Data-driven Intelligent Systems: Practical Assignments #3

Due on Fri, May 29, 2019

Task 1

Decision Trees ID3, 30 points.
The following table lists 8 observations from a poll asking customers whether or not they would wait in a queue for a restaurant. Two attributes were identified that may play a role on the decision, namely the weather (is it sunny or rainy?) and whether or not the person is hungry. Compute the corresponding decision tree by hand and answer the accompanying questions.

obs	weather	hungry	wait?
1	sunny	yes	+
2	sunny	no	+
3	rainy	no	-
4	rainy	no	-
5	sunny	no	+
6	rainy	yes	+
7	rainy	yes	-
8	sunny	yes	+

- For which kind of data you can use a decision tree?
- What is the learning strategy in a decision tree?
- What is *entropy* in this context and why is it necessary to compute?

Task 2

Decision Tree in Python, 30 points. In the material folder you can find a Jupyter notebook providing a demonstration of the computational steps of decision trees in Python. Run the demo (play around, change, etc.) and explain what the *Gini index* is, used here in this example.

Task 3

Simple Perceptron, 10 points.
Design a perceptron at hand (you can include a pen & paper drawing in your results) with inputs x_A and x_B, which implements the boolean function $f(x_A, x_B)$, representing the propositional formula $A \wedge \neg B$.
Explain all relevant parameters in your network.
Hint: Use the value 0 *for false and* 1 *for true, and the activation function* $\varphi(x) = \max(sign(x), 0)$.

Task 4

Backpropagation, 30 points.
Learning with the backpropagation algorithm: Our task is to train an MLP that approximates an unknown function underlying a certain data set as well as possible.
The given data set consists of input and output tuples $\{\mathbf{x}^{(d)}, \mathbf{y}^{(d)}\}$ for each data point with $d = 1, \ldots, 10000$.
The $n = 5$ components of the input tuple $\mathbf{x}^{(d)} = (x_1, \ldots, x_n)^{(d)} \in \mathbb{R}^n$ are within the interval $[0, 100]$, while

the $m = 1$ outputs $\mathbf{y}^{(d)} = (y_1, \ldots, y_m)^{(d)} \in \mathbb{R}^m$ are within the interval $[0, 10]$ (so in this case it's a scalar!) and have been produced by the unknown function with noise.
Unfortunately, the given pseudo-code contains several errors. Correct the errors in the following sketch of a solution and do necessary completions!

Proposed solution:

Definition of the network architecture: 5 inputs, 3 hidden neurons, 1 output neuron. One bias neuron with activation "-1", which is connected with all neurons.

Initialisation of all weights v_{ij} and w_{jk} with random values in the interval [0,1]

```
BEGIN {iterate}
    all temporary weight-updates: $\Delta v_{ij} = \Delta w_{jk} = 0$
    overall sum-of-squares training error: $e^{(train)} = 0$
    BEGIN {for each data point $d$}
        activations of input neurons $i$: $x_i = x_i^{(d)}$
        activations of hidden neurons $j$: $a_j = 1/(1 + \exp(-\sum_i v_{ij}\, x_i))$
        activations of output neurons $k$: $y_k = 1/(1 + \exp(-\sum_j w_{jk}\, a_j))$
        error for output neurons $k$: $\delta_k = y_k \cdot (1 - y_k) \cdot (y_k^{(d)} - y_k)^2$
        error for hidden neurons $j$: $\delta_j = a_j \cdot (1 - a_j) \cdot \sum_k \delta_k w_{jk}$
        cumulate updates for weights: $\Delta w_{jk} += \delta_k \cdot a_j$ , $\Delta v_{ij} += \delta_j \cdot x_i$
        sum overall error: $e^{(train)} += (y_k^{(d)} - y_k)$
    END {for each data point}
    adapt weights: $v_{ij} += \Delta v_{ij}$ , $w_{jk} += \Delta w_{jk}$
END {iterate} STOP IF {$e^{(train)} = 0$}
```