SCHEDAE INFORMATICAE
VOLUME 20 (2011) DOI 10.4467/20838476SI.11.004.0290

# On the Mean Squared Error of Hierarchical Estimator

Stanisław Brodowski
Faculty of Physics, Astronomy, and Applied Computer Science,
Jagiellonian University, Reymonta 4, 30-059 Kraków, Poland
e-mail: *stanislaw.brodowski@uj.edu.pl*

**Abstract.** In this paper a new theorem about components of the mean squared error of Hierarchical Estimator is presented. Hierarchical Estimator is a machine learning meta-algorithm that attempts to build, in an incremental and hierarchical manner, a tree of relatively simple function estimators and combine their results to achieve better accuracy than any of the individual ones. The components of the error of a node of such a tree are: weighted mean of the error of the estimator in a node and the errors of children, a non-positive term that decreases below 0 if children responses on any example differ and a term representing relative quality of an internal weighting function, which can be conservatively kept at 0 if needed. Guidelines for achieving good results based on the theorem are briefly discussed.

**Keywords:** Hierarchial Estimator, hierarchical model, regression, function approximation, classifier error.

## 1. Introduction

Machine learning is one of the classical topics in computer science [1, 2]. This paper presents some theoretical findings about a machine learning solution concerned with supervised learning called Hierarchical Estimator. That meta-algorithm, presented in [3], arranges many simple, possibly relatively inaccurate, function estimators (approximators) into a tree structure and combines their results in an attempt to obtain one more accurate.

The basic general task of the mentioned technique is to predict values of a random variable $Y$ with possible values in $\mathcal{Y} \subset \mathbb{R}^r$ being presented with values of another

variable $X$ (with possible values in $\mathcal{X} \subset \mathbb{R}^p$) and knowing some set (or series) of values of $X$ paired with values of $Y$ called training set $\mathcal{D} = \{(x^{(k)}, y^{(k)}), k \in \{1 \ldots |\mathcal{D}|\}, \forall k : x^{(k)} \in \mathcal{X}, y^{(k)} \in \mathcal{Y}\}$. This may be done by the approximating function $f : \mathcal{X} \longrightarrow \mathcal{Y}$, such that:

$$Y = f(X) + \varepsilon, \tag{1}$$

where $\varepsilon$ is some error variable of certain properties (e.g. having 0 mean).

Because joint probability $P_{X,Y}$ (so also $\varepsilon$) is not available, usually minimizing of a loss function, e.g. squared loss over $\mathcal{D}$, is attempted instead [4].

As mentioned above, the main task is prediction, so working on examples not being in the training set is required. If a solution works well only on the training set, but poorly on unseen examples, it is described as having low *generalization*. If the technique that is used is parametric, i.e. first some model is selected and then parameters optimized, low generalization is often a result of selecting a too complicated model [5, 6, 7].

### 1.1. Similar solutions

Hierarchical Estimator attempts to combine many less accurate estimators into more accurate one, so it is loosely related to the Theory of Weak Learnability [8]. Its execution may be seen as building a problem model in an incremental manner – starting from a simple one and increasing complexity. Because some parts of it guide the creation and working of others, it may be considered hierarchical. It creates a tree structure that is automatically adapted to the problem being learned, so its similarity to well known AdaBoost [9] is at most moderate. Another difference is that while original AdaBoost sets the weight of component models for all examples, Hierarchical Estimator assigns different weights to the experts based on the example being evaluated. This makes it more similar to Hierarchical Mixture of Experts (see [10]), but its operation differs significantly, even when constructive algorithms like [11] are considered. For example, HME has expert nodes in leaf nodes, while Hierarchical Estimator has them in all nodes and they all solve some subproblem of the original problem. The outputs of internal nodes can be used both for evaluating the result and, after additional processing and possibly including other variables, for weighting results of component estimators. This also constitutes the most significant of many differences between Hierarchical Estimator and regression trees M5 [12].

Probably the most similar solution to Hierarchical Estimator is Hierarchical Classifier [13], based on similar premises. Its details are strongly connected to the classification task though and that forces many differences [3]. Hierarchical Estimator is designed for predicting continuously-valued number or vector outputs, so its scope is different than that of Hierarchical Classifier. The meta-algorithm nature of Hierarchical Estimator is also more explicit than in the case of Hierarchical Classifier.

## 1.2. Hierarchical Estimator

### 1.2.1. Basic definitions

In a very general sense, Hierarchical Estimator is a function $HE : \mathcal{X} \longrightarrow \mathcal{Y}$, that uses a tree structure where node indices are from some index set $I$. Let $N(i)$ be the number of children (possibly 0) of the valid tree node with index $i$ (called for the sake of brevity "node $i$").

Two functions are assigned to each valid node [3]:

1. a function estimator (approximator) $g_i : \mathcal{X} \longrightarrow \mathcal{Y}$ that solves some subproblem of the original problem, in [3] simple neural networks are used for this task,
2. *competence function* $C_i : \{0, \ldots, N(i)\} \times \mathcal{X} \longrightarrow [0, 1]$ which values are used as weights for results of children nodes and the result of the estimator in node $i$ when the value of the estimator for a given example is calculated, as described in Eq. (2).

We assign to each child a number among its siblings. $P : I \times \mathbb{N} \to I$ is the function that returns the global node index of a child based on the parent's index and that child number, i.e. $P(i, j)$ gives the index of the $j$-th child of node $i$.

