PERGAMON

AVAILABLE AT
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Neural Networks 16 (2003) 261–269

Neural Networks

www.elsevier.com/locate/neunet

Clustering ensembles of neural network models

Bart Bakker*, Tom Heskes

SNN, University of Nijmegen, Geert Grooteplein 21, 6525 EZ Nijmegen, The Netherlands

Received 25 January 2002; accepted 26 July 2002

Abstract

We show that large ensembles of (neural network) models, obtained e.g. in bootstrapping or sampling from (Bayesian) probability distributions, can be effectively summarized by a relatively small number of representative models. In some cases this summary may even yield better function estimates. We present a method to find representative models through clustering based on the models' outputs on a data set. We apply the method on an ensemble of neural network models obtained from bootstrapping on the Boston housing data, and use the results to discuss bootstrapping in terms of bias and variance. A parallel application is the prediction of newspaper sales, where we learn a series of parallel tasks. The results indicate that it is not necessary to store all samples in the ensembles: a small number of representative models generally matches, or even surpasses, the performance of the full ensemble. The clustered representation of the ensemble obtained thus is much better suitable for qualitative analysis, and will be shown to yield new insights into the data.

Keywords: Clustering; Bootstrapping; Bias/variance analysis; Multitask learning; Deterministic annealing; Expectation-maximization algorithm; Multilayered perceptron; Nonlinear model

1. Introduction

In neural network analysis we often find ourselves confronted with a large ensemble of models trained on one database. One example of this is resampling, which is a popular approach to try and obtain better generalization performance with nonlinear models. Individual models are trained on slightly different samples of the available data set, which are generated e.g. by bootstrapping or cross-validation (Efron & Tibshirani, 1993). Another example can be found in the Bayesian approach, which creates a probability distribution over all possible models, which may be sampled from by Markov Chain Monte Carlo (MCMC) procedures (Neal, 1996).

In both cases a considerable number of network representations, or models, may be needed to catch the fine nuances of the system. For complex problems, the number of models may even be too high to keep a good overview of the ensemble and special transformations will be required to summarize it, preferably without loss of information. The clustering procedure that we will present will help to understand such problems better, and may be a valuable tool in their analysis.

Clustering can be seen as the representation of a large collection W by a smaller collection of representative entities, M. The components of W and M may be anything: locations on a map, people, words, models, etc. The type of the elements may even vary from set W to set M, or within sets (Cadez, Gaffney, & Smyth, 2000). The only requirement is that there exists a distance function $D(W, M)$ indicating how much sets W and M differ from each other, or rather how much information is lost in the conversion from W to M.

Since W and M may contain any kind of elements, the method of clustering may well be used to find a workable representation of any oversized ensemble of (neural network) models. Taking the elements of W to be the models in the original (large) ensemble, represented in the case of neural networks by their weights, biases and overall structure (number of hidden layers, transfer functions, etc.) M can be a smaller set of networks best representing the features contained in W.

In Section 2 we will review the clustering method outlined by Rose, Gurewitz, and Fox (1990). Their method, based on the principles of deterministic annealing as first described by Jaynes (1957), was shown to yield good results

* Corresponding author. Tel.: +31-24-361-4230.
E-mail addresses: bartb@mbfys.kun.nl (B. Bakker), tom@mbfys.kun.nl (T. Heskes).

0893-6080/03/$ - see front matter © 2003 Elsevier Science Ltd. All rights reserved.
PII: S0893-6080(02)00187-9

for the clustering of two-dimensional data with a Euclidean distance function. We will generalize this method for use with other data types and distance functions, and use it to cluster models (e.g. neural networks). In Section 3 we describe the algorithms that we use to implement the method.

Although clustering in weight space has been used in network analysis (Rüger & Ossen, 1996), model clustering as described in this article is, to the best of our knowledge, a new approach. In our approach the distance function $D(W, M)$ is based on model outputs instead of model parameters. We feel this approach is more intuitive, since model outputs on a known database provide a more direct representation of the models' characteristics than the more abstract model parameters themselves.

