Expert Systems With Applications 96 (2018) 14–24

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Classification using hierarchical mixture of discriminative learners: How to achieve high scores with few resources?

Elvis Togban*, Djemel Ziou

Département d'Informatique, Faculté des Sciences, Université de Sherbrooke, 2500 Bl. de l'université, Sherbrooke, Québec J1K2R1, Canada

ARTICLE INFO

Article history:
Received 31 July 2017
Revised 16 November 2017
Accepted 22 November 2017
Available online 23 November 2017

Keywords:
Discriminative learner
Hierarchical mixture of experts
Input-dependent weight
Model selection

ABSTRACT

In many real-world classification tasks, discriminative models are widely applied since they achieve good predictive scores. In this paper, we propose a generalized framework for the well-studied Hierarchical mixture of experts (HME) model. HME combines hierarchically several discriminative models through a set of input-dependent weights. We derive from our generalized framework, two models as examples of how we can reduce the number of experts used. Those two examples are based on the choice of the weights functions. We choose a Gaussian-based and a linear softmax as weights, restricting our study to a two-level tree. Experiments on synthetic and real-world datasets show that our models can efficiently reduce the number of experts and outperform some state-of-art algorithms.

1. Introduction

Classification is one of the most important tasks performed by humans and machine systems. It consists in assigning an object based on its attributes to a predefined class or group. Classification is used in many real-world applications such as retrieval (Ksantini, Ziou, Colin, & Dubeau, 2008), object recognition (Vidal-naquet & Ullman, 2003) and fraud detection (Caudill, Ayuso, & Guillén, 2005). It can be done with either a generative model or a discriminative model. The first goal of a generative model is to describe the data by relying on the distribution that generates the attributes and user can rely on it to classify an object (Liu & Wechsler, 2002). In contrast, discriminative models aim to determine the class boundaries directly. In this work, we will focus on the discriminative models, since for the classification tasks, they usually outperform the generative ones (Ng & Jordan, 2002). In the case of complex relationship between the classes and the attributes, previous research use either Kernel method (Kung, 2014; Shawe-Taylor & Cristianini, 2004) or artificial neuronal networks (ANN) (Bishop, 1995; Nguyen, Yosinski, & Clune, 2015; Zhang, Yin, Guo, Yu, & Xiao, 2014) or ensemble methods. On the one hand, a limitation of kernel method is that it is global, viewing the data as a whole and ignoring the local specificities such as the geometric structures. On the other hand, the "*black box*" nature of ANN makes it difficult to establish a cause and effect relationship between local specificities and the predicted results. One possible approach to go beyond these limitations is to break down a complex task into easier tasks. This is how most of real-life complex tasks are processed by living organisms: complex objects are recognized by the brain as a combination of simple objects (Tsunoda, Yamane, Nishizaki, & Tanifuji, 2001), a company generally splits a project into several areas of expertise and the detection of nationwide financial fraud is done dividedly in homogeneous regions. This process can be tiered, one task is split into several subtasks, these latter are in turn split and so on and so forth. Since these are data which are segmented according to this paradigm, we will use the word "*mixture*" to refer to the tasks and the relationships between them. In both generative (Bdiri, Bouguila, & Ziou, 2014) and discriminative (Jordan & Jacobs, 1993) learning literature, hierarchical mixture models have been proposed to model the complex tasks in a *divide-and-conquer* fashion. Specifically in the context of discriminative learning, Hierarchical Mixture of Experts (HME) was introduced by Jordan and Jacobs (1993). The basic idea behind the HME model is to recursively divide a complex problem in a set of sub-problems identified by regions. The spatial limitation of each region is a set of hyperplanes described by an input-dependent weight which indicates how a data sample is inside this region. Each component distribution is a discriminative model acting as an *expert*. HME model follows the same process as the tree-based classification algorithms by recursively splitting the input space in several regions and sub-regions. However, classification tree algorithms such as CART (Breiman, Friedman, Stone, & Olshen, 1984) and C4.5 (Quinlan, 1993) use a '*hard-split*' of the input space. In contrast, HME model softly splits the input data through a set of

* Corresponding author.
E-mail addresses: E.Togban@usherbrooke.ca (E. Togban), Djemel.Ziou@usherbrooke.ca (D. Ziou).

https://doi.org/10.1016/j.eswa.2017.11.046
0957-4174/

weights and can also be viewed as an attractive way to combine classifiers (i.e an ensemble method).

A challenging task for human resources, when considering the formation of a project team, is choosing a small number of experts while keeping a high level outcome. There are several motivations behind the reduction of the number of experts such as project duration, financial cost and available tools (e.g softwares). To implement this reduction, a subset of experts is selected. Each expert receives a subset of tasks, where the cardinality depends on the complexity of the task and his skills. Translating such a paradigm to the data analysis by *HME* leads to the selection of: 1) a suitable number of experts by using learning methods; 2) a region on which each expert operates by choosing appropriate weighting functions; 3) the best structure of the HME. In this paper, we will deal with 2).

The previous works in the *HME* literature select a suitable number of experts by a time consuming procedure which involves the evaluation of a given criterion for several models (Ahmed & Campbell, 2011; Bishop & Svenskn, 2003; Fritsch, Finke, & Waibel, 1997). Xu, Jordan, and Hinton (1995) proposed a mixture model without hierarchical structure using a Gaussian-based weights function and can be viewed as an alternative model to a one-level HME. With this model, Ramamurti and Ghosh (1996) use 15 experts to approximate a function; while for the same result, they applied a binary HME tree using 256 experts. Moreover, Moerland proposed several alternatives to a one-level HME by choosing appropriate weighting functions which describe complex boundaries (Moerland, 2000, chap 4). Its results illustrate that with these alternative models, the number of experts can be reduced while keeping a high classification score. On the one hand, in the HME model, only weights describing a linear boundary were applied. However, the use of this type of weights will not be adequate if for example, the tasks exhibit several local specificities (e.g. each region has its own covariance). On the other hand, when a more complex weight was used, previous works were limited to a model without hierarchical structure leading to a coarse split of tasks (Abbasi, Shiri, & Ghatee, 2016b; Moerland, 2000; Shahbaba & Neal, 2009). Nevertheless, a hierarchical splitting can lead to a better refined distribution of tasks.

To address these limitations, we go a step further by developing a generalized hierarchical mixture of discriminative learners where a specific weighting function can be used at each level of the hierarchy. The goal is to keep the advantages of the tree-structured architecture while including the benefit of different weighting functions. As examples of how the number of experts can be reduced, we derive two new models from the generalized one according to the choice of the weights. We show by experiments on several benchmark datasets that these models can use few experts while keeping a high classification score. Moreover, these models outperform some relevant state-of-art algorithms. The remaining of this paper is organized as follows. In Section (2), we introduce the Generalized Hierarchical mixture of discriminative learners, its relationships with some models and the strategy of reduction of the number of experts. In Section (3), we present the learning algorithms. Finally, some experimental results are shown in Section (4), followed by a conclusion.

2. Generalized hierarchical mixture of discriminative learners

2.1. The model

Let $x \in \mathbb{R}^D$ and $y \in \{1 \cdots C\}$ denote respectively the input variable and the corresponding class label. A parametric discriminative model is defined by the distribution $p(y|x, \theta)$ of the class label y, conditioned on the variable x and a set of parameters θ. The goal of this model is to predict the class label $\hat{y}$ given a new $\hat{x}$. However, there is at least two cases where this prediction task will fail. The first case occurs when the data are incomplete i.e there is some missing data or non-observable features in the process of explanation of y. For example, we want to predict the presence or absence (modeled by y) of a flu disease given the explanatory variable x. We have the observed variables (x, y) but we do not have information about the weather season (*spring, summer, autumn, winter*) during which these variables have been observed. Since the weather influences the presence/absence of flu (Davis, Rossier, & Enfield, 2012), it will be hard to correctly make a prediction with this incomplete data. Therefore, we can introduce a multinomial variable z denoting the weather season. Through a mixture model of four components, we can take into account this issue where each mixture weight will identify the data belonging to each season and the corresponding component distribution will learn the presence/absence of flu in that season. The second case occurs when the relationship between x and y is too complex to be understood by the distribution $p(y|x, \theta)$. For example, in a binary classification problem, when one class is included in the other, a mixture of several discriminative models can be used to locally approximate this complex class boundary. Formally, a mixture of discriminative model can generally be defined as:

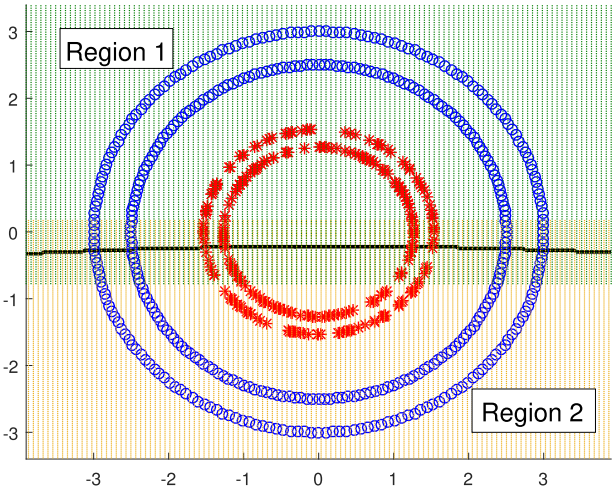


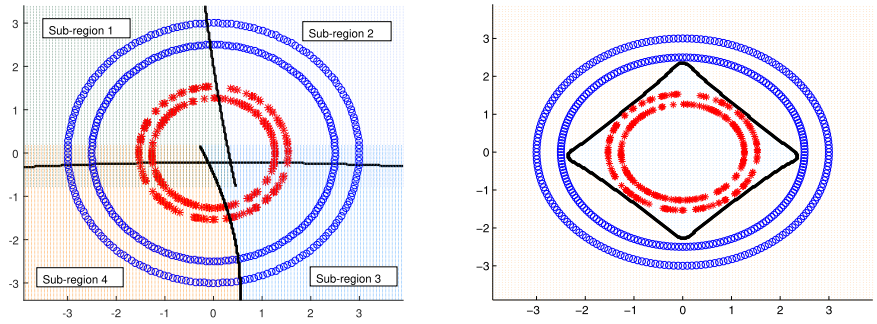
Fig. 1. Division in regions. The black line denotes the boundary of the regions.

$$p(y|x, \Psi) = \sum_{i=1}^{K} \pi_i(x)\, p(y|x, \theta_i);$$
$$s.t\ \pi_i(x) \geq 0;\ \sum_{i=1}^{K} \pi_i(x) = 1 \quad (1)$$

where Ψ is the set of all parameters, K the number of mixture components and θ_i is the parameters of the ith component distribution $p(y|x, \theta_i)$. Basically, the input space containing x is softly split into K *regions* and the probability that the data instance x lie in the ith region is given by $\pi_i(x)$. Then, each component $p(y|x, \theta_i)$ discriminates among the data belonging to the ith region. The mixture weight $\pi_i(x)$ is an input-dependent parametric function describing the spatial boundary of the ith region (Fig. 1).

Now suppose that each region i described by $\pi_i(x)$ and associated with $p(y|x, \theta_i)$ can be split into M_i sub-regions. Therefore, $p(y|x, \Psi)$ can be written as:

$$p(y|x, \Psi) = \sum_{i=1}^{K} \pi_i(x) \sum_{j=1}^{M_i} \omega_{j|i}(x)\, p\big(y|x, \theta_{ij}\big)$$
$$s.t\ \pi_i(x) \geq 0;\ \omega_{j|i}(x) \geq 0$$



(a) Division in sub-regions.

(b) Class boundary generated by 4 classifiers.

Fig. 2. Illustration of the HMD splitting process and the final decision boundary.

$$s.t \sum_{i=1}^{K} \pi_i(x) = 1;\ \sum_{j=1}^{M_i} \omega_{j|i}(x) = 1 \tag{2}$$

where $\omega_{j|i}(x)$ denotes the probability that the data instance x lie in the j^{th}*sub-region* given the fact that it lies in the i^{th} *region*. The numbers K and $\{M_i\}_{i=1}^{K}$ depend on the data, the choice of the component distributions and the weights. By repeating this reasoning on each sub-region, we can derive a hierarchical model where each level is composed of a set of mixtures (see Fig. 2). Each weighting function $\omega_{j|i}(x)$ acts like a window selecting the data to be discriminated by the component $p(y|x, \theta_{ij})$.

At each level of the hierarchy, the mixture weights can take many forms as long as their output is a probability. We suggest that they can be defined as follows:

$$\pi_i(x) = \frac{\mathcal{F}_i(x)}{\sum_{l=1}^{K} \mathcal{F}_l(x)};\ \omega_{j|i}(x) = \frac{\mathcal{G}_{j|i}(x)}{\sum_{l=1}^{M_i} \mathcal{G}_{l|i}(x)}$$

where $\mathcal{F}_l(x)$ and $\mathcal{G}_{l|i}(x)$ are two positive parametric functions with different forms. To simplify parameter estimation, $\mathcal{F}_l(x)$ and $\mathcal{G}_{l|i}(x)$ can be taken as an exponential function: $\mathcal{F}_l(x) \equiv \exp(\zeta_l(x))$, $\mathcal{G}_{l|i}(x) \equiv \exp\left(\vartheta_{l|i}(x)\right)$ where $\zeta_l(x)$ and $\vartheta_{l|i}(x)$ are input-dependent. Therefore, the form of π_i and $\omega_{j|i}$ depend on the choice of $\zeta_l(x)$ and $\vartheta_{l|i}(x)$. We called the proposed model and described in Eq. (2) generalized hierarchical mixture of discriminative learners (HMD). This model can be viewed as: 1) a tree where the leaves are the final component distributions and the branches are the mixture weights. Each inner node (id. for the root) receives the weighted sum of its child outputs and represents a specific region (mode) of the data. 2) a hierarchical manner to combine multiple local classifiers defined by $p(y|x, \theta_{ij})$. In the remaining of this paper, each component distribution is chosen as a multinomial logistic regression and we have for $i = 1 \cdots K,\ j = 1 \cdots M_i$:

$$p\left(y = c|x;\ \theta_{ij}\right) = \begin{cases} \frac{e^{\theta_{ijc}^T \tilde{x}}}{1+\sum_{h=1}^{C-1} e^{\theta_{ijh}^T \tilde{x}}} & c \neq C \\ \frac{1}{1+\sum_{h=1}^{C-1} e^{\theta_{ijh}^T \tilde{x}}} & c = C \end{cases} \tag{3}$$

where C is the number of class labels, $\theta_{ijh} \in \mathbb{R}^{D+1}$ and $\tilde{x} = \begin{pmatrix} 1 \\ x \end{pmatrix}$. The choice of this classifier is motivated by the fact that linear models are easily interpretable. However, our model can use other complex models like ANN as local classifiers.

2.2. Relationships between HMD and other models

2.2.1. Relation to hierarchical mixture of experts (HME)

Hierarchical Mixture of Experts (Jordan & Jacobs, 1993) (HME) is a well-known discriminative hierarchical mixture model. Let us recall that we can obtain the *HME* model from Eq. (2) by setting ζ_l and $\vartheta_{l|i}$ as linear functions of the input x. The weights $\pi_i(x)$ and $\omega_{j|i}(x)$ take then the following expressions:

$$\pi_i(x) = \frac{\exp\left(\eta_i^T \tilde{x}\right)}{\sum_{l=1}^{K} \exp\left(\eta_l^T \tilde{x}\right)};\ \omega_{j|i}(x) = \frac{\exp\left(\nu_{ij}^T \tilde{x}\right)}{\sum_{l=1}^{M_i} \exp\left(\nu_{il}^T \tilde{x}\right)} \tag{4}$$

Where η_l and $\nu_{il} \in \mathbb{R}^{D+1}$. Although our model (*HMD*) shares some similarities with the *HME* model, there are significant differences. Both *HMD* and *HME* models use a *divide-and-conquer* strategy to tackle complex classification tasks. A set of input-dependent weights is used by the two models to affect each classifier to a specific sub-task. The major difference between our model and the HME one resides in the way of affecting a sub-task to a classifier. In the HME model, the weights $\pi_i(x)$ and $\omega_{j|i}(x)$ have the same form (a linear softmax function). In contrast, at each level of the *HMD* tree, the form of the weighting functions can be different from one level to another. Several forms of the weighting functions can be chosen and placed each one of them at a specific level according to their properties. By doing so, we can combine the advantages of different types of weighting functions and improve the repartition of the tasks among the classifiers. Therefore, our model is not restricted to the use of linear softmax as weights. Specifically, if each level of the tree has a semantic meaning, a specific weighting function can be defined for each one. In addition, a more complex weighting function can be used at all levels of the tree. Considering these characteristics, our model can be viewed as a generalization of the *HME* model. The component distributions (classifiers) in the HME literature is called "experts" and we will keep this name for our model too.

2.2.2. Relation to localized mixture of experts (LME) and others

The *LME* model was introduced by Xu et al. (1995) and can be obtained from Eq. (1) by giving to $\pi_i(x)$ the following expression:

$$\pi_i(x) = \frac{\alpha_i g(x, \mu_i, \Sigma_i)}{\sum_{l=1}^{K} \alpha_l g(x, \mu_l, \Sigma_l)};\ i = 1 \cdots K$$

$$\sum_{l=1}^{K} \alpha_l = 1;\ \alpha_l > 0 \tag{5}$$

where $g(x, \mu_l, \Sigma_l)$ is the Gaussian distribution with mean μ_l and covariance matrix Σ_l. LME is equivalent to a one level *HMD* where

$\zeta_i = \ln(\alpha_i) - \frac{1}{2}\left[(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i) + \ln|\Sigma_i| + D\ln(2\pi)\right]$. By using this weighting function, *LME* leads to more local regions than the one obtained by *HME*. However *LME* loose the advantage of the hierarchical structure even if its ability to capture heterogeneous covariance Σ_l improves the classification score. Moreover, when more complex weights (e.g ANN) were used, previous works on discriminative mixtures have no hierarchical structure (Abbasi et al., 2016b; Masoudnia & Ebrahimpour, 2014; Yuksel, Wilson, & Gader, 2012). All these complex weights can be defined according to the expressions of ζ_i and the models that used them can be viewed as an instance of *HMD* with a specific local classifiers (e.g ANN).