DEFINITION 1 (Hierarchical Estimator node response). *The recursive formula for retrieving response of some node $i$ on $k$-th example is [3]:*

$$\widetilde{g}_i(x^{(k)}) = \sum_{j=1}^{N(i)} \widetilde{g}_{P(i,j)}(x^{(k)}) \cdot C_i(j, x^{(k)}) + C_i(0, x^{(k)}) \cdot g_i(x^{(k)}), \tag{2}$$

*where*

$$\sum_{j=0}^{N(i)} C_i(j, x^{(k)}) = 1. \tag{3}$$

DEFINITION 2 (Hierarchical Estimator response). *The Hierarchical Estimator response for a given example is the response of the root (its index denoted here as $r$):*

$$HE(x) = \widetilde{g}_r(x). \tag{4}$$

For a leaf $C_i(0, x^{(k)}) = 1$ and $\widetilde{g}_i(x^{(k)}) = g_i(x^{(k)})$.

A more compact version of the definition arises when we identify the result of the estimator in the given node $g_i(x^{(k)})$ with a result of a "virtual" zeroth child $\widetilde{g}_{P(i,0)}$:

$$\widetilde{g}_i(x^{(k)}) = \sum_{j=0}^{N(i)} \widetilde{g}_{P(i,j)}(x^{(k)}) \cdot C_i(j, x^{(k)}). \tag{5}$$

When aggregating the result, an example is first propagated down the tree starting from the root. Weights are proposed for a given example and each child node

by function $C$ and only those children that achieved a non-zero value are used. This means that the example is not propagated through the whole tree, but only certain paths and branches. The propagation along a given path stops if it reaches a node in which $C_i(0, x^{(k)}) = 1$, usually, but not necessarily, a leaf node.

It is very important that function $C$ depends on the example being evaluated. Therefore, although for any given example the response of Hierarchical Estimator is a weighted mean of the response of some nodes, the whole Hierarchical Estimator is not a linear combination of the estimators in the nodes.

### 1.2.2. Useful terms – competence

In the discussion about Hierarchical Estimator two more definitions will be very useful [3]:

DEFINITION 3 (Competence area). *A competence area is the set of all feature vectors that a given node* may *possibly be required to evaluate.*

DEFINITION 4 (Competence set). *A competence set contains all* examples *from a given set (also if that set is only known from the context of the term use) that fall into the competence area of the node.*

An example can fall into the competence set or competence area of a node if this is a valid feature vector and the node is root, or if for some given set $\mathcal{S}$ (a set of all possible vectors for the competence area) and the given node being a $jth$ child of node $i$, the competence function from node $i$ is non-zero. In the latter case the competence set is designated as $\mathcal{S}_{P(i,j)}$ and follows:

$$\mathcal{S}_{P(i,j)} = \{(x^{(k)}, y^{(k)}) | (x^{(k)}, y^{(k)}) \in \mathcal{S} \wedge C_i(j, x^{(k)}) > 0\}. \tag{6}$$

This can be also applied to "virtual" child 0.

### 1.2.3. Learning

The whole structure of Hierarchical Estimator is found while learning from examples, so at least a brief description of the learning algorithm is needed for full understanding the consequences of theoretical findings described in this article.

The procedure of learning Hierarchical Estimator on a training set $\mathcal{D}$ is:

1. Create a root node and make $\mathcal{D}$ its training set.

2. Build a function estimator (possibly simple) in the processed node (later called node $i$).

3. Compute $E(\mathcal{S}_i, g_i)$ – the mean squared error or some other error measure – for the given node and its competence set (which is not necessarily identical to training set $\mathcal{D}_i$). If it is smaller than some preset value ("the goal") stop the algorithm for this branch. If, on the other hand, this error its greater than that of its parent (on the same set), stop the algorithm for this branch, but also delete this node.

4. If the solution is becoming too complex with respect to some preset parameter (usually maximum tree depth) stop the algorithm (for this branch). This condition is placed to limit the learning time.

5. Build

   (a) Training sets for the children nodes $\{\mathcal{D}_{P(i,1)} \dots \mathcal{D}_{P(i,N(i))}\}$ ($N(i)$ also needs to be found). This is usually done by creating a function $U_i$, such that $(x^{(k)}, y^{(k)}) \in \mathcal{D}_{P(i,j)} \Leftrightarrow U_i(j, x^{(k)}, y^{(k)}) > 0$. Because competence sets generally should overlap (as indicated in [3]) training sets usually also will.

   (b) Competence function $C_i$.

   As these tasks are closely related, they are usually performed together [3].

6. Run this algorithm for the children of the given node from point 2.

In [3], the creation of competence function $C$ and dividing the training set is based primarily on the responses of the estimator in the node. Usually it involves some form of fuzzy clustering e.g. Fuzzy C-Means [14] with the cluster number selection technique, described in [15]. For example, in the simplest, but not very effective form, outputs of the estimator in the node are fuzzy-clustered, each cluster constitutes one training set for a child and the competence function value is the membership of the given example in the given cluster. In one of the more sophisticated methods, both outputs of the estimator in the node and true values are clustered by means of fuzzy clustering. Then a corellation matrix is made between clusters in estimator outputs and clusters in true values. Finally, the rows of such matrix are clustered. The competence function is based on finding the memberships of a given example in each row, by using memberships of the example in response clusters, the corellation matrix and a chosen set of fuzzy operators, and then combining this information, again using fuzzy operators, with the memberships of each row in final clusters. The training sets are found in a similar way, but information about true values is also used. Paper [3] presents this method in detail and in two variants as well as one other method.

It should be mentioned that because the solution presented in [3] uses Artificial Neural Networks as estimators in the nodes, the data given as their input should be adequatly prepared, normalized among others. This is sometimes not a trivial task [16, 17] though usually standard normalization procedures are used.

### 1.2.4. Details

As it can be seen from definitions above, certain important details have to be determined separately. This concerns not only the selection of estimators in nodes (like in many other solutions, e.g. AdaBoost) but also the exact form of the competence function and creating training sets for successor nodes. Several versions of such details are described in [3] and their performance evaluated on several datasets. They are inspired by the theorems cited in Section 2.2. and their proofs.

