Artificial Neural Networks and Deep Learning Summer 2018
Christian Borgelt and Christoph Doell from 2018.06.26

Exercise Sheet 8

Exercise 31 Momentum Term

We consider the gradient descent variant that uses a momentum term to accelerate the training, that is, we consider weight changes according to $\Delta w_t = -\frac{\eta}{2} \nabla_w e|_{w_t} + \alpha \, \Delta w_{t-1}$, $t = 0, 1, 2, 3, \ldots$. Suppose this training rule is applied for an error function that exhibits an infinitely extended constant slope. Do the weight changes increase without limit? If not, what is the limit for the step width?

Exercise 32 Deep Learning: Autoencoder

As discussed in the lecture, an autoencoder is a 3-layer perceptron with as many output neurons as input neurons, which is supposed to map its inputs to (reconstructions of) its inputs. We assume that in the hidden layer as well as in the output layer the activation function is a rectified maximum (or ramp function) and that neither dropout training nor a restriction of the number of active neurons is used. In this case: why is it inappropriate to use as many neurons in the hidden layer as there are neurons in the input (or output) layer?
Hint: Consider with which (very simple) parameters such a network may map its inputs entirely unchanged to its outputs.

Exercise 33 Deep Learning: Autoencoder

Excercise 32 showed a problem that may be encountered when training an autoencoder. In the lecture, (up to) four approaches were considered with which this problem can be tackled/prevented. Which are these methods and why do they (help to) overcome this problem?

Exercise 34 Learning to Play Games

In the lecture it was briefly considered how deep learning artificial neural networks led to a program that could play the Asian board game of Go and that managed to defeat a top ranked human Go player. In a coarse analogy, in this exercise we consider the simple board game of Tic-Tac-Toe. How could one train an artificial neural network to be able to play this board game? How could one encode the board as input and the move to be made as the output of this network? Would it be useful to employ a convolutional neural network?