ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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

Spring semester 2019

Prof. Dr. Jens Honore Walther
Dr. Julija Zavadlav
ETH Zentrum, CLT
CH-8092 Zürich

Solution Set 7

Issued: 15.04.2019

Question 1: Shedding light in Backpropagation

a)

$$\frac{\partial z_j^k}{\partial w_{ij}^k} = \frac{\partial}{\partial w_{ij}^k}\left(\sum_{i'=1}^{h_{k-1}} w_{i'j}^k\, o_{i'}^{k-1}\right) = o_i^{k-1} \tag{1}$$

b)

$$\frac{\partial z_i^{k+1}}{\partial z_j^k} = \frac{\partial}{\partial z_j^k}\left(\sum_{j'=1}^{h_k} w_{j'i}^{k+1} o_{j'}^k\right) = \frac{\partial}{\partial z_j^k}\left(\sum_{j'=1}^{h_k} w_{j'i}^{k+1}\sigma(z_{j'}^k)\right) = w_{ji}^{k+1}\,\sigma'(z_j^k) \tag{2}$$

c)

$$\delta_j^k = \sum_{i=1}^{h_{k+1}} \delta_i^{k+1}\frac{\partial z_i^{k+1}}{\partial z_j^k} = \sum_{i=1}^{h_{k+1}} w_{ji}^{k+1}\,\delta_i^{k+1}\sigma'(z_j^k) = \sigma'(z_j^k)\sum_{i=1}^{h_{k+1}} w_{ji}^{k+1}\,\delta_i^{k+1} \tag{3}$$

d)

$$\frac{\partial E_s}{\partial w_{ij}^k} = \frac{\partial E_s}{\partial z_j^k}\frac{\partial z_j^k}{\partial w_{ij}^k} = \delta_j^k\frac{\partial z_j^k}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1} = o_i^{k-1}\sigma'(z_j^k)\sum_{i=1}^{h_{k+1}} w_{ji}^{k+1}\,\delta_i^{k+1} \tag{4}$$

e) The update equation of the Gradient Descent optimizer takes the following form

$$\Delta w_{ij}^k = -\eta\frac{1}{N}\sum_{s=1}^{N}\frac{\partial E_s}{\partial w_{ij}^k}, \tag{5}$$

where η is called the step-size parameter or the learning rate. In neural networks, the objective is not convex with respect to the weights. As a consequence, Gradient Descent does not guarantee convergence to a global optimum. More sophisticated techniques, like stochastic optimization, adaptive learning rates, exploiting momentum, etc. can accelerate the training procedure and converge to better solutions.

f) 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.

g) It is possible that the error gradients become very small or close to zero as we increase the number of layers, due to the activation functions saturating. This might be the case when using **tanh** or **sigmoid** activation functions. These activation functions have a limited Image set $\sigma(\cdot) \in [0, 1]$ or $[-1, 1]$ and the gradients close to the boundary of these sets are

converging to zero. In Backpropagation the effect is multiplied as we increase the number of layers since we propagate the error gradients many more layers back. This slows down the training. Using rectifier linear unit activation functions, whose domain is $[0, \infty]$ and the gradient does not saturate, might be a good option in such a case. Supervising the training procedure and checking the values of the gradients at each epoch/batch, is important for successfully training a neural network.

Question 2: Regression with `TensorFlow`

a) Solution can be found in the script `nn_sinus_solution.py`

b) Without noise, the sinus signal can be recovered with low MSE error. The gradient descent optimizer is sensitive to the learning rate, which has to be tuned.

c) Increasing the data noise increases the difficulty of the task. Still, for small noise levels, the signal can be recovered.

d) Neural networks are also applied in time series. Assuming we have time series data describing the evolution of a signal in time, we can train a neural network to use the value of the signal at time t to predict its value at time instant $t+\Delta t$. The accuracy might be increased in case of longer-term correlations and data dependencies, if we do not only use the signal value at time t but rather values from a window $\{t, t-\Delta t, \ldots, t-d\Delta t\}$ to infer the signal value at time step $t + \Delta t$.