## 2. The error structure of Hierarchical Estimator

### 2.1. Preliminary notions

In this section, several notions will be used that were not explained above, because their scope is more limited. For convenience they are grouped here. Most of them appear in a similar form as in [3].

- $\mathcal{S} = \{(x^{(k)}, y^{(k)}) | k = 1, \ldots, |\mathcal{S}|\}$ – is used for a set of examples on which the estimator (or a given node) is evaluated. $|\mathcal{S}|$ is the size of that set. Please recall that each $x^{(k)} \in \mathcal{X} \subset \mathbb{R}^p, y^{(k)} \in \mathcal{Y} \subset \mathbb{R}^r$
- $|\mathcal{S}_{P(i,j)}|$ is the size of the set $\mathcal{S}_{P(i,j)}$, a competence set of $j$-th child of node $i$ within set $\mathcal{S}$.
- $e((x^{(k)}, y^{(k)}), g)$ – a squared error of estimator $g$ on example $(x^{(k)}, y^{(k)})$

$$e((x^{(k)}, y^{(k)}), g) = \sum_{l=1}^{r} (y_l^{(k)} - g(x^{(k)})_l)^2. \tag{7}$$

  This notation can be shortened as in:

$$e_{(i,j)}(k) = e((x^{(k)}, y^{(k)}), g_{P(i,j)}), \tag{8}$$
$$\widetilde{e}_{(i,j)}(k) = e((x^{(k)}, y^{(k)}), \widetilde{g}_{P(i,j)}), \tag{9}$$
$$e_{(i)}(k) = e((x^{(k)}, y^{(k)}), g_i), \tag{10}$$

- $\eta_{i,j}(k)$ – a short way for denoting the difference between the target function value on $k$-th example of a given set and the result of $j$-th child of node $i$ for this example;

$$\eta_{i,j}(k) = g_{P(i,j)}(x^{(k)}) - y^{(k)}, x^{(k)} \in \mathcal{S}_{P(i,j)} \ ,$$
$$\widetilde{\eta}_{i,j}(k) = \widetilde{g}_{P(i,j)}(x^{(k)}) - y^{(k)}, x^{(k)} \in \mathcal{S}_{P(i,j)}. \tag{11}$$

The error function can be easily created from $\eta$:

$$\widetilde{e}_{(i,j)}(k) = \sum_{l=1}^{r} \widetilde{\eta}_{i,j}(k)_l^2. \tag{12}$$

→ $E(\mathcal{S}, g)$ – a mean squared error of estimator $g$ on the set $\mathcal{S}$

$$E(\mathcal{S}, g) = \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} e((x^{(k)}, y^{(k)}), g). \tag{13}$$

→ $\mathbf{1}_{C_i}$ is a characteristic (indicator) function of competence set (or area)

$$\mathbf{1}_{C_i}(j, x^{(k)}) = \begin{cases} 1, & C_i(j, x^{(k)}) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Note, that $C_i$ multiplied by $\mathbf{1}_{C_i}$ is still $C_i$.

→ $n_{ik}$ is the number of such $j$ for which $C_i(j, x^{(k)}) > 0$, so it is the number of children actually used on a given example (possibly including "virtual" child 0).

→ $n_i^{max}$ is its maximum on the whole set $\mathcal{S}$: $n_i^{max} = max_{k:(x^{(k)}, y^{(k)}) \in \mathcal{S}} \ n_{ik}$.

→ $n_i$ is used if $n_{ik}$ is constant over all examples that are considered, so the $k$ index can be omitted.

### 2.2. Existing theorems about the error of Hierarchical Estimator

In [3] several facts were proved about the Hierarchical Estimator squared error. For the purpose of this article the first of them is of most interest.

THEOREM 1. *For any node $i$ in Hierarchical Estimator suppose that:*

*$\mathcal{S}$ is a competence set of node $i$,*

*for each example in set $\mathcal{S}$, $n_{ik}$ is constant:*

$$\forall k : (x^{(k)}, y^{(k)}) \in \mathcal{S}, \quad \sum_{j=0}^{N(i)} \mathbf{1}_{C_i}(j, x^{(k)}) = n_i, \tag{15}$$

*where $n_i > 0$,*

*$C_i$ fulfills*

$$\frac{1}{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|}\sum_{l=1}^{r}\left(\sum_{j:C_i(j,x^{(k)})>0}\widetilde{\eta}_{i,j}(k)_l\cdot C_i(j,x^{(k)})\right)^2 \leq \frac{1}{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|}\sum_{l=1}^{r}\left(\frac{1}{n_i}\sum_{j:C_i(j,x^{(k)})>0}\widetilde{\eta}_{i,j}(k)_l\right)^2. \tag{16}$$

*Then*

$$E(\mathcal{S},\widetilde{g}_i)\leq E(\mathcal{S},g_i)-\sum_{j=1}^{N(i)}\frac{|\mathcal{S}_{P(i,j)}|}{|\mathcal{S}|\cdot n_i}\left(E(\mathcal{S}_{P(i,j)},g_i)-E(\mathcal{S}_{P(i,j)},\widetilde{g}_{P(i,j)})\right). \tag{17}$$

In other words, if we always use $n_i$ children for an example (or one less, but use the estimator in a given node) and errors achieved when the given competence function is used are no greater than if the same children were used, but weighted equally, the error is no greater than the error of the estimator diminished by differences between its error on competence sets of children and the children errors on that sets.

It is not a suprising result, but one of the corollaries proved in the article [3] (Corollary 2) states that the final inequality can be easily made strict – it is enough that the used children nodes have different errors on one example.

