Pattern Recognition Letters 36 (2014) 161–170

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Learning ensemble classifiers via restricted Boltzmann machines

Chun-Xia Zhang [a,*], Jiang-She Zhang [a], Nan-Nan Ji [a], Gao Guo [b]

[a] Institute of Statistical Decision and Machine Learning, Faculty of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an Shaanxi 710049, China
[b] Department of Applied Mathematics, School of Science, Xi'an University of Technology, Xi'an Shaanxi 710054, China

ARTICLE INFO

Article history:
Received 5 May 2013
Available online 25 October 2013

Communicated by A. Marcelli

Keywords:
Ensemble classifier
Bagging
Restricted Boltzmann machine
Deep learning
Majority voting
Diversity

ABSTRACT

Recently, restricted Boltzmann machines (RBMs) have attracted considerable interest in machine learning field due to their strong ability to extract features. Given some training data, an RBM or a stack of several RBMs can be used to extract informative features. Meanwhile, ensemble learning is an active research area in machine learning owing to their potential to greatly increase the prediction accuracy of a single classifier. However, RBMs have not been studied to work with ensemble learning so far. In this study, we present several methods for integrating RBMs with bagging to generate diverse and accurate individual classifiers. Taking a classification tree as the base learning algorithm, a thoroughly experimental study conducted on 31 real-world data sets yields some promising conclusions. When using the features extracted by RBMs in ensemble learning, the best way is to perform model combination respectively on the original feature set and the one extracted by a single RBM. However, the prediction performance becomes worse when the features detected by a stack of 2 RBMs are also considered. As for the features detected by RBMs, good classification can be obtained only when they are used together with the original features.

1. Introduction

An ensemble classifier tries to solve a classification problem by combining the predictions of a collection of base classifiers. Due to its advantage in greatly improving the generalization ability of a learning system, ensemble classifier strategies are rapidly growing and attracting more and more attention from pattern recognition and other various domains over the past two decades (Bauer and Kohavi, 1999; Breiman, 1996; Freund and Schapire, 1997; Kuncheva, 2004; Rokach, 2009, 2010; Zhou, 2012). Ensemble classifier has different names in the literature, such as combination of multiple classifiers, committee of classifiers, mixture of experts, classifier fusion, classifier ensembles, etc. Up to now, ensemble learning techniques have been proven to be quite versatile in a broad range of real applications such as face recognition (Zhang and Zhou, 2010), disease diagnosis (Plumpton et al., 2012; Sun et al., 2012), hand gesture recognition (Wang et al., 2012) and so on.

Generally speaking, the construction of an ensemble classifier can be broken into two steps, that is, classifier generation and classifier fusion. So far, many approaches for creating a diverse set of base classifiers as well as for fusing their predictions have been developed. For some comprehensive surveys about these approaches, one can refer to Kuncheva (2004), Polikar (2006), Rokach (2009, 2010), and Zhou (2012).

Much evidence (Kuncheva, 2004; Rodríguez et al., 2006; Zhang and Zhang, 2010; Zhang and Zhou, 2013) has indicated that diversity and accuracy of the ensemble members are two critical ingredients for an ensemble classifier to have much better generalization ability than its any constituent member. In order to generate diverse and accurate base classifiers in practice, there are two types of commonly used methods. One is to train a set of classifiers from different executions of the same learning algorithm by injecting randomness into the learning algorithm or manipulating the training instances, the input features and the outputs (Breiman, 1996, 2001; Freund and Schapire, 1997; Ho, 1998; Rodríguez et al., 2006; Wang et al., 2013; Zhang and Zhang, 2010). The classifiers obtained in this way are often called *homogeneous* since they are of the same type. The other is to apply some different learning algorithms to the same data set and the derived classifiers are usually called *heterogeneous* (Džeroski and Ženko, 2004; Zhang and Duin, 2009). Among the commonly used ensemble methods, bagging (Breiman, 1996), boosting (Freund and Schapire, 1997), random forest (Breiman, 2001) and random subspace (Ho, 1998) are all the techniques to generate homogeneous classifiers.

With the rapid development of ensemble learning, there are also a lot of hybrid ensemble methods which make use of two or more aforementioned techniques (Baumgartner and Serpen, 2012; Rokach, 2009, 2010; Zhou, 2012; Zhu, 2010). In the domain of ensemble learning, the research on classifier generation usually concentrates on how to produce homogeneous classifiers because the heterogeneous classifiers are relatively easier to obtain. In

* Corresponding author. Tel.: +86 029 82663004; fax: +86 029 82668551.
E-mail address: cxzhang@mail.xjtu.edu.cn (C.-X. Zhang).

0167-8655/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2013.10.009

the current work, we will also focus on studying homogeneous classifiers.

When combining multiple classifiers into a more accurate classifier, many fusion methods such as mean rule, majority voting, weighted average, decision templates, etc. are available. Currently, there has been a certain amount of literature about them (Kuncheva, 2004; Polikar, 2006; Rokach, 2009, 2010; Zhang and Duin, 2011) and interested readers can consult these references for more details. Owing to the fact that majority voting is simple but effective, we will utilize it to merge the predictions of base classifiers.

Bagging (Breiman, 1996) and random subspace (Ho, 1998) may be the two most intuitive and simplest ensemble learning techniques to implement. These two methods both integrate their base classifiers via simple majority voting. The difference between them only lies in how to utilize the given training set to generate a series of different base classifiers. *Bagging*, short for *bootstrap aggregating*, trains its base classifiers by applying a base learning algorithm to some bootstrap samples of the given training data. In other words, bagging executes modifications in the example space in order to create data sets for training base classifiers. As for random subspace (Ho, 1998), it performs modifications in the feature space to generate new training sets for its constituent members. The obtained different training sets are then input into the given learning algorithm to train its base classifiers. Put in another way, random subspace utilizes the strategy of randomly selecting some feature subsets to produce diversity among its individual classifiers.

Besides random subspace method, there also exist many other procedures (Cheng et al., 2006; Crawford and Kim, 2009; García-Pedrajas et al., 2007; Hothorn and Lausen, 2003; Loeff et al., 2008; Rodríguez et al., 2006; Sun et al., 2012) which try to yield diverse base classifiers through manipulating input features. In general, these methods work well when combined with some feature transformation or feature extraction techniques such as LDA (Linear Discriminant Analysis), PCA (Principal Component Analysis), ICA (Independent Component Analysis) and so on. For example, Rodríguez et al. (2006) developed a new classifier combination technique called rotation forest. This method uses PCA to do feature axis rotation for each base classifier so that diversity is promoted. To encourage individual accuracy, it attempts to keep all principal components and also utilizes the whole data set to train each base classifier. Up to now, many experiments (De Bock and Poel, 2011; Rodríguez et al., 2006, 2011; Wang et al., 2012) have shown that rotation forest performs better than bagging (Breiman, 1996), AdaBoost (Freund and Schapire, 1997) and random forest (Breiman, 2001). Since ICA often suffer from the small sample size problem as well as the choice of its independent components if it is applied to solve high-dimensional pattern recognition tasks, Cheng et al. (2006) brought forward a random independent subspace method with the aim to overcome the two problems. Hothorn and Lausen (2003) suggested a double-bagging technique to combine the advantages of bagging and LDA. Based on the philosophy of boosting (i.e., putting more effort on difficult instances), García-Pedrajas et al. (2007) proposed using nonlinear projections to achieve both accuracy and diversity of individual classifiers. Moreover, some scholars (Loeff et al., 2008; Crawford and Kim, 2009) explored the application of manifold learning techniques in multi-classifier systems.

Recently, restricted Boltzmann machines (RBMs) (Bengio, 2009; Fischer and Igel, 2012, 2013; Hinton, 2010) have attracted considerable interest in machine learning community because of their strong representation ability. RBMs are probabilistic graphical models that can be interpreted as stochastic neural networks. With the increase in computational power and the development of faster learning algorithms such as those based on contrastive divergence (Brügge et al., 2013; Hinton, 2002; Tieleman, 2008; Tieleman and Hinton, 2009), RBMs have been proven effective in a range of artificial intelligence tasks such as collaborative filtering, dimension reduction, sparse and over-complete representation (Bengio et al., 2007; Bengio, 2009; Bengio et al., 2012; Cai et al., 2012; Hinton and Salakhutdinov, 2006; Larochelle et al., 2012). In particular, an RBM can extract informative features from a complex data set in an unsupervised manner by introducing hidden units. In other words, an RBM can be used as a nonlinear feature detector. Moreover, some researchers (Bengio, 2009; Hinton et al., 2006; Hinton and Salakhutdinov, 2006) found that a deep belief network composed of several RBMs can detect more abstract features which are very helpful for pattern recognition. For this reason, RBMs have gradually occupy key position in the field of deep learning.

