren

Max van den Hoven