This theorem and its proof brought some more detailed information on what is needed for the solution to work properly.

The assumption 15 about the constant number of used children is inconvenient (though necessary for the given form of the theorem), so modified version of the theorem was proved, exchanging it for another (considered weaker by the author) [3]:

THEOREM 2. *Consider node $i$ and example set $\mathcal{S}$.*
*Here points (15) and (16) from Theorem 1 are replaced by:*

$$\frac{1}{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|}\sum_{l=1}^{r}\left(\sum_{j:C_i(j,x^{(k)})>0}(\widetilde{\eta}_{i,j}(k)_l\cdot C_i(j,x^{(k)})\right)^2 \leq \frac{1}{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|}\sum_{l=1}^{r}\left(\frac{1}{n_i^{max}}\sum_{j:C_i(j,x^{(k)})>0\wedge j>0}\widetilde{\eta}_{i,j}(k)_l\frac{n_i^{max}-n_{ik}+1}{n_i^{max}}\widetilde{\eta}_{i,0}(k)_l\right)^2. \tag{18}$$

*And*

$$\forall k:(x^{(k)},y^{(k)})\in\mathcal{S}\,,\quad C_i(0,x^{(k)})>0, \tag{19}$$

*The conclusion is then*

$$E(\mathcal{S},\widetilde{g}_i)\leq E(\mathcal{S},g_i)-\sum_{j=1}^{N(i)}\frac{|\mathcal{S}_{P(i,j)}|}{|\mathcal{S}|\cdot n_i^{max}}\left(E(\mathcal{S}_{P(i,j)},g_i)-E(\mathcal{S}_{P(i,j)},\widetilde{g}_{P(i,j)})\right). \tag{20}$$

The assumption 15 about the constant number of used children is replaced by a requirement that the estimator in a given (parent) node is always used (19). It may be in many cases less restricting than that of the first theorem and possibly also more technical, as we can use arbitrary small values of the competence function for that node. The thesis changed accordingly and can be called somewhat weaker. The sketch of the proof is also in [3], the technical details are in [18]. These two theorems laid foundation for several corollaries, also proved in [3]. One of the most important (apart from the one mentioned above, about strict inequality) states that if the conditions of this theorem are met and each child node gives better *average* results than its parent on the *child's* competence set, then adding nodes to the tree decreases the error on the respective set (on which the conditions are met).

Unfortunately, strictly meeting assumptions of those theorems is not easy on examples that were not used for training. However, it was not established that those are necessary conditions, just sufficient ones, so, for example, it is not perfectly clear what really happens if one or more assumptions are not met. That is why a bit more detailed analysis is attempted in this paper.

### 2.3. The new theorem concerning error components

The theorems from [3] mentioned several conditions sufficient for the solution to work well and pointed at several places in which the inequality in Theorem 1 might be made strict, but did not formally answer the question about performance of the solution when not all conditions are perfectly met or how large the difference can be.

The theorem presented below tries to formally shed some light on this. As Theorem 1 was the basic one, the new theorem is a modification of that one.

Below is the additional notation, that was not needed for the previous theorems, but is necessary now.

→ $\tau$ is the notation corresponding to the assumption given in (16) – the relative quality condition on function $C_i$.

$$\tau = \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \sum_{l=1}^{r} \tau_{kl},$$

where $\tau_{kl}$ is just the difference between the error on example $k$ on the coordinate $l$ when the actual competence function $C$ is used, and the error in the case the same estimators would be used (as $\mathbf{1}_{C_i}$ is nonzero if and only if $C_i$ is nonzero) for evaluation of that example, but the results were weighted equally,

$$\tau_{kl} = \left( \sum_{j=0}^{N(i)} \widetilde{\eta}_{i,j}(k)_l \cdot C_i(j, x^{(k)}) \right)^2 - \left( \frac{1}{n_i} \sum_{j=0}^{N(i)} \widetilde{\eta}_{i,j}(k)_l \cdot \mathbf{1}_{C_i}(j, x^{(k)}) \right)^2 . \quad (21)$$

→ The notation $\delta$ is used to describe one of the error components, its full meaning will be better explained during the proof.

$$\delta = \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|S|} \sum_{l=1}^{r} \delta_{kl}$$

$$\delta_{kl} = \left( \sum_{j=0}^{N(i)} \widetilde{\eta}_{i,j}(k)_l \cdot \mathbf{1}_{C_i}(j, x^{(k)}) \cdot \mathbf{1}_{C_i}(j, x^{(k)}) \right)^2 - \left( \sum_{j=0}^{N(i)} \widetilde{\eta}_{i,j}(k)_l^2 \mathbf{1}_{C_i}(j, x^{(k)})^2 \right) \left( \sum_{j=0}^{N(i)} \mathbf{1}_{C_i}(j, x^{(k)})^2 \right). \tag{22}$$

According to the Cauchy-Schwarz inequality, $\delta_{kl}$ is never positive.

The new theorem is:

THEOREM 3. *For any node $i$ in Hierarchical Estimator suppose that:*

*$\mathcal{S}$ is a competence set of node $i$.*

*As in Theorem1, for each example in set $\mathcal{S}$, $n_{ik}$ is constant:*

$$\forall k : (x^{(k)}, y^{(k)}) \in \mathcal{S}, \quad \sum_{j=0}^{N(i)} \mathbf{1}_{C_i}(j, x^{(k)}) = n_i, \tag{23}$$

*where $n_i > 0$.*

*Then*

$$E(\mathcal{S}, \widetilde{g}_i) = \frac{1}{|\mathcal{S}| \cdot n_i} \sum_{j=0}^{N(i)} |\mathcal{S}_{P(i,j)}| \cdot E(\mathcal{S}_{P(i,j)}, \widetilde{g}_{P(i,j)}) + \tau + \frac{1}{n_i^2} \delta. \tag{24}$$

The first term is not suprising – mean of errors of children weighted by sizes of their competence sets within the main set, but there are two more. One that is never positive ($\delta$, it is usually negative) and another one, that corresponds to quality of competence function $C$ relative to a function that chooses the same estimators for each example, but weights them equally ($\tau$). This one can quite easily be kept 0.

