Neural Networks 23 (2010) 144–153

Contents lists available at ScienceDirect

Neural Networks

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

A Hopfield Neural Network for combining classifiers applied to textured images

Gonzalo Pajares [a,*], María Guijarro [b], Angela Ribeiro [c]

[a] Dpt. Ingeniería del Software e Inteligencia Artificial, Facultad Informática, Universidad Complutense, 28040 Madrid, Spain

[b] Centro Superior de Estudios Felipe II. Ingeniería Técnica en Informática de Sistemas, 28300 Aranjuez, Madrid, Spain

[c] Artificial Perception Group (GPA), Instituto de Automática Industrial (IAI), Spanish National Research Council (CSIC), Arganda del Rey, Madrid, Spain

ARTICLE INFO

Article history:
Received 29 June 2008
Received in revised form 14 November 2008
Accepted 15 July 2009

Keywords:
Hopfield Neural Network
Classifier combination
Fuzzy classifier
Bayesian classifier
Unsupervised
Textured images

ABSTRACT

In this paper we propose a new method for combining simple classifiers through the analogue Hopfield Neural Network (HNN) optimization paradigm for classifying natural textures in images. The base classifiers are the Fuzzy clustering (FC) and the parametric Bayesian estimator (BP). An initial unsupervised training phase determines the number of clusters and estimates the parameters for both FC and BP. Then a decision phase is carried out, where we build as many Hopfield Neural Networks as the available number of clusters. The number of nodes at each network is the number of pixels in the image which is to be classified. Each node at each network is initially loaded with a state value, which is the membership degree (provided by FC) that the node (pixel) belongs to the cluster associated to the network. Each state is later iteratively updated during the HNN optimization process taking into account the previous states and two types of external influences exerted by other nodes in its neighborhood. The external influences are mapped as consistencies. One is embedded in an energy term which considers the states of the node to be updated and the states of its neighbors. The other is mapped as the interconnection weights between the nodes. From BP, we obtain the probabilities that the nodes (pixels) belong to a cluster (network). We define these weights as a relation between states and probabilities between the nodes in the neighborhood of the node which is being updated. This is the classifier combination, making the main finding of this paper. The proposed combined strategy based on the HNN outperforms the simple classifiers and also classical combination strategies.

1. Introduction

Nowadays the advanced technological nature of aerial images demands solutions for different image-based applications. The natural texture classification is one of such applications because of the high image spatial resolutions. The areas where textures are suitable include agricultural crop ordination, forest areas determination, urban identifications and damage evaluation in catastrophes or dynamic path planning during rescue missions or intervention services also in catastrophes (fires, floods, etc.) among others. This justifies the choice of the textured images as the data where the proposed approach is to be applied. The next step is to select the features. The behaviour of different features has been also studied in texture classifications, where the set of features describes each pattern (Hanmandlu, Madasu, & Vasikarla, 2004; Puig & García, 2006; Wagner, 1999). There are pixel-based (Kumar, Ghosh, & Crawford, 2001; Rud, Shoshany, Alchanatis, & Cohen, 2006; Yu, Li, Zhang, & Feng, 2002) and region-based approaches (Maillard, 2003; Puig & García, 2006; Randen & Husøy, 1999; Smith & Burns, 1997; Wagner, 1999). A pixel-based approach tries to classify each pixel as belonging to one of the clusters. The region-based identifies patterns of textures within the image and describes each pattern by applying filtering (laws masks, Gabor filters, Wavelets, etc.); it is assumed that each texture displays different levels of energy allowing its identification at different scales. The aerial images used in our experiments do not display texture patterns. This implies that textured regions cannot be identified. In this paper we focus on the pixel-based category. Taking into account that we are classifying multi-spectral textured images, we use as the attribute vector the spectral components, i.e. the Red, Green and Blue (RGB) colour mapping. The RGB map performs better than other colour representations (Drimbarean & Whelan, 2003); we have verified this assertion in our experiments, justifying its choice.

An important issue reported in the literature is that the combination of classifiers outperforms the simple classifiers (Cao, Shridhar, & Ahmadi, 1995; Kittler, Hatef, Duin, & Matas, 1998; Kong & Cai, 2007; Kumar, Ghosh, & Crawford, 2002; Kuncheva, 2003; Valdovinos, Sánchez, & Barandela, 2005). Particularly, the studies in Deng and Zhang (2005) and Partridge and Griffith (2002) report

* Corresponding author. Tel.: +34 1 3947546; fax: +34 1 3947547.
E-mail addresses: pajares@fdi.ucm.es (G. Pajares), mguijarro@cesfelipesegundo.com (M. Guijarro), aribeiro@iai.csic.es (A. Ribeiro).

0893-6080/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2009.07.019

the advantages of using combined classifiers against simple ones. This is because each classifier produces errors on a different region of the input pattern space (Alexandre, Campilho, & Kamel, 2001).

Nevertheless, the main problem is: which is the best strategy for combining simple classifiers? This is an issue still open. Indeed in Kuncheva (2003) it is stated that there is not best combination method.

In the pattern classification literature, various classifiers have been developed from different methodologies, which motivate studies on combining multiple classifiers for a better performance. Xu, Krzyzak, and Sun (1992) introduce two tasks to be considered for the combination: (a) how many and what type of classifiers should be used for a specific application; (b) how to combine the results from single classifiers. The first task has been studied in Kuncheva (2004) concluding that the number and the type of classifiers depend on the specific applications. In this paper we put the emphasis on the second task.

In the machine learning context the expert combination is a classical combination model (Xu & Amari, 2008) which has been used in various classification problems. In this model, a selector makes use of a separate classifier, which determines the participation of the experts in the final decision for an input pattern. This architecture has been proposed in the neural network context. The experts are neural networks, which are trained so that each network is responsible for a part of the feature space. The selector uses the output of another neural network called the gating network (Kuncheva, 2004). The input of the gating network is the pattern to be classified and the output is a set of outputs determining the competences for each expert. These competences are used together during the decision with the classifier outputs provided by the experts. The (cooperative) modular neural networks can be considered under this category (Auda & Kamel, 1997, 1998; Poirazi, Neocleous, Pattichis, & Schizas, 2004; Zhao, 2008). Usually, a modular neural network contains two steps: task decomposition and combining decision. The first is mainly concerned with the training phase where a complex learning task can be accomplished by dividing it into some subtasks assigned to several experts. The second is the decision process where the outputs provided by the experts are conveniently combined for making the decision. Hence, modular neural networks can be well-suited for training. Nevertheless, our proposed approach makes the main contribution on the decision phase and it could be also used for combining the outputs of the experts following the same scheme of this paper. We have used the outputs provided by two simple classifiers instead of the ones provided by modular neural networks because our goal is the combination and this can be achieved with simple classifiers. The combination for other kind of outputs, including those provided by the modular networks, is also feasible under our approach. The simple classifiers are: the *Fuzzy Clustering* (FC) [10, 41] and the *Bayesian estimator Parametric* approach (BP) (Duda, Hart, & Stork, 2001).

Our proposed classifier consists of two phases: training and decision. We have designed an unsupervised classifier for the training phase based on the procedure described in Duda et al. (2001) where a partition of the training patterns is established through the FC and then validated by the Sum-of-Squared Error criterion (SE). Once the partition is validated the number of clusters is known. So, the supervised FC and BP methods estimate their parameters which are stored in a *Knowledge Base* (KB) to be recovered during the decision phase.

During the decision phase, we build as many Hopfield Neural Networks as number of available clusters. The number of nodes at each network is the number of pixels in the image which is to be classified. Each node at each network is initially loaded with a state value, which is the membership degree (provided by FC) that the node (pixel) belongs to the cluster associated to the network. Each state is later iteratively updated during the HNN optimization process taking into account the previous states and two types of external influences exerted by other nodes in its neighborhood. The external influences are mapped as consistencies. One embedded in an energy term which considers the states of the node to be updated and the states of its neighbors. The other is mapped as the inter-connection weights between the nodes. From BP, we obtain the probabilities that the nodes (pixels) belongs to a cluster (network). We define these weights as a relation between states and probabilities between the nodes in the neighborhood of the node which is being updated. This is the classifier combination, making the main finding of this paper.