Based on the above analysis, we envisage that some classifiers with strong generalization ability can be trained if integrating RBM with ensemble learning in a suitable manner. This will provide us a new effective way to construct ensemble classifiers for solving some difficult classification tasks. To the best of our knowledge, however, there is no literature on the study of this issue so far. In order to fill this gap, we propose in this paper several possible ways to merging RBMs with bagging to generate diverse and accurate individual classifiers. The experimental studies conducted on 31 benchmark data sets reveal some promising conclusions, among which a classification tree is adopted as the base learning algorithm. When using the features extracted by RBMs in ensemble learning, the best way is to perform model combination respectively on the original feature set and the one extracted by a single RBM. However, the prediction performance becomes worse when the features detected by a stack of 2 RBMs are also considered. As for the features detected by RBMs, good classification can be obtained only when they are used together with the original features.

The rest of this paper is organized as follows. Section 2 provides a brief introduction of RBM. This is followed by introducing several ways for integrating RBMs with bagging to generate multiple base classifiers in Section 3. Section 4 conducts some experiments to examine and compare the performance of the proposed methods. Finally, conclusions and some future work are offered in Section 5.

2. Brief introduction of RBM

2.1. Principle of RBM

An restricted Boltzmann machine (RBM) (Bengio, 2009; Fischer and Igel, 2012, 2013; Hinton, 2010) is a parameterized generative model representing a probability distribution. Given some training data, learning an RBM means adjusting the RBM parameters such that the probability distribution represented by the RBM fits the training data as well as possible. After successful learning, an RBM can provide a closed-form representation of the distribution underlying the training data, which facilitates the following inferences related to the given data. From a structural viewpoint, an RBM is a type of Markov random field (MRF) that consists of a visible and a hidden layer as shown in Fig. 1. For these two layers, there are connections between the hidden and visible units but not connections between two units within the same layer. The visible units constitute the first layer and correspond to the components of an observation (e.g., one visible unit for each feature of an input pattern). The hidden units model dependencies between the components of observations (e.g., dependencies between features). Thus, an RBM can be deemed as a non-linear feature detector.

Owing to the increase in computational power and the development of faster learning algorithms such as those based on contrastive divergence (Brügge et al., 2013; Hinton, 2002; Tieleman, 2008; Tieleman and Hinton, 2009), RBMs can now be applied to more interesting problems. Particularly, RBMs have received a lot of

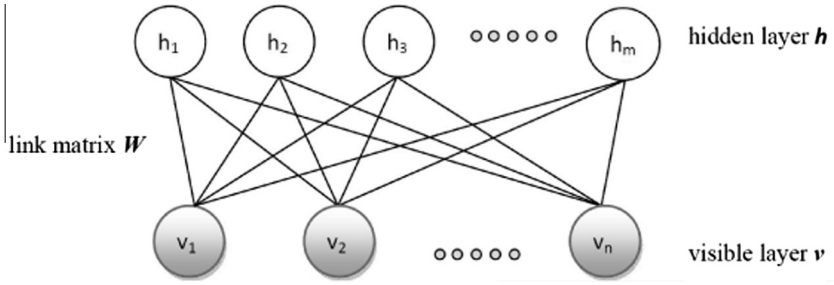

Fig. 1. Schematic diagram of an RBM with n visible units and m hidden units.

attention after being proposed as building blocks of multi-layer architectures called deep belief networks (DBNs) (Bengio, 2009; Hinton et al., 2006; Hinton and Salakhutdinov, 2006). The idea is that the hidden units extract relevant features from the observations. These features can serve as input to another RBM. By stacking RBMs in this way, one can learn features from features in the hope of arriving at a high-level representation. Hence, the features extracted by single or stacked RBMs can serve as input to a supervised learning system.

Fig. 1 provides a schematic diagram of a standard RBM model, i.e., a visible layer $\mathbf{v}$ containing n units to represent observable data and a hidden layer $\mathbf{h}$ composed of m units to capture dependencies between observed variables. The link matrix $\mathbf{W}$ is a real-valued matrix whose element w_{ij} indicates the weight between the visible unit v_i and the hidden unit h_j.

Originally, RBMs were developed using binary visible and hidden units, but many other types of units can be used. Welling et al. (2005) has pointed out that the units in an RBM can be any one in the exponential family, such as softmax units, Gaussian units, binomial units, etc. In order to simplify later discussions, we assume that all the visible and hidden units are binary variables, namely, $\mathbf{v} \in \{0,1\}^n$, $\mathbf{h} \in \{0,1\}^m$. In RBMs with the structure that is illustrated in Fig. 1, the joint probability distribution of $(\mathbf{v},\mathbf{h})$ is given by the Gibbs distribution $P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\exp(-E(\mathbf{v},\mathbf{h}))$ with the energy function

$$E(\mathbf{v},\mathbf{h}) = -\sum_{i=1}^{n} a_i v_i - \sum_{j=1}^{m} b_j h_j - \sum_{i=1}^{n}\sum_{j=1}^{m} v_i w_{ij} h_j, \tag{1}$$

and Z is a normalization factor and $Z = \sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}))$. In the above formula (1), v_i and h_j are the binary states of visible unit i and hidden unit j, respectively, a_i, b_j are their biases and w_{ij} is the weight between v_i and h_j. Here, $\{w_{ij}, a_i, b_j, 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}$ are real-valued parameters that need to be estimated.

2.2. Learning algorithm of RBM

Since RBM is a generative model, its parameters can be optimized by performing stochastic gradient ascent on the log-likelihood of training data. The probability that the network assigns to a training instance (visible vector) is given by summing over all possible hidden vectors, i.e.,

$$P(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h})). \tag{2}$$

The derivative of the log probability of a training vector with respect to a weight is simple and it can be expressed as

$$\frac{\partial \log P(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}, \tag{3}$$

where $\langle\cdot\rangle_{data}$ and $\langle\cdot\rangle_{model}$ denote expectations under the distribution specified by $P(\mathbf{h}|\mathbf{v})$ and $P(\mathbf{v},\mathbf{h})$, respectively. This leads to a very simple learning rule for performing stochastic gradient steepest ascent in the log probability of the training data, namely,

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}). \tag{4}$$

Here, ϵ is a learning rate which needs to be specified in advance by the user.

Because there are no direct connections between hidden units in an RBM, it is very easy to get an unbiased sample of $\langle v_i h_j \rangle_{data}$. Given a randomly selected training vector $\mathbf{v}$, the binary state h_j of each hidden unit j ($j = 1, 2, \ldots, m$) is set to be 1 with probability

$$P(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_{i=1}^{n} v_i w_{ij}\right), \tag{5}$$

where $\sigma(x)$ is the logistic sigmoid function $1/(1+\exp(-x))$. Similarly, it is also easy to obtain an unbiased sample of the state of a visible unit, given a hidden vector $\mathbf{h}$,

$$P(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_{j=m}^{n} h_j w_{ij}\right), \tag{6}$$

since there are no direct connections between visible units. However, getting an unbiased sample of $\langle v_i h_j \rangle_{model}$ is much more difficult. Although it can be done by starting at any random state of the visible units and performing alternating Gibbs sampling until the system reaches steady state, it is time-consuming. Notice that one iteration of alternating Gibbs sampling consists of updating all the hidden units in parallel using Eq. (5) followed by updating all the visible units in parallel using Eq. (6).

Due to the lack of fast learning algorithm, RBM had not received much attention for a long period of time even though it has good properties. Until 2002, Hinton developed a much faster learning procedure based on contrastive divergence (CD), which set off an upsurge in studying RBM and its relevant applications (Bengio et al., 2007, 2012; Brügge et al., 2013; Hinton et al., 2006; Hinton and Salakhutdinov, 2006; Larochelle et al., 2012). The algorithm starts by setting the states of the visible units to a training vector. Then, the binary states of the hidden units are all computed in parallel using Eq. (5). Once binary states have been determined for the hidden units, a "reconstruction" is produced by setting each v_i to 1 with a probability given by Eq. (6). Consequently, the changes in a weight is then given by

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon}), \tag{7}$$

where $\langle v_i h_j \rangle_{recon}$ indicates a distribution obtained by running alternating Gibbs sampling, initialized at the data v_i, for 1 step. For the biases a_i and b_j, a simplified version of the same learning rule that utilizes the states of individual units instead of pairwise products should be used. Put in another way, Hinton (2002) proposed to execute just 1 full step of alternating Gibbs sampling to approximate the expectation $\langle\cdot\rangle_{model}$. Even though the learning is only crudely approximating the gradient of the log probability of the training data, it has proven to behave well enough in many significant

applications (Bengio et al., 2007; Hinton, 2002; Hinton et al., 2006; Hinton, 2010; Larochelle et al., 2012; Welling et al., 2005). In order to make this paper self-contained, we listed the pseudo-code for RBM's fast learning algorithm in Algorithm 1.