*Proof.* The proof is analogical to that of Theorem 1. First, we take squared error definitions (including Eq. (8) and (13)):

$$E(\mathcal{S}, \widetilde{g}_i) = \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|S|} \sum_{l=1}^{r} \left( \widetilde{g}_i(x^{(k)})_l - y_l^{(k)} \right)^2,$$

and apply the main equation for the response (2) to them:

$$E(\mathcal{S}, \widetilde{g}_i) = \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|S|} \sum_{l=1}^{r} \left( \left( \sum_{j=0}^{N(i)} \left( \widetilde{g}_{P(i,j)}(x^{(k)}) \right)_l \cdot C_i(j, x^{(k)}) \right) - y_l^{(k)} \right)^2$$

As the sum of $C_i(j, x^{(k)})$ by definition (Eq. (3)) equals to 1 for each example, we can expand the equation and then collapse with a convenient notation of $\eta$ (see Eq. (11)):

$$\begin{aligned} E(\mathcal{S}, \widetilde{g}_i) &= \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \sum_{l=1}^{r} \left( \sum_{j=0}^{N(i)} (\widetilde{g}_{P(i,j)}(x^{(k)}))_l \cdot C_i(j, x^{(k)}) - \sum_{j=0}^{N(i)} C_i(j, x^{(k)}) \cdot y_l^{(k)} \right)^2 \\ &= \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \sum_{l=1}^{r} \left( \sum_{j=0}^{N(i)} ((\widetilde{g}_{P(i,j)}(x^{(k)}))_l - y_l^{(k)}) \cdot C_i(j, x^{(k)}) \right)^2 \\ &= \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \sum_{l=1}^{r} \left( \sum_{j=0}^{N(i)} \widetilde{\eta}_{i,j}(k)_l \cdot C_i(j, x^{(k)}) \right)^2 . \end{aligned}$$

We can extract the term $\tau$ using its definition – Eq. (21):

$$\begin{aligned} E(\mathcal{S}, \widetilde{g}_i) &= \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \sum_{l=1}^{r} \left( \sum_{j=0}^{N(i)} \widetilde{\eta}_{i,j}(k)_l \cdot C_i(j, x^{(k)}) \right)^2 \\ &= \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \sum_{l=1}^{r} \left( \left( \frac{1}{n_i} \sum_{j=0}^{N(i)} \widetilde{\eta}_{i,j}(k)_l \cdot \mathbf{1}_{C_i}(j, x^{(k)}) \right)^2 + \tau_{kl} \right) \\ &= \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \sum_{l=1}^{r} \left( \frac{1}{n_i} \sum_{j=0}^{N(i)} \widetilde{\eta}_{i,j}(k)_l \cdot \mathbf{1}_{C_i}(j, x^{(k)}) \right)^2 + \tau. \end{aligned} \tag{25}$$

Because values $\mathbf{1}_C$ are 0 or 1, raising them to any power greater than 0 does not change them:

$$E(\mathcal{S}, \widetilde{g}_i) = \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \sum_{l=1}^{r} \frac{1}{n_i^2} \left( \sum_{j=0}^{N(i)} \widetilde{\eta}_{i,j}(k)_l \cdot \mathbf{1}_{C_i}(j, x^{(k)}) \cdot \mathbf{1}_{C_i}(j, x^{(k)}) \right)^2 + \tau. \tag{26}$$

At this point we can apply notation $\delta$ (22)

$$\begin{aligned} &E(\mathcal{S}, \widetilde{g}_i) = \\ &= \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \sum_{l=1}^{r} \frac{1}{n_i^2} \left( \left( \sum_{j=0}^{N(i)} \widetilde{\eta}_{i,j}(k)_l^2 \cdot \mathbf{1}_{C_i}(j, x^{(k)})^2 \right) \left( \sum_{j=0}^{N(i)} \mathbf{1}_{C_i}(j, x^{(k)})^2 \right) + \delta_{kl} \right) + \tau. \end{aligned} \tag{27}$$

The fact that, according to the Cauchy-Schwarz inequality, $\delta_{kl}$ is never positive is quite important here.

Assumption (23) requires that $\sum_{j=0}^{N(i)} \mathbf{1}_{C_i}(j, x^{(k)}) = \sum_{j=0}^{N(i)} \mathbf{1}_{C_i}(j, x^{(k)})^2 = n_i$. So we can write:

$$E(\mathcal{S}, \widetilde{g}_i) = \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \sum_{l=1}^{r} \frac{1}{n_i^2} \left( \left( \sum_{j=0}^{N(i)} \widetilde{\eta}_{i,j}(k)_l^2 \cdot \mathbf{1}_{C_i}(j, x^{(k)})^2 \right) n_i + \delta_{kl} \right) + \tau, \tag{28}$$

then extract $\delta$, concurrently simplifying $1/n_i^2 \cdot n_i$ to $1/n_i$

$$E(\mathcal{S}, \widetilde{g}_i) = \frac{1}{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \sum_{l=1}^{r} \frac{1}{n_i} \left( \sum_{j=0}^{N(i)} \widetilde{\eta}_{i,j}(k)_l^2 \cdot \mathbf{1}_{C_i}(j, x^{(k)})^2 \right) + \tau + \frac{1}{n_i^2}\delta, \tag{29}$$

then reorder sums and factors:

$$E(\mathcal{S}, \widetilde{g}_i) = \sum_{j=0}^{N(i)} \frac{1}{|\mathcal{S}| \cdot n_i} \sum_{k=1}^{|\mathcal{S}|} \mathbf{1}_{C_i}(j, x^{(k)})^2 \cdot \sum_{l=1}^{r} \widetilde{\eta}_{i,j}(k)_l^2 + \tau + \frac{1}{n_i^2}\delta.$$