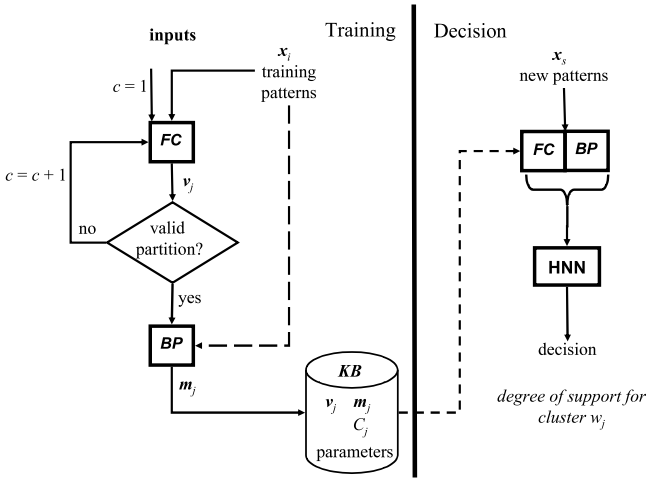

Fig. 1. Architecture of the hybrid classifier based on the HNN approach: Training and classification phases.

The paper is organized as follows. In Section 2 the architecture of the unsupervised classifier is explained, where the most significant details are given for both classifiers during the training (including the validation criterion) and decision phases. The *HNN* approach proposed for combining the base classifiers is also explained. In Section 3 we give details about the performance of the proposed strategy applied to natural textured images. Finally, in Section 4 the conclusions are presented.

2. Design of the automatic unsupervised combined classifier

The system works in two phases: training and classification. The training patterns are supplied to the *FC* classifier which automatically establishes a partition assuming a number of clusters c. Initially, c is set to 1. Taking into account the cluster centers $\boldsymbol{v}_j$ with $j = 1, \ldots, c$ obtained for such partition, we apply a validation criterion for determining if the partition can be considered as valid. If the partition is not validated, a new partition is tried by setting $c = c+1$ until validation. This makes the process unsupervised (Duda et al., 2001). Once the partition is validated, the c clusters and associated patterns are supplied to *BP*, which estimates a probability density function with the mean ($\boldsymbol{m}_j$) and covariance matrix (C_j) as parameters. *FC* assumes, as parameters, the fuzzy membership degrees and cluster centers previously computed. All parameters are stored in the *KB*. During the decision phase the estimated parameters are recovered from *KB*, so that the new patterns $\boldsymbol{x}_s$ supplied to the system can be classified as belonging to a cluster w_j, based on the decision made through the final supports after the optimization process through the *HNN*.

2.1. Training process

During the training phase, we start with the observation of a set X of n training samples, i.e. $X = \{x_1, x_2, \ldots, x_n\} \in \Re^d$, where d is the data dimensionality, i.e. $d = 3$ in this approach because we use the three RGB spectral components for each pixel to be classified. Each pattern is to be assigned to a given cluster w_j, where the number of possible clusters is c, i.e. $j = 1, 2, \ldots, c$.

(a) Fuzzy Clustering (FC)

This process receives the input training patterns and establishes a partition assuming known the number of clusters c (Asuncion & Newman, 2008; Kuncheva, 2004; Zimmermann, 1991). *FC* computes for each $x_i \in X$ at the iteration t its membership grade μ_i^j and updates the cluster centers, $v_j \in \Re^d$ as follows,

$$\mu_i^j(t+1) = \frac{1}{\sum_{r=1}^{c} \left(d_{ij}(t)/d_{ir}(t)\right)^{2/(m-1)}};$$
$$v_j(t+1) = \frac{\sum_{i=1}^{n} \mu_i^j(t)^m x_i}{\sum_{i=1}^{n} \mu_i^j(t)^m} \tag{1}$$

$d_{ij}^2 \equiv d^2\left(x_i, v_j\right)$ is the squared Euclidean distance. The number m is called the exponent weight (Bezdek, 1981; Zimmermann, 1991). The stopping criterion of the iteration process is achieved when $\left\|\mu_i^j(t+1) - \mu_i^j(t)\right\| < \varepsilon \forall ij$ or a number $t_{\max}$ of iterations is reached, set to 50 in our experiments; ε has been fixed to 0.01 (see Section 3.1).

The method requires the initialization of the cluster centers, so that Eq. (1) can be applied at the iteration $t = 1$. With such purpose we apply the pseudorandom procedure described in Balasko, Abonyi, and Feil (2008):

(1) Perform a linear transform $Y = f(X)$ of the training sample values so that they range in the interval [0, 1].
(2) Initialize $v = 2D\bar{\boldsymbol{M}} \circ \boldsymbol{R} + D\bar{\boldsymbol{m}}$, where $\bar{\boldsymbol{m}}$ is the mean vector for the transformed training samples values in Y and $\bar{\boldsymbol{M}} = \max(abs(Y - \bar{\boldsymbol{m}}))$, both of size $1 \times d$; $D = \begin{bmatrix}1 & \ldots & 1\end{bmatrix}^t$ with size $c \times 1$; $\boldsymbol{R}$ is a $c \times d$ matrix of random numbers in [0, 1]; the operation $\circ$ denotes the element by element multiplication.

Once the fuzzy clustering process is carried out, a partition of the input training samples is obtained, where each cluster w_j has associated its center $\boldsymbol{v}_j$.

(b) Validation of the partition

Several works have studied this topic (see Volkovich, Barzily, & Morozensky, 2008 and related references). Nevertheless, we have found acceptable the performance achieved by the *Sum-of-Squared Error Criterion* (*SE*) Duda et al., 2001, which uses the information provided by the *FC*, exactly the cluster centers $\boldsymbol{v}_j$ as follows,

$$SE(v_j, x, c) = \sum_{j=1}^{c} \sum_{x \in w_j} \left\|x - v_j\right\|^2. \tag{2}$$

SE decrease rapidly until to reach the best partition (equivalently the best number of clusters $c = \hat{c}$) decreasing much more slowly thereafter until it reaches zero when the number of clusters is equal to the number of samples, i.e. $c = n$. Given a number of clusters c, we compute the delta function $\Delta^{SE}(c) = SE(c+1) - SE(c)$, $c = 1, \ldots, G-1$ which represents the difference value for two consecutive numbers of clusters, where G is the last number of clusters evaluated. Now we normalize these differences to range in [0, 1] as follows,

$$\hat{\Delta}^{SE}(c) = \frac{\Delta^{SE}(c)}{\sum_{i=1}^{G-1} \Delta^{SE}(c)}. \tag{3}$$

Starting from $c = 1$, a partition is validated for a given number of clusters $\hat{c}$ when we find $\hat{\Delta}^{SE}(\hat{c}) < U$; U has been set to 0.1 after the experiments described in Section 3.1.

(c) Parametric Bayesian (BP) estimation

BP receives the validated partition, supplied by *FC*. Assuming known the distribution (Gaussian) for each cluster w_j, the probability density function is expressed as follows,

$$p\left(\boldsymbol{x}|w_j\right) = \frac{1}{(2\pi)^{d/2}\left|C_j\right|^{1/2}} \exp\left[-\frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{m}_j\right)^t C_j^{-1}\left(\boldsymbol{x} - \boldsymbol{m}_j\right)\right] \tag{4}$$

where the parameters to be estimated are the mean, $\boldsymbol{m}_j$ and the covariance, C_j; both for each cluster w_j with n_j samples. The estimation is carried out through maximum likelihood (Duda et al., 2001), obtaining,

$$\boldsymbol{m}_j = \frac{1}{n_j}\sum_{k=1}^{n_j} \boldsymbol{x}_k \qquad C_j = \frac{1}{n_j - 1}\sum_{k=1}^{n_j}\left(\boldsymbol{x}_k - \boldsymbol{m}_j\right)\left(\boldsymbol{x}_k - \boldsymbol{m}_j\right)^t \tag{5}$$

where t denotes transpose.

2.2. Combining the classifiers: The Hopfield Neural Network

2.2.1. A review on the HNN

Given a new sample $\boldsymbol{x}_s$, Fig. 1, the problem is to decide which the cluster it belongs is. We make the decision by combining the outputs of the *FC* and *BP* classifiers through the *HNN* optimization paradigm. The *HNN* paradigm initially proposed by Hopfield and Tank (1985, 1986) has been widely used for solving optimization problems. This implies fixing two characteristics (Joya, Atencia, & Sandoval, 2002): its activation dynamics and an associated energy function which decreases as the network evolves.

The *HNN* is a recurrent network containing feedback paths from the outputs of the nodes back into their inputs so that the response of such a network is dynamic. This means that after applying a new input, the output is calculated and fed back to modify the input. The output is then recalculated, and the process is repeated again and again. Successive iterations produce smaller and smaller output changes, until eventually the outputs become constant, i.e. at this moment the network achieves an acceptable stability.