2.2.3. Number of experts in HME/LME

In the HME/LME literature, a suitable number of experts is determined by model selection methods (Ahmed & Campbell, 2011; Bishop & Svenskn, 2003) or by a growing and pruning process (Fritsch et al., 1997; Ramamurti & Ghosh, 1996). These approaches can be time consuming because they require the evaluation of a given criterion for several models. In this paper, we are not looking for the optimal number of experts. Rather, we will limit ourself to a suboptimal solution, which is computationally efficient. The idea is to reduce the number of experts by using appropriate weighting functions. Indeed, if the tasks are well distributed among the experts, a waste of resources can be avoided. For example, in the case of *LME* model, Ramamurti and Ghosh (1996) show that for a regression task, one could use very few experts to achieve the same scores as a *HME* model using a wide number of experts. In their study, they approximate a function with *LME* using 15 experts; while they obtained the same score with a binary *HME* tree using 256 experts. Moreover, in our experiment, we performed a classification task on the two-spirals problem (Lang, 1988) with *LME* using 24 experts and we reached the 100% classification score (on the training set) outperforming the one obtained using a binary *HME* tree of 1024 experts (Waterhouse & Robinson, 1994). This result confirms the earlier research on the use of *LME* for classification (Moerland, 1999). However, a hierarchical splitting leads to a better refined distribution of tasks rather than a coarse one when we do not have hierarchical structure. In the following sections, we will give two examples of how our model can reduce the number of experts by an appropriate combination of weighting functions while keeping the hierarchical structure.

2.3. Choice of the weights

Since the weighting functions are used in our model to classify the data in a number of sub-populations (modes or regions in the data space), they can be modeled either by a parametric (Abbasi et al., 2016b) or nonparametric approach (Shahbaba & Neal, 2009). On the one hand, a parametric model summarizes the data by a set of parameters with fixed size which is independent of the number of training instances. The effectiveness of this type of model depends on how the assumption made on the parametric function form is right. To have a better result, one can use a more flexible function (e.g. ANN Abbasi et al., 2016b) and obtain by the same way an almost black-box model. Therefore, there is a trade-off between the flexibility of the parametric function and its black-box nature. On the other hand, in a nonparametric model, the number of parameters is linked to the size of training instances. The nonparametric models are more flexible than parametric ones and do not assume a specific form for the underlying function. However, nonparametric models are slower to train and prone to overfit. Consequently, in this work in order to reduce the training time, we choose a parametric approach to model the weighting functions. There is at least two strategies that can be applied to place the weighting functions in the HMD structure. Using the first strategy, each region can be modeled by a mixture of heterogeneous distributions (i.e. each subregions belonging to each region can follow its own distribution) and a procedure similar to the one applied in El-Zaart and Ziou (2007) can be used to select each of them. Then, the weighting functions are chosen as the posteriori probability to belong to a region. The second strategy is simpler than the first one and the regions at each level of the hierarchy are split with the same form of weighting functions. For the sake of simplicity, we will focus in this work on the second strategy.

Consider the Gaussian-based function used in *LME* model (Eq. (5)) which can be rewritten as:

$$\pi_i(x) = \frac{e^{-\frac{x^T\Sigma_i^{-1}x+\ln|\Sigma_i|}{2}}.e^{\ln(\alpha_i)-\frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i+\mu_i^T\Sigma_i^{-1}x}}{\sum_{l=1}^{K} e^{-\frac{x^T\Sigma_l^{-1}x+\ln|\Sigma_l|}{2}}.e^{\ln(\alpha_l)-\frac{1}{2}\mu_l^T\Sigma_l^{-1}\mu_l+\mu_l^T\Sigma_l^{-1}x}} \tag{6}$$

Defining $a_i = \mu_i^T\Sigma_i^{-1}$ and $b_i = \ln(\alpha_i) - \frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i$, this expression becomes

$$\pi_i(x) = \frac{e^{-\frac{x^T\Sigma_i^{-1}x+\ln|\Sigma_i|}{2}}.e^{\eta_i^T\tilde{x}}}{\sum_{l=1}^{K} e^{-\frac{x^T\Sigma_l^{-1}x+\ln|\Sigma_l|}{2}}.e^{\eta_l^T\tilde{x}}} \tag{7}$$

where $\eta_l = (a_l, b_l)^T$. Assuming that $\Sigma_i = \Sigma, \forall\, i = 1\cdots K$, then the quadratic term $x^T\Sigma_l^{-1}x$ is canceled and the expression of $\pi_i(x)$ in Eq. (7) is equivalent to the one defined in Eq. (4). Therefore, the Gaussian-based weight describes a linear boundary when the data present in the regions share the same covariance. In contrast, a curved boundary is described when the regions are heterogeneous (each region has its own covariance matrix, see Fig. 3). Based on these facts, we derive two examples of how our model can use appropriate weighting functions to reduce the number of experts. The first example uses a different form of weights at each level of the *HMD* tree (*Hybrid weights*) and the second one uses the same form of weights through all the levels (*Homogeneous weights*). Through experiments, we will see which one of these two examples is efficient. In the remaining of this paper, we will restrict ourself to a *HMD* tree composed of two levels but the same principle can be applied to a deeper tree.

2.3.1. Hybrid weights

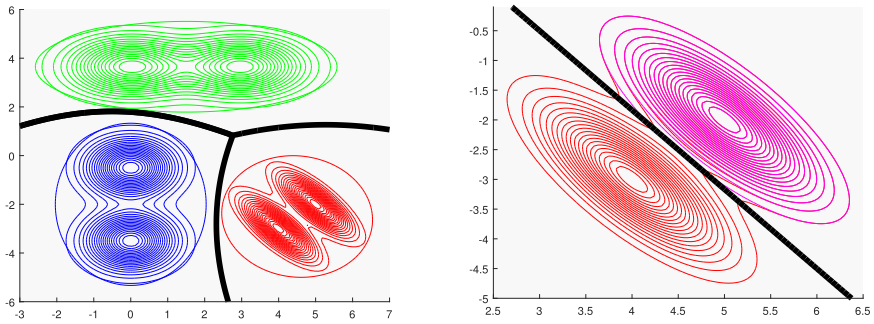
For this first example, we assume that the regions described by $\{\pi_i(x)\}_{i=1}^{K}$ can be heterogeneous and then we choose the expression given by Eq. (5) for the top-level weighting functions. In fact, by using the Gaussian-based function, we can promote its local effect and then use few experts (the depth of the tree is reduced Ramamurti & Ghosh, 1996). By contrast, at the lower level, since the data present in a region represent a task less complex than the original one, we assume that the sub-regions share the same covariance. Instead of using the Gaussian-based function at the lower-level which will lead to the estimation of Σ ($\frac{D^2+D}{2}$ elements), we use its equivalent given by:

$$\omega_{j|i}(x) = \frac{\exp\left(\delta_{ij}^T\tilde{x}\right)}{\sum_{h=1}^{M_i}\exp\left(\delta_{ih}^T\tilde{x}\right)};\ i = 1\cdots K;\ j = 1\cdots M_i \tag{8}$$

where $\delta_{ij} \in \mathbb{R}^{D+1}$. We call this first example *HMD1* model.

2.3.2. Homogeneous weights

In the discriminative mixture literature, many weighting functions describing nonlinear boundaries are used to split the input space such as Gaussian-based function (Moerland, 1999; Ramamurti & Ghosh, 1996; Xu et al., 1995) and neuronal networks (Abbasi et al., 2016b; Moerland, 1999) (see Yuksel et al., 2012 for more weighting functions). At the best of our knowledge, these weighting functions are just applied for mixture model without hierarchical structure. The *HMD* model allows to extend, for example, the use of the Gaussian-based weighting functions to all the levels.



(a) Regions with different covariance matrices.

(b) Regions sharing the same covariance matrix.

Fig. 3. Illustration of the boundaries described by the Gaussian-based weighting function.

Contrary to the first example, we want to capture (if they exist) heterogeneous covariances in each region described by $\pi_i(x)$. Following the notations adopted in the Section (2.2), for fixed i, the lower-level weighting function is expressed as follows:

$$\omega_{j|i}(x) = \frac{\beta_{ij} g(x, \xi_{ij}, \sigma_{ij})}{\sum_{l=1}^{M_i} \beta_{il} g(x, \xi_{il}, \sigma_{il})};\ j = 1 \cdots M_i;$$
$$s.t \sum_{l=1}^{M_i} \beta_{il} = 1;\ \beta_{il} > 0 \tag{9}$$

For the sake of simplicity, we take here σ_{ij} as diagonal covariance matrix. The top-level weight $\pi_i(x)$ keeps the same expression as defined for the first example (*HMD1*). The new model obtained is called *HMD2*. Assuming that $\sigma_{ij} = \sigma$ (all sub-regions share the same covariance), we can convert *HMD2* into *HMD1* but the inverse is not obvious.

