Feed Forward and Backprop

Aziz Temirkhanov

LAMBDA lab
Faculty of Computer Science
Higher School of Economics

February 26, 2024

Outline

1 Introduction

Introduction

Empirical Risk Minimization

Usually, when solving ML problems, one is seeking for solution to ERM
ERM:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} = \mathcal{L}(\theta; x, y) \tag{1}$$

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(g_\theta(x_i), y_i) \tag{2}$$

Networks

Assume, we have:

- $x \in \mathbb{R}^d$ - data features
- $y \in \mathbb{R}$ - target
- $\theta \in \Theta$ or $w \in W$ - network parameters
- $f_\theta(x) = x_1 \times \theta_1 + x_2 \times \theta_2 + ... + x_n \times \theta_n$ - a network parameterized by θ
- $\{(x_i, y_i)\}_{i=1}^{\ell}$ - training set
- $\mathcal{L}(\hat{y}, y)$ - loss function

To minimize empirical risk one can use gradient descent (usually, its stochastic version). Thus, the loss function (and f itself, of course) should be differentiable

Backpropagation

Q: How to compute $\nabla(L)$?

A: Use chain rule!

- $y = f(g(x)) \implies \frac{dy}{dx} = \frac{df}{dg} \times \frac{dg}{dx}$
- $y = f(g_1(x), g_2(x), ..., g_n(x)) \implies$

$$\frac{dy}{dx} = \sum_{i=1}^{n} \frac{df}{dg_i} \times \frac{dg_i}{dx}$$

Fully-Connected Netwroks

Now, let:

$x \in \mathbb{R}^{d_0}, \qquad f_\theta(x) = \langle x, \theta \rangle = \theta^T \times x, \theta \in \mathbb{R}^{d_0}$

Define: $\{z_i\}_{i=1}^k$ **- logit, where** k **is a number of layers**

- $z_1 = \theta_1 x, \qquad \theta_1 \in \mathbb{R}^{d_0 \times d_1}$
- $z_2 = \theta_2 z_1, \qquad \theta_2 \in \mathbb{R}^{d_0 \times d_2}$
- $z_n = \theta_n z_{n-1}, \qquad \theta_n \in \mathbb{R}^{d_0 \times d_n}$

$$z_n = \theta_n \theta_{n-1} ... \theta_2 \theta_1 x, \theta \in \mathbb{R}^{d_{n0}},$$

– still a linear function

Fully-Connected Networks

Let's add non-linearity!

activation function

$\sigma(\cdot) : \mathbb{R} \mapsto \mathbb{R}$

Now, one can re-define z_i as:

$$z_i = \sigma(z_{i-1}\theta_i + b_i)$$

$b_i \in \mathbb{R}^{d^i}$ - a bias vector to turn linear transform to affine

Fully-Connected Networks

To sum up:

linear layer

$f(x;\theta, b) = \theta x + b$ or $f(x; W, b) = Wx + b$

hidden(latent) representation or logit

$z_i = (z_i^1, ..., z_i^{d_i})$

non-linearity

$\sigma(\cdot) = \sigma(z_i)$

Activation functions

Sigmoid

$\sigma(x) = \frac{1}{1+\exp^{-x}}$ $\sigma'(x) = \sigma(x)(1 - \sigma(x))$

tanh

$tanh(x) = \frac{e^x - e^{-x}}{x^x + e^{-x}}$ $tanh'(x) = 1 - tanh^2(x)$

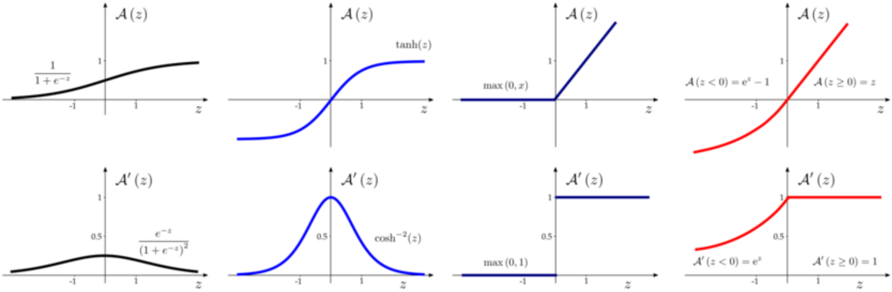
ReLU

$ReLU(x) = \max(0, x)$

Leaky ReLU

$LeakyReLU(x) = \max(\alpha x, 0)$

Activation functions



Classification

Consider a multi-class classification problem:

$$\{(x_i, y_i)\}_{i=1}^{\ell}, \quad y_i \in \{1, ..., C\}, \text{where } C \text{ is the number of classes}$$

Then, solve this optimization problem:

$$\min_{\theta} \mathcal{L}(f(\theta; x_i), y_i) = \frac{1}{n} \sum_{i=1}^{n} [f(\theta; x_i) \neq y_i] \tag{3}$$

Softmax

The derivative of indicator function either does not exist or equals to zero

$$\mathrm{p} = \begin{pmatrix} p(\hat{y}_i = 1) \\ p(\hat{y}_i = 2) \\ \vdots \\ p(\hat{y}_i = C) \end{pmatrix} \quad (4)$$

$$z_i = \sigma(\theta_i z_{i-1} + b_i) \tag{5}$$

$$z_n = \theta_n \times z_{n-1} + b_n, z_n \in \mathbb{R}^{d_n} \tag{6}$$

$$p = \{p_i\}_{i=1}^{C}, \quad p_i \leq 0, \quad \sum_{i=1}^{C} p_i = 1 \tag{7}$$

Softmax

$$p(y = k) = p_k = softmax(z)_k = \frac{\exp z_k}{\sum_{j=1}^{C} \exp z_j} \tag{8}$$

Let us now use the **maximum likelihood estimation** to train the network:

$$-\sum_{k=1}^{C} [k = y] \log p_k = -\log p_y \mapsto \min \tag{9}$$

$$L = -\frac{1}{l} \sum_{i=1}^{l} \sum_{k=1}^{C} [y_i = k] \log p_k^{(i)} \mapsto \min \tag{10}$$

Also, $-\log p_y$ is called Negative Log Likelihood, and that is a special case of Cross Entropy Loss

Log Softmax

$$logsoftmax(z_k) = \log \frac{\exp z_k}{\sum_{j=1}^{C} \exp z_j} = \log \exp z_k - \log \sum_{j=1}^{C} \exp z_j \tag{11}$$

$$logsoftmax(z_k) = z_k - \log \sum_{j=1}^{C} \exp z_j \tag{12}$$

$$logsoftmax(z_k) = z_k - \max_m z_m - \log \sum_{j=1}^{C} \exp z_j - \max_m z_m \tag{13}$$

Cross-Entropy

$$CrossEntropyLoss(z, y) = -\sum_{i=1}^{C} q_i \log p_i, \tag{14}$$

where $q_i = [i = y]$


Thank you