The expression of the distance in terms of model outputs, combined with a suitable distance formula (e.g. sum-squared distance), further allows us to lower the computation time required to perform clustering considerably. We accomplish this by expressing not only the distance, but also the representative models themselves in terms of their outputs. We will show in Section 3 how this leads to an algorithm that is much faster than one finding representative models in terms of their model parameters. The final result, however, needs to be in the form of representative *models*, and not *model outputs*. In Section 4 we discuss several methods to make the translation from model outputs to model parameters. We compare the methods in terms of computation time and quality of the obtained representatives.

We apply our method on two databases in Section 6.2. The first, containing the well-known Boston housing data, serves as a benchmark problem to study the effect of bootstrapping. We apply the clustering algorithm that is described in this article on ensembles of models obtained through bootstrapping, and use the resulting clusters to illustrate the effect of bootstrapping in terms of bias/variance reduction.

The other database concerns the prediction of single-copy newspaper sales in the Netherlands. This can be represented as a series of parallel tasks (outlets at different locations), which we optimize through multitask learning. In multitask learning (Heskes, 1998) one is presented with a (preferably large) number of parallel tasks, e.g. predicting student test results for students in a series of schools, or survival analysis on patients in a series of hospitals. Such a series of tasks can be represented by a series of models, each trained on one of the tasks. Although any one of these models may be overfitting its training data to some extent, the ensemble of all models may yield a good estimation of the underlying function. We show that a clustered representation of this ensemble can give valuable new insights into the nature of the problem. Also, the knowledge about the grouping of the models into tasks may allow the clustered solution to reach a significantly lower prediction error than the full ensemble.

2. Clustering by deterministic annealing

2.1. Notation

Suppose we have N_W models, fully characterized through their parameters $\mathbf{w}_i$, $i = 1, \ldots, N_W$. We define N_M other models, which we will refer to as cluster centers, denoted $\mathbf{m}_\alpha$, $\alpha = 1, \ldots, N_M$. $D(\mathbf{w}_i, \mathbf{m}_\alpha)$ is the distance of model i to cluster center α. The distance need not be symmetric, i.e. $D(\mathbf{w}, \mathbf{m})$ may be different from $D(\mathbf{m}, \mathbf{w})$.[1] We do, however, require that $D(\mathbf{w}, \mathbf{m}) \geq 0$ and $D(\mathbf{m}, \mathbf{m}) = 0$. The models considered in the present article are feedforward models that are optimized through supervised training.

We assume distances of the form

$$D(\mathbf{w}_i, \mathbf{m}_\alpha) = \sum_\mu d(y(\mathbf{w}_i, \mathbf{x}^\mu), y(\mathbf{m}_\alpha, \mathbf{x}^\mu)), \tag{1}$$

for some distance measure $d(y_1, y_2)$, where $y(\mathbf{w}_i, \mathbf{x}^\mu)$ is the output of a model with parameters $\mathbf{w}_i$ on an input $\mathbf{x}^\mu$. Each model is supposed to have the same input. Since, once the inputs $\mathbf{x}^\mu$ are given, the clustering procedure depends on the models $\mathbf{w}_i$ only through the outputs $y(\mathbf{w}_i, \mathbf{x}^\mu)$, we can compute these outputs in advance.