The connection weights between the nodes in the network net_j may be considered to form a matrix T^j. To illustrate the Hopfield networks in more detail, consider the special case of a Hopfield network with a symmetric matrix. The input to the ith node comes from two sources: external inputs and inputs from the other nodes. The total input u_i^j to node i is then

$$u_i^j = \sum_{k \neq i} T_{ik}^j \mu_k^j + \theta_i^j \tag{6}$$

where the μ_k^j value represents the output of the kth node; T_{ik}^j is the weight of the connection between nodes i and k; and θ_i^j represents an external input bias value which is used to set the general level of excitability of the network. There are two kinds of Hopfield networks (Haykin, 1994; Joya et al., 2002) namely, (1) the analog ones in which the states of the neurons are allowed to vary continuously in an interval, such as $[-1, +1]$ and; (2) the discrete ones in which these states are restricted to the binary

values −1 and +1. The drawback of these binary networks is that they oscillate between different binary states, and settle down into one of many locally stable states. Hopfield has shown that analog networks perform better since they have the ability to smooth the surface of the energy function which prevents the system from being stucked in minor local minima Hopfield and Tank (1985, 1986).

For analog Hopfield networks the total input into a node is converted into an output value by a sigmoid monotonic activation function instead of the thresholding operation for discrete Hopfield networks (Qiao, Peng, & Xu, 2001). The dynamic of a node is defined by

$$\frac{\mathrm{d}u_i^j}{\mathrm{d}t} = -\frac{u_i^j}{R_i} + \sum_{k\neq i} T_{ik}^j u_k^j + \theta_i^j \quad \text{where } \mu_k^j = g\left(u_k^j\right) \ \forall k \tag{7}$$

$g\left(u_i^k\right)$ is the sigmoid activation function, and R_i is a time constant which can be set to 1 for simplicity (Haykin, 1994; Yu & Tsai, 1992). We have chosen the sigmoid activation function to be the hyperbolic tangent function (Haykin, 1994), $g\left(u_k^j\right) = \tanh\left(u_k^j/\beta\right)$. This function is differentiable, smooth and monotonic. A detailed discussion about the settings of the time step $\mathrm{d}t$ and gain β^{-1} can be found in Joya et al. (2002). As $\mathrm{d}t$ increases, the probability that the energy falls into a local minimum also increases. According to some experiments carried out by Joya et al. (2002) where this parameter has been set to values in the range 1 to 10^{-2}, the best performance is achieved with the minimum value (i.e. 10^{-2}), hence we have fixed it to 10^{-3} which is an order of magnitude less than the experimented in Joya et al. (2002). The way to avoid that a continuous network cannot find a solution due to the existence of local minimum and makes the network converge up to a solution state is to decrease β along the simulation, theoretically until $\beta = 0$. This strategy reminds a simulated annealing process starting from high enough β, then the network evolves until a stable state (which is not a solution) is reached, then β is decreased and the network evolves again up to a new stable state, and so on; the process ends when β becomes zero and at this moment, the stable state reached should be a global minimum. According to the results reported in Kasetkasem and Varshney (2002) and Starink and Backer (1995), we have tested the following scheduling strategy $\beta(t) = \beta_0/\log(t+1)$ where t is the iteration number. We have computed β_0 as follows (Laarhoven van & Aarts, 1989): (1) we select eight pairs of images, where the nodes have been initialized; now we compute the initial energy; (2) we choose an initial β, that permits about 80% of all transitions to be accepted (i.e. transitions that decrease the energy function), and this value is changed until this percentage is achieved; (3) we compute the M transitions ΔE_i and we look for a value for β for which $\frac{1}{M}\sum_{k=1}^{M}\exp(-\Delta E_k/\beta) = 0.8$, after rejecting the higher order terms of the Taylor expansion of the exponential, $\beta = 5\langle\Delta E_k\rangle$, where $\langle\cdot\rangle$ is the mean value. In our experiments, we have obtained $\langle\Delta E_k\rangle = 0.77$, giving $\beta_0 = 3.85$. In the work of Starink and Backer (1995) a simulated annealing scheduling is used with $\beta_0 = 2$, i.e. with the same order of magnitude. Taking into account that $\beta(t) = 0$, $t \to +\infty$ and considering $t = 10^6$ we obtain $\beta = 0.28$, i.e. $\beta^{-1} = 3.59$. In our image classification approach, we have carried out different experiments by applying the above scheduling and also assuming fix the gain without apparent improvement in the final results. Hence we set the gain to 3.59 during the full process.

The model provided in Eq. (7) is the classical Hopfield circuit (Hopfield & Tank, 1985, 1986; Kosko, 1992) which follows from the Cohen–Grossberg dynamical systems (Cohen & Grossberg, 1983). In Kosko (1992) the global stability of these systems is proven under the positivity assumption $\mathrm{d}g/\mathrm{d}t > 0$ and considering that the coefficient in the left term of Eq. (7) is also positive. Because g is the hyperbolic tangent function the first condition is true. Although some recent studies carried out by Lee and Chuang (2005) in Hopfield Neural Networks have been addressed for solving the problem of optimal asymmetric associative memories, we have found acceptable the classical approach studied in Cohen and Grossberg (1983) or Qiao et al. (2001) where it is shown that a recurrent network is stable if the matrix is symmetrical with zeros on its diagonal, that is, if $T_{ik}^j = T_{ki}^j$ for all i and k and $T_{ii}^j = 0$ for all neurons i. Additionally, towards the global stability also contributes that the bias θ_i^j varies slowly. In our design this is also true according to the discussion in Section 2.2.2(d). The stability of the Hopfield Neural Network has also been studied under different perspectives in Qiao et al. (2001) or Zhao (2004). Hence, it belongs to the important class of feedback neural networks models that are globally stable.

The quantity describing the state of each network net_j, called energy, is defined as follows,

$$E^j(t) = -\frac{1}{2}\sum_i\sum_{k\neq i} T_{ik}^j\mu_i^j\mu_k^j - \sum_i \theta_i^j\mu_i^j + \beta^j\sum_i\int_0^{j\mu_i} g^{-1}\left(\mu^j\right)\mathrm{d}\mu^j. \tag{8}$$

The total energy for the c available networks is,

$$E(t) = \sum_{j=1}^{c} E^j(t). \tag{9}$$

According to the results reported in Joya et al. (2002), the integral term in (8) is bounded by $\beta^j \ln 2 \approx 0.19$ when μ_i^j is +1 or −1 and is null when μ_i^j zero is. In our experiments, we have verified that this term does not contribute to the network stability and only the energy is increased in a very little quantity with respect to the other two terms in (8), hence for simplicity we have removed it from the equation (8).

The continuous Hopfield model described by the system of nonlinear first-order differential equation (7) represents a trajectory in phase space, which seeks out the minima of the energy function in (9).

2.2.2. Combined classifier by HNN

As mentioned before, for each cluster w_j, we build a network of nodes, net_j. Each node i in net_j is associated to the pixel location $i \equiv (x, y)$ in the image, which is to be classified; the node i in net_j is initialized with the membership degree μ_i^j provided by *FC* according to Eq. (1), but mapped linearly to the range $[-1, +1]$ instead of $[0, +1]$. These membership degrees are the initial network states associated to the nodes. As it is known, the simple *FC* method classifies each pixel i as belonging to the cluster w_j according to the maximum network state value associated to the pixel i in the j networks, i.e. $i \in w_j$ if $\mu_i^j > \mu_i^h, \forall\, j \neq h$. Through the *HNN* optimization process each node state is updated (reinforced or punished) based on the influences exerted by the nodes in its neighborhood and also according to its own state value. The first is an external influence exerted over the node which is being updated and the second is considered as a self-influence. The goal is to make better decisions based on more stable state values through the *HNN* optimization process.

An important issue addressed in neural computation for image applications is referred to how sensory elements in a scene perceive the objects, i.e. how the scene analysis problem is addressed. To deal with real-world scenes some criterion for grouping elements in the scene is required. In the work of Wang (2005) a list of major grouping principles is exhaustively studied. In our image

classification approach we apply the following three principles: *proximity*, labeled pixels that lie close in space tend to group; *similarity*, labeled pixels with similar values tend to group; *connectedness*, labeled pixels that lie inside the same connected region tend to group. These principles define a neighborhood. Now the problem is to build an energy function where the influences exerted by the nodes k in the neighborhood over a node i are mapped as data and contextual consistencies. The self-influence is mapped as self-data information. This is explained later.