3. Parameters estimation using maximum likelihood (ML)

Let $\mathbf{Obs} = \{\mathbf{X}, \mathbf{Y}\}$ be a set of observed data where $\mathbf{X} = \{\vec{x_1}, \cdots, \vec{x_N}\} \in \mathbb{R}^{D \times N}$ is a set of i.i.d. vectors and $\mathbf{Y} = (y_1, \cdots, y_N)^T \in \mathbb{R}^N$ their equivalent class labels; N is the number of observations. Since our generalized model (Eq. (2)) is a mixture model, EM algorithm (Dempster, Laird, & Rubin, 1977) can be used for the training process. Let $\mathcal{L}$ be the *incomplete-log-likelihood*:

$$\mathcal{L} = \sum_{n=1}^{N} \ln \left[\sum_{i=1}^{K} \pi_i(x_n) \sum_{j=1}^{M_i} \omega_{j|i}(x_n)\, p(y_n|x_n, \theta_{ij}) \right] \tag{10}$$

As is usual in the EM literature, we introduce a set of '*hidden*' binary random variables for the n^{th} data: $z_i^{(n)}$ and $z_{j|i}^{(n)}$ respectively for the top-level and the lower-level of the tree. If x_n belongs to the region i then $z_i^{(n)} = 1$ and 0, otherwise. Given the region i, if x_n belongs to the sub-region j then $z_{j|i}^{(n)} = 1$ and 0, otherwise. The *complete-log-likelihood* can be written as follows:

$$\mathcal{L}^c = \sum_{n=1}^{N} \sum_{i=1}^{K} z_i^{(n)} \sum_{j=1}^{M_i} z_{j|i}^{(n)} \ln \left[\pi_i(x_n) \omega_{j|i}(x_n)\, p\big(y_n|x_n, \theta_{ij}\big) \right] \tag{11}$$

In the E-step we compute the following expectation:

$$Q\big(\Psi, \Psi^{(t)}\big) = \sum_{n=1}^{N} \sum_{i=1}^{K} h_i^{(n)} \sum_{j=1}^{M_i} h_{j|i}^{(n)} \ln \left[\pi_i(x_n) \omega_{j|i}(x_n)\, p\big(y_n|x_n, \theta_{ij}\big) \right] \tag{12}$$

where $\Psi^{(t)}$ is the parameters at the tth iteration; $h_i^{(n)}$ and $h_{j|i}^{(n)}$ are respectively the expected values of $z_i^{(n)}$ and $z_{j|i}^{(n)}$ and their expressions are given by

$$h_i^{(n)} = \frac{\pi_i(x_n) \sum_{j=1}^{M_i} \omega_{j|i}(x_n)\, p\big(y_n|x_n, \theta_{ij}\big)}{\sum_{i=1}^{K} \pi_i(x_n) \sum_{j=1}^{M_i} \omega_{j|i}(x_n)\, p\big(y_n|x_n, \theta_{ij}\big)} \tag{13}$$

and

$$h_{j|i}^{(n)} = \frac{\omega_{j|i}(x_n)\, p\big(y_n|x_n, \theta_{ij}\big)}{\sum_{j=1}^{M_i} \omega_{j|i}(x_n)\, p\big(y_n|x_n, \theta_{ij}\big)} \tag{14}$$

Eq. (12) can be split in three parts where each one containing respectively the parameters of the top and lower level weighting functions as well as the experts parameters. The following equations give the maximization operations performed in the M-Step:

$$G_1^{(t+1)} = argmax_{G_1} \sum_{n=1}^{N} \sum_{i=1}^{K} h_i^{(n)} \ln \left[\pi_i(x_n) \right] \tag{15}$$

$$G_2^{(t+1)} = argmax_{G_2} \sum_{n=1}^{N} \sum_{i=1}^{K} h_i^{(n)} \sum_{j=1}^{M_i} h_{j|i}^{(n)} \ln \left[\omega_{j|i}(x_n) \right] \tag{16}$$

$$\theta_{ij}^{(t+1)} = argmax_{\theta_{ij}} \sum_{n=1}^{N} h_i^{(n)} h_{j|i}^{(n)} \ln \left[p\big(y_n|x_n, \theta_{ij}\big) \right] \tag{17}$$

where $G_1^{(t+1)}$ and $G_2^{(t+1)}$ are respectively the parameters of the top-level and the lower-level weighting functions at the $(t+1)$th iteration. Following the definition of this general EM framework for our hierarchical model, we derive in the next subsections the learning algorithms for the *HMD1* and *HMD2* models.

3.1. Parameters estimation for the HMD1 model

In the case of *HMD1*, $G_1 = \{\alpha_i, \mu_i, \Sigma_i\}_{i=1}^{K}$ and $G_2 = \left\{\delta_{ij}\right\}_{ij=11}^{KM_i}$. Substituting the expression of $\pi_i(x_n)$ (Eq. (5)) in Eq. (15) will lead to $3 \times K$ nonlinear optimization problems. To avoid this computational burden, the EM algorithm can be performed on the joint probability $p(y_n|x_n, \Psi) p(x_n|\Psi)$ where $P(x_n|\Psi) = \sum_{l=1}^{K} \alpha_l g_l(x_n, \mu_l, \Sigma_l)$ (Moerland, 1999; Xu et al., 1995). This is equivalent to replace the expression of $\pi_i(x_n)$ in Eqs. (10)–(15) by $\alpha_i g_i(x_n, \mu_i, \Sigma_i)$. Once this substitution done, we obtain the

Algorithm 1. Learning algorithm for the *HMD1* model.

1. Initialization:
   (a) set the parameters $\{\mu_i, \Sigma_i\}_{i=1}^{K}$ through the K-means algorithm
   (b) choose randomly the parameters $\{\alpha_i\}_{i=1}^{K}$ and normalize them.
   (c) choose randomly the parameters $\{\delta_{ij}\}_{ij=11}^{KM_i}$ and $\{\theta_{ijc}\}_{ijc=111}^{KM_iC}$
2. E-step: Update $Q\left(\Psi, \Psi^{(t)}\right)$ via equation (12)
3. M-step: Maximize $Q\left(\Psi, \Psi^{(t)}\right)$ with respect to current value of $\Psi^{(t)}$ and update Ψ via equations (16–20).
4. Repeat steps 2 and 3 until convergence.

following update equations for the top-level weight parameters: For $i \in \{1 \cdots K\}$

$$\alpha_i^{(t+1)} = \frac{1}{N}\sum_{n=1}^{N} h_i^{(n)}; \quad \mu_i^{(t+1)} = \frac{\sum_{n=1}^{N} h_i^{(n)} x_n}{\sum_{n=1}^{N} h_i^{(n)}}$$

$$\Sigma_i^{(t+1)} = \frac{\sum_{n=1}^{N} h_i^{(n)} \left(x_n - \mu_i^{(t)}\right)^T \left(x_n - \mu_i^{(t)}\right)}{\sum_{n=1}^{N} h_i^{(n)}} \tag{18}$$

By substituting Eq. (3) and the expression of $\omega_{j|i}(x_n)$ (Eq. 8) in Eqs. (16) and (17), we can estimate the parameters δ_{ij} and θ_{ijc} via a standard nonlinear optimization algorithm (e.g quasi-newton methods) and the gradients are expressed as:

$$\nabla_{\delta_{ij}} Q\left(\Psi, \Psi^{(t)}\right) = \sum_{n=1}^{N} h_i^{(n)} \left[h_{j|i}^{(n)} - \omega_{j|i}(x_n)\right] \tilde{x}_n i \in \{1 \cdots K\} \tag{19}$$

$$\nabla_{\theta_{ijc}} Q\left(\Psi, \Psi^{(t)}\right) = \sum_{n=1}^{N} h_i^{(n)} h_{j|i}^{(n)} \left[v_{cn} - p\left(y_n = c|x_n; \theta_{ij}\right)\right] \tilde{x}_n \quad i \in \{1 \cdots K\},\ j \in \{1 \cdots M_i\} \tag{20}$$

where v_{cn} are binary variables indicating membership of n^{th} data to the c^{th} class and $\tilde{x}_n = (1, x_n)^T$. It is interesting to see that Eq. (19) is similar to the gradient of the weighted multinomial regression logistic where $h_i^{(n)}$ is the weight, $h_{j|i}^{(n)}$ the "*label*" and $\omega_{j|i}(x_n)$ the probability to belong to the jth sub-region. Algorithm 1 summarizes the HMD1 learning procedure.

3.2. Parameters estimation for the HMD2 model

For the *HMD2* model, $G_1 = \{\alpha_i, \mu_i, \Sigma_i\}_{i=1}^{K}$ and $G_2 = \left\{\beta_{ij}, \xi_{ij}, \sigma_{ij}\right\}_{ij=11}^{KM_i}$. The expressions for the top-level weight parameters and the gradient with respect to θ_{ijc} are the same as in Eqs. (18) and (20). The weighting function $\omega_{j|i}(x_n)$ as defined in Eq. (9) can be rewritten in the form of softmax function:

$$\omega_{j|i}(x_n) = e^{\eta_{ij}} / \sum_{l=1}^{M_i} e^{\eta_{il}}$$

where $\eta_{ij} = \ln\left(\beta_{ij}\right) - \frac{1}{2}\left[\left(x_n - \xi_{ij}\right)^T \sigma_{ij}^{-1}\left(x_n - \xi_{ij}\right) + \ln\left|\sigma_{ij}\right| + D\ln(2\pi)\right]$. To ensure that β_{ij} and σ_{ijd} are positive we use the following re-parametrization: $\beta_{ij} = e^{\beta'_{ij}}$ and $\sigma_{ijd} = e^{\sigma'_{ijd}}$. By replacing this expression in Eq. (16), the parameters of $\omega_{j|i}(x_n)$ can be estimated with a nonlinear optimization algorithm and the gradients with respect to β'_{ij}, ξ_{ijd} and σ'_{ijd} are given by:

For $i = 1 \cdots K$, $j = 1 \cdots M_i$ and $d = 1 \cdots D$

$$\nabla\beta'_{ij} = \sum_{n=1}^{N} h_i^{(n)} \left[h_{j|i}^{(n)} - \omega_{j|i}(x_n)\right] \tag{21}$$

Algorithm 2. Learning algorithm for the *HMD2* model.

