# Additional Pseudocode for the article "Algorithms for Manipulation of Level Sets of Nonparametric Density Estimates"

Jussi Klemelä

June 25, 2003

## 1 An algorithm for forming a level set tree

We give in the following an algorithm for forming the level set tree. This algorithm is a typical tree-creating algorithm. The algorithm starts by finding roots of the tree. Then we travel to the upper levels of the tree. When there is a branching of the tree, we will travel to one of the branches and will put those nodes, which are starting nodes of other branches, to a stack. Once we reach a leaf node, we took a node from the stack and start again traveling upwards. The algorithm calls a procedure **partition** for partitioning a set of gridpoints. Note that the tree structure of a multitree is uniquely determined by giving the parent node for every node,

1. Input for the algorithm is an evaluation tree. Also, an input is a grid of levels of the level set tree to be calculated: $0 < Q_1 < \cdots < Q_L$.

2. Output of the algorithm is a level set tree. With every node of the level set tree there is associated a set of gridpoints (disconnected components of level sets) and a level. Tree structure of the tree is determined by giving parent for every node and giving for root nodes an appropriate label.

3. Internal data structure of the algorithm is a stack $S$ of nodes of the level set tree.

ALGORITHM **form** for forming a level set tree

```
1. partition the level set with level $Q_1$ to minimally disconnected components, these disconnected components are the root nodes of level set tree LST
2. put these components of level $Q_1$ to stack $S$
3. if we have more than one level ($L > 1$)
    (a) while stack $S$ is not empty
        i. take a component $C$ from stack $S$, assume that the level of this component is $Q_l$
        ii. make intersection with component $C$ and the level set of level $Q_{l+1}$
        iii. if component $C$ is the same as this intersection, then
            A. change the level of component $C$ to be $Q_{l+1}$
            B. if $Q_{l+1} < Q_L$, then put component $C$ back to stack $S$
        iv. else (intersection is a genuine subset of component $C$)
            A. partition the intersection to disconnected components, assign the parent of these new components to be component $C$, add these new components to level set tree LST
            B. if $Q_{l+1} < Q_L$, then put new components to stack $S$
        v. end if
    (b) end while
4. end if
5. return LST
```

In step 3(a)iii.A we have economized the level set tree by not taking such nodes to the tree whose associated set is the same but only the level is changing.

In step 1 and step 3(a)iv.A we have called a procedure **partition** which is the heart of the algorithm. Next section describes this procedure.

## 2 An algorithm for calculating evaluation tree for the weighted average of two functions

Here is the pesudocode for the algorithm **treeadd** for making a weighted average of two evaluation trees.

1. **Input** of the algorithm
2. **Output** of the algorithm

```
ALGORITHM **treeadd**($tr1$, $tr2$, $p$)

1. **go** through the leafs of tree $tr1$.
2. Consider leaf $curleaf$ of tree $tr1$. We grow $tr1$ by making $curleaf$ the
   root node of a subtree of $tr1$. Initialize stacks: $Sadd[1] = curleaf$ and
   $Str2[1]$=root of $tr2$.
3. **while** stacks are not empty
   (a) Take from stack $Sadd$ node $curleaf$ of new tree and from stack
       $Str2$ node $nodeoftr2$ of tr2.
   (b) **while** $nodeoftr2$ is not a leaf
       i. let $s$ be the split point and $k$ the direction of node $nodeoftr2$
       ii. **if** $s$ splits the set of the current node $curleaf$, **then**
           A. create left child $newleft$ and right child $newright$ for
              $curleaf$
           B. Denote $val$ = value of $curleaf$, $val.left$ = value of left
              child of $nodeoftr2$, $val.right$ = value of right child of
              $nodeoftr2$.
           C. Calculate mean: annotate left child $newleft$ with $(1-p) \cdot$
              $val + p \cdot val.left$ and annotate right child $newright$ with
              $(1-p) \cdot val + p \cdot val.right$.
           D. Put right child $newright$ to the stack $Sadd$ and the right
              child of node $nodeoftr2$ to the stack $Str2$.
           E. Set $nodeoftr2$=left child of node $nodeoftr2$.
       iii. **else if** $s$ is smaller than the left endpoint of rectangle of the
            current node $curleaf$, in direction $k$, **then** set $nodeoftr2$=left
            child of node $nodeoftr2$.
       iv. **else** (if $s$ is larger than the right endpoint of rectangle of the
           current node $curleaf$, in direction $k$, **then** set $nodeoftr2$=right
           child of node $nodeoftr2$.
   (c) **end while**
4. **end while**
```

```
5. end go
6. return $tr1$
```

Note that in the definition of the evaluation tree only leaf nodes were annotated with values of the function. In the above algorithm we have annotated also other nodes, in step 3.(b).ii.