Ludwig-Maximilians-Universitaet Muenchen 20.05.2021
Institute for Informatics
Prof. Dr. Volker Tresp
Rajat Koner
Andreas Lohrer
Sebastian Schmoll

Machine Learning
Summer 2021
Exercise Sheet 5

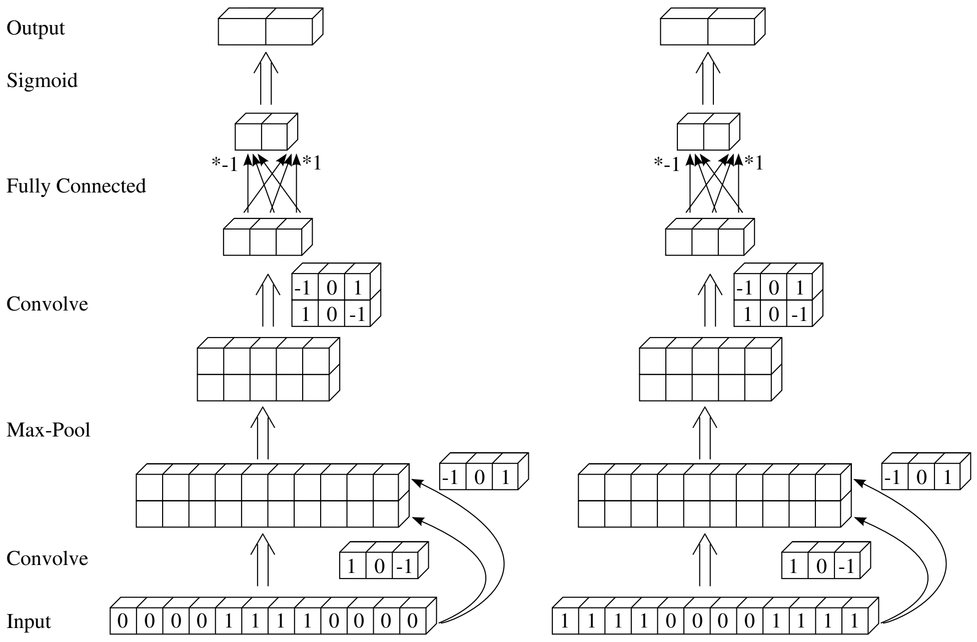
Exercise 5-1 Convolutional Neural Networks

In this exercise we address a convolutional neural network (CNN) with one-dimensional input. While two-dimensional CNNs can be used for example for grayscale images, one-dimensional CNNs could be used for time-series such as temperature or humidity readings. Concepts for the 1D-case are equivalent to 2D networks. We interpret data in our network as three-dimensional arrays where a row denotes a feature map, a column denotes a single dimension of the observation, and the depth of the array represents different observations. As we will only work with a single input vector, the depth will always be one.

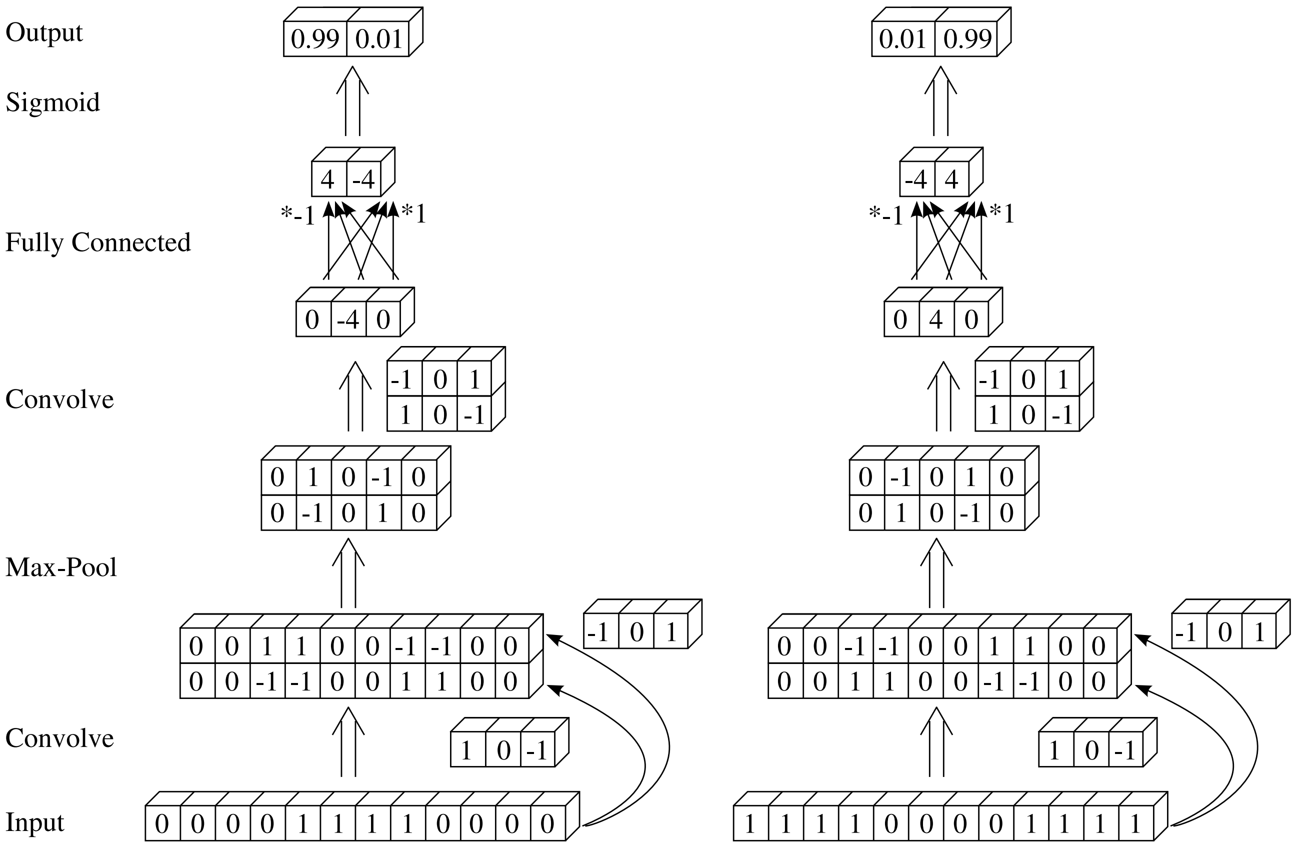
Let the following CNN be given:

- Input I: Matrix of size $1 \times 12 \times 1$. We therefore have an input with twelve dimensions consisting of a single feature map.
- First convolutional layer with filters $F_0^1 = (-1, 0, 1)$ and $F_1^1 = (1, 0, -1)$ that generates two output feature maps from a single input feature map. Use *valid* mode for convolutions.
- Max-pooling layer with stride 2 and filter size 2. Note that max-pooling pools each feature map separately.
- Convolutional layer with convolutional kernel $F_0^2 = ((-1, 0, 1), (1, 0, -1))$ of size $2 \times 3 \times 1$.
- Fully connected layer that maps all inputs to two outputs. The first output is calculated as the negative sum of all its inputs, and the second layer is calculated as the positive sum of all its inputs.
- Sigmoidal activation function

Calculate the response of the CNN for the two inputs $(0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0)$ and $(1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1)$.



Possible Solution



Exercise 5-2 Basis Functions of Neural Networks

Given a test vector $\mathbf{x}_i$, the output of a neural network is defined as

$$f(\mathbf{x}_i) = \sum_{h=0}^{M_\phi - 1} w_h \phi_h(\mathbf{x}_i, \mathbf{v}_h).$$

The weights of the neurons can be learned by employing the back-propagation rule with sample-based gradient descent. In the lecture neural networks with sigmoid neurons have been introduced, but it is possible to employ different basis functions:

a) Which properties do these basis functions have to fulfill?

b) Can a linear combination $\phi(\mathbf{x}_i, \mathbf{v}_h) = z_h = \sum_{j=0}^{M} v_{h,j} x_{i,j}$ be suitable for this?

c) Is the number of parameters for $\phi(\mathbf{x}_i, \mathbf{v}_h)$ limited? Could several different basis functions be used for the same neural network?

Possible Solution

a) The basis function of a neural network should be **continuous** and **partial differentiable** (side note: it is sufficient to be almost everywhere partial differentiable). Furthermore, it comes in handy if the amount of inflection points as well as maxima and minima are minimized (for the case that one does not want to learn a periodic function). The approach by applying a logistic function resembles the functionality of cerebral interconnections. There we have some stables regions which change their value only on a given range rapidly (and not for the entire domain of a function).

b) The linear combination is only suitable for single-layered NNs. If the setup of the NN contains linear neurons in serial, then the neurons neutralize one each other:

$$\begin{aligned} f(x_i) &= \sum_{h=0}^{M_\phi - 1} w_h \phi(\mathbf{x}_i, \mathbf{v}_h) = \sum_{h=0}^{M_\phi - 1} w_h \sum_{j=0}^{M} v_{h,j} x_{i,j} = \\ &= \sum_{h=0}^{M_\phi - 1} \sum_{j=0}^{M} w_h v_{h,j} x_{i,j} = \sum_{j=0}^{M} \left(\sum_{h=0}^{M_\phi - 1} w_h v_{h,j} \right) x_{i,j} = \\ &= \sum_{j=0}^{M} u_j x_{i,j} \end{aligned}$$

The effect is the same as the one of a simple perceptron.

c) The amount of parameters for $\phi(\mathbf{x}_i, \mathbf{v}_h)$ are not limited - one can see that if we take the radial basis function (RBF) into account:

$$\phi(x_{i,j}, \mu_j, \sigma_j) = \exp - \frac{\|x_{i,j} - \mu_j\|^2}{2\sigma_j^2} ,$$

which are typically used as a first layer in a 2-layer RBF network in order to propagate the input signal. In the second layer, one can use a linear combination of the signals coming from the first layer. In total this architecture is similar to a gaussian kernel density estimator. Of course it is possible to combine basis functions in a more complex way.

Exercise 5-3 PyTorch - Convolutional Neural Network

On the course website you will find an ipython notebook leading you through the implementaion of a Convolutional Neural Network in PyTorch for classifying handwritten digits from the MNIST dataset. Fill in the missing parts and compare the results to the ones from the Feedforward Neural Network we used last week. What do you observe?