1. Initialization:
   (a) do the initialization step 1.a and 1.b of algorithm (1).
   (b) For each i, choose randomly the parameters $\{\beta_{ij}\}_{j=1}^{M_i}$ and normalize them.
   (c) Given the data relying on the i^{th} region, do a k-means to set the parameters $\{\xi_{ij}\}_{j=1}^{M_i}$ and $\{\sigma_{ij}\}_{j=1}^{M_i}$
   (c) choose randomly the parameters $\{\theta_{ijc}\}_{ijc=111}^{KM_iC}$
2. E-step: Update $Q\left(\Psi, \Psi^{(t)}\right)$ via equation (12)
3. M-step: Maximize $Q\left(\Psi, \Psi^{(t)}\right)$ with respect to current value of $\Psi^{(t)}$ and update Ψ via equations (16–18) and equations (20–23). 4. Repeat steps 2 and 3 until convergence.

$$\nabla\xi_{ijd} = \sum_{n=1}^{N} h_i^{(n)} \left[h_{j|i}^{(n)} - \omega_{j|i}(x_n)\right] \frac{\left(x_{n,d} - \xi_{ij,d}\right)}{\sigma_{ijd}} \tag{22}$$

$$\nabla\sigma'_{ijd} = \sum_{n=1}^{N} h_i^{(n)} \left[h_{j|i}^{(n)} - \omega_{j|i}(x_n)\right] \left[\frac{\left(x_{n,d} - \xi_{ij,d}\right)^2}{\sigma_{ijd}} - 1\right] \tag{23}$$

where the lower-script d denotes the d^{th} dimension in the input space. Algorithm 2 summarizes the HMD2 procedure.

3.3. How to choose the tree structure?

Although we argue that an appropriate choice of weighting functions can allow to reduce the number of experts, the problem of selecting a suitable tree architecture remains. This can be done by a priori choice of K and M_i or by using criteria like AIC or by cross-validation. The last two options will be very time consuming when the number of experts grows since we have to evaluate all possible combinations of K and M_i. Another simple approach consists in setting K and M_i to a larger number and deleting iteratively the components that do not contribute enough to the overall result. Given Eqs. (13) and (13) we define $S_i = \sum_{n=1}^{N} h_i^{(n)}/N$ and $S_{ij} = \sum_{n=1}^{N} h_i^{(n)} h_{j|i}^{(n)}/N$ as indicator of how region i and expert ij contribute to the overall result. An expert ij (respectively a top-level component i) can be removed if S_{ij} is below a certain threshold. The latter one can be unique for all the levels or level-specifc or can be increase (resp. decrease) from one iteration to another. This procedure is summarized as follows:

1. Set $K = k_{\max}$, $M_i = m_{\max}$ and choose a threshold
2. Run Algorithms 1 or 2
3. Delete the components according to the values of S_i and S_{ij}
4. Update the threshold if it has to vary.
5. Repeat (2)–(4) until there is no change in the structure.

4. Experimental results

To assess the performance of our models, we have considered twelve datasets from UCI collection (Lichman, 2013) and IDA benchmark repository. These datasets are chosen based on the fact that linear classifiers can not achieve a good score on them. Table 1 summarizes the key features of these datasets. For all the dataset, we do not apply any pre-processing methods other than the simple standardization (subtract the mean and divide by the standard deviation). The reported accuracies are evaluated on the test sets and the training times are in seconds.

Usually the nonlinear optimization method used to estimate the parameters in the *HME* model is the *iteratively re-weighted least*

Table 1
Datasets employed in the experimental study.

Datasets	#instances	#features	#classes
banana	5300	2	2
Thyroid	7200	21	3
Forensic Glass	214	9	6
Pen-digit	10 992	16	10
Vowel	990	13	11
Cancer (Breast)	277	9	2
Heart	270	13	2
Magic	19,020	10	2
Balance scale	625	4	3
Waveform	5000	21	3
Vehicle	946	18	4
Satimage	6 435	4	6

squares (IRLS) method. However, in the context of classification with *HME*, it is reported that IRLS perform poorly (Chen, Xu, & Chi, 1999). Consequently, we turn to the use of a limited-memory quasi-Newton algorithm named L-BFGS (Nocedal, 1980). It is worth recalling that each expert is defined as a multinomial logistic regression and several studies have shown the efficiency of L-BFGS in fitting this kind of problem (Malouf, 2002). Especially, L-BFGS outperforms IRLS when it is used to learn a Mixture of Experts (Chen et al., 1999). Moreover, L-BFGS allows us to avoid a heavy computation of the inverse Hessian matrices by approximating them. We used the '*minFunc*' implementation (Schmidt, 2012) of L-BFGS with the convergence tolerance set to 1×10^{-3} over 200 maximum update cycles. We set the EM convergence tolerance to 1×10^{-4} over 50 maximum update cycles. Moreover, since the initialization is random, we run all experiments twice to avoid poor local maxima.

4.1. Comparison between HME, LME, HMD1 and HMD2 models

In this subsection, we compare experimentally, *HME, LME, HMD1* and *HMD2* models. These four models are implemented using MATLAB and the same type of experts is used for all of them (Eq. (3)). The goal is to show how our models can reduce the number of experts just by using an appropriate combination of weighting functions. We ran these algorithms on five top datasets described in Table 1 and 10-fold cross-validation were used for these datasets except for *Pen-digit* for which we applied a 5-fold cross-validation. In the first experiment, we used a range from 4 to 10 experts with $M_i = 2$ and the average accuracy and training time are summarized in Table 2. In the second experiment, we used the procedure of model selection described in Section (3.3) with $k_{\max}$ and $m_{\max}$ set to 10 and the threshold is set to 0.01. The accuracy as well as the average of selected number of experts are summarized in Table 3.

4.1.1. Accuracy

The results differ from one dataset to another and due to space limitations, a full analysis cannot be given here. First, we will discuss the results of the first experiment Table 2. For ***Forensic Glass***, when 8 experts are used, *HME* model achieves a score of 67.74% and this score fells by 3% when the number of experts is reduced to 4. In contrast, with 4 experts, *HMD1, HMD2* and *LME* achieve at least a score of 68%. However, *LME* needs ten experts to achieve a similar score than *HMD1* and *HMD2* using six experts. In contrast, even with ten experts, the accuracy obtained with *HME* is low compared to the one achieved with *HMD1* or *HMD2* using four experts. Similar results are obtained on the ***Vowel*** dataset where in general, *LME* uses at least two more experts to achieve an accuracy close to that obtained by *HMD1* and *HMD2*. For ***Banana***, *LME* and *HMD1* achieve the highest accuracies. However, *HMD1* slightly outperforms *LME* in general. *HME* needs on average ten experts to achieve the accuracy obtained by *HMD1* using four experts. Moreover, *HME* needs more than ten experts to achieve the accuracy of *HMD2* using six experts. Furthermore, *LME* used around two or three experts more than *HMD1* and *HMD2* to achieve the same accuracy. In average, the standard deviation of both *HMD1* and *HMD2* is lower than the one of *HME* and *LME*. However, *HMD1* has often a higher accuracy than *HMD2*.

For the second experiment Table 3, we notice that for all the five datasets (except banana in the case of *HMD1*), our models have selected a smaller number of experts than the two other methods. In terms of accuracy, *HMD2* achieves at least similar results to *LME* and has been outperformed by *HME* only on ***Thyroid*** dataset. In contrast, *LME* and *HME* outperform *HMD1* respectively on ***Vowel*** and ***Thyroid*** datasets. However, we notice that the number of experts selected for *HMD1* is higher than the one of *HMD2*. This difference is due to the fact that, the selected number of component M_i for *HMD1* is always greater than the one of *HMD2* although *HMD1* has a smaller *K*. In fact, in the case of *HMD2*, several experts have a very small value of S_{ij} which is not the case for *HMD1*. A similar observation was also reported in Ramamurti and Ghosh (1996) when comparing the Gaussian-based weighting function to the linear softmax. Therefore, we suggest the use of two thresholds: one for *Si* and another for *Sij*. Especially, in the case of *HMD1*, the threshold for *Sij* can be increased from one iteration to another in order to delete more experts. However, a maximum value has to be imposed to the threshold of *Sij* to avoid the suppression of all experts in *HMD1*.

From these results, we can guess that the combination of several forms of weighting functions is good for the classification. Moreover, these results confirm that the use of hierarchical mixture model can be useful for certain collections. It will be interesting to understand the characteristics of these collections.

