Johannes Gutenberg-Universität Mainz
Institut für Informatik
Data Mining
Prof. Dr. Stefan Kramer

Machine Learning - Sheet 7

08.06.2017
Deadline: 15.06.2017 - 23:55

Task 1: Perceptron *(5 Points)*

We want to gain some understanding of the functions that can be represented by perceptrons.

(1) Design a two-input perceptron that implements the Boolean function $(A, B) \mapsto A \wedge \neg B$. Sketch the decision boundary and the four possible values for (A, B).

(2) Design a two-layer network of perceptrons that implements the exclusive-OR (XOR) function $(A, B) \mapsto A \oplus B$. Explain how you came up with the weights, sketch the decision boundaries for all intermediate steps.

Handwritten Digit Recognition

Suppose that we want to train an artificial neural network that can recognize handwritten digits. A simple design is sketched in Figure 1. In the next two exercises, we will take a closer look at the various operations that are needed for this network.

The network is supposed to take grayscale pixel values as inputs (each image will correspond to a row-vector with values between 0 and 1). The target values will be encoded as one-hot vectors. The network is supposed to output probabilities $p_0, \ldots, p_9$ for the ten digit classes.

For performance reasons, we want to train the network on *minibatches*. That means that in each iteration, instead of a single row-vector, the network will get a matrix with N training instances (one instance per row).

The network will be composed of a few types of basic operators. In the forward pass, each operator will take some matrices as inputs, and produce a matrix as output (this matrix will usually be denoted by A, for "activations"). The final output will be compared to the expected target values, and a loss E will be calculated. In the backward pass, each operator will receive the backpropagated errors as matrix B, and then use B to compute the partial derivatives of the loss E with respect to each input.

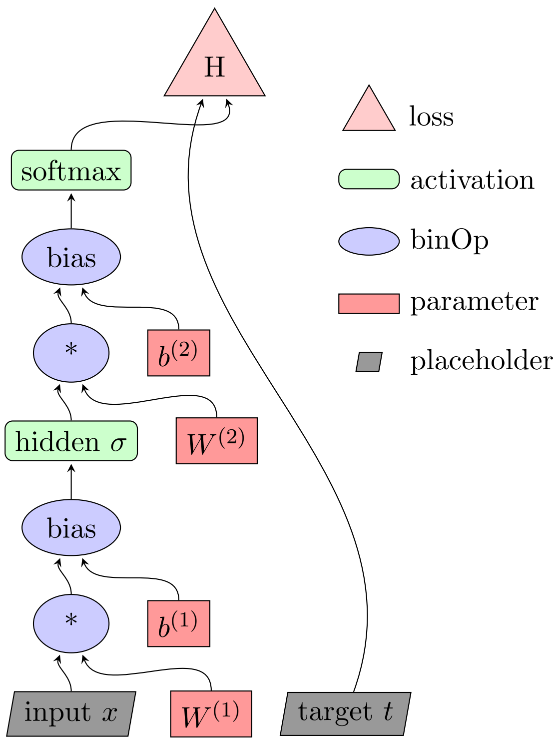

Figure 1: Simple ANN for handwritten digit classification.

Task 2: Bipartite Connections, Biases, Sigmoid Layers *(7 Points)*

In this exercise, we focus on the most common components of artificial neural networks. In the following, let n, m, N be some natural numbers.

(1) **Full bipartite connection layer:** Let $D \in \mathbb{R}^{N \times n}$ and $W \in \mathbb{R}^{n \times m}$ be two matrices. Suppose that in the forward pass, the matrix multiplication node gets D and W as inputs, and outputs their matrix product $A := DW$, that is, $A \in \mathbb{R}^{N \times m}$ with $A_{ij} := \sum_{q=1}^{n} D_{iq} W_{qj}$. In the backward pass, the node receives the backpropagated error $B \in \mathbb{R}^{N \times m}$ with $B_{ij} = \frac{\partial E}{\partial A_{ij}}$. Prove[1]:

$$\frac{\partial A_{ij}}{\partial W_{kl}} = D_{ik}\delta_{jl}, \qquad \frac{\partial E}{\partial W_{kl}} = (D^\top B)_{kl}, \qquad \frac{\partial A_{ij}}{\partial D_{kl}} = \delta_{ik} W_{lj}, \qquad \frac{\partial E}{\partial D_{kl}} = (BW^\top)_{kl}.$$

