Machine Learning June 30, 2022

Exercise 1 Basics

Answer the following questions:

1. In general, we can distinguish between two forms of machine learning. Which are these and how can we characterize them?
2. Which traditional forms of machine learning do you know?
3. Define the following terms: Hyperparameter, Bias, Classifier, Dataset, Clustering, Loss-Function, Class, Parameter, Datapoint, Variance, Regression, Training

Exercise 2 Traditional Machine Learning Procedures

Familiarize yourself with the provided script `polynomial_fit.ipynb`. Discuss strengths and weaknesses of the methods shown.

Understanding Neural Networks

Let $\vec{x} = (x_1, \dots, x_n)$ be the input vector und $\vec{g}(\vec{x}) = (\vec{g}(\vec{x})_1, \dots, \vec{g}(\vec{x})_m)$ the output vector of a neural network. We denote the loss function (Cost-Funktion) with C and with L the number of layers. The weights between layer $l-1$ and l and between node j of layer l and node k of layer $l-1$ are written as $W^l = w^l_{jk}$. The number of nodes per layer l is given by σ_l. Figure 1 shows an example for such a neural network. The total function g of the layer can be split along the function of the individual layers f^l. We can thus write

$$\vec{g}(\vec{x}) = \vec{f}^L(W^L \vec{f}^{L-1}(W^{L-1} \dots \vec{f}^1(W^1\vec{x}) \dots)) \tag{1}$$

We inspect the first layer of our example in figure 1. Here, the result at the first hidden layer will be

$$t^1_j = f^1\left(w^1_{j1}x_1 + w^1_{j2}x_2 + \dots + w^1_{jn}x_n\right) \tag{2}$$

$$t^1_j = f^1\left(\sum_{k=0}^{n} w^1_{jk}x_k\right) \tag{3}$$

or generally speaking for a larger neural network

$$t^{l+1}_j = f^{l+1}\left(\sum_{k=0}^{\sigma_l} w^{l+1}_{jk} t^l_k\right). \tag{4}$$

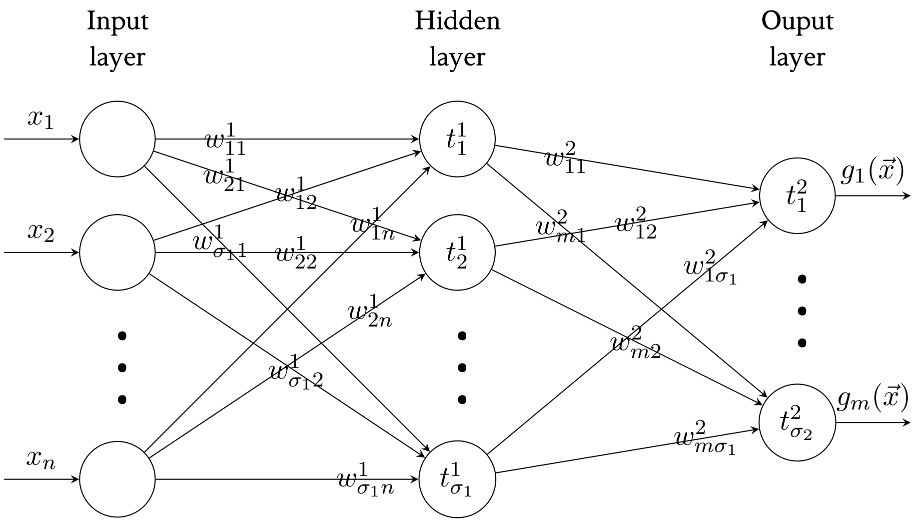

Figure 1: Example of a neural network with only one hidden layer.

Exercise 3 Building Neural Networks

1. Build a neural network with one input node and one output node and no hidden layers. This network should return double its input value. Assume $f(x) = x$. How do we need to chose the weights?
2. Extend the previous example to a vector, such that the result is of the same size buth with entries doubled.
3. Build a neural network with no hidden layers and an input vector $(x_1, \dots, x_n)$ that puts out the sum of all entries of this vector. Still assume that $f(x) = x$.
4. Build a neural network with no hidden layer that outputs the scalar product of two input vectors $v = (v_1, \dots, v_n)$ and $w = (w_1, \dots, w_n)$. Assume that all weights are exactly unity $w^l_{jk} = 1$. How does the output layer look like? How do we need to choose f?
5. Build a neural network that

Exercise 4 Backpropagation

Backpropagation computes the gradient

$$\frac{\partial C}{\partial w^l_{jk}}(\vec{g}(\vec{x}), \vec{y}) = \frac{\partial \vec{g}}{\partial w^l_{jk}}(\vec{x}) \cdot \vec{\nabla}_y C\left(\vec{g}(\vec{x}), \vec{y}\right) \tag{5}$$

at fixed input $\vec{x}$ and desired output $\vec{y}$ but with varying weights w^l_{jk}. We could calculate this derivative by applying the chain rule iteratively to equation 1 but this would be extremely inefficient.
Given a function

$$G(x) = a(b(c(\dots z(x) \dots))) \tag{6}$$

calculate the derivative with respect to x. How can we generalize this behaviour for the general case

$$x^l = f^l(x^{l-1}) \tag{7}$$

where $l \in \{0, \dots, n\}$? What happens if we assume $f^l(x) = A^l g(x)$ where A is a matrix?