Algorithm 1. Pseudo-code for RBM's fast learning algorithm based on CD

- **Input**: a training vector $\mathbf{x}_0$; number of hidden units m; learning rate ϵ; maximum number of iterations T.
- **Output**: weight matrix W; bias vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)^T$ for visible units; bias vector $\mathbf{b} = (b_1, b_2, \ldots, b_m)^T$ for hidden units.
- **Training phase**:

  Initialization: set the starting state of visible units as $\mathbf{v}_1 = \mathbf{x}_0$; let the initial value of W, $\mathbf{a}$ and $\mathbf{b}$ to be small random values.

  For $t = 1, 2, \ldots, T$

  For $j = 1, 2, \ldots, m$ (for all hidden units)
  compute $P(h_{1j} = 1|\mathbf{v}_1)$, i.e., $P(h_{1j} = 1|\mathbf{v}_1) = \sigma\left(b_j + \sum_i v_{1i} w_{ij}\right)$;
  sample $h_{1j} \in \{0, 1\}$ from the conditional distribution $P(h_{1j}|\mathbf{v}_1)$.
  EndFor

  For $i = 1, 2, \ldots, n$ (for all visible units)
  compute $P(v_{2i} = 1|\mathbf{h}_1)$, i.e., $P(v_{2i} = 1|\mathbf{h}_1) = \sigma\left(a_i + \sum_j w_{ij} h_{1j}\right)$;
  sample $v_{2i} \in \{0, 1\}$ from the conditional distribution $P(v_{2i}|\mathbf{h}_1)$.
  EndFor

  For $j = 1, 2, \ldots, m$ (for all hidden units)
  compute $P(h_{2j} = 1|\mathbf{v}_2)$,i.e., $P(h_{2j} = 1|\mathbf{v}_2) = \sigma(b_j + \sum_i v_{2i} w_{ij})$;
  EndFor

  Update each parameter according to the following formulae
  - $W \leftarrow W + \epsilon(P(\mathbf{h}_{1\cdot} = 1|\mathbf{v}_1)\mathbf{v}_1^T - P(\mathbf{h}_{2\cdot} = 1|\mathbf{v}_2)\mathbf{v}_2^T)$;
  - $\mathbf{a} \leftarrow \mathbf{a} + \epsilon(\mathbf{v}_1 - \mathbf{v}_2)$;
  - $\mathbf{b} \leftarrow \mathbf{b} + \epsilon(P(\mathbf{h}_{1\cdot} = 1|\mathbf{v}_1) - P(\mathbf{h}_{2\cdot} = 1)|\mathbf{v}_2)$.

  EndFor

In Algorithm 1, it should be noted that the number of visible units n is equivalent to the dimension of the training vector $\mathbf{x}_0$. The symbol $P(\mathbf{h}_{k\cdot} = 1|\mathbf{v}_k)$ $(k = 1, 2)$ denotes an m-dimensional column vector with its jth element being $P(h_{kj} = 1|\mathbf{v}_k)$. Moreover, notice that the rule for updating the parameters in an RBM as listed in Algorithm 1 is based on ordinary gradient ascent method. Because an RBM is essentially a stochastic neural network, some skills (such as weight decay regularization and momentum) employed in training a neural network can also be borrowed here. Take the weight parameter w_{ij} as an example, let us consider the following update rule

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \underbrace{\epsilon \frac{\partial}{\partial w_{ij}^{(t)}} (\log P(\mathbf{v})) - \lambda w_{ij}^{(t)} + \alpha \Delta w_{ij}^{(t-1)}}_{=\Delta w_{ij}^{(t)}}. \tag{8}$$

If the nonnegative constants λ and α are set to zero, we have vanilla gradient ascent. The constant ϵ is the learning rate. Considering that it is desirable to strive for models with weights having small absolute values, we can optimize an objective function consisting of the log-likelihood minus half of the norm of the parameters $||W||^2/2$ weighted by λ. This method is called weight decay, and penalizes weights with large magnitude. It leads to the $-\lambda w_{ij}^{(t)}$ term in the above update rule (8). As pointed out by Hinton (2010), using weight decay in an RBM has many other advantages such as avoiding overfitting to the training data, improving the learning efficiency of contrastive divergence algorithm and so on. Meanwhile, the update rule can be further extended by a momentum term, $\Delta w_{ij}^{(t-1)}$, weighted by the parameter α. Using a momentum term helps against oscillations in the iterative update procedure and can speed up the learning process, as is seen in feed-forward neural network training. In applications, the parameters λ and α are also need to be tuned. With regard to the corresponding update rule for visible and hidden biases, the formulae are similar to (8) with the exception that they do not involve the weight decay component. The main reason is that the biases are less likely to cause overfitting and they sometimes need to be quite large.

Although Algorithm 1 is designed for the RBMs with binary visible and hidden units, it can be easily extended to the other cases such as Gaussian visible units, Gaussian visible and hidden units and so on. Interested readers can refer to Hinton (2010) and Tran et al. (2011) for more details. In the present study, we will use the case that an RBM has Gaussian visible units and binary hidden units because the observed data are often described by some continuous random variables. Under this condition, the common solution is to first normalize each feature of the training data to have zero mean and unit variance. Then, we only need to change the way to compute the conditional distribution $P(v_{2i} = 1|\mathbf{h}_1)$ in Algorithm 1. Under Gaussian visible units, holding $\mathbf{h}$ fixed, the distribution $P(v_i|\mathbf{h})$ should be computed as

$$P(v_i|\mathbf{h}) = \mathcal{N}\left(a_i + \sum_{j=1}^{m} w_{ij} h_j, 1\right), \tag{9}$$

where $\mathcal{N}(x, 1)$ is a Gaussian distribution with mean x and unit variance.

It is worth to mention that Algorithm 1 summarizes the main steps for updating the weights and biases of an RBM on a single training vector. In practice, this is feasible but time-consuming, especially for large-scale training sets since the parameters need to be updated on each training case. To make the algorithm more efficient, the common solution is to divide the training set into some "mini-batches". This allows matrix–matrix multiplies to be used which is very advantageous on GPU boards or in Matlab. Under this circumstance, it is helpful to divide the total gradient computed on a mini-batch by the size of the mini-batch. In this way, we do not have to change the learning rate when the size of a mini-batch is changed. As for the size of mini-batches, it should be made not too large. Many experiments (Bengio et al., 2007; Bengio, 2009; Bengio et al., 2012; Hinton, 2010) have demonstrated that one mini-batch containing 10 to 100 cases performs in general satisfactory.

When training an RBM, some parameters (i.e., the number of hidden units, the learning rate, weight-cost coefficient, etc.) involved in it should be specified in advance. The optimal setting of them often depends on the problem to be resolved. Fortunately, Hinton (2010) and Bengio (2012) have provided some practical recommendations about how to set appropriate values for these parameters.

2.3. Using RBMs for classification

In addition, we will briefly discuss how RBMs can be used for classification here since the focus of this study is to construct good classifiers. Generally speaking, there are three obvious ways of using RBMs to handle classification tasks. The first is to use the features learned by a single or a stacked of some RBMs as the input for some standard discriminative methods such as neural network,

logistic regression, etc. This is probably the most important way of using RBMs, especially when many layers of features are learned unsupervised before starting on the discriminative training. The second method is to train a separate RBM on each class. After training, the free energy of a test vector can be computed for each class-specific RBM. Then, the log probability that the RBM trained on each class can be calculated accordingly and the test vector can be assigned to the class that has the highest log probability. However, the free energies can not be used directly for classification because each class-specific RBM may have a different, unknown normalization factor. Based on the free energies obtained from all the class-specific RBMs, Hinton (2010) suggested to train another softmax model on a separate training set to predict the class that the test vector belongs to. And the third method (Larochelle et al., 2012) is to train a joint density model using a single RBM that has two parts of visible units with one part representing input data and the other part representing the label of input. Once the training is completed, each possible label is tried in turn with a test vector and the one that gives the lowest free energy is chosen as the most likely class. In the current study, we will make use of the first approach to apply RBMs to tackle classification problems.