4.1.2. Training time

For the case where $M_i = M$, let t_{br} be the top layer branching factor and N_e the number of experts ($N_e = Mt_{br}$). We solved the following number of nonlinear optimization problems: $[N_e + t_{br} + 1]T$ for *HME*, N_eT for *LME*, $[N_e + t_{br}]T$ for *HMD1* and $[N_e + 3t_{br}]T$ for *HMD2* where T is the number of EM iterations. Although *LME* solved the lowest number of nonlinear optimization problems, it is not always the fastest algorithm. In fact, we notice in our experiments that for all models, the training time is governed by T and the number of updates cycle in L-BFGS. Generally, *HME* converged after a large number of EM iterations which explain its high training time compared to other models. From Table 2, we can see that in average, *HMD2* is faster than other methods. This is explained by the fact that *HMD2* converged after a fewer number of EM iterations than other three methods. However, when N_e grows (e.g. $K = 10$ and $M_i = 10$), the number of nonlinear optimizations becomes the major factor in the evaluation of the training time. Consequently, with larger N_e, the fastest method in ascending order is *LME, HMD1, HMD2* and *HME*. Moreover, for the four models, the training time grows almost linearly with the number of training instances *N*. These results suggest that we can use *HMD2* to reduce the training time when fewer experts are used. Furthermore, *HMD1* appears to be a compromise between the accuracy and the training time when we are in presence of a large number of experts.

4.2. Comparison of HMD1 and HMD2 with some state-of-art algorithms

For the sake of comparison with some relevant state-of-art methods for nonlinear data, we conducted our experiments on the

Table 2
Average accuracies and training times for *HME, LME, HMD1* and *HMD2*. The number of experts used is preceded by #. The best result for each number of experts is set in bold.

Models	Average accuracy (%)				Average Training time (sec)			
	#4	#6	#8	#10	#4	#6	#8	#10
HME	84.95 ± 13.96	87.52 ± 14.27	87.93 ± 11.9	87.99 ± 11.79	92.21 ± 133.95	112 ± 196.23	165.55 ± 267.46	214.72 ± 345.85
LME	85.75 ± 12.63	88.03 ± 11.96	88.49 ± 11.24	89.16 ± 10.79	19.44 ± 32.2	23.86 ± 37.24	**25.59** ± 38.43	36.94 ± 60.88
HMD1	**87.41** ± 11.24	**89.09** ± 10.97	**91.19** ± 8.27	**91.32** ± 8.53	15.74 ± 19.6	24.70 ± 34	36.18 ± 51.34	39.78 ± 63.24
HMD2	86.99 ± 11.97	88.91 ± 10.46	90.25 ± 8.82	90.9 ± 8.95	**9.32** ± 7.99	**13.14** ± 11.84	33.06 ± 48.44	**31.4** ± 37.13

Table 3
Average accuracies and selected number of experts for *HME, LME, HMD1* and *HMD2*. The average number of experts used is preceded by #. The best result for each data is set in bold.

Datasets	HME		LME		HMD1		HMD2	
	#Exp.	Acc.	#Exp.	Acc.	#Exp.	Acc.	#Exp.	Acc.
Banana	16	90.17 ± 1.13	34.2	90.3 ± 0.81	22.8	90.36 ± 1.19	10.8	90.13 ± 0.88
Thyroid	40.2	99.01 ± 0.32	13.15	97.57 ± 0.39	11	98.32 ± 0.36	6.29	97.78 ± 0.48
Glass	20.5	66.85 ± 13.41	22.1	68.6 ± 10.3	15.23	71.51 ± 7.46	9.57	73.46 ± 6.47
Pen-digit	37	98.17 ± 0.27	29	98.56 ± 0.24	22.6	98.16 ± 0.28	18.4	98.53 ± 0.34
Vowel	26.3	92.22 ± 3.09	30	97.37 ± 0.98	18.58	94.95 ± 1.3	13.56	96.46 ± 1.19

twelve aforementioned datasets with a 5-cross validation for each of them.

4.2.1. Comparison with some popular state-of-art algorithms

The popular state-of-art methods used are: **Kernel Logistic Regression** (*KLR*) applied with radial Basis function (RBF), **SVM** (*C-SVC* trained with *libsvm* Chang & Lin (2011)), **Random Subspace** (*RS*) (Ho, 1998), a version of Adaboost called **Adaboost.M1** (Freund, Schapire et al., 1996), **Multi-Layer Perceptron** (*MLP*) trained with backpropagation, **Radial Basis Function Network** (*RBFN*) and **C4.5** tree (Quinlan, 1993). We implemented our own MATLAB code for the KLR. For the remaining state-of-art methods, we adopted the implementation provided in Weka 3.9 (Hall et al., 2009). To reduce overfitting, we add *l2-regularization* on the experts of both *HMD1* and *HMD2*. We also set the threshold of S_i to 0.1 and the one of S_{ij} to 0.01. In the case of *HMD1*, after each iteration of the procedure described in Section (3.3), we increase the threshold of S_{ij} by multiply it by 1.6 with 0.1 as maximum value for this threshold. Our experiments show that these thresholds work well in practice.

In our experiments with *Adaboost.M1* and *Random Subspace*, we first chose a logistic regression as '*weak learners*' but the scores obtained were poor. Consequently, we chose unpruned C4.5 trees as '*weak learners*'. It is worth recalling that C4.5 can produce a highly nonlinear boundary between the classes and that logistic regression used as experts in our models can produce only linear boundary. Our main goal for these two methods is to compare them with our models with regard to the number of classifiers used as well as the accuracy obtained on the test datasets. Therefore, we started by setting the number of C4.5 trees as equal to the number of experts used for our methods. When the classification accuracy obtained was very poor we tested a range from 10 to 100 trees with step size 10. For the other state-of-art methods, the performance is only evaluated in terms of accuracy on the test datasets. *MLP* is trained with backpropagation, one hidden layer and we used from 2 to 20 hidden units (sigmoid function) and we kept the configuration that gives the highest accuracy. *RBFN* is trained with one hidden layer and we used from 2 to 14 hidden units. For the C4.5 algorithm, we used a pruned tree and a range from 1 to 10 with step size of 1 for the minimum number of instances per leaf. The configuration giving the highest accuracy is retained.

Table 4 summarized the results where the numbers between the brackets refer respectively to the number of trees used for *Adaboost.M1* and *Random Subspace*, the number of hidden units used for the neural net methods, the type of Kernel used for SVM and the average number of experts used for our models. The results show that both *HMD1* and *HMD2* outperform *Adaboost.M1* on seven datasets. Specifically, even with 100 C4.5 trees on *Cancer* and *Vehicle* datasets, *Adaboost.M1* does not outperform our methods (around four experts or less were used). A similar result is noticed when comparing *RS* to our models. In fact, *HMD1* and *HMD2* outperform *RS* respectively on seven and ten datasets. Therefore, our models can use few linear classifiers to achieve a high accuracy compared to *Adaboost.M1* and *RS* using nonlinear classifiers. However, the training time of our models can be expensive since we have to learn the weighting functions several times[1]. With regard to MLP and RBFN, *HMD1* and *HMD2* outperform both of them on at least eight datasets. KLR outperforms our models on ***Vowel*** and ***Pen-digit*** datasets. By contrast, in our experiments, *SVM* has never outperformed our models. The results obtained on *Banana, Satimage* and *Thyroid* datasets prove that a pruned C4.5 tree can be better than an ensemble of unpruned C4.5 trees. This pruned C4.5 tree achieves a better score than *HMD1* and *HMD2* just on the *Thyroid* dataset. From Table 4, we notice that, in average, *HMD1* and *HMD2* achieve the highest accuracy followed by *MLP*.

4.2.2. Comparison with some recent ensemble methods

We compared our models with some recent ensemble methods (Abbasi, Shiri, & Ghatee, 2016a; 2016b; Cruz, Sabourin, Cavalcanti, & Ren, 2015; Fossaceca, Mazzuchi, & Sarkani, 2015; Kheradpisheh, Sharifizadeh, Nowzari-Dalini, Ganjtabesh, & Ebrahimpour, 2014; Kotsiantis, 2011; Li, Zou, Hu, Wu, & Yu, 2013; Meo, Bachar, & Ienco, 2012; Rahman & Verma, 2013; Ykhlef & Bouchaffra, 2015). DCE-CC dynamically select the base-classifiers (Nearest-neighbor and C4.5) according to their confidence (Li et al., 2013). A similar approach is used by META-DES (Cruz et al., 2015) where five meta-features and meta-classifiers are used to dynamically select the most competent base-classifier. MFSE (Kheradpisheh et al., 2014) is built in 3 stages. In the first stage, an optimal subset of features selected by training several MLP classifiers. In the second stage, MLP is trained on each selected subset of features and the parameters obtained are used to initialize each experts in the next stage. Finally, ME is trained where the weighting functions (MLP) are fed with the whole features and each MLP expert is fed with

[1] We do not provide a comparison of the times, as they cannot be directly compared to m-code execution times since Weka is written in a highly optimized Java code.

Table 4
Comparison of HMD1 (with diagonal covariance), HMD2, Adaboost.M1, Random Subspace, MLP, RBFNN, KLR, SVM and C4.5. The accuracies are in percentage and the numbers following '±' are the standard deviations. The best score per dataset is set in bold. The last column displays the average accuracy and standard deviation obtained by each method (the best is set in bold). The mark "•" indicates that the average is done on the available results.