Suppose each network net_j has N nodes. Initially the states of the nodes are the analog membership degrees μ_i^j supplied by *FC*.

(a) Consistency from the data.

During the optimization process the initial states μ_i^j are modified trying to achieve the network stabilization. Now, the goal is to map the data consistency between nodes i and k in net_j into the *consistency* coefficient $r_{ik}^j(t)$ at each iteration t. Given the node i we consider its m-connected neighborhood, N_i^m, under the grouping criterion established by the proximity and connectedness principles according to Wang (2005); m could be 4, 8, 24, 48 or any other value taking into account only horizontal, vertical or diagonal directions. A typical value is 8, corresponding to a central pixel and its 8 neighbors.

For each node i, only consistencies can be established between nodes k, where $k \in N_i^m$ and $i \neq k$ otherwise if $k \notin N_i^m$ it is assumed that there is not consistency between nodes i and k. This is justified under the hypothesis that only local relations can be established between labeled pixels. Two nodes i, k where $k \in N_i^m$ are said consistent if they have similar data information. Otherwise they should be inconsistent.

The data consistency between the nodes i, k is mapped into the consistency coefficient as follows

$$r_{ik}^j(t) = \begin{cases} 1 - \left|\mu_i^j(t) - p_k^j\right| & k \in N_i^m,\ i \neq k \\ 0 & k \notin N_i^m \text{ or } i = k \end{cases} \tag{10}$$

where p_k^j is a support supplied by *BP*, it is the probability that a node (pixel) k with attributes $\boldsymbol{x}_k$ belongs to the cluster w_j, computed through Eq. (4), i.e. $p_k^j \equiv p\left(x_k|w_j\right)$. These probability values are mapped linearly to range in $[-1, +1]$ instead of $[0, +1]$. From (10) we can see that $r_{ik}^j(t)$ ranges in $[-1, +1]$. We can see that the influence exerted by the node k over the node i will be positive (reward) or negative (punishment). Hence, a positive data consistency will contribute towards the network stability. As one can see, the information comes from the membership degrees and the probabilities supplied by FC and BP respectively. The consistency coefficient is the key of the classifier combination proposed because it involves the supports provided by the simple classifiers.

(b) Consistency from the contextual information.

In some existing works dealing with images, such as in Kasetkasem and Varshney (2002), the inter-pixel dependence is described by defining a kind of consistency which is achieved under the consideration of contextual information. We make use of this concept and apply it to our *HNN* approach. Given the node i at the pixel location (x, y) with state value μ_i^j and a set of nodes $k \in N_i^m$ with state values μ_k^j, a measurement of contextual consistency between the node i and its k neighbors can be expressed as,

$$E_i^j(t) = \sum_{j \in N_i^m} \mu_i^j \mu_k^j. \tag{11}$$

This term represents an inter-state relation between the nodes in net_j. It also represents a kind of external influence exerted by the nodes k over the node i. As μ_i^j, μ_k^j range in $[-1, +1]$, given μ_i^j the term $E_i^j(t)$ will be maximum when the μ_k^j values are close to μ_i^j. Indeed, assuming that under the 8 neighborhood μ_i^j and μ_k^j take simultaneously values of $+1$ or -1, $E_i^j(t) = 8$, i.e. reaches its maximum value. On the contrary, if $\mu_i^j = +1$ and all $\mu_k^j = -1$ or $\mu_i^j = -1$ and all $\mu_k^j = +1$, $E_i^j(t) = -8$, i.e. its minimum value. It is worth noting that (11) can be regarded as an implementation of the Gibbs potential in a neighborhood under the Markov Random Fields framework (Kasetkasem & Varshney, 2002).

Table 1
Behavior of the energy term against data and contextual consistencies.

$r_{ik}^j(t)$	μ_i^j	μ_k^j	$E_C^j(t)$	$r_{ik}^j(t)$	μ_i^j	μ_k^j	$E_C^j(t)$
+	+	+	−	−	+	+	+
+	+	−	+	−	+	−	+
+	−	+	+	−	−	+	+
+	−	−	−	−	−	−	+

Once data and contextual consistencies are specified, we search for an energy function such that the energy is low when both consistencies are high and vice versa. This energy is expressed as,

$$E_C^j(t) = -\frac{A}{2} \sum_i \sum_{j \in N_i^m} \left\{ \left[sgn\left(r_{ik}^j(t)\right) \right]^{v+1} r_{ik}^j(t) - \delta_{ik}^j \right\} \mu_i^j \mu_k^j$$

$$\text{where } sgn\left(r_{ik}^j(t)\right) = \begin{cases} -1 & r_{ik}^j(t) \leq 0 \\ +1 & r_{ik}^j(t) > 0 \end{cases} \quad \text{and} \tag{12}$$

$$\delta_{ik}^j = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases}$$

where A is a positive constant to be defined later, sgn is the *signum function* and v is the number of negative values in the set $C \equiv \left\{r_{ik}^j(t), \mu_i^j, \mu_k^j\right\}$, i.e. given $S \equiv \{s \in C / s < 0\} \subseteq C$, $v = card(S)$; δ_{ik}^j is introduced to cancel the self-contribution of the node i because it is considered later under the self-data information.

Table 1 shows the behaviour of the energy term $E_C^j(t)$ against data and contextual consistencies. As one can see, the energy decreases as the data and the states are both simultaneously consistent (rows 1 and 4 in the left part of Table 1); otherwise under any inconsistency the energy increases. We have considered that data inconsistencies have higher priority than contextual ones; so under this criterion if $r_{ik}^j(t) < 0$ then the energy increases.

(c) Self-data information.

We have analyzed the inter-relations between nodes in a given neighborhood, based on data and contextual consistencies. This implies that the state for each node evolves according to the information provided by the majority in the neighborhood, ignoring its own information. This may lead to an incorrect state for the node under consideration. To overcome this drawback we assume that each node must contribute to the evolution of its own state through the self-data information. The self-data information is modeled as a kind of self-consistency based on the hypothesis that a node in the net_j with the maximum membership degree μ_i^j must be labeled as belonging to the cluster w_j and vice versa. This implies that a node with high/low membership degree must have a high/low state value simultaneously. Under this assumption, the self-consistency is mapped as an energy function as follows,

$$E_B^j(t) = -B \sum_i \mu_i^j \mu_i^j. \tag{13}$$

The constant B is a positive number to be defined later. So, if the node i has a low/high μ_i^j value, it implies that E_B at each iteration is minimum, as expected.

(d) Derivation of the connection weights and the external inputs for the HNN.

Assuming data and contextual consistencies, Eq. (12), and self-data information, Eq. (13), we derive the energy function of the Eq. (14), which is to be minimized by optimization under the HNN framework.

$$E^j(t) = E^j_C(t) + E^j_B(t) = -\frac{A}{2}\sum_i \sum_{j\in N_i^m} \left\{ \left[sgn\left(r^j_{ik}(t)\right)\right]^{v+1} \times r^j_{ik}(t) - \delta^j_{ik} \right\} \mu^j_i \mu^j_k - B\sum_i \mu^j_i \mu^j_i. \quad (14)$$

By comparison of the expressions (8) and (14) without the integral term in (8), it is easy to derive the connection weights and the external input bias as follows:

$$T^j_{ik} = A\left[sgn\left(r^j_{ik}(t)\right)\right]^{v+1} r^j_{ik}(t) - \delta^j_{ik}; \quad \theta^j_i = B\mu^j_i. \quad (15)$$

According to the discussion in Section 2.2.1, to ensure the convergence to stable state Cohen and Grossberg (1983), symmetrical inter-connection weights and no self-feedback are required, i.e. we see that by setting $A = B = 1$ both conditions can easily be derived from (14). Also, the external input bias θ^j_i must vary slowly to ensure the network stability. Because, the network is loaded initially with the membership degrees provided by FC, the network optimization process starts with a high degree of stability and these values change slowly. Additionally, the definition of the neighborhood establishes that only small numbers of neurons are inter-connected. It is also well-known that this contributes to the stability (Zhao, 2004).

The energy in (14) represents a trade-off between the data and contextual information coming from the spatial environment surrounding the node (pixel) i and also its self-data information. The constants A and B could be fixed so that they tune the influence of each term in (14). We have carried out several experiments verifying that in our approach the above setting is appropriated.