3. Novel ensemble classifier generation methods via RBMs

Inspired by the powerful representation capability of RBM as well as the success of ensemble learning worked with other feature extraction techniques, in this section we propose several ways for generating ensemble classifiers via RBMs. Considering the fact that bagging (Breiman, 1996) is simple, easy to implement whereas having good performance, we will mainly study the manners how it can be combined with RBMs. Nevertheless, the combination methods discussed here can be easily extended to other ensemble learning techniques such as boosting (Freund and Schapire, 1997; Sammut and Webb, 2011) and random forest (Breiman, 2001; Sammut and Webb, 2011).

To facilitate the descriptions, we first introduce some notations. Let $\mathscr{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ denote a training set containing N independent instances, in which each case $(\mathbf{x}_i, y_i)$ is described by an input feature vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip}) \in \mathbf{R}^p$ and a class label y_i which takes value from the label space $\Phi = \{1, 2, \ldots, J\}$. Meanwhile, let X be an $N \times p$ matrix composed of the values of p input features for each training instance and Y be an N-dimensional column vector containing the outputs of each training instance in $\mathscr{L}$. In this way, $\mathscr{L}$ can be expressed as concatenating X and Y horizontally, that is, $\mathscr{L} = [X\ Y]$. Denote by $C_1, C_2, \ldots, C_L$ the base classifiers included into an ensemble classifier, say, C^*. When predicting the label y of an instance $\mathbf{x}$, we utilize majority voting to fuse the outputs of $C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_L(\mathbf{x})$. The methods to integrate RBMs with bagging to generate diverse and accurate base classifiers are described as follows.

1RBM: Since an RBM can act as a feature extractor, it is thus feasible to use the feature matrix X to train an RBM. Suppose the matrix composed of the new feature values to be X', we can perform bagging on the data set $[X'\ Y]$ to create multiple base classifiers.

Org+1RBM: Because an RBM is an unsupervised learning method, some features extracted by it may be not useful for solving the classification problem. In this procedure, we use the original features as well as those obtained by RBM to constitute a new feature space. Subsequently, bagging is applied to the data set $[X\ X'\ Y]$. Actually, this method executes data combination according to the terminology developed by Ting et al. (1999).

2Bag: When there are several data sets, we can also use each of them to generate some classifiers and then integrate their predictions in a suitable manner. In this approach, we respectively apply bagging to $[X\ Y]$ and $[X'\ Y]$ to obtain two sets, say, $\mathscr{C}$ and $\mathscr{C}'$, of base classifiers. Given a test instance $\mathbf{x}$, the classifiers in $\mathscr{C}$ can be directly used to produce their predictions. As for the classifiers $\mathscr{C}'$, we need to use the estimated parameters of the RBM to transform $\mathbf{x}$ into the feature space that X' lies in. Assume the new feature vector to be $\mathbf{x}'$, put it as the input of the base classifiers in $\mathscr{C}'$ and the corresponding predictions can be obtained. Virtually, this approach implements model combination.

2RBM: Recently, deep belief nets (DBN) (Bengio et al., 2007; Bengio, 2009; Hinton et al., 2006; Hinton and Salakhutdinov, 2006) have attracted significant interest in the field of artificial intelligence. Through modeling the deep architecture of brain, DBNs possess strong ability to learn high-level or more abstract representations. A DBN is actually a composition of several simple learning modules such as RBMs. It can be trained efficiently by the greedy layer-wise fashion. The main idea of a DBN is as follows. Based on the training matrix X, we can learn an RBM. Then, the output of the first RBM is used as the input of another RBM, which is a key characteristic of DBNs. The parameters included into the second RBM can be similarly trained by the CD algorithm as listed in Algorithm 1. This process can be repeated for several times to learn a deep, hierarchical model.

Motivated by the idea of deep learning, we can use the output of the first RBM, X', as the input of another RBM to extract the second layer of features. After implementing this process, suppose that we obtain the feature matrix X'', bagging is then performed on the data set $[X''\ Y]$ to yield diverse base classifiers.

1+2RBM: In this procedure, we employ the features extracted by the first as well as the second RBM to constitute a new feature set. Put in another way, this method uses $[X'\ X''\ Y]$ as the input of bagging technique.

Org+2RBM: This approach is similar to the previous "Org+1RBM" with the difference lies in that the new feature set is composed as the original features as well those extracted by the second RBM. In other words, this algorithm adopts the set $[X\ X''\ Y]$ as the training set of bagging technique.

Org+1+2RBM: In this method, the training set of bagging is utilized as $[X\ X'\ X''\ Y]$. The feature set contains the original features as well as those extracted by the first and second RBMs.

3Bag: Similar to "2Bag", this method perform model combination. However, except for the base classifiers generated by implementing bagging on $[X\ Y]$ and $[X'\ Y]$, respectively, 3Bag also includes some classifiers created by providing $[X''\ Y]$ as the input of bagging algorithm.

4. Experimental studies

4.1. Experimental data

In this section, we carry out some experiments to examine the performance of the previously proposed methods and compare them across 31 real-world data sets. These sets were selected from the UCI machine learning repository (Frank and Asuncion, 2010) and PRDatasets contained in PRTools which is a free Matlab toolbox for pattern recognition developed by Duin et al. (2007). To ensure a thorough assessment of a method's performance, the chosen data sets vary in size, dimensionality of features and class distributions. At the same time, these data sets were selected to contain only continuous input features so that the unit type for each layer of an RBM can be easier to be determined. Table 1 summarizes the main characteristics of the data sets used in our experiments. For each data set, it lists the number of total instances, number of features and that of classes.

4.2. Experimental setup

Considering the fact that bagging works well to improve the performance of an instable learning algorithm such as neural

Table 1
Summary of characteristics for the used data sets.

Data set	Size	Features	Classes	Data set	Size	Features	Classes
Autompg	398	6	2	Mfeat_zer	2000	47	10
Balance	625	4	3	Musk	5698	166	2
Gamma	19,020	10	2	Pendigits	10,992	16	10
German	1000	24	2	Pima	768	8	2
Gesture	793	84	11	Satimage	6435	36	6
Haberman	306	3	2	Seeds	210	7	3
Heart	270	13	2	Segment	2310	19	7
House	506	13	3	Sonar	208	60	2
Imox	192	8	4	Twonorm	7400	20	2
Isolet	7797	617	26	Vowelc	990	12	11
Madelon	2600	500	2	Waveform	5000	21	3
Mfeat_fac	2000	216	10	Waveformnoise	5000	40	3
Mfeat_fou	2000	76	10	Wdbc	569	30	2
Mfeat_kar	2000	64	10	Wine	178	13	3
Mfeat_mor	2000	6	10	Wpbc	192	32	2
Mfeat_pix	2000	240	10				

networks and decision trees (Breiman, 1996), a classification tree CART (Breiman et al., 1984) was always adopted as the base learning algorithm in all the ensemble methods. In order to make the comparison complete, CART was also taken into account. In the meantime, bagging performed on the given training set was taken as the baseline. The experiments were all conducted in Matlab software with version 7.7. The CART was implemented by the "treefit" contained in the "Stat" package in Matlab. The parameters included into this algorithm were set to be its default values. That is, impure nodes to be split must have at least 10 instances. The tree is fully grown until its stopping criterion is met and it is unpruned. The implementations of the compared methods were realized in Matlab by writing programs according to their respective pseudo-codes.

Note that in the literature of combining homogeneous classifiers created by bagging, one bootstrap sample (e.g., one iteration) corresponds to one base classifier. If the error of a bagging ensemble is plotted as a function of the ensemble size, we can observe that the largest error reduction generally occurs at the first several iterations. Thus, we set the ensemble size in our experiments to be 25. Although larger ensemble size may result in better performance, the improvement achieved at the cost of additional computational complexity is trivial in comparison with that obtained with just a few iterations. With respect to the parameters related to RBMs, their values were set up as follows. The number of hidden units for the two RBMs was respectively taken as 30 and 10. In order to accelerate the learning speed of RBMs, the corresponding training set was made into mini-batches with each containing 10 instances. The maximum number of epoches for learning weights and biases of RBMs was 50. Note that one epoch corresponds to traverse the training set one time for updating the parametric values. The learning rate was chosen as 0.05. In the light of the proposals presented by Hinton (2010), we employed a momentum 0.5 for the initial five epoches and 0.9 for the other epoches with the aim to further enhance the learning speed. Meanwhile, a weight-cost coefficient 0.0002 for L_2 weight-decay was applied to the weights w_{ij} to avoid over-fitting. For each data set, fine-tuning the parameters of RBMs may lead to better results. However, we set them to be the same for all sets due to two reasons. On one hand, adjusting the optimal parameters for each data set is time-consuming. On the other hand, our main aim is to find out an effective procedure to integrate RBMs with bagging by conducting a thoroughly experimental study over 31 benchmark data sets. In addition, it was found that the RBMs behave well under the current parametric setting according to the pre-experiments that we carried out with various values for the parameters pertinent to RBM.