2.2. The derivation of 'free energy'

We define variables $p_{i\alpha}$ as the probability that model i belongs to cluster α. The distances between models $\mathbf{w}_i$ and cluster centers $\mathbf{m}_\alpha$ are given by $D(\mathbf{w}_i, \mathbf{m}_\alpha)$. We assume that the models $\mathbf{w}_i$ are given, but that the probabilities $p_{i\alpha}$ and cluster centers $\mathbf{m}_\alpha$ are still free to choose. One of the goals of clustering is to put the cluster centers such as to minimize the average distance of the models to the cluster centers, i.e. to find a low average energy

$$E(\{m\}, \{p\}) = \sum_{i\alpha} p_{i\alpha} D(\mathbf{w}_i, \mathbf{m}_\alpha), \tag{2}$$

where $\{m\}$ and $\{p\}$ represent the full sets $\mathbf{m}_\alpha$ and $p_{i\alpha}$ $\forall_{i\alpha}$. In this framework the average energy is the average over the distances between models and cluster centers, weighted by $p_{i\alpha}$. For fixed cluster centers $\mathbf{m}_\alpha$ minimizing the average energy would correspond to assigning each model to its nearest cluster center with probability one. A proper way to regularize this is through a penalty term of the form

$$S(\{p\}) = -\sum_{i\alpha} p_{i\alpha} \log p_{i\alpha}, \tag{3}$$

the discrete version of the Shannon entropy, which is the only quantity which is positive, increases with increasing uncertainty, and is additive for independent sources of uncertainty (Jaynes, 1957). Maximizing $S(\{p\})$ therefore favors a state of total chaos, i.e. $p_{i\alpha} = p_{i'\alpha'}$ $\forall_{i,i',\alpha,\alpha'}$, which

[1] Note that $D(\mathbf{w}, \mathbf{m})$ is not a distance function in the mathematical sense, but rather a measure of the difference between $\mathbf{w}$ and $\mathbf{m}$. However, since the concept of clustering is intuitively best understood in terms of positions and distances, we will still refer to $D(\mathbf{w}, \mathbf{m})$ as a distance.

corresponds to the notion that we have no prior knowledge about the structure of the clusters.

We introduce a regularization parameter T, weighting the two different terms, to arrive at the 'free energy'

$$F(\{m\},\{p\}) = E(\{m\},\{p\}) - TS(\{p\}).$$

Minimizing $F(\{m\},\{p\})$ can be seen as a search for the best compromise between a low average distance (minimizing $E(\{m\},\{p\})$) and keeping a reasonable amount of chaos in the system (maximizing $S(\{p\})$). For any choice of model centers $\mathbf{m}_\alpha$ the probabilities $p_{i\alpha}$ minimizing the free energy $F(\{m\},\{p\})$ read (under the constraint $\sum_\alpha p_{i\alpha} = 1\ \forall_i$)

$$p_{i\alpha}(m) = \frac{e^{-\beta D(\mathbf{w}_i,\mathbf{m}_\alpha)}}{\sum_{\alpha'} e^{-\beta D(\mathbf{w}_i,\mathbf{m}_{\alpha'})}} \tag{4}$$

with $\beta = 1/T$. Substitution of this result into the free energy then yields

$$F(\{m\}) = F(\{m\},\{p(\{m\})\}) = \sum_i \log \sum_\alpha e^{-\beta D(\mathbf{w}_i,\mathbf{m}_\alpha)}. \tag{5}$$

Eq. (5) is equivalent to the result presented in Rose et al. (1990). The main difference between our derivation and the derivation made by Rose et al. is the role of the parameter β. Here, as in Buhmann and Kühnel (1993), it is just a regularization parameter that can be chosen in advance. In Rose et al. (1990) an average energy $\langle E\rangle$ is defined, such that (like in statistical physics theory) β is a Lagrange multiplier that must be tuned to ensure that $\langle E\rangle$ stays constant.

3. Annealing and the EM algorithm

The annealing process finds cluster centers $\mathbf{m}_\alpha$ minimizing the free energy $F(\{m\})$ for increasing values of β. We start with β close to zero and a large number of cluster centers $\mathbf{m}_\alpha$.[2] Such a low value of β will strongly favor the entropy part of the free energy, resulting in a solution where all $\mathbf{m}_\alpha$ are identical (so $p_{i\alpha} = p_{i'\alpha'}\ \forall_{i,i',\alpha,\alpha'}$.) When the new $\mathbf{m}_\alpha$ have been found, we increase β and again minimize the free energy. These steps are repeated until the balance between average energy and entropy has shifted enough to warrant multiple clusters; at this point a *phase transition* occurs, and the cluster centers are divided over two separate solutions. More and more clusters will appear when β is increased further, until we have reached a satisfactory number of clusters and we terminate the process.