The Eq. (7) describes the time evolution of the network, the total input to the neuron μ^j_i is computed by solving the Eq. (7) with the Runge–Kutta method. Finally, the state μ^j_k is also computed according to Eq. (7). As we can see, the energy in (14) is obtained by considering the state values and a kind of attractiveness derived from both, data and contextual consistencies. The derivation of an energy function with attractiveness between fixed points has been well-addressed in the work of Müezzinoglu, Güzelis, and Zurada (2005) for discrete Hopfield memories preserving symmetrical weights and without self-feedback. Hence, we can assume that under the attractiveness of data and contextual consistencies, our analog Hopfield approach performs appropriately.

(d) Summary of the Image classifier combination procedure.

After mapping the energy function onto the Hopfield Neural Network, the image classification process is achieved by letting the networks evolve so that they reach stable states, i.e. when no change occurs in the states of its nodes during the updating procedure. The whole image classification procedure can be summarized as follows:

1. *Initialization*: create a network net_j for each cluster w_j. For each net_j create a node i at each pixel location (x, y) from the image to be classified; $t = 0$ (iteration number); load each node with the state value μ^j_i, i.e. the membership degree provided by *FC*, Eq. (1); compute T^j_{ik} and θ^j_i through Eq. (15); set $\varepsilon = 0.01$ (a constant to accelerate the convergence, see Section 3.1(c)); $t_{\max} = 20$ (maximum number of iterations allowed); set the constant values as follows: $R_i = 1$; $\beta = 0.28$; $dt = 10^{-3}$. Define nc as the number of nodes that change their state values at each iteration.

2. *HNN process*: set $t = t + 1$ and $nc = 0$; *for* each node i in net_j compute $u^j_i(t)$ using the Runge–Kutta method and update $\mu^j_i(t)$, both according to (7) and *if* $\left|\mu^j_i(t) - \mu^j_i(t-1)\right| > \varepsilon$ *then* $nc = nc + 1$; when all nodes i have been updated, *if* $nc \neq 0$ and $t < t_{\max}$ then go to step 2 (new iteration), *else* stop.

3. *Outputs*: μ^j_i updated for each node; it is the degree of support for the cluster w_j (Fig. 1). The node i is classified as belonging to the cluster with the greatest degree.

3. Comparative analysis and performance evaluation

To assess the validity and performance of the proposed approach we describe the tests carried out according to both processes: unsupervised training and decision. Nevertheless, before we give details about the setting of some free parameters involved in the method.

3.1. Setting free parameters

We have used several data sets for setting the free parameters, these are: (1) nine data sets from the Machine Learning Repository (Asuncion & Newman, 2008): (bupa, cloud, glass, imageSegm, iris, magi4, thyroid, pimaIndians and wine); (2) three synthetic data sets manually generated with different number of clusters and (3) four data sets coming from outdoor natural images, also with different number of clusters.

(a) Parameters involved in the FC training phase

They are the exponential weight m in Eq. (1) and the convergence parameters ε and $t_{\max}$ used there. The number of clusters and the distribution of the patterns on the clusters are known. We apply the following cross-validation procedure (Duda et al., 2001). We randomly split each set into two parts. The first (90% of the patterns) is used as the training set. The other set (validation set) is used to estimate the global classification error based on the single *FC* classifier. We set $m = 2.0$ (which is a usual value) and vary ε from 0.01 to 0.1 in steps of 0.15 and estimate the cluster centers and membership degrees for each training set. Then, we compute the error rate for each validation set. The maximum error was obtained with $\varepsilon = 0.1$ for 10 iterations and the minimum with $\varepsilon = 0.01$ and 47 iterations. Fixed those values, we vary m from 1.1 to 4.0 in steps of 0.1 and estimate once again the cluster centers and the membership degrees with the training set. Once again the validation sets are used for computing the error rates, the minimum error value is obtained for $m = 2.0$. The settings are finally fixed to $m = 2.0$, $\varepsilon = 0.01$ and $t_{\max} = 50$ (expanding the limit of 47).

(b) Validation threshold

From the same data sets the threshold U, used for estimating the best number of clusters according to Eq. (3), is set as follows. We apply for each data set the unsupervised training procedure described in Section 2.1 starting from $c = 1$ to $c = G$, where G is fixed to 8 because the maximum number of clusters is found in the textured images and it is assumed as acceptable. We compute $\hat{\Delta}^{SE}(c)$ for each partition according to the Eq. (3) verifying that for the true number of clusters, $\hat{c}$, the corresponding $\hat{\Delta}^{SE}(\hat{c})$ is always less than 0.1, hence U is finally set to this value.

(c) HNN convergence

The ε used for accelerating the convergence in the HNN process is set to 0.01 by using the validation set for the four natural images mentioned above. Verifying, that $t_{\max} = 20$ suffices.

3.2. Unsupervised training: estimating the best partition

We have available a set of 36 digital aerial images acquired during May in 2006 from the Abadin region located at Lugo (Spain). They are multi-spectral images with 512×512 pixels in size. The

Table 2
Behaviour of $\hat{\Delta}^{SE}(c)$.

c	1	2	3	4	5	6	7
Δ	$c1$–$c2$	$c2$–$c3$	$c3$–$c4$	$c4$–$c5$	$c5$–$c6$	$c6$–$c7$	$c7$–$c8$
$\hat{\Delta}^{SE}(c)$	0.4951	0.2433	0.1221	**0.0728**	0.0611	0.0381	0.0301

images were taken during different days from an area with several natural textures. We select randomly 12 images from the set of 36 available. Each image is down sampled by two, so that the number of training samples provided by each image is the number of pixels. The total number of samples is $n = 12 \times 256 \times 256 = 786432$.

Table 2 shows the behaviour of $\hat{\Delta}^{SE}(c)$, Eq. (3), against the number of clusters c ranging from $c = 1$ to $c = G - 1$, where G is set to 8 in our experiments as described above. We can see that with $U = 0.1$ the condition $\hat{\Delta}^{SE}(c) < U$ is fulfilled for $\hat{c} = 4$, which is finally selected as the number of clusters estimated.

Once the number of clusters is established, the cluster centers obtained for *FC* ($\boldsymbol{v}_j$) and *BP* ($\boldsymbol{m}_j$) and also the covariance matrices C_j for *FC* are stored in *KB*.

3.3. Classification: Comparative analysis

The remaining 24 images from the set of 36 are used as images for testing. Four sets, S0, S1 S2 and S3 of 6 images each one, are processed during the test according to the strategy described below. The images assigned to each set are randomly selected from the 24 images available.

(a) Design of a test strategy

In order to assess the validity and performance of the proposed approach we have designed a test strategy with two purposes: (1) to verify the performance of our approach as compared against some existing strategies; (2) to study the behaviour of the method as the training (i.e. the learning) increases.

Our proposed combined *HNN* (HN) method is compared against the base classifiers used for the combination. It is also compared against classical combiners that apply the decision rules described below (Kittler et al., 1998; Kuncheva, 2004). Let a pixel i to be classified. *FC* and *BP* provide the membership degree μ_i^j and probability p_i^j respectively, that the pixel i belongs to the cluster w_j. After applying a rule a new support s_i^j is obtained for that pixel of belonging to w_j as follows: (a) Mean rule (ME) $s_i^j = \left(\mu_i^j + p_i^j\right)/2$; (b) Maximum rule (MA) $s_i^j = \max\left\{\mu_i^j, p_i^j\right\}$; (c) Minimum rule (MI) $s_i^j = \min\left\{\mu_i^j, p_i^j\right\}$ and (d) Product rule (PD) $s_i^j = \mu_i^j p_i^j$. These rules have been studied in terms of reliability (Cabrera, 2006). Yager (1988) proposed a multi-criteria decision making approach based on fuzzy sets aggregation. So, HN is also compared against the fuzzy aggregation (FA) where the final support that the pixel i belongs to the cluster w_j is given by the following aggregation rule,

$$s_i^j = 1 - \min\left\{1, \left(\left(1-\mu_i^j\right)^a + \left(1-p_i^j\right)^a\right)^{1/a}\right\} \quad a \geq 1. \tag{16}$$

The parameter a has been fixed to 4 by applying a cross-validation procedure similar to the described in Section 3.1.

Given the supports, according to each rule, the decision about the pixel i is made as follows: $i \in w_j$ if $s_i^j > s_i^k \; \forall w_k \; w_k \neq w_j$.