In this form it is easy to apply the definition of the squared error (9) and the observation about $\eta$ (12), remembering that raising $\mathbf{1}_C$ to a positive power does not change it:

$$E(\mathcal{S}, \widetilde{g}_i) = \sum_{j=0}^{N(i)} \frac{1}{|\mathcal{S}| \cdot n_i} \sum_{k=1}^{|\mathcal{S}|} \mathbf{1}_{C_i}(j, x^{(k)}) \cdot \widetilde{e}_{(i,j)}(k) + \tau + \frac{1}{n_i^2}\delta \tag{30}$$

and use the fact that $\mathbf{1}_C$ is a characterstic function of $\mathcal{S}_{P(i,j)}$ to apply Eq. (13)

$$E(\mathcal{S}, \widetilde{g}_i) = \frac{1}{|\mathcal{S}| \cdot n_i} \sum_{j=0}^{N(i)} |\mathcal{S}_{P(i,j)}| \cdot E(\mathcal{S}_{P(i,j)}, \widetilde{g}_{P(i,j)}) + \tau + \frac{1}{n_i^2}\delta.$$

This ends the proof.

Analogically to Corollary 2 in [3], we can show that $\delta = 0$ is in fact a rather special case, so in most cases it is negative.

COROLLARY 1 (Of $\delta$). *$\delta_{kl}$ is zero only if all errors of used approximators are the same:* $\delta_{kl} = 0 \Longrightarrow \forall j, o : \mathbf{1}_{C_i}(j, x^{(k)}) > 0 \wedge \mathbf{1}_{C_i}(o, x^{(k)}) > 0 \quad \widetilde{\eta}_{i,o}(k)_l = \widetilde{\eta}_{i,j}(k)_l$

*Proof.* Because non-positiveness of $\delta_{kl}$ (22) comes from the Cauchy-Schwarz theorem, it could only be 0 if the two vectors for which it is applied were linearly dependent. In the case of two real non-null vectors, one of them would have to be identical to the second one, just scaled by some number. This should apply to vectors $\left(\mathbf{1}_{C_i}(j, x^{(k)})\right)_{j=0}^{j=N(i)}$ and $\left(\widetilde{\eta}_{i,j}(k)_l \cdot \mathbf{1}_{C_i}(j, x^{(k)})\right)_{j=0}^{j=N(i)}$, so each $\widetilde{\eta}_{i,j}(k)_l$ should have the same value, which is the thesis of the corollary.

#### Change of the error in the node during adding a subtree

For assessing the plausibility of Hierarchical Estimator, the following observation, based on Theorem 3, may be of use. If the assumptions of Theorem 3 hold, then:

$$E(\mathcal{S}, \widetilde{g}_i) - E(\mathcal{S}, g_i) = -\sum_{j=0}^{N(i)} \frac{|\mathcal{S}_{P(i,j)}|}{|\mathcal{S}| \cdot n_i} \left( E(\mathcal{S}_{P(i,j)}, g_i) - E(\mathcal{S}_{P(i,j)}, \widetilde{g}_{P(i,j)}) \right) + \tau + \frac{1}{n_i^2}\delta. \tag{31}$$

That equation may be used to describe the difference of the error in the node with $(E(\mathcal{S}, \widetilde{g}_i))$ and without $(E(\mathcal{S}, g_i))$ the subtree rooted in it, in the manner similar to the thesis of Theorem 3. One of the components of that difference $(1/n_i^2 \cdot \delta)$ is never positive (see Eq. 22), and is negative if only children errors differ on some examples, as indicated by Corollary 1. Another one ($\tau$, a relative quality of the competence function, Eq. 21) can be kept at 0 if needed. If the whole difference is negative, the existence of the subtree decreases the solution error on the given set. Of course, for this to happen, the remaining component, a mean of differences between the mean errors of the estimator in the node and the mean errors of children of the node (with their subtrees, if they have them) on the children competence sets, should not cause increase greater than the decrease caused by $\tau + 1/n_i^2 \cdot \delta$. On the training set, this increase is guaranteed to be non-positive (see pt. 3 in the learning algorithm in Sect. 1.2.3.). Keeping it low on unknown examples is one of the main concerns when creating competence functions and dividing training set [3].

The proof begins with addition of the term $E(\mathcal{S}, g_i)$ to both sides:

$$E(\mathcal{S}, \widetilde{g}_i) = E(\mathcal{S}, g_i) - \sum_{j=0}^{N(i)} \frac{|\mathcal{S}_{P(i,j)}|}{|\mathcal{S}| \cdot n_i} \big(E(\mathcal{S}_{P(i,j)}, g_i) - E(\mathcal{S}_{P(i,j)}, \widetilde{g}_{P(i,j)})\big) + \tau + \frac{1}{n_i^2}\delta.$$

