Machine Learning I WS2016/17

Lecturer: Prof. Dr. Bethge

1. Decision Tree Learning & Overfitting

Generate binary data with the following MATLAB code:

```
nn=50; nt=500;
w=randn(1,nn);
x=sign(randn(nn,nt));
y=(1+sign(w*x))/2;
```

such that each output value of y is a nonlinear function of the vector $\mathbf{x}$ given by the corresponding column of x.

Write a program that learns a binary decision tree that predicts the output y based on the relative frequencies obtained with the decision tree (as explained in the lecture). In the beginning, the set of all leaf nodes consists only of the root node that contains all training examples. The main loop consists of iterating through all leaf nodes to determine which of the leaf nodes is the best to be split into two new leaf nodes. To find the best one, we need to determine for each of them how much the conditional entropy could be decreased. That is we need to know for each leaf node which new attribute would lead to the largest reduction of the conditional entropy and how much this reduction would be. If we do book keeping about the potential entropy reduction for each node, we only need to determine the potential entropy reduction for the two new leaf nodes that are obtained whenever we split a former leaf node into two new ones. In addition, we sort the data of the former leaf node to its two new descendants. This splitting process can be continued as long as the entropy reduction is positive.

You can think of the leafs of the decision tree to define a partition $\{L_k\}_{k=1}^n$ of the sample space of the random vector $\mathbf{x}$. That is, for all possible random vectors $\mathbf{x}$ there is exactly one k^* for which it holds $\mathbf{x} \in L_{k^*}$ and $\mathbf{x} \notin L_j$ for all $j \neq k^*$. In this way a model distribution $\hat{p}(y|\mathbf{x} \in L_k)$ is constructed that assigns different probabilities to the two possible outputs $y \in \{0, 1\}$ depending on which event of the events L_k are observed. During learning of the tree we seek to minimize the conditional entropy

$$\sum_{k=1}^{n} \hat{p}(\mathbf{x} \in L_k) \sum_{y \in \{0,1\}} \hat{p}(y|\mathbf{x} \in L_k) \log_2 \hat{p}(y|\mathbf{x} \in L_k)$$

Use the same code as shown above to generate test data in addition to the training data. For each new split of a former leaf node into two new ones, use the test data to verify whether the cross entropy of the model distribution $\hat{p}(y|\mathbf{x} \in L_k)$ defined by the learned decision tree

$$\sum_{k=1}^{n} p(\mathbf{x} \in L_k) \sum_{y \in \{0,1\}} p(y|\mathbf{x} \in L_k) \log_2 \hat{p}(y|\mathbf{x} \in L_k)$$

is reduced or not. Note that we here use p to refer to the probabilities defined by the relative frequencies on the test set whereas $\hat{p}$ was determined by the relative frequencies on the training set. In contrast to the entropy, which by construction of the decision tree learning algorithm can only decrease with each new split of a leaf node, the cross entropy on test data can increase. Whenever this happens, it indicates that the decision tree is "overfitted" to the training data. Let $\kappa_1, \kappa_2, \ldots$ denote the attribute selected for the first, second, ... split.

Submit the following plots: (a) amplitude of the weight $|w_{\kappa_j}|$ as a function of j, (b) entropy as a function of the number of splits, (c) cross entropy on test data as a function of the number of splits. In addition, upload your code to Ilias.