In order to verify the behaviour of each method as the learning degree increases, we have carried out the experiments according to the following three STEPs:

STEP 1: given the images in S0 and S1, classify each pixel as belonging to a cluster, according to the number of clusters fixed during the training phase. Compute the percentage of successes according to the ground truth defined for each cluster at each image. The classified pattern samples from S1 are added to the previous training samples and a new training process is carried out, but assuming known the number of clusters. This training process is a partial process of the defined in Fig. 1 because the number of clusters does not need to be estimated. This implies that only the parameters associated to each classifier are updated. The set S0 is used as a pattern set in order to verify the performance of the training process as the learning increases. Note that it is not considered for training.

STEPs 2 and 3: perform the same process but using the sets S2 and S3 respectively instead of S1; S0 is also processed as before.

As one can see the number of training samples added at each STEP is $6 \times 512 \times 512$ because this is the number of pixels classified during the STEPs 1 to 3 belonging to the sets S1, S2 and S3.

To verify the performance for each method we have built a ground truth for each image under the supervision of the expert human criterion. Based on the assumption that the automatic training process determines four classes, we classify each image pixel with the simple classifiers obtaining a labeled image with four expected clusters. For each class we build a binary image, which is manually touched up until a satisfactory classification is obtained under the human supervision.

Fig. 2(a) displays an original image belonging to the set S0; (b) displays the correspondence between the classes and the color assigned to the corresponding cluster center according to a color map previously defined; (c) labeled image for the four clusters obtained by our proposed HN approach. The labels are artificial colors identifying each cluster. The correspondence between labels and the different textures is: 1 — yellow with forest vegetation; 2 — blue with bare soil; 3 — green with agricultural crop vegetation; 4 — red with buildings and man made structures.

Fig. 3 displays the binary ground truth images for each cluster, where the white pixels represent the corresponding cluster; (a)–(d) represent the clusters numbers 1–4 respectively identified in Fig. 2.

(b) Results

Table 3 shows the percentage of error during the classification for the different classifiers. For each STEP from 1 to 3, we show the results obtained for both sets of tested images S0 and either S1 or S2 or S3. These percentages are computed as follows. Let I_N^r an image r ($r = 1, \ldots, 6$) belonging to the set *SN* ($N = 0, 1, 2, 3$); i is the node at the location (x, y) in I_N^r. An error counter E_N^r is initially set to zero for each image r in the set *SN* at each STEP and for each classifier. Based on the corresponding decision process, each classifier determines the cluster to which the node i belongs, $i \in w_j$. If the same pixel location on the corresponding ground truth image is black then the pixel is incorrectly classified and $E_N^r = E_N^r + 1$. The error rate of the image I_N^r is: $e_N^r = E_N^r/Z$, where Z is the image size, i.e. 512×512. The average error rate for the set *SN* at each STEP is: $\bar{e}_N = \frac{1}{6}\sum_{r=1}^{6} e_N^r$ and the standard deviation $\bar{\sigma}_N = \sqrt{\frac{1}{5}\sum_{r=1}^{6}\left(e_N^r - \bar{e}_N\right)^2}$. In Table 3 they are displayed as percentages, i.e. $\tilde{e}_N = 100\bar{e}_N$ and $\tilde{\sigma}_N = 100\bar{\sigma}_N$. The numbers in square brackets in the row *HN* indicates the rounded and averaged number of iterations required by each set at each STEP.

(d) Discussion

Based on the error rates displayed in Table 3, we can see that in general, the proposed HN approach outperforms the other methods and achieves the less error rates for STEP 3 in both sets S0 and S1. All strategies achieve the best performance in the STEP 3. Of particular interest is the improvement achieved for the set S0 in STEP 3 with respect its results in STEPs 1 and 2. Based on the above observations, we can conclude that the learning improves

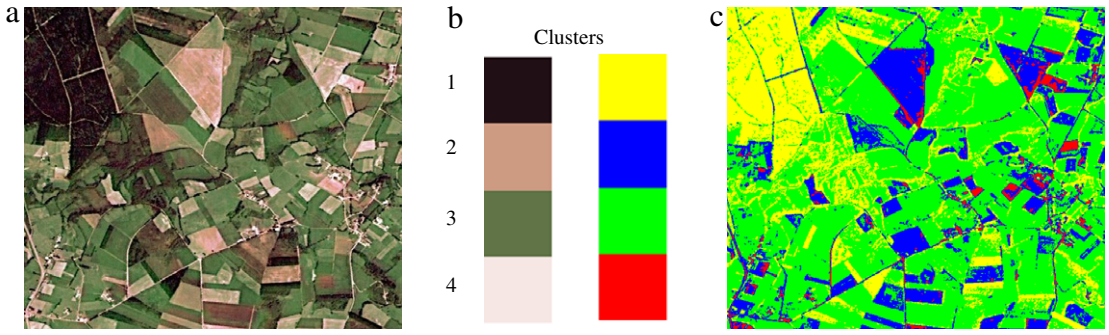

Fig. 2. (a) Original image belonging to the set S0; (b) correspondence between labels and clusters; (c) labeled image with the four clusters according to (b).

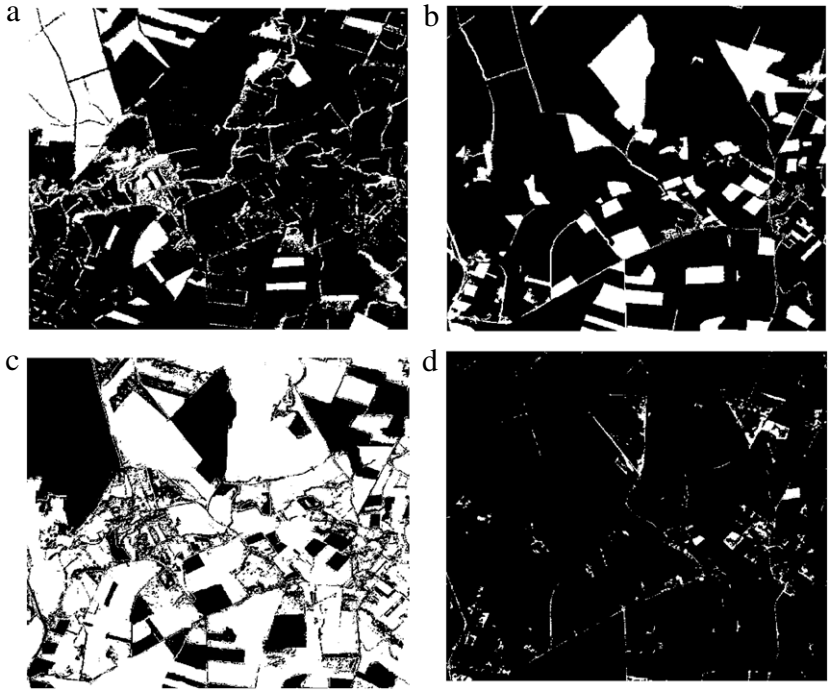

Fig. 3. Ground truth images (a)–(d) for the four clusters 1–4 respectively displayed in Fig. 2.

Table 3
Average percentages of error and standard deviations at each STEP for the four sets of tested images S0, S1, S2 and S3.

		STEP 1				STEP 2				STEP 3			
$\tilde{e}_N$: Average percentage of error		S0		S1		S0		S2		S0		S3	
$\tilde{\sigma}_N$: Standard deviation of error (%)		$\tilde{e}_0$	$\tilde{\sigma}_0$	$\tilde{e}_1$	$\tilde{\sigma}_1$	$\tilde{e}_0$	$\tilde{\sigma}_0$	$\tilde{e}_2$	$\tilde{\sigma}_2$	$\tilde{e}_0$	$\tilde{\sigma}_0$	$\tilde{e}_3$	$\tilde{\sigma}_3$
Hopfield Neural Network	[iterations]	[9]		[10]		[9]		[8]		[7]		[8]	
Fuzzy combination	HN (Hopfield Neural Network)	20.4	1.6	21.6	1.5	18.0	1.2	17.3	1.0	14.8	0.8	17.1	0.8
Combination rules	FA (Yager aggregation)	25.5	2.2	26.8	2.1	24.1	1.9	24.4	1.8	21.5	1.6	20.8	1.5
	MA (Maximum rule)	31.2	2.9	30.7	2.7	28.4	2.8	27.5	2.6	26.9	2.1	26.8	1.9
	MI (Minimum rule)	37.1	3.1	36.9	2.9	32.2	3.3	35.2	2.8	30.9	2.4	28.5	2.3
	ME (Mean rule)	29.1	2.6	28.6	2.2	25.3	2.3	26.4	2.2	25.5	1.9	24.3	1.7
	PR (Product rule)	29.5	2.7	29.1	2.3	25.8	2.4	27.0	2.4	25.2	2.1	25.1	1.8
Simple classifiers	FC (Fuzzy clustering)	32.1	2.8	30.2	2.6	27.1	2.3	27.4	2.3	26.0	2.1	25.9	2.0
	BP (Bayesian Parametric)	30.2	2.7	29.1	2.5	26.1	2.2	26.4	2.2	25.2	2.0	24.7	1.8

the results, i.e. better decisions can be made as the learning increases. This also affects to the HN approach because the network initializations carried out by the FC are better. Moreover, the improved results achieved by the BP also affect the convergence and accuracy through the consistency coefficients according to Eq. (10), consequently the number of iterations required during the optimization decreases. A detailed analysis for groups of classifiers is the following:

(1) *Simple classifiers*: the best performance is achieved by *BP* as compared to *FC*. This suggested us a swapping in the roles of the simple classifiers, so that the networks were initialized with the probabilities instead of the membership degrees, which were used in the consistency coefficient instead of probabilities. The results were similar to those reported here. This means that the HNN optimization strategy has the ability for fusing the information coming from FC and BP independently of their roles.

(2) *Combined rules*: the mean and product rules achieve both similar averaged errors. The performance of the mean is slightly better than the product. This is because, as reported in Tax,

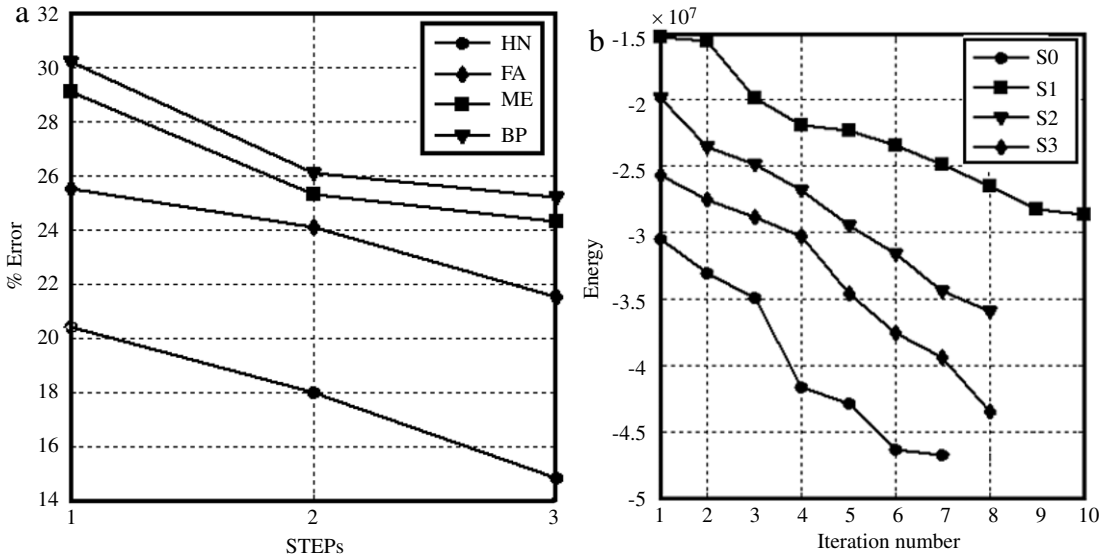

Fig. 4. (a) percentage of error for HN, FA, ME and BP against the three STEPs; (b) energy behaviour for S0 to S3 against the number of iterations.

van Breukelen, Duin, and Kittler (2000), combining classifiers which are trained in independent features spaces result in improved performance for the product rule, while in completely dependent feature spaces the performance is the same. We think that this occurs in our RGB feature space because of the high correlation among the R, G and B spectral components (Cheng, Jiang, Sun, & Wang, 2001; Littmann & Ritter, 1997). High correlation means that if the intensity changes, all the three components will change accordingly.

(3) *Fuzzy combination*: this approach outperforms the simple classifiers and the combination rules. Nevertheless, this improvement requires the convenient adjusting of the parameter a, with other values the results get worse.

For clarity, in Fig. 4(a) the performance of the proposed HN approach for the set SP0 is displayed against FA and the best methods at each group, i.e. ME and BP. Fig. 4(b) shows the energy behaviour for the four sets (S1, S2, S3 and S0 in STEP 3) against the averaged number of iterations required to reach the convergence. The energy decreases as the optimization process increases, as expected according to the Eq. (14). Similar slopes can be observed for the sets S0, S2 and S3. On the contrary, the slope for S1 is smoother; this explains the greater number of iterations required for this set during the convergence.

Overall, the results show that the combined approaches perform favourably for the data sets used. The MA and ME fusion methods provide best results than the individual ones. This means that combined strategies are suitable for classification tasks. This agrees with the conclusion reported in Kuncheva (2003) or Kittler et al. (1998) about the choice of classifiers. Moreover, as the learning increases through STEPs 1 to 3 the performance improves and the number of iterations for S0 decreases. This means that the learning phase is important and that the number of samples affects the performance.

The main drawback of the HNN approach is its execution time, which is greater than for the other methods. Table 4 displays the averaged execution time per image in seconds at each STEP. The same number of iterations displayed in Table 3 is also reproduced. All tests have been implemented in MATLAB and executed on a Pentium M, 1.86 GHz with 1 GB RAM. On average, the execution time per iteration and per image is 14.1 s.

According to the test strategy, three unsupervised training processes have been carried out according to the learning procedure described in Section 3.3(a) based on the *FC* approach. Hence, Table 5 displays the number of training patterns used during the three processes, the number of iterations and the execution time required for convergence. These three processes are: *A*: the initial training where the partition is established; *B*: the training in the STEP 1, which includes the initial training samples plus the samples classified from the set S1; and *C*: the training in the STEP 2, with the previous samples plus the samples classified coming from the set S2.

The times for HNN and FC are not comparable because they are consumed in different phases, i.e. training and decision respectively.

Table 4
Number of iterations and execution times at each STEP for the four sets of testing images S0, S1, S2 and S3.

	STEP 1		STEP 2		STEP 3	
	S0	S1	S0	S2	S0	S3
Number of iterations	9	10	9	8	7	8
Execution time (s)	20.1	23.8	19.9	18.7	16.6	18.8

Table 5
Number of patterns, iterations and execution times for the FC process for the three learning phases carried out.

	Initial training	STEP 1	STEP 2
Number of patterns	786432	2359296	3932160
Number of iterations	18	21	23
Execution time (s)	15.8	55.5	101.3

4. Conclusions

In this paper we have proposed the combination of simple classifiers based on the well-founded HNN paradigm. The network states are updated based on both: self-influence and external influences exerted by neighbors nodes. The external influence embedded in the Eq. (10) can be easily extended if more classifiers are available for combination. We only must define new consistency coefficients, one for each classifier available. Also this scheme could be applied for combining the outputs of modular neural networks during the decision phase, when they provide continuous outputs.

The overall design is an unsupervised classifier from two classical supervised classifiers FC and BP; BP provides the supports

for the network initialization and then these supports are updated under the combination strategy considering the probabilities provided by BP. The roles between FC and BP can be easily interchanged.

The proposed approach outperforms the simple and classical combined strategies. Nevertheless, the main drawback is its execution time during the decision phase. This could be partially avoided through parallel implementations taking into account that only neighboring nodes are inter-connected and the updating process depends only on a reduced neighborhood.

Acknowledgments

The authors would like to thank SITGA (Servicio Territorial de Galicia) in collaboration with the Dimap company (http://www.dimap.es/) for the original aerial images supplied and used in this paper. The authors are also grateful to the referees for their constructive criticism and suggestions on the original version of this paper.

References

Alexandre, L. A., Campilho, A. C., & Kamel, M. (2001). On combining classifiers using sum and product rules. *Pattern Recognition Letters*, *22*, 1283–1289.

Asuncion, A., & Newman, D. J. (2008). *UCI machine learning repository*. Irvine, CA: University of California, School of Information and Computer Science. Available on-line [http://www.ics.uci.edu/~mlearn/MLRepository.html].

Auda, G., & Kamel, M. (1997). CMNN: Cooperative modular neural networks for pattern recognition. *Pattern Recognition Letters*, *18*, 1391–1398.