We find cluster centers $\mathbf{m}_\alpha$ (for each value of β) through minimization of the free energy $F(\{m\})$. This corresponds to solving a series of coupled equations:

$$\frac{\partial F(\{m\})}{\partial \mathbf{m}_\alpha} = \sum_i p_{i\alpha} \frac{\partial D_{i\alpha}}{\partial \mathbf{m}_\alpha} = 0\ \forall_\alpha, \tag{6}$$

where the equations for different $\mathbf{m}_\alpha$ are interdependent through the normalization of $p_{i\alpha}$, which is a function of all $\mathbf{m}_\alpha$. We solve this system of equations using an expectation-maximization (EM) algorithm, a full description of which can be found in (Rubin, 1991).

In the expectation step of the EM algorithm the probabilistic assignments $p_{i\alpha}$, as given by Eq. (4), are calculated. In the maximization step we find new cluster centers $\mathbf{m}'_\alpha$ such that:

$$\begin{aligned}\mathbf{m}'_\alpha &= \operatorname*{argmax}_{\mathbf{m}_\alpha} \sum_i p_{i\alpha} F(\{m\})\ \forall_\alpha \\ &= \operatorname*{argmax}_{\mathbf{m}_\alpha} \sum_i p_{i\alpha} D(\mathbf{w}_i,\mathbf{m}_\alpha)\ \forall_\alpha.\end{aligned} \tag{7}$$

A solution for $\mathbf{m}'_\alpha$ can be obtained from any gradient descent algorithm on $\sum_i p_{i\alpha} D(\mathbf{w}_i,\mathbf{m}_\alpha)$ starting from the current $\mathbf{m}_\alpha$.

Our aim is to summarize an ensemble of neural networks through a smaller set of networks with a similar architecture, and Eq. (7) is expressed in terms of the parameters of these networks. However, the above description also applies to 'model free' clustering, solely based on the outputs on examples $\mathbf{x}^\mu$. The maximization step for model free clustering (where we now maximize with respect to $\boldsymbol{y}_\alpha = [y(\mathbf{m}_\alpha,\mathbf{x}^1), y(\mathbf{m}_\alpha,\mathbf{x}^2),\ldots, y(\mathbf{m}_\alpha,\mathbf{x}^N)]$) can be very simple: for suitable distance functions (e.g. the sum-squared error or the cross-entropy error[3]) it is simply

$$\mathbf{y}'_\alpha = \sum_i p_{i\alpha}\mathbf{y}_i. \tag{8}$$

Model based clustering can still benefit from this simplified maximization step. We simply ignore the model parameters $\mathbf{w}_i$ during (parts of) the clustering process, and use the corresponding model outputs $\mathbf{y}_i$ for model free clustering.

Obviously, the 'cluster outputs' need to be translated back to model parameters in the end (through optimization on the cluster center outputs $\mathbf{y}_\alpha$ and inputs $\mathbf{x}^\mu$), but we may still gain a tremendous speed-up of the annealing process. In Section 4 we discuss several alternative procedures to combine model free clustering and translation back to model parameters.

4. Computational issues

Deterministic annealing. The annealing process starts at $\beta = 10$ (for the data examined in the present article). We first allow two clusters, which are initiated by adding random noise to the average of the model outputs over a full ensemble of models. After the EM algorithm described in Section 3 has converged, we either accept the new cluster (the single cluster centered around the average model output is split into two clusters), or we 'merge' the two

[2] The number of cluster centers is infinite in theory; in practice we implement a sufficiently large number of clusters at each point in the process. The exact choice for this number will be elaborated in Section 4.