In order to evaluate the performance of each considered classification method on every data set, we adopted 10 runs of stratified 10-fold cross-validation to achieve the task since there are no separate training and test set available for use. For each data set $\mathscr{L}$ of size N, one realization of the stratified 10-fold cross-validation involves the following steps.

(1) Randomly split $\mathscr{L}$ into 10 subsets with approximately equal sizes and each of them having roughly the same class proportion as that in $\mathscr{L}$. Combine 9 out of the 10 subsets into a training set $\mathscr{L}_{train}$ and use the remaining one as a test set $\mathscr{L}_{test}$.
(2) Concerning the convenience to learn the parameters for RBMs having Gaussian visible units, each feature of $\mathscr{L}_{train}$ is normalized to have mean 0 and variance 1. The same transformation is then applied to the test set $\mathscr{L}_{test}$. To simplify notations, $\mathscr{L}_{train}$ and $\mathscr{L}_{test}$ are continued to denote the training and test sets.
(3) Provide $\mathscr{L}_{train}$ as the input of each classification algorithm to train a single classifier C for CART or multiple base classifiers $C_1, C_2, \ldots, C_L$ for each ensemble method. Here, we must mention that for the method 2Bag, we generated approximately the same number of base classifiers respectively with $[X\ Y]$ and $[X'\ Y]$ so that its ensemble size is identical to that of the other methods in order to make the comparison more fair. Regarding 3Bag, the similar procedure was followed.
(4) Utilize each classifier to predict the class label y of each test instance $\mathbf{x}$ in $\mathscr{L}_{test}$. As for each ensemble algorithm, make use of majority voting mechanism to fuse the outputs of their constituent members. Then, compute the test error for each classifier.
(5) Alternate the role of 10 subsets until each of them is used for tested once and the obtained test errors were averaged over 10 folds.

4.3. Results and discussions

Table 2 reports the means and standard deviations of the test errors for each algorithm. In this table, the "Tree" column contains the results of CART and the "1Bag" presents the results of executing bagging on the original training set. In order to facilitate the comparisons, the smallest error for each data set was typed in boldface and the second smallest one was underlined. Furthermore, we performed a one-tailed paired-sample t-test to check whether the best algorithm significantly outperforms the other algorithms on each data set. For each algorithm listed in Table 2, if it is followed by a symbol "•", then the difference between its

Table 2
Means and standard deviations of prediction error (expressed in %) for each method on the benchmark data sets.

Data set	Tree	1Bag	1RBM	Org+1RBM	2Bag	2RBM	1+2RBM	Org+2RBM	Org+1+2RBM	3Bag
Autompg	12.91 ± 4.72	11.66 ± 4.57•	14.35 ± 4.97	11.74 ± 4.59•	11.94 ± 4.33•	17.72 ± 5.88	14.25 ± 4.88	**11.29 ± 4.24**	11.61 ± 4.76•	12.51 ± 4.76
Balance	21.01 ± 4.02	15.71 ± 3.39	33.80 ± 13.78	14.39 ± 3.91•	16.64 ± 4.15	41.42 ± 14.67	33.96 ± 13.70	14.93 ± 4.00•	**14.34 ± 4.14**	22.23 ± 6.86
Gamma	17.81 ± 0.84	**12.49 ± 0.71**	16.88 ± 0.91	12.59 ± 0.70•	12.99 ± 0.70	25.54 ± 1.70	16.83 ± 0.90	12.57 ± 0.72•	12.65 ± 0.70•	14.43 ± 0.75
German	29.12 ± 4.54	**23.48 ± 3.37**	26.18 ± 3.33	25.62 ± 3.62	23.50 ± 3.05•	28.12 ± 4.14	25.89 ± 3.53	24.63 ± 4.23	25.13 ± 3.77	23.72 ± 3.30•
Gesture	46.71 ± 5.96	31.78 ± 5.51•	40.33 ± 4.82	32.14 ± 5.70•	**31.58 ± 5.38**	51.88 ± 5.48	40.61 ± 5.20	32.65 ± 5.25•	31.62 ± 5.60•	35.16 ± 5.03
Haberman	31.16 ± 8.01	30.45 ± 6.69	31.74 ± 6.84	30.10 ± 7.01•	**28.92 ± 6.17**	32.15 ± 7.34	31.54 ± 7.01	30.46 ± 7.19•	29.59 ± 6.62•	29.76 ± 5.78•
Heart	24.22 ± 7.80	18.56 ± 7.64•	18.56 ± 6.48•	18.48 ± 6.85•	**17.41 ± 7.03**	25.04 ± 8.09	18.70 ± 7.97•	19.78 ± 7.78	18.52 ± 6.60•	17.67 ± 7.44•
House	27.86 ± 5.15	21.35 ± 6.08•	24.31 ± 6.09	21.32 ± 6.16•	21.59 ± 6.11•	27.66 ± 6.83	24.60 ± 6.62	**21.26 ± 5.97**	21.42 ± 6.34•	22.38 ± 6.40•
Imox	7.87 ± 5.91	**6.04 ± 4.90**	15.14 ± 8.55	7.61 ± 5.29	6.94 ± 5.80•	24.07 ± 10.77	15.30 ± 7.85	7.35 ± 5.40	8.12 ± 5.88	9.27 ± 7.32
Isolet	18.69 ± 1.29	8.41 ± 0.98	15.57 ± 1.37	7.97 ± 0.95	**7.23 ± 0.99**	28.65 ± 1.76	15.77 ± 1.57	8.59 ± 0.88	8.07 ± 0.92	8.79 ± 1.08
Madelon	24.92 ± 3.42	**18.31 ± 2.28**	33.03 ± 2.46	19.49 ± 2.66	21.28 ± 2.22	43.52 ± 3.05	33.52 ± 2.75	18.35 ± 2.33•	19.30 ± 2.37	24.98 ± 2.74
Mfeat_fac	11.56 ± 2.19	5.51 ± 1.57	6.83 ± 1.73	5.07 ± 1.51	**4.05 ± 1.32**	9.10 ± 1.94	6.81 ± 1.72	5.48 ± 1.55	5.31 ± 1.55	4.48 ± 1.40
Mfeat_fou	26.20 ± 2.79	19.70 ± 2.61	24.17 ± 2.69	**18.64 ± 2.54**	18.72 ± 2.43•	30.50 ± 2.95	24.30 ± 2.63	19.10 ± 2.41•	18.90 ± 2.48•	19.82 ± 2.48
Mfeat_kar	17.56 ± 2.69	8.33 ± 1.92	9.35 ± 2.37	6.90 ± 2.16	**5.24 ± 1.68**	12.45 ± 2.52	9.31 ± 2.31	7.75 ± 2.14	7.29 ± 2.04	5.80 ± 1.98
Mfeat_mor	33.64 ± 2.50	**30.11 ± 2.53**	34.22 ± 3.67	30.24 ± 2.68•	30.62 ± 2.82•	35.55 ± 3.70	33.85 ± 3.98	30.22 ± 2.69•	30.35 ± 2.99•	31.43 ± 2.96
Mfeat_pix	13.19 ± 2.14	6.12 ± 1.72	5.58 ± 1.97	5.51 ± 1.93	3.91 ± 1.52•	8.20 ± 2.24	5.52 ± 1.80	5.74 ± 1.78	5.23 ± 1.75	**3.69 ± 1.42**
Mfeat_zer	33.02 ± 2.74	24.79 ± 2.31	28.02 ± 2.58	25.05 ± 2.25	**23.99 ± 2.16**	31.51 ± 2.91	27.88 ± 2.49	25.21 ± 2.23	24.71 ± 2.43	25.15 ± 2.28
Musk	3.20 ± 0.58	2.28 ± 0.51•	3.02 ± 0.63	**2.20 ± 0.49**	2.30 ± 0.48•	5.71 ± 0.83	3.08 ± 0.65	2.26 ± 0.52•	2.23 ± 0.51•	2.87 ± 0.54
Pendigits	4.18 ± 0.59	1.95 ± 0.38	3.61 ± 0.62	1.93 ± 0.40	**1.54 ± 0.38**	12.19 ± 2.84	3.63 ± 0.58	1.96 ± 0.44	1.99 ± 0.46	2.17 ± 0.43
Pima	28.87 ± 5.28	**23.99 ± 3.87**	27.09 ± 4.70	24.49 ± 4.40•	24.92 ± 4.12	29.58 ± 5.17	27.19 ± 4.93	25.17 ± 4.53	24.92 ± 4.26•	25.25 ± 4.05
Satimage	13.84 ± 1.39	9.53 ± 1.16	10.05 ± 1.10	**8.97 ± 1.03**	8.99 ± 1.06•	14.13 ± 1.30	10.48 ± 1.08	9.61 ± 1.13	9.00 ± 1.06•	9.91 ± 1.14
Seeds	7.71 ± 5.93•	7.14 ± 5.72•	9.76 ± 6.58	7.52 ± 6.14•	**7.05 ± 5.96**	11.10 ± 6.97	9.62 ± 6.70	7.48 ± 5.99•	7.43 ± 6.07•	8.00 ± 6.34•
Segment	4.35 ± 1.50	2.86 ± 1.12•	6.27 ± 1.66	2.79 ± 1.03•	**2.76 ± 0.99**	14.72 ± 2.96	6.31 ± 1.80	3.03 ± 1.13	2.84 ± 0.94•	3.80 ± 1.30
Sonar	29.54 ± 10.17	19.87 ± 9.32•	19.96 ± 10.52•	19.57 ± 9.46•	18.41 ± 9.46•	22.27 ± 8.87	19.42 ± 9.25•	20.40 ± 8.64	19.21 ± 9.24•	**18.09 ± 9.33**
Twonorm	15.79 ± 1.28	4.38 ± 0.70	2.83 ± 0.53	2.54 ± 0.55•	2.68 ± 0.54	2.74 ± 0.57	2.82 ± 0.53	2.66 ± 0.72•	**2.53 ± 0.50**	2.59 ± 0.52•
Vowelc	23.41 ± 4.41	11.24 ± 3.08	16.67 ± 3.80	9.56 ± 2.98	**7.97 ± 2.75**	28.66 ± 6.18	17.06 ± 3.88	9.75 ± 2.75	9.65 ± 3.13	9.65 ± 3.01
Waveform	24.68 ± 2.03	16.78 ± 1.50	14.98 ± 1.54	14.70 ± 1.55	**14.29 ± 1.52**	22.26 ± 2.64	14.93 ± 1.51	16.73 ± 1.64	14.82 ± 1.41	14.85 ± 1.56
Waveformnoise	25.12 ± 1.94	17.33 ± 1.59	15.71 ± 1.41	15.17 ± 1.52	**14.66 ± 1.32**	23.75 ± 4.20	15.86 ± 1.62	16.84 ± 1.67	15.02 ± 1.42	15.16 ± 1.48
Wdbc	7.65 ± 3.62	4.76 ± 2.74	4.69 ± 2.70	4.27 ± 2.57•	**4.01 ± 2.51**	5.68 ± 3.08	4.60 ± 2.82•	4.71 ± 2.76	4.36 ± 2.71•	4.08 ± 2.65•
Wine	9.08 ± 6.50	4.15 ± 5.01•	4.94 ± 5.13	4.39 ± 5.02•	**3.58 ± 4.39**	10.50 ± 7.39	5.51 ± 4.79	4.09 ± 4.34•	4.55 ± 4.88•	4.09 ± 4.50•
Wpbc	35.29 ± 9.64	25.80 ± 6.04•	26.24 ± 7.09	25.78 ± 7.39•	**24.54 ± 5.60**	27.09 ± 6.51	26.97 ± 6.76	25.93 ± 6.10	25.07 ± 6.99•	24.68 ± 5.28•

