ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Models, Algorithms and Data (MAD): Introduction to Computing

Spring semester 2021

Prof. Dr. Jens Honore Walther
Dr. Georgios Arampatzis
ETH Zentrum, CLT
CH-8092 Zürich

Solution Set 10

Issued: 21.05.2021

Question 1: Shedding light in Backpropagation

a)

$$\frac{\partial a_j^k}{\partial W_{ij}^k} = \frac{\partial}{\partial W_{ij}^k}\left(\sum_{i'=1}^{n_{k-1}} W_{i'j}^k \, z_{i'}^{k-1}\right) = z_i^{k-1} \tag{1}$$

b)

$$\frac{\partial a_i^{k+1}}{\partial a_j^k} = \frac{\partial}{\partial a_j^k}\left(\sum_{j'=1}^{n_k} W_{j'i}^{k+1} z_{j'}^k\right) = \frac{\partial}{\partial a_j^k}\left(\sum_{j'=1}^{n_k} W_{j'i}^{k+1} \varphi(a_{j'}^k)\right) = W_{ji}^{k+1} \, \varphi'(a_j^k) \tag{2}$$

hence the error gradient becomes

$$\delta_j^k = \sum_{i=1}^{h_{k+1}} \delta_i^{k+1} \frac{\partial a_i^{k+1}}{\partial a_j^k} = \sum_{i=1}^{n_{k+1}} W_{ji}^{k+1} \, \delta_i^{k+1} \varphi'(a_j^k) = \varphi'(a_j^k) \sum_{i=1}^{n_{k+1}} W_{ji}^{k+1} \, \delta_i^{k+1} \tag{3}$$

c) Simply replacing the previously found terms yields:

$$\frac{\partial E_n}{\partial W_{ij}^k} = \frac{\partial E_n}{\partial a_j^k} \frac{\partial a_j^k}{\partial W_{ij}^k} = \delta_j^k \frac{\partial a_j^k}{\partial W_{ij}^k} = \delta_j^k z_i^{k-1} = z_i^{k-1} \varphi'(a_j^k) \sum_{i=1}^{n_{k+1}} W_{ji}^{k+1} \, \delta_i^{k+1} \tag{4}$$

The update equation of the gradient descent optimizer takes the following form:

$$\Delta W_{ij}^k = -\eta \frac{1}{N} \sum_{n=1}^{N} \frac{\partial E_n}{\partial W_{ij}^k}, \tag{5}$$

where η is called the step-size parameter or *learning rate*. In neural networks, the objective function is often non-convex. As a consequence, gradient descent does not guarantee convergence to a global minimum, thus more sophisticated techniques, such as stochastic optimization, adaptive learning rates, exploiting momentum, etc. are used to accelerate the training procedure and converge to a better solution.

d) The Logistic or Sigmoid function $f(x) = \frac{1}{1+e^{-x}}$ satisfies this relationship. It is commonly used in practice due to its simple derivative form.

e) It is possible that the error gradients become very small or near-zero as we increase the number of layers, due to the activation functions saturating. This might be the case when using **sigmoid** or **tanh** activation functions. These activation functions have a limited image set $\sigma(\cdot) \in [0, 1]$ or $\tanh(\cdot) \in [-1, 1]$ and the gradients close to the boundary of these sets are reduced to (near-) zero. In backpropagation, the effect is multiplied as we increase the number of layers since we propagate the error gradients many more layers back, slowing down the training. Using rectifier linear unit activation functions (ReLU), whose image domain is $[0, \infty]$ might be a good option in such cases, as the gradient does not saturate. Monitoring the training procedure and checking the values of the gradients is important for successfully training a neural network.