	Models \ Datasets	Glass	Cancer	Banana	Balance	Pen.	Magic	Veh.	Wav.	Sat.	Heart	Thy.	Vow.	Avg.
Our models	HMD1	75.25 ± 3.40 (5)	**76.19** ± 3.29 (2.6)	**90.36** ± 0.29 (5)	95.18 ± 2.1 (7.6)	98.91 ± 0.12 (4)	**86.81** ± 0.96 (5)	**83.57** ± 0.8 (3.8)	**86.8** ± 1.38 (4.2)	85.24 ± 0.69 (2)	83.7 ± 6.2 (2.4)	97.36 ± 0.42 (2)	94.75 ± 1.27 (3.4)	**87.71**
	HMD2	75.71 ± 3.00 (5)	**76.19** ± 3.03 (3.2)	90.04 ± 0.32 (4.8)	94.07 ± 1.26 (8)	98.92 ± 0.28 (4)	85.08 ± 0.7 (4)	83.45 ± 1.13 (4.2)	**86.7** ± 1.39 (4.2)	85.24 ± 0.69 (2)	**84.44** ± 5.94 (4)	97.36 ± 0.42 (2)	94.65 ± 0.77 (4)	87.65
Classifiers combination	Adaboost.M1	**76.97** ± 6.03 (60)	68.37 ± 5.15 (100)	88.26 ± 0.93 (20)	78.72 ± 3.58 (4)	98.91 ± 0.23 (10)	86.2 ± 3.15 (10)	77.49 ± 2.68 (100)	84.64 ± 1.20 (10)	83.96 ± 0.86 (2)	81.41 ± 4.22 (80)	99.63 ± 0.22 (4)	96.21 ± 1.39 (100)	85.06
	RS	75.48 ± 5.61 (100)	74.66 ± 3.86 (100)	70.85 ± 1.41 (60)	84.06 ± 2.46 (90)	97.67 ± 1.26 (60)	85.51 ± 2.63 (100)	75 ± 2.35 (90)	84.83 ± 1.06 (10)	84.5 ± 0.76 (100)	83.26 ± 3.94 (80)	97.67 ± 1.26 (60)	94.19 ± 2.02 (100)	83.97
ANN	MLP	69.87 ± 5.88 (18)	74.36 ± 5.16 (2)	89.79 ± 0.46 (17)	**96.24** ± 2.03 (18)	95.02 ± 0.26 (20)	86.52 ± 0.47 (16)	82.57 ± 2.76 (15)	85.44 ± 0.88 (5)	**85.99** ± 0.7 (14)	81.30 ± 4.84 (10)	97.36 ± 0.42 (14)	89.9 ± 2.5 (18)	86.19
	RBFN	54.87 ± 8.55 (14)	72.83 ± 5.28 (3)	89.96 ± 0.87 (14)	85.14 ± 2.79 (14)	98.32 ± 0.32 (14)	83.39 ± 0.61 (14)	70.78 ± 2.54 (14)	85.78 ± 1.1 (5)	85.5 ± 2.54 (5)	78.52 ± 2.54 (14)	94.03 ± 0.55 (14)	90.83 ± 3.39 (14)	82.49
Kernel Methods	KLR	67.29 ± 5.86	69.32 ± 2.41	90.13 ± 0.21	91.68 ± 2.3	**99.40** ± 0.15	79.86 ± 0.28	67.96 ± 1.98	81.41 ± 1.04	83.01 ± 0.57	56.66 ± 1.01	94 ± 0.5	**99.09** ± 0.75	81.65
	SVM	65.23 ± 5.54 (RBF)	74.38 ± 3.93 (RBF)	90.31 ± 0.73 (RBF)	91.68 ± 2.07 (lin.)	97.77 ± 0.3 (lin)	65.74 ± 0.13 (RBF)	79.49 ± 2.67 (lin)	86.15 ± 1.14 (RBF)	84.79 ± 0.82 (lin)	83.59 ± 4.98 (lin)	92.58 ± 0.07	88.1 ± 3.26	83.31
	C4.5	66.82 ± 7.26	74.74 ± 5.17	89.04 ± 0.98	78.24 ± 2.59	96.35 ± 0.37	85.07 ± 0.51	72.80 ± 2.64	76.89 ± 1.2	85.08 ± 0.74	81.30 ± 6.13	**99.64** ± 0.21	83.14 ± 4.21	82.42

the selected subset of features. MARK-ELM (Fossaceca et al., 2015) combines by boosting different kernel-based classifiers each one of them is trained on a subset of the training data. Non-Uniform Layered Cluster Oriented Ensemble Classifier (NULCOEC) (Rahman & Verma, 2013) partitioned the data into several clusters at different layers. Then, a set of base classifiers (SVM with RBF kernel) is trained on each cluster and a majority vote is used for the final decision. A genetic algorithm is used to optimize the number of clusters and layers. In Kotsiantis (2011), they combine bagging, boosting, rotation forest and random subspace methods to classify the data. The most recent ensemble methods RTQRT-ME and R-RTQRT-ME (Abbasi et al., 2016b; 2016a) used MLP as weighting functions and experts and introduced a regularization term to increase the diversity of the experts. For this experiment, we used the same number of experts (five) as in Abbasi et al. (2016b, 2016a) and the structure of the tree is determined by a 5-cross validation. We report the accuracy obtained for each dataset by the some recent ensemble methods in Table 5.

HMD1 obtains the best accuracy on seven datasets while R-RTQRT-ME outperforms on two datasets. HMD1 and R-RTQRT-ME have the same accuracy on the magic dataset. HMD2 outperforms all other methods on the heart datasets. The improvement made by our models vary from 1% to 30%. For example, on vehicle dataset, HMD1 improves the accuracy near 16.4% relative to NULCOEC (Rahman & Verma, 2013), 10.8% relative to MARK-ELM (Fossaceca et al., 2015), 10.32% relative to LODE (Meo et al., 2012), 4.6% relative to META-DES (Cruz et al., 2015), 3.6% relative to MFSE (Kheradpisheh et al., 2014) and less than 0.5% compared to RTQRT-ME and R-RTQRT-ME. However, the improvement made by HMD1 when we compared to the most efficient method (R-RTQRT-ME) is not greater than 2%. Indeed, we used five logistic regressions in our models while R-RTQRT-ME used five MLP classifiers (with at least five hidden units). Therefore, it is very interesting to see that by an appropriate choice of weighting functions, we can outperform a combination of a strong classifier (MLP) by a combination of the simple logistic regression. As shown in the previous subsection, HMD1 is better than HMD2 in terms of accuracy. The good results obtained by HMD1 is certainly due to the fact that the chosen weighting functions collaborate together to affect more efficiently the tasks to each classifier. It will be interesting to investigate on other strategies mentioned above for the choice of the weighting functions.

5. Conclusion

In this paper, we proposed a generalized hierarchical mixture model for discriminative learners (*HMD*) to classify nonlinear data while reducing the number of experts (classifiers) used. The main idea behind our model is to divide a complex task into several sub-tasks through a set of weighting functions. We derive two examples from the generalized *HMD*, named *HMD1* and *HMD2*. The *HMD1* model used different type of weighting functions at each level of the tree (Hybrid weighting function) while the *HMD2* used the same type of weighting function across the tree. By doing so, these two models allow an efficient repartition of the tasks among the experts and in the same time reduce the numbers of experts used. Some experiments were conducted on both synthetic and real-world datasets to show the ability of our models to reduce the number of experts. Moreover, both *HMD1* and *HMD2* models outperform some relevant state-of-art algorithms. However, *HMD1* is more accurate than *HMD2*, even though the training time of the latter is the shortest in the presence of few number of experts. Future work will be devoted to the construction of efficient learning algorithms which will allow to reduce the training time (especially for very large datasets) while increasing the classification scores. Moreover, investigations will be made to compare several

Table 5
Comparison of our models with some recent ensemble methods.

Datasets	Methods	Acc. (%)
Glass	META-DES (Cruz et al., 2015)	66.87
	MFSE (Kheradpisheh et al., 2014)	72.22
	DCE-CC (Li et al., 2013)	72.48
	NULCOEC (Rahman & Verma, 2013)	**91.82**
	RTQRT-ME (Abbasi et al., 2016b)	72.87
	R-RTQRT-ME (Abbasi et al., 2016a)	74.76
	HDM1	75.25
	HMD2	75.71
Cancer	BABOROR (Kotsiantis, 2011)	73.16
	RTQRT-ME (Abbasi et al., 2016b)	75.74
	R-RTQRT-ME (Abbasi et al., 2016a)	**81.02**
	HDM1	**81.24**
	HMD2	79.80
Banana	R-RTQRT-ME (Abbasi et al., 2016a)	89.81
	HDM1	**90.36**
	HMD2	90
Balance	DCE-CC (Li et al., 2013)	75.18
	RTQRT-ME (Abbasi et al., 2016b)	92.96
	ISCG-Ranking (Ykhlef & Bouchaffra, 2015)	78.88
	HDM1	**94.08**
	HMD2	93.44
Pen-digit	MFSE (Kheradpisheh et al., 2014)	98.27
	RTQRT-ME (Abbasi et al., 2016b)	97.13
	R-RTQRT-ME (Abbasi et al., 2016a)	97.84
	HDM1	**98.91**
	HMD2	98.25
Magic	ISCG-Ranking (Ykhlef & Bouchaffra, 2015)	82.33
	R-RTQRT-ME (Abbasi et al., 2016a)	**86.82**
	HDM1	**86.81**
	HMD2	85.82
Vehicle	META-DES (Cruz et al., 2015)	82.75
	MARK-ELM (Fossaceca et al., 2015)	76.54
	MFSE (Kheradpisheh et al., 2014)	83.73
	NULCOEC (Rahman & Verma, 2013)	70.87
	LODE (Meo et al., 2012)	77.03
	RTQRT-ME (Abbasi et al., 2016b)	87.11
	R-RTQRT-ME (Abbasi et al., 2016a)	**87.29**
	HDM1	**87.35**
	HMD2	86.17
Waveform	BABOROR (Kotsiantis, 2011)	84.29
	RTQRT-ME (Abbasi et al., 2016b)	87.96
	R-RTQRT-ME (Abbasi et al., 2016a)	**89.72**
	HDM1	87.26
	HMD2	87
Satimage	MFSE (Kheradpisheh et al., 2014)	85.33
	RTQRT-ME (Abbasi et al., 2016b)	87.44
	R-RTQRT-ME (Abbasi et al., 2016a)	**89.61**
	HDM1	86.39
	HMD2	86.2
Heart	META-DES (Cruz et al., 2015)	84.80
	DCE-CC (Li et al., 2013)	80
	BABOROR (Kotsiantis, 2011)	81.74
	RTQRT-ME (Abbasi et al., 2016b)	84.41
	R-RTQRT-ME (Abbasi et al., 2016a)	87.4
	HDM1	88.15
	HMD2	**88.52**
Thyroid	ISCG-Ranking (Ykhlef & Bouchaffra, 2015)	95.44
	HDM1	**98.58**
	HMD2	97.57
Vowel	BABOROR (Kotsiantis, 2011)	94.6
	HDM1	**97.37**
	HMD2	96.76