performance and that of the best algorithm is not significant. Otherwise, the corresponding method behaves significantly worse than the best one. For example, 1Bag has the best performance on German data set. In the meantime, its advantage over the other methods is statistically significant except for 2Bag and 3Bag.

From the results listed in Table 2, we can observe that 2Bag has the best performance among all the compared methods. It achieves the best and second best result respectively on 16 and 6 data sets. Meanwhile, it performs not significantly worse than the best algorithm in 10 cases. Moreover, 1Bag and Org+1RBM are seen to perform best on some data sets (i.e., 6 and 3 ones, respectively). As for the algorithms (i.e., 1RBM, 2RBM and 1+2BM) just based on the features extracted by RBM, their performance seems to be almost always worse than the other methods.

In order to use statistical analysis to draw some conclusions relating to the relative performance of the considered approaches, we employed some statistics utilized by Webb (2000) and Zhang and Zhang (2010) to achieve the purpose. Table 3 provides the comparative summaries of the mean test errors computed previously for each method across all the used data sets, and it has the following format. The first two rows respectively list the average ranks and means of error for each method. Here, the best method evaluated in terms of these two statistics has been highlighted in bold face. In the following descriptions, *row* indicates the mean error on a data set for the algorithm labeled in a row and *col* stands for the mean error on a data set for the algorithm labeled in a column. Rows labeled as $\dot{r}$ present the geometric mean of error ratios col/row. Rows labeled as s give the "win-tie-loss" statistic, where the three values, respectively, refer to the number of data sets for which $col < row$, $col = row$ and $col > row$. Rows labeled as q report the test p-values of a two-tailed sign test based on the win-tie-loss record. If the value of q is smaller than the given significance level α, the difference between the two compared algorithms is significant and otherwise it is not significant.

With regard to the average rank for each method, it was computed in the following way. For each data set, the methods are sorted according to their performance. The best approach has rank 1, the second rank 2, and so on. If several methods have the same result, they are assigned an average value. For each method, its average rank is calculated as the mean across all the data sets.

From the results summarized in Table 3, we can make the following conclusions and discussions:

- It is noteworthy that 2Bag achieves the best performance in terms of all the used statistics. Especially, the average rank for 2Bag is much smaller than that of the other algorithms. The geometric means of error ratios computed for 2Bag over each other method all favor it. Given the significance level $\alpha = 0.05$, the sign test results based on the win-tie-loss records also show that the advantage of 2Bag over all the other methods but 1Bag and Org+1RBM is statistically significant.
- If only taking 1Bag, 1RBM and 2Bag into consideration, it can be found that performing model combination respectively on the original feature space and the space transformed by 1 RBM is beneficial. Although one RBM can detect some informative features, they are not enough for solving the original classification problem. In the process of extracting features, some discriminative information may be lost.
- Take Org+1RBM and 2Bag into comparison, the superiority of 2Bag show that model combination is better than data combination. This conclusion is consistent with that reported by Ting et al. (1999).
- All ensemble methods but 2RBM outperform a single classifier, which demonstrates the ability of ensemble learning to boost the performance of a classifier. The worst behavior of 2RBM

Table 3
Comparative summaries for each method across all the considered data sets.

Method		Tree	1Bag	1RBM	Org+1RBM	2Bag	2RBM	1+2RBM	Org+2RBM	Org+1+2RBM	3Bag
Average rank		8.7097	4.2742	7.4032	3.0968	**2.3065**	9.5161	7.3871	4.5645	3.2581	4.4839
Mean of error		20.33	14.35	17.56	14.09	**13.69**	22.69	17.62	14.39	14.06	14.72
	$\dot{r}$		0.6552	0.8147	0.6289	0.5939	1.1216	0.8185	0.6471	0.6319	0.6553
Tree	s		31-0-0	23-0-8	31-0-0	31-0-0	12-0-19	23-0-8	31-0-0	30-0-1	27-0-4
	q		<0.0001	0.0107	<0.0001	<0.0001	0.2810	0.0107	<0.0001	<0.0001	<0.0001
1Bag	$\dot{r}$			1.2435	0.9599	0.9065	1.7119	1.2492	0.9877	0.9645	1.0001
	s			5-1-25	20-0-11	21-0-10	1-0-30	6-0-25	14-0-17	21-0-10	13-0-18
	q			0.0003	0.1496	0.0708	<0.0001	0.0009	0.7201	0.0708	0.4731
1RBM	$\dot{r}$				0.7719	0.7290	1.3767	1.0046	0.7943	0.7757	0.8043
	s				31-0-0	31-0-0	1-0-30	15-0-16	25-0-6	31-0-0	31-0-0
	q				<0.0001	<0.0001	<0.0001	>0.9999	0.0009	<0.0001	<0.0001
Org+1RBM	$\dot{r}$					0.9444	1.7834	1.3014	1.0289	1.0048	1.0419
	s					20-0-11	0-0-31	1-0-30	9-0-22	13-0-18	11-0-20
	q					0.1496	<0.0001	<0.0001	0.0294	0.4731	0.1496
2Bag	$\dot{r}$						1.8884	1.3780	1.0895	1.0640	1.1033
	s						0-0-31	0-0-31	8-0-23	8-1-22	3-0-28
	q						<0.0001	<0.0001	0.0107	0.0161	<0.0001
2RBM	$\dot{r}$							0.7297	0.5769	0.5634	0.5842
	s							30-0-1	31-0-0	31-0-0	31-0-0
	q							<0.0001	<0.0001	<0.0001	<0.0001
1+2RBM	$\dot{r}$								0.7906	0.7721	0.8806
	s								25-0-6	31-0-0	31-0-0
	q								0.0009	<0.0001	<0.0001
Org+2RBM	$\dot{r}$									0.9766	1.0126
	s									22-0-9	14-1-16
	q									0.0294	0.8555
Org+1+2RBM	$\dot{r}$										1.0369
	s										9-1-21
	q										0.0428