[3] $$\begin{aligned}&d(y(\mathbf{w}_i,\mathbf{x}^\mu), y(\mathbf{m}_\alpha,\mathbf{x}^\mu)) \\ &= y(\mathbf{w}_i,\mathbf{x}^\mu)\log\frac{y(\mathbf{w}_i,\mathbf{x}^\mu)}{y(\mathbf{m}_\alpha,\mathbf{x}^\mu)} + [1-y(\mathbf{w}_i,\mathbf{x}^\mu)]\log\frac{1-y(\mathbf{w}_i,\mathbf{x}^\mu)}{1-y(\mathbf{m}_\alpha,\mathbf{x}^\mu)}.\end{aligned}$$

clusters back into one. This decision is based on a comparison of the distance between the two cluster centers to the distance between the models in the corresponding clusters. If the former is relatively small, the two cluster centers merge, otherwise a new cluster is accepted. For the initial, low value of β, the two clusters always merge. The annealing parameter β is increased by 1 after each merging or acceptation of a new cluster. After the first phase transition (when the algorithm has accepted two separate clusters), we split each of the corresponding representative model outputs in two by adding random noise. Thus we allow the next transition into three or four clusters to occur. Generally, only one new cluster is gained in each transition. When the algorithm does 'skip' intermediate numbers of clusters (e.g. goes from three clusters to five clusters after one increment of β), the increase in β (Δ_β) is temporarily lowered until all intermediate cluster numbers are found, or Δ_β drops below 0.01. This steady growth from one to a maximum number of clusters can be compared to 'greedy mixture learning' (Vlassis & Likas, 2002).

When to retrain. A downside of model free clustering of elements that are in fact models is that cluster centers are found directly in the space of model outputs. Such cluster centers generally do not have a generating model of the same architecture as the models in the ensemble. We therefore need a retraining step to translate the cluster centers back to actual models. To retrain a 'cluster model', we split the model's outputs, which were found in the clustering process, and the corresponding inputs into a training set and a validation set. Retraining then proceeds in exactly the same way that the original networks were trained: we minimize the mean squared error on the training set, and stop training when the error on the validation set starts to increase. We can make several choices for the point in the process where we make this translation:

1. *Retrain after clustering*. The simplest option is to perform the entire clustering process solely in terms of model outputs, and train back the actual representative models at the end, when all cluster centers are found. In doing this however, we disrupt the clustering we found: the models obtained through retraining may not be the optimal representatives, even though the cluster centers in terms of outputs were. Therefore, after retraining we could implement a short reclustering sequence, where now in each EM-step we retrain the networks to fit the cluster center outputs. Model free clustering then serves as an 'initialization' for model based clustering.
2. *Cluster in terms of parameters*. The other extreme is to perform clustering in model space entirely, varying only the parameters of the representative models. Although this makes each step considerably more time consuming, it does prevent the cluster centers from wandering into areas of output space that cannot be reached by the actual models. Cluster centers obtained from the previous approach that do stray into this area may be very difficult to approximate by the applied models. In this case the final approximations may not only take a lot of extra time, but (e.g. due to local minima near the initialization for retraining the model) may also yield qualitatively inferior results.
3. *Retrain during clustering*. A reasonable compromise might be to retrain the model after every N EM-steps, where N is large enough not to unacceptably slow down the process, but small enough to keep the cluster centers in the right area. N can be kept constant throughout the clustering process, or vary, when for example we retrain for each new value of β, or each time a new cluster is accepted. In Section 6.3 we compare several approaches in terms of computation time and accuracy.