strategies for the choice of the weighting functions. We will also investigate on possible situations where our generalized framework will face some identifiability issues.

Acknowledgment

The authors would like to thank the reviewers for their valuable comments and important suggestions. This work is supported by research grants from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Abbasi, E., Shiri, M. E., & Ghatee, M. (2016a). A regularized rootquartic mixture of experts for complex classification problems. *Knowledge-Based Systems, 110*, 98–109.
Abbasi, E., Shiri, M. E., & Ghatee, M. (2016b). Root-quatric mixture of experts for complex classification problems. *Expert Systems with Applications, 53*, 192–203.
Ahmed, N., & Campbell, M. (2011). Variational bayesian learning of probabilistic discriminative models with latent softmax variables. *IEEE Transactions on Signal Processing, 59*(7), 3143–3154.
Bdiri, T., Bouguila, N., & Ziou, D. (2014). Object clustering and recognition using multi-finite mixtures for semantic classes and hierarchy modeling. *Expert Systems with Applications, 41*(4, Part 1), 1218–1235.
Bishop, C. M. (1995). *Neural networks for pattern recognition*. New York, NY, USA: Oxford University Press, Inc.
Bishop, C. M., & Svenskn, M. (2003). Bayesian hierarchical mixtures of experts. In *Proceedings of the nineteenth conference on uncertainty in artificial intelligence*. In *UAI'03* (pp. 57–64). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. (1984). *Classification and regression trees*. CRC press.
Caudill, S. B., Ayuso, M., & Guillén, M. (2005). Fraud detection using a multinomial logit model with missing information. *Journal of Risk and Insurance, 72*(4), 539–550.
Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology, 2*, 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
Chen, K., Xu, L., & Chi, H. (1999). Improved learning algorithms for mixture of experts in multiclass classification. *Neural networks, 12*(9), 1229–1252.
Cruz, R. M., Sabourin, R., Cavalcanti, G. D., & Ren, T. I. (2015). Meta-des: A dynamic ensemble selection framework using meta-learning. *Pattern Recognition, 48*(5), 1925–1935.
Davis, R. E., Rossier, C. E., & Enfield, K. B. (2012). The impact of weather on influenza and pneumonia mortality in New York city, 1975–2002: a retrospective study. *PLoS One, 7*(3), 1–8.
Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, 1–38.
El-Zaart, A., & Ziou, D. (2007). Statistical modelling of multimodal sar images. *International Journal of Remote Sensing, 28*(10), 2277–2294.
Fossaceca, J. M., Mazzuchi, T. A., & Sarkani, S. (2015). Mark-elm: Application of a novel multiple kernel learning framework for improving the robustness of network intrusion detection. *Expert Systems with Applications, 42*(8), 4062–4080.
Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *ICML: 96* (pp. 148–156).
Fritsch, J., Finke, M., & Waibel, A. (1997). Adaptively growing hierarchical mixtures of experts. In *Advances in neural information processing systems* (pp. 459–465).
Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The weka data mining software: An update. *ACM SIGKDD Explorations Newsletter, 11*(1), 10–18.
Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(8), 832–844.
Jordan, M., & Jacobs, R. (1993). Hierarchical mixtures of experts and the EM algorithm. In *Proceedings of 1993 international joint conference on neural networks, 1993. IJCNN '93-nagoya: 2* (pp. 1339–1344).
Kheradpisheh, S. R., Sharifizadeh, F., Nowzari-Dalini, A., Ganjtabesh, M., & Ebrahimpour, R. (2014). Mixture of feature specified experts. *Information Fusion, 20*, 242–251.
Kotsiantis, S. (2011). Combining bagging, boosting, rotation forest and random subspace methods. *Artificial Intelligence Review, 35*(3), 223–240.
Ksantini, R., Ziou, D., Colin, B., & Dubeau, F. (2008). Weighted pseudometric discriminatory power improvement using a Bayesian logistic regression model based on a variational method. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 30*(2), 253–266.
Kung, S. Y. (2014). *Kernel methods and machine learning*. Cambridge University Press.
Lang, K. J. (1988). Learning to tell two spirals apart. In *Proc. of 1988 connectionist models summer school* (pp. 52–59).
Li, L., Zou, B., Hu, Q., Wu, X., & Yu, D. (2013). Dynamic classifier ensemble using classification confidence. *Neurocomputing, 99*, 581–591.
Lichman, M. (2013). UCI machine learning repository. http://archive.ics.uci.edu/ml.
Liu, C., & Wechsler, . (2002). Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *IEEE Transactions on Image Processing, 11*(4), 467–476.

Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th conference on natural language learning-volume 20* (pp. 1–7). Association for Computational Linguistics.
Masoudnia, S., & Ebrahimpour, R. (2014). Mixture of experts: A literature survey. *Artificial Intelligence Review, 42*(2), 275–293.
Meo, R., Bachar, D., & Ienco, D. (2012). Lode: A distance-based classifier built on ensembles of positive and negative observations. *Pattern Recognition, 45*(4), 1409–1425.
Moerland, P. (1999). Classification using localized mixture of experts. In *Artificial neural networks, 1999. ICANN 99. ninth international conference on (conf. publ. no. 470): 2* (pp. 838–843).
Moerland, P. (2000). *Mixture models for unsupervised and supervised learning*. École Polytechnique Fédérale de Lausanne, Computer Science Department Ph.D. thesis.
Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems* (pp. 841–848).
Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 427–436).
Nocedal, J. (1980). Updating quasi-newton matrices with limited storage. *Mathematics of Computation, 35*(151), 773–782.
Quinlan, J. R. (1993). *C4.5: Programs for machine learning*: 1. Morgan kaufmann.
Rahman, A., & Verma, B. (2013). Ensemble classifier generation using non-uniform layered clustering and genetic algorithm. *Knowledge-Based Systems, 43*, 30–42.
Ramamurti, V., & Ghosh, J. (1996). *Structurally adaptive localized mixtures of experts for non-stationary environments*. Department of Electrical and Computer Engineering. UT Austin. TX 78712-1084, USA.
Schmidt, M. (2012). Minfunc: Unconstrained differentiable multivariate optimization in matlab. URL http://www.di.ens.fr/mschmidt/Software/minFunc.html.
Shahbaba, B., & Neal, R. (2009). Nonlinear models using dirichlet process mixtures. *Journal of Machine Learning Research, 10*(Aug), 1829–1850.
Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. New York, NY, USA: Cambridge University Press.
Tsunoda, K., Yamane, Y., Nishizaki, M., & Tanifuji, M. (2001). Complex objects are represented in macaque inferotemporal cortex by the combination of feature columns. *Nature Neuroscience, 4*(8), 832–838. doi:10.1038/90547.
Vidal-naquet, M., & Ullman, S. (2003). Object recognition with informative features and linear classification. In *In ICCV* (pp. 281–288).
Waterhouse, S., & Robinson, J. (1994). Classification using hierarchical mixtures of experts. In *Neural networks for signal processing [1994]IV. proceedings of the 1994 IEEE workshop* (pp. 177–186).
Xu, L., Jordan, M. I., & Hinton, G. E. (1995). An alternative model for mixtures of experts. In *Advances in neural information processing systems* (pp. 633–640).
Ykhlef, H., & Bouchaffra, D. (2015). Induced Subgraph Game for Ensemble Selection. In *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)* (pp. 636–643).
Yuksel, S. E., Wilson, J. N., & Gader, P. D. (2012). Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems, 23*(8), 1177–1193.
Zhang, Y., Yin, Y., Guo, D., Yu, X., & Xiao, L. (2014). Cross-validation based weights and structure determination of chebyshev-polynomial neural networks for pattern classification. *Pattern Recognition, 47*(10), 3414–3428.