Auda, G., & Kamel, M. (1998). Modular neural network classifiers: A comparative study. *Journal of Intelligent and Robotic Systems*, *21*, 117–129.

Balasko, B., Abonyi, J., & Feil, B. (2008). *Fuzzy clústering and data analysis toolbox for use with Matlab*, Veszprem University, Hungary (available from URL: http://www.fmt.vein.hu/softcomp/fclusttoolbox/FuzzyCl'usteringToolbox.pdf).

Bezdek, J. C. (1981). *Pattern recognition with fuzzy objective function algorithms*. New York: Kluwer, Plenum Press.

Cabrera, J. B. D. (2006). On the impact of fusion strategies on classification errors for large ensembles of classifiers. *Pattern Recognition*, *39*, 1963–1978.

Cao, J., Shridhar, M., & Ahmadi, M. (1995). Fusion of classifiers with fuzzy integrals. In *Proceedings of the third international conference on document analysis and recognition*. 1, (pp. 108–111).

Cheng, H. D., Jiang, X. H., Sun, Y., & Wang, J. (2001). Color image segmentation: Advances and prospects. *Pattern Recognition*, *34*(12), 2259–2281.

Cohen, M. A., & Grossberg, S. G. (1983). Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, *13*, 815–826.

Deng, D., & Zhang, J. (2005). Combining multiple precision-boosted classifiers for indoor-outdoor scene classification. *Information Technology and Applications*, *1*(4–7), 720–725.

Drimbarean, A., & Whelan, P. F. (2003). Experiments in colour texture analysis. *Pattern Recognition Letters*, *22*(4), 1161–1167.

Duda, R. O., Hart, P. E., & Stork, D. S. (2001). *Pattern classification*. New York: Wiley & Sons.

Hanmandlu, M., Madasu, V. K., & Vasikarla, S. (2004). A fuzzy approach to texture segmentation. In *Proc. of the IEEE international conference on information technology: Coding and computing* (pp. 636–642).

Haykin, S. (1994). *Neural networks: A comprehensive foundation*. New York: Macmillan College Publishing Co.

Hopfield, J. J., & Tank, D. W. (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics*, *52*, 141–152.

Hopfield, J. J., & Tank, D. W. (1986). Computing with neural circuits: A model. *Science*, *233*, 625–633.

Joya, G., Atencia, M. A., & Sandoval, F. (2002). Hopfield Neural Networks for optimization: Study of the different dynamics. *Neurocomputing*, *43*, 219–237.

Kasetkasem, T., & Varshney, P. K. (2002). An image change detection algorithm based on Markov random field models. *IEEE Transactions on Geoscience Remote Sensing*, *40*(8), 1815–1823.

Kittler, J., Hatef, M., Duin, R. P. W., & Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(3), 226–239.

Kong, Z., & Cai, Z. (2007). Advances of research in fuzzy integral for classifier's fusion. In *Proceedings of the 8th ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing*. 2 (pp. 809–814).

Kosko, B. (1992). *Neural networks and fuzzy systems: A dynamical systems approach to machine intelligence*. Englewood Cliffs: Prentice-Hall.

Kumar, S., Ghosh, J., & Crawford, M. M. (2002). Hierarchical fusion of multiple classifiers for hyperspectral data analysis. *Pattern Analysis and Applications*, *5*, 210–220.

Kumar, S., Ghosh, J., & Crawford, M. M. (2001). Best-bases feature extraction for pairwise classification of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, *39*(7), 1368–1379.

Kuncheva, L. I. (2004). *Combining pattern classifiers: Methods and algorithms*. New York: Wiley.

Kuncheva, L. I. (2003). "Fuzzy vs non-fuzzy" in combining classifiers designed by boosting. *IEEE Transactions on Fuzzy Systems*, *11*(6), 729–741.

Laarhoven van, PMJ, & Aarts, EHL (1989). *Simulated annealing: Theory and applications*. Holland: Kluwer Academic.

Lee, D. L., & Chuang, T. C. (2005). Designing asymmetric Hopfield-type associative memory with higher order Hamming stability. *IEEE Transactions on Neural Networks*, *16*(6), 1464–1476.

Littmann, E., & Ritter, H. (1997). Adaptive color segmentation—A comparison of neural and statistical methods. *IEEE Transactions on Neural Networks*, *8*(1), 175–185.

Maillard, P. (2003). Comparing texture analysis methods through classification. *Photogrammetric Engineering and Remote Sensing*, *69*(4), 357–367.

Müezzinoglu, M. K., Güzelis, C., & Zurada, J. M. (2005). An energy function-based design method for discrete Hopfield associative memory with attractive fixed points. *IEEE Transactions on Neural Networks*, *16*(2), 370–378.

Partridge, D., & Griffith, N. (2002). Multiple classifier systems: Software engineered, automatically modular leading to a taxonomic overview. *Pattern Analysis and Applications*, *5*, 180–188.

Poirazi, P., Neocleous, C., Pattichis, C. S., & Schizas, C. N. (2004). Classification capacity of a modular neural network implementing neurally inspired architecture and training rules. *IEEE Transactions Neural Networks*, *15*(3), 597–612.

Puig, D., & García, M. A. (2006). Automatic texture feature selection for image pixel classification. *Pattern Recognition*, *39*(11), 1996–2009.

Qiao, H., Peng, J., & Xu, Z. B. (2001). Nonlinear Measures: A new approach to exponential stability analysis for Hopfield-type neural networks. *IEEE Transactions on Neural Networks*, *12*(2), 360–370.

Randen, T., & Husøy, J. H. (1999). Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(4), 291–310.

Rud, R., Shoshany, M., Alchanatis, V., & Cohen, Y. (2006). Application of spectral features' ratios for improving classification in partially calibrated hyperspectral imagery: A case study of separating Mediterranean vegetation species. *Journal Real-Time Image Processing*, *1*, 143–152.

Smith, G., & Burns, I. (1997). Measuring texture classification algorithms. *Pattern Recognition Letters*, *18*, 1495–1501.

Starink, J. P., & Backer, E. (1995). Finding point correspondences using simulated annealing. *Pattern Recognition*, *28*(2), 231–240.

Tax, D. M. J., van Breukelen, M., Duin, R. P. W., & Kittler, J. (2000). Combining multiple classifiers by averaging or by multiplying? *Pattern Recognition*, *33*, 1475–1485.

Valdovinos, R. M., Sánchez, J. S., & Barandela, R. (2005). Dynamic and static weighting in classifier fusion. In J. S. Marques, N. Pérez de la Blanca, & P. Pina (Eds.), *Lecture notes in computer science*, *Pattern recognition and image analysis* (pp. 59–66). Berlin: Springer-Verlag.

Volkovich, Z., Barzily, Z., & Morozensky, L. (2008). A statistical model of cluster stability. *Pattern Recognition*, *41*(7), 2174–2188.

Wagner, T. (1999). Texture analysis. In B. Jähne, H. Haußecker, & P. Geißler (Eds.), *Signal processing and pattern recognition*: *Vol. 2. Handbook of computer vision and applications*. Academic Press.

Wang, D. (2005). The time dimension for scene analysis. *IEEE Transactions on Neural Networks*, *16*(6), 1401–1426.

Xu, L., & Amari, S-I. (2008). In J. R Rabuñal-Dopico, J. Dorado, & A. Pazos (Eds.), *Combining classifiers and learning mixture-of-experts. Encyclopedia of artificial intelligence* (pp. 318–326). IGI Global (IGI) publishing company.

Xu, L., Krzyzak, A., & Sun, C. Y. (1992). Several methods for combining multiple classifiers and their applications in handwritten character recognition. *IEEE Transactions on Systems, Man and Cybernetics*, *22*, 418–435.

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, *18*(1), 183–190.

Yu, H., Li, M., Zhang, H. J., & Feng, J. (2002). Color texture moments for content-based image retrieval. *Proceedings of the International Conference on Image Processing*, *3*, 24–28.

Yu, S. S., & Tsai, W. H. (1992). Relaxation by the Hopfield Neural Network. *Pattern Recognition*, *25*(2), 197–209.

Zhao, H. (2004). Global asymptotic stability of Hopfield Neural Network involving distributed delays. *Neural Networks*, *17*, 47–53.

Zhao, Z.-Q. (2008). A novel modular neural network for imbalanced classification problems, *Pattern Recognition Letters*, doi:10.1016/j.patrec.2008.06.002.

Zimmermann, H. J. (1991). *Fuzzy set theory and its applications*. Norwell: Kluwer Academic Publishers.