can be explained as follows. Note that the parameters of RBMs are learned in an unsupervised way. After executing two feature transformations with RBMs, the extracted features contain less discriminative information for classification.

- With respect to the four methods (i.e., 2RBM, 1+2RBM, Org+2RBM and Org+1+2RBM) that make use of the features detected by 2 RBMs, we can observe that original features can help to improve the classification ability of RBMs to some extent. This conclusion tells us that simultaneously using the original feature set together one or two sets obtained by RBMs is generally better than using only one transformed feature set.
- Through comparing 2Bag and 3Bag, it can be seen that performing model combination based on the original feature set and that learned by 1 RBM is profitable. Nevertheless, the classification performance becomes worse when also considering the feature set extracted by 2 RBMs.

In the light of the above-mentioned analysis, one can find that stacking 2 RBMs seems to be unhelpful and even deleterious to the performance of bagging under the current circumstances. This phenomenon is inconsistent with the results reported in the literature of deep learning (Hinton et al., 2006; Hinton and Salakhutdinov, 2006), the reasons may be explained as follows. In the experiments conducted by the other scholars, the results output from a stacked of RBMs are generally used to initialize a neural network which is then fine-tuned with the feed-forward learning algorithm. Just as pointed out by Bengio (2009), this type of unsupervised pre-training executed by stacking RBMs acts as a regularizer to help finding better minima of the optimized criterion. In our experiments, however, we directly adopted the features extracted by RBMs as the input of bagging method in which a classification tree is used as the base learning algorithm. In a word, RBMs are used in different manner in the experiments conducted by other scholars and us. In the meantime, the used classification method is also different.

Furthermore, we also utilized Wilcoxon signed-ranks test to evaluate whether there is significant difference between the comparative algorithms. The reason in doing so is that outliers (exceptionally goog/bad performances on a few data sets) have less effect on the Wilcoxon than on the t-test (Demšar, 2006). In the following part, we briefly present the steps to carry out a Wilcoxon signed-ranks test. Suppose that there are n objects to be observed by two algorithms, let us denote the difference value of the two algorithms' observation on the ith objects to be d_i, $i = 1, 2, \ldots, n$. The differences are ranked according to their absolute values, ranks of the tied values are averaged. Let R^+ stand for the sum of the ranks of the objects on which the difference value of the two algorithms' observations are greater than zero, and R^- denote the sum of the opposite. Ranks of $d_i = 0$ are evenly split between R^+ and R^-. The values of R^+ and R^- can be calculated as

$$R^+ = \sum_{d_i>0} rank(|d_i|) + \frac{1}{2}\sum_{d_i=0} rank(|d_i|), \tag{10}$$

$$R^- = \sum_{d_i<0} rank(|d_i|) + \frac{1}{2}\sum_{d_i=0} rank(|d_i|). \tag{11}$$

Let T be the smaller of the sums, namely, $T = \min(R^+, R^-)$. The significance value that T should be equal or less than for rejection of a null hypothesis can be retrieved by querying the critical value table. Most books on statistics include such a table for n up to 25 (or sometimes more). For a larger number of data sets, the statistic

$$Z = \frac{T - \frac{1}{4}n(n+1)}{\sqrt{\frac{1}{24}n(n+1)(2n+1)}} \tag{12}$$

is distributed approximately normally. With significance level $\alpha = 0.05$, the null hypothesis can be rejected if Z is smaller than -1.96. In the rest of this section, Wilcoson signed-ranks test was conducted between 2Bag and each of the other comparative algorithms and the results were summarized in Table 4. Given 31 data sets and $\alpha = 0.05$, we should reject the null hypothesis that there is no significant difference between the two compared algorithms if $T \leqslant 148$. In Table 4, the "(+)" symbol signifies that 2Bag is quantitatively better than the comparative algorithm under consideration.

The results in Table 4 indicate that 2Bag has significant advantage over each of its rivals. Moreover, the values of R^+ and R^- reveal that several methods (i.e., Tree, 1RBM, 2RBM and 1+2RBM) have larger test errors than 2Bag on each considered data set since the corresponding values of R^+ are equal to zero.

5. Conclusions and future work

In multi-classifier systems, it has proven that good ensemble classifiers can be generally obtained via manipulating input features assisted by some feature transformation techniques such as PCA, ICA and so on. Although much evidence has proven that an RBM can act as a useful feature extractor, it has not been applied to construct ensemble classifiers so far. In this paper, we investigated some possible ways for integrating RBMs with bagging to generate diverse and accurate individual classifiers. In the light of the detailed analysis of classification error, the experimental studies carried out on 31 benchmark data sets yield the following conclusions.

- When using the features extracted by RBMs in ensemble learning, the best way is to perform model combination respectively on the original feature set and the one extracted by a single RBM. Nevertheless, the prediction performance becomes worse when also considering the feature set extracted by stacking 2 RBMs, namely, 3Bag performs worse than 2Bag.
- As for the features detected by RBMs, good classification results can be obtained only when they are utilized together with the original features. Because RBMs are trained in an unsupervised manner, the extracted features may contain less discriminative information. For example, the classifier trained by using the feature set learned by a stack of 2 RBMs (i.e., 2RBM) performs even worse than one classification tree in some cases.
- It is also possible to execute data combination through constitute a large feature set which contain the original features as well as those extracted by a single or a stack of 2 RBMs. However, this method is outperformed by model combination.

Table 4
Wilcoxon signed-ranks test results between 2Bag and each of the other methods.

	Tree	Org	1RBM	Org+1RBM	2RBM	1+2RBM	Org+2RBM	Org+1+2RBM	3Bag
R^+	0	112	0	120	0	0	92	107.5	13
R^-	496	384	496	376	496	496	404	388.5	483
T	0(+)	112(+)	0(+)	120(+)	0(+)	0(+)	92(+)	107.5(+)	13(+)

In the current study, we focused on studying the ways to integrate RBMs with bagging technique. Although the approaches discussed here can be easily extended to other ensemble learning techniques such as boosting and random forest, how they will perform in these situations is one of our interested future tasks. On the other hand, the parameters of RBMs (especially for 2 RBMs) are currently learned in an unsupervised way. It is interesting to study how to effectively utilize the information provided by class labels so that some discriminative features can be extracted. This may further improve the performance of ensemble classifiers to some extent. Moreover, diversity among individual classifiers is widely recognized to play a key role in the success of an ensemble. We believe that better ensemble classifiers can be attained if RBM feature extraction is fused with random feature selection to further enhance diversity. Nevertheless, how to integrating them properly and how many features should be randomly chosen are some aspects deserved to be further investigated.

Acknowledgements

This research was supported by the National Basic Research Program of China (973 Program, No. 2013CB329406), the National Natural Science Foundations of China (Nos. 11201367, 61075006), the Major Research Project of the National Natural Science Foundation of China (No. 91230101), the Research Fund for the Doctoral Program of Higher Education of China (No. 20100201120048), the Fundamental Research Funds for the Central Universities of China and the Foundation of Shaanxi Provincial Department of Education in China (No. 09JK615).