Then, we can transform the left side according to Theorem 3 (Eq. 24), achieving:

$$\begin{aligned} &\frac{1}{|\mathcal{S}| \cdot n_i} \sum_{j=0}^{N(i)} |\mathcal{S}_{P(i,j)}| \cdot E(\mathcal{S}_{P(i,j)}, \widetilde{g}_{P(i,j)}) + \tau + \frac{1}{n_i^2}\delta = \\ &= E(\mathcal{S}, g_i) - \sum_{j=0}^{N(i)} \frac{|\mathcal{S}_{P(i,j)}|}{|\mathcal{S}| \cdot n_i} \big(E(\mathcal{S}_{P(i,j)}, g_i) - E(\mathcal{S}_{P(i,j)}, \widetilde{g}_{P(i,j)})\big) + \tau + \frac{1}{n_i^2}\delta \quad . \end{aligned} \tag{32}$$

Next we can subtract the term $\tau + \frac{1}{n_i^2}\delta$ from both sides and arrange the sums differently on the right side:

$$\begin{aligned} &\frac{1}{|\mathcal{S}| \cdot n_i} \sum_{j=0}^{N(i)} |\mathcal{S}_{P(i,j)}| \cdot E(\mathcal{S}_{P(i,j)}, \widetilde{g}_{P(i,j)}) = \\ &= \sum_{j=0}^{N(i)} \frac{|\mathcal{S}_{P(i,j)}|}{|\mathcal{S}| \cdot n_i} \big(E(\mathcal{S}_{P(i,j)}, g_i) - \big(E(\mathcal{S}_{P(i,j)}, g_i) - E(\mathcal{S}_{P(i,j)}, \widetilde{g}_{P(i,j)})\big)\big) \\ &= \frac{1}{|\mathcal{S}|} \sum_{j=0}^{N(i)} \left( \frac{1}{n_i} \cdot |\mathcal{S}_{P(i,j)}| E(\mathcal{S}_{P(i,j)}, g_i) \right) \\ &\quad - \sum_{j=0}^{N(i)} \frac{|\mathcal{S}_{P(i,j)}|}{|\mathcal{S}| \cdot n_i} \left(E(\mathcal{S}_{P(i,j)}, g_i) - E(\mathcal{S}_{P(i,j)}, \widetilde{g}_{P(i,j)})\right). \end{aligned}$$

We just extracted another term that is on the right side of (31) $-\sum_{j=0}^{N(i)} \frac{|\mathcal{S}_{P(i,j)}|}{|\mathcal{S}| \cdot n_i} \big(E(\mathcal{S}_{P(i,j)}, g_i) - E(\mathcal{S}_{P(i,j)}, \widetilde{g}_{P(i,j)})\big)$, so we can cancel it out of the

equation, which then achieves the form:

$$\frac{1}{|\mathcal{S}|}\sum_{j=0}^{N(i)}\left(\frac{1}{n_i}\cdot|\mathcal{S}_{P(i,j)}|E(\mathcal{S}_{P(i,j)},g_i)\right)=E(\mathcal{S},g_i).$$

Again, we will transform the left side. As $\mathbf{1}_{C_i}$ is a characteristic function of $\mathcal{S}_{P(i,j)}$, we can expand the mean squared error using definitions (10) and (13). Then rearrange sums again:

$$\begin{aligned}\frac{1}{|\mathcal{S}|}\sum_{j=0}^{N(i)}\left(\frac{1}{n_i}\cdot|\mathcal{S}_{P(i,j)}|E(\mathcal{S}_{P(i,j)},g_i)\right)&=\frac{1}{|\mathcal{S}|}\sum_{j=0}^{N(i)}\left(\frac{1}{n_i}\sum_{k=1}^{|S|}e_{(i)}(k)\cdot\mathbf{1}_{C_i}(j,x^{(k)})\right)\\&=\frac{1}{|\mathcal{S}|}\sum_{k=1}^{|S|}\left(\frac{1}{n_i}\cdot e_{(i)}(k)\cdot\sum_{j=0}^{N(i)}\mathbf{1}_{C_i}(j,x^{(k)})\right).\end{aligned}$$

Because the assumption (23): $\sum_{j=0}^{N(i)}\mathbf{1}_{C_i}(j,x^{(k)})=n_i$ still holds, we may use the definition of the mean square error (13) and get

$$\frac{1}{|\mathcal{S}|}\sum_{k=1}^{|S|}\left(\frac{1}{n_i}\cdot e_{(i)}(k)\cdot\sum_{j=0}^{N(i)}\mathbf{1}_{C_i}(j,x^{(k)})\right)=\frac{1}{|\mathcal{S}|}\sum_{k=1}^{|S|}(e_{(i)}(k))=E(\mathcal{S},g_i).$$

So the Equation (31) is true.

**Change of the error during tree growing.**

The last observation will be described informally here, but the analogical corollary with the more formal proof can be found in [3] (Corollaries 5 and 6). It concerns a change of the error of the whole tree when a new subtree is added for a given node. Obviously in such a case $E(\mathcal{S},\widetilde{g}_i)$ changes from $E(\mathcal{S},g_i)$ to a different value, as described by Eq. (31). This causes a change in one of the $E(\mathcal{S}_{P(u,j)},\widetilde{g}_{P(u,j)})$ of its parent $u$, proportionally to the size of the competence set. The same thing happens one level up and the change is propagated to the root and the whole estimator.

## 3. Discussion

The theorem proved in this article specifically gives the components of the mean squared error for Hierarchical Estimator:

1. The error of the estimator in nodes $E(\mathcal{S},g_i)$, both in leaves (where they are $E(\mathcal{S},\widetilde{g}_i)$) and internal nodes.

2. The relative quality of competence function $\tau$. This quality is measured with respect to the reference function that selects the same children as the assessed one, but weights them equally (and has, by definition, $\tau = 0$).

3. $\delta$ which is never positive and is negative if only children results differ on an example, so usually reduces the error.

It requires the number of used estimators in a given node to be constant. This can be easily forced by always using $n_i$ estimators that are considered best and possibly giving some of them very low weights. However, this can influence the term $\tau$, so developing a theorem lifting the requirement seems to be urgent. A possible way to do that may be to reuse the technique from Theorem 2 presented in [3].