(2) **Bias layer:** Suppose that $D \in \mathbb{R}^{N \times n}$ and $b \in \mathbb{R}^n$. Suppose that $A \in \mathbb{R}^{N \times n}$ with $A_{ij} = D_{ij} + b_j$. Let $B \in \mathbb{R}^{N \times n}$ be the backpropagated error, that is: $B_{ij} = \frac{\partial E}{\partial A_{ij}}$. Show:

$$\frac{\partial A_{ij}}{\partial D_{kl}} = \delta_{ik}\delta_{jl}, \qquad \frac{\partial E}{\partial D_{kl}} = B_{kl}, \qquad \frac{\partial A_{ij}}{\partial b_k} = \delta_{kj}, \qquad \frac{\partial E}{\partial b_k} = \sum_i B_{ik}.$$

(3) **Sigmoid layer:** The input is $D \in \mathbb{R}^{N \times n}$. The activation is $A \in \mathbb{R}^{N \times n}$ with $A_{ij} := \sigma(D_{ij})$, where $\sigma(t) := 1/(1 + e^{-t})$ is the *sigmoid function*. Let $B \in \mathbb{R}^{N \times n}$ be the backpropagated error, i.e., $B_{ij} = \frac{\partial E}{\partial A_{ij}}$. Show that $\sigma'(t) = \sigma(t)(1 - \sigma(t))$, and $\frac{\partial E}{\partial D_{kl}} = B_{kl} A_{kl}(1 - A_{kl})$.

Task 3: Softmax and Cross-Entropy Loss *(8 Points)*

We want to use our network for classification, therefore we need an output layer that can output probabilities for multiple possible classes. A *softmax* layer is a common choice. The goal of this exercise is to understand the interaction between the softmax output layer and the *categorical cross entropy* loss.

1. For a vector $v \in \mathbb{R}^n$, we define the *softmax* function as follows:

$$Z(v) := \sum_{j=1}^{n} e^{v_j}, \qquad S_i(v) := \frac{e^{v_i}}{Z(v)}, \qquad \mathrm{softmax}(v) := (S_i(v))_{i=1}^{n} \equiv (S_1(v), \ldots, S_n(v)).$$

   Prove:

   - For all $n \in \mathbb{N}$ and $v \in \mathbb{R}^n$, the function $i \mapsto S_i(v)$ is a probability mass function.
   - For $c \in \mathbb{R}$, $v, w \in \mathbb{R}^n$ with $w_i = v_i + c$, it holds: $S_j(w) = S_j(v)$.
   - For all i, j it holds: $\frac{\partial S_j(v)}{\partial v_i} = S_j(v)(\delta_{ij} - S_i(v))$

2. For $n \in \mathbb{N}$ and two vectors $p, q \in [0,1]^n$ with $\sum_i p_i = \sum_i q_i = 1$, we define the *cross entropy* by

$$H(p, q) := -\sum_{i=1}^{n} p_i \log(q_i).$$

   Now suppose that $t \in [0,1]^n$ with $\sum_i t_i = 1$ is some target distribution, and $v \in \mathbb{R}^n$ are some activations. Compute $\frac{\partial H(t,q)}{\partial q_j}$, then set $q_j := S_j(v)$ and prove:

$$\frac{\partial H(t, \mathrm{softmax}(v))}{\partial v_i} = S_i(v) - t_i.$$

3. Why would designers of TensorFlow or Caffe include softmax_cross_entropy_with_logits and SoftmaxWithLossLayer in their frameworks? Explain the differences in the setup of the training and test phases, discuss the advantages.

[1] Here, δ_{ij} stands for the Kronecker Delta. This means: δ_{ij} is 1 iff $i = j$, and 0 otherwise. Hint: if $\xi(i)$ is some expression that depends on the index i, then $\sum_i \xi(i)\delta_{ij} = \xi(j)$.