5. Bootstrapping and multitask learning

In bootstrapping, instead of training one model on the complete (training) data, we resample the data set multiple times, and train one model for each resampling. Resampling is done by drawing N samples from a training set of N instances, where each instance may be drawn more than once. In this way, the ensemble of models reflects the variability of the data under review. Breiman (1996) describes how bootstrap ensembles can be used for prediction through a procedure called *bagging* (bootstrap aggregating). If the data contains only global similarities, the models in the ensemble will be centered around one average model, and bootstrapping only serves to obtain a more unbiased version of this model. If, however, the data contains multiple local similarities, bootstrapping provides a way to include corresponding local summaries in the final predictions, which is impossible for any one model. In both cases, instead of using the entire ensemble, we may need only the local summaries, and a way to weigh them. This is provided by the clustering algorithm described in the previous sections. The cluster based prediction on a new input $\mathbf{x}^\nu$ reads:

$$\tilde{y}^\nu = \sum_\alpha p_\alpha y(\mathbf{m}_\alpha, \mathbf{x}^\nu), \qquad \text{with } p_\alpha = \frac{1}{N}\sum_{i=1}^{N} p_{i\alpha}, \tag{9}$$

a weighted sum over the representative model outputs, where the weights are determined by the 'effective' number of models in each cluster.

Multitask learning, or 'learning to learn', makes use of the fact that a series of similar tasks may share a common underlying structure. Instead of learning each of these tasks separately, the tasks are forced to share their knowledge, and learn from each other. One way to implement this is a 'hard sharing' of part of the model parameters (Caruana, 1997). In this article we allow knowledge sharing through clustering (see (Cadez et al., 2000) for a similar approach). For each of the tasks we train one or more models, which will generally be strongly biased to the part of the training set

corresponding to that task. Task clustering can then be applied to represent strongly similar tasks by a common cluster center. A predicted new output for task t, $\tilde{y}_t^\nu$, changes slightly to include knowledge of the partition of the data into tasks:

$$\tilde{y}_t^\nu = \sum_\alpha p_{t\alpha} y(\mathbf{m}_\alpha, \mathbf{x}_t^\nu), \qquad \text{with } p_{t\alpha} = \frac{1}{n_t} \sum_{i \in t} p_{i\alpha}, \tag{10}$$

where n_t is the number of models trained for task t, and where the weights $p_{t\alpha}$ for task t depend only on the corresponding models trained on this task.

Note that we apply the same model clustering procedure to two very different topics. The models obtained in bootstrapping are interchangeable, and cannot a priori be assigned to different clusters. The models for multitask learning are each trained on a specific task, and can therefore be 'labeled'. This difference is apparent in Eqs. (9) and (10), where in the former all models contribute, whereas in the latter a model's contribution depends on its label.

6. Results

6.1. Description of the data

Boston housing. The Boston housing problem concerns the prediction of housing values in the suburbs of Boston, based on 13 inputs including e.g. per capita crime rate and nitric oxides concentration. The database contains 506 examples. For more information, see (Harrison & Rubinfeld, 1978). We preprocessed the data by scaling each variable (both input and output) to have zero mean and unity variance.

Prediction of newspaper sales. We also apply our method on a database of single-copy sales figures for one of the major Dutch newspapers. The database contains the numbers of newspapers sold on 156 consecutive Saturdays, at 343 outlets in The Netherlands. Inputs include recent sales (-4 to -6 weeks), last year's sales (-51 to -53 weeks), weather information (temperature, wind, sunshine, precipitation quantity and duration) and season (cosine and sine of scaled week number). The responses are the realized sales figures. All covariates and responses are scaled per outlet to have zero mean and unit standard deviation.

6.2. Clustering and prediction

Methods. For both databases we trained ensembles of neural networks. These networks had one hidden layer and biases on the hidden and output units. The hidden units had hyperbolic tangents as transfer functions; the output units were linear. For the Boston housing data we trained ensembles of models with numbers of hidden units varying from 1 to 14 (one type of model in each ensemble). For the newspaper data we only trained networks with two hidden units, since we know from past experience that these yield the best results for this particular data set.

The clustering algorithm was applied to 10 independent ensembles of 50 models, obtained by bootstrapping. To create an ensemble of models we made a random 2:1 split of