Perhaps the most important conclusion that could be drawn from the theoretical considerations above, especially Theorem 3 and Corollary 1, is that the mean squared error of the whole will be lower than the weighted mean of errors of the involved children nodes and estimator in the node if more than one of them are used and they have non-identical errors. Though a similar conclusion may be drawn from Theorems in [3] here it is described a bit more precisely. This decrease in the error can be reinforced if the competence function is able to assign greater weights to the children that give lower errors, but it is not necessary.

An improvement over the theoretical basis from [3] allows to draw the following conclusion, stronger than before. According the observation from the end of the previous section and other theorems, adding a subtree to a node in the existing tree can lower the mean squared error of the whole Hierarchical Estimator even if we are not able to *assure* that all children nodes have lower errors on their competence sets than their parent, or that the competence function offers gain over the reference function ($\tau$ close to 0). It is just enough that the loss on them does not exceed the gain from $\delta$. Theorems proved in [3] did not allow to state it so clearly.

Such a conclusion is significant because it is generally not easy to *guarantee* that a child node has the lower error on examples that were not available during training. Mostly because it is a difficult task for a competence function to assign the examples to the right estimators, i.e. the ones that would made low errors on them. Failing to do that increases the errors of the approximators that actually received the example. Another, though maybe easier to avoid, problem is that in a given node there may not be any function estimators (in children nor the approximator in the node) that would perform well on a given example because of e.g. generalization problems.

Based on these conclusions, one may try to formulate practical guidelines for construction of detailed solutions, in a manner similar to [3]. For example:

1. It would be good if the competence area represented a truly smaller and somewhat separated problem, i.e. if the child was able to achieve greater accuracy without a significant threat of overfitting, increased learning time or a competence function assigning "wrong" examples.

2. An example should be evaluated by more than one child (possibly including "virtual", the estimator in a given node) so that $\delta$ could be negative.

3. It is better if the children have different errors from each other on the given example rather than similar, to make $\delta$ even lower.

4. Choosing the right children by the competence function seems to be a more important task than assigning them exact weights, because the solution can work well also if $\tau$ are 0 – all chosen children are weighted equally. Still, negative average $\tau$ can decrease the error.

Unsurprisingly, those guidelines are very similar to those from [3]. Some of them are approximated in [3] as a requirement that examples within one competence area should be similar (guideline 1) while training sets should be rather dissimilar (1 and 3), and further considerations about what similarity measure to use follow.

An important trait of all error components found in the theorem described in this article is that they can be directly measured during training and validating, so it is possible to measure where the error comes from, at least to some degree. Refinements of the solution could even automatically use such measures to improve the solution performance.

## 4. References

[1] Bishop C.; *Pattern recognition and machine learning*, Springer, Berlin, Heidelberg, New York, 2006.

[2] Hand D., Mannila H., Smyth P.; *Principles of Data Mining*, MIT Press, 2001.

[3] Brodowski S., Podolak I. T.; *Hierarchical Estimator*, Expert Systems with Applications, 38(10), 2011, pp. 12237–12248.

[4] Hastie T., Tibshirani R., Friedman J.; *The Elements of Statistical Learning*, Springer, Berlin, Heidelberg, New York, 2001.

[5] Russell S. J., Norvig P.; *Artificial Intelligence: A Modern Approach*, Pearson Education, 2003.

[6] Christiani N., Shawe-Taylor J.; *Support Vector Machines and other kernel based learning methods*, Cambridge University Press, 2000.

[7] Scholkopf B., Smola A.; *Learning with kernels*, MIT Press, Cambridge, 2002.

[8] Schapire R.; *The Strength of Weak Learnability*, Machine Learning, 5(2), 1990, pp. 197–227.

[9] Freund Y., Schapire R.; *A decision theoretic generalization of online learning and an application to boosting*, Journal of Computer and System Sciences, 55, 1997, pp. 119–139.

[10] Jordan M., Jacobs R.; *Hierarchical mixtures of experts and the EM algorithm*, Neural Computation, 1994, pp. 181–214.

[11] Saito K., Nakano R.; *A constructive learning algorithm for an HME*, IEEE International Conference on Neural Networks, 3, 1996, pp. 1268–1273.

[12] Quinlan J.; *Learning with continuous classes*, Proceedings of the 5-th Australian Conference on Artificial Intelligence, 1992, pp. 343–348.

[13] Podolak I.; *Hierarchical classifier with overlapping class groups*, Expert Systems with Applications, 34(1), 2008, pp. 673–682.

[14] Pal N., Bezdek J.; *On cluster validity for the fuzzy c-means model*, IEEE Transactions on Fuzzy Systems, 3(3), 1995, pp. 370–379.

[15] Brodowski S.; *A Validity Criterion for Fuzzy Clustering*, in: Jedrzejowicz P., Nguyen N. T., Hoang K. (ed.), *Computational Collective Integlligence – ICCCI 2011*, Springer, Berlin, Heidelberg, 2011.

[16] Bielecki A., Bielecka M., Chmielowiec A.; *Input Signals Normalization in Kohonen Neural Networks*, Lecture Notes in Artificial Intelligence, 5097, 2008, pp. 3–10.

[17] Barszcz T., Bielecka M., Bielecki A., Wójcik M.; *Wind turbines states classification by a fuzzy-ART neural network with a stereographic projection as a signal normalization*, Lecture Notes in Computer Science, 6594, 2011, pp. 225–234.

[18] Brodowski S.; *Adaptujący się hierarchiczny aproksymator*, Master's thesis, Jagiellonian University, 2007.

*Received May 19, 2010*