Table 1. Data sets used in the experimental section

<table>
<thead>
<tr>
<th>name</th>
<th>dataset size</th>
<th>nr. attributes</th>
<th>source</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADULT</td>
<td>48842</td>
<td>123</td>
<td>[1]</td>
</tr>
<tr>
<td>IJCNN</td>
<td>141691</td>
<td>22</td>
<td>[25]</td>
</tr>
<tr>
<td>COD-RNA</td>
<td>488565</td>
<td>8</td>
<td>[29]</td>
</tr>
</tbody>
</table>

5 Experiments

In this section we test empirically the validity of our method and the heuristics proposed. Recall that the main goal of this work is to devise a classifier where the basis vectors of the decision rule depend on the input example $x$. Our approach is to partition the input space and use a different perceptron in each region. The most straightforward strategy is to divide the feature space into two disjoint regions. The first experiment is designed to show that already in this very simple setting we can gain in accuracy by using two different perceptrons in the regions.

In the second experiment we study how well the tree-based approach compares with other budgeted online learning algorithms. We do not expect them to outperform these in accuracy. Instead our main argument is that we can achieve a similar performance by using a considerably smaller number of basis vectors. While our trees can grow very large, and may hence contain a large number of leaf nodes, the total number of kernel computations for each example is bounded by the user-specified budget $B$. We set $B = 100$ in these experiments. A more thorough study of the effects of $B$ is left for a longer version of this paper.

We compare our methods with the Forgetron [11] and Projectron [23] algorithms, and the Stoptron, a simple budget perceptron baseline that learns a kernel perceptron but stops updating after budget is reached. The data sets we use are publicly available at [7]. Table 1 shows details of the data sets. Each data set was split into training (50%), testing (25%), and validation (25%) sets. In the experiments that follow we use different portions of the datasets, which will be appropriately described in the text.

In each case we use a Gaussian kernel. The $\gamma$ parameter of the kernel was optimized for each data by running the basic perceptron algorithm on the training data using different values of $\gamma$ and choosing the one with the best performance on the validation set.

5.1 Comparing Splitting Functions

This experiment compares the performance of using a single Stoptron against using two Stoptrons at the leaves of a single split, varying and thus comparing the different splitting criteria. The setting is as follows: we start training a Stoptron on the training data. Once the number of basis vectors hits $B$ (set to 100), we split the node using each of the techniques described above. At this point training is stopped, and we switch to testing. For each splitting method,
the resulting model is applied to the test data. We make a pass over the entire test data, and two STOPTRONS are grown on either side of the splits.

The results in Figure 1. show that splitting in combination with STOPTRON outperforms the global STOPTRON, supporting our hypothesis that splitting may be very useful. Moreover, the results also suggest that it is important to do the split in a smart way. This is indicated by the relatively poor performance of the completely random RNDW-RDNS heuristic in case of ADULT and IJCNN. Interestingly, RNDW-OPTS performs quite well, suggesting that if the basis vectors are selected carefully, any balanced linear branching function will have a reasonable performance. Finally, our WPQ method is the only one that consistently shows good performance across all three datasets.

5.2 Algorithm Comparison

This experiment measures the trade-off between classification time and classification accuracy. We use the number of basis vectors as a proxy for classification