References

Bauer, E., Kohavi, R., 1999. An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Mach. Learn. 36 (1–2), 105–139.
Baumgartner, D., Serpen, G., 2012. A design heuristic for hybrid classification ensembles in machine learning. Intell. Data Anal. 16 (2), 233–246.
Bengio, Y., 2009. Learning deep architectures for AI. Found. Trends Mach. Learn. 2 (1), 1–127.
Bengio, Y., 2012. Practical recommendations for gradient-based training of deep architectures. In: Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol. 7700. Springer, Berlin, pp. 437–478.
Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., 2007. Greedy layer-wise training of deep networks. Advances in Neural Information Processing System, vol. 19, pp. 153–160.
Bengio, Y., Chapados, N., Delalleau, O., Larochelle, H., Saint-Mleux, X., Hudon, C., Louradour, J., 2012. Detonation classification from acoustic signature with the restricted Boltzmann machine. Comput. Intell. 28 (2), 261–288.
Breiman, L., 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140.
Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.
Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Chapman & Hall, CRC Press, New York.
Brügge, K., Fischer, A., Igel, C., 2013. The flip-the-state transition operator for restricted Boltzmann machines. Mach. Learn. 93 (1), 53–69.
Cai, X.G., Hu, S., Lin, X.L., 2012. Feature extraction using restricted Boltzmann machine for stock price prediction. In: Proc. IEEE Int. Conf. Comput. Sci. Autom. Eng., pp. 80–83.
Cheng, J., Liu, Q.S., Lu, H.Q., Chen, Y.W., 2006. Ensemble learning for independent component analysis. Pattern Recognit. 39 (1), 81–88.
Crawford, M., Kim, W., 2009. Manifold learning for multi-classifier systems via ensembles. In: Benediktsson, J.A., Kitller, J., Roli, F. (Eds.), MCS, Lecture Notes in Computer Science, vol. 5519. Springer, Berlin, pp. 519–528.
De Bock, K.W., Poel, D.V., 2011. An empirical evaluation of rotation-based ensemble classifiers for customer churn prediction. Expert Syst. Appl. 38 (10), 12293–12301.
Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. 7, 1–30.
Duin, R.P.W., Juszczak, P., Paclík, P., Pękalska, E., Ridder, D., Tax, D.M.J., Verzakov, S., 2007. PRTools4: a matlab toolbox for pattern recognition. Delft University of Technology, Delft.
Džeroski, S., Ženko, B., 2004. Is combining classifiers with stacking better than selecting the best one? Mach. Learn. 54 (3), 255–273.
Fischer, A., Igel, C., 2012. An introduction to restricted Boltzmann machines. In: Alvarez et al. (Eds.), CIARP, Lecture Notes in Computer Science, vol. 7441. Springer, Berlin, pp. 14–36.
Fischer, A., Igel, C., 2014. Training restricted Boltzmann machines: an introduction. Pattern Recognit. 47 (1), 25–39.
Frank, A., Asuncion A., 2010. UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml.
Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Sys. Sci. 55 (1), 119–139.
García-Pedrajas, N., García-Osorio, C., Fyfe, C., 2007. Nonlinear boosting projections for ensemble construction. J. Mach. Learn. Res. 8, 1–33.
Hinton, G.E., 2002. Training products of experts by minimizing contrastive divergence. Neural Comput. 14 (8), 1771–1800.
Hinton, G.E., 2010. A practical guide to training restricted Boltzmann machines. Department of Computer Science, University of Toronto, Montreal.
Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. Science 313, 504–507.
Hinton, G.E., Osindero, S., Teh, Y.W., 2006. A fast learning algorithm for deep belief nets. Neural Comput. 18 (7), 1527–1554.
Ho, T.K., 1998. The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. 20 (8), 832–844.
Hothorn, T., Lausen, B., 2003. Double-bagging: combining classifiers by bootstrap aggregation. Pattern Recognit. 36 (6), 1303–1309.
Kuncheva, L.I., 2004. Combining Pattern Classifiers, Methods and Algorithms. Wiley Interscience, John Wiley & Sons, New Jersey.
Larochelle, H., Mandel, M., Pascanu, R., Bengio, Y., 2012. Learning algorithms for the classification restricted Boltzmann machine. J. Mach. Learn. Res. 13, 643–669.
Loeff, N., Forsyth, D., Ramachandran, D., 2008. ManifoldBoost: stagewise function approximation for fully-, semi-, and un-supervised learning. In: Proc. 25th Int. Conf. Mach. Learn., Helsinki, Finland, pp. 600–607.
Plumpton, C.O., Kuncheva, L.I., Oosterhof, N.N., Jognston, S.J., 2012. Naive random subspace ensemble with linear classifiers for real-time classification of fMRI data. Pattern Recognit. 45 (5), 2101–2108.
Polikar, R., 2006. Ensemble based systems in decision making. IEEE Circuits Syst. Mag. 6 (3), 21–45.
Rodríguez, J.J., Kuncheva, L.I., Alonso, C.J., 2006. Rotation forest: a new classifier ensemble method. IEEE Trans. Pattern Anal. Mach. Intell. 28 (10), 1619–1630.
Rodríguez, J.J., Díez-Pastor, J.F., García-Osorio, C., 2011. Ensembles of decision trees for imbalanced data. In: Sansone, C., Kittler, J., Roli, F. (Eds.), MCS, Lecture Notes in Computer Science, vol. 6713. Springer, Berlin, pp. 76–85.
Rokach, L., 2009. Taxonomy for characterizing ensemble methods in classification tasks: a review and annotated bibliography. Comput. Stat. Data Anal. 53 (12), 4046–4072.
Rokach, L., 2010. Ensemble-based classifiers. Artif. Intell. Rev. 33 (1–2), 1–39.
Sammut, C., Webb, G.I., 2011. Encyclopedia of Machine Learning. Springer-Verlag, New Work.
Sun, Z.L., Zheng, C.H., Gao, Q.W., Zhang, J., Zhang, D.X., 2012. Tumor classification using eigengene-based classifier committee learning algorithm. IEEE Signal Process Lett. 19 (8), 455–458.
Tieleman, T., 2008. Training restricted Boltzmann machines using approximations to the likelihood gradient. In: Proc. 25th Int. Conf. Mach. Learn., Helsinki, Finland, pp. 1064–1071.
Tieleman, T., Hinton, G.E., 2009. Using fast weights to improve persistent contrastive divergence. In: Proc. 26th Int. Conf. Mach. Learn., Montreal, Canada, pp. 1033–1040.
Ting, K.M., Low, B.T., Witten, I.H., 1999. Learning from batched data: models combination vs data combination. J. Knowl. Inf. Syst. 1 (1), 83–106.
Tran, T., Phung, D.Q., Venkatesh, S., 2011. Mixed-variate restricted Boltzmann machines. J. Mach. Learn. Res. 20, 213–229.
Wang, G.W., Zhang, C.X., Zhuang, J., 2012. An application of classifier combination methods in hand gesture recognition. Math. Prob. Eng.. Article ID 346951, 17 pages.
Wang, Z., Jie, W.B., Chen, S.C., Gao, D.Q., 2013. Random projection ensemble learning with multiple empirical kernels. Knowledge-Based Syst. 37, 388–393.
Webb, G.I., 2000. Multiboosting: a technique for combining boosting and wagging. Mach. Learn. 40 (2), 159–196.
Welling, M., Rosen-Zvi, M., Hinton, G.E., 2005. Exponential family harmoniums with an application to information retrieval. Advances in Neural Information Processing System, vol. 17. Cambridge, MIT Press, pp. 1481–1488.
Zhang, C.X., Duin, R.P.W., 2009. An empirical study of a linear regression combiner on multi-class data sets. In: Benediktsson, J.A., Kitller, J., Roli, F. (Eds.), MCS, Lecture Notes in Computer Science, vol. 5519. Springer, Berlin, pp. 478–487.
Zhang, C.X., Duin, R.P.W., 2011. An experimental study of one- and two-level classifier fusion for different sample sizes. Pattern Recognit. Lett. 32 (14), 1756–1767.
Zhang, C.X., Zhang, J.S., 2010. A variant of rotation forest for constructing ensemble classifiers. Pattern Anal. Appl. 13 (1), 59–77.
Zhang, Y., Zhou, Z.H., 2010. Cost-sensitive face recognition. IEEE Trans. Pattern Anal. Mach. Intell. 32 (10), 1758–1769.
Zhang, M.L., Zhou, Z.H., 2013. Exploring unlabeled data to enhance ensemble diversity. Data Min. Knowl. Discovery 26 (1), 98–129.
Zhou, Z.H., 2012. Ensemble Methods: Foundations and Algorithms. Chapman & Hall/CRC, Boca Raton.
Zhu, D., 2010. A hybrid approach for efficient ensembles. Decis. Support Syst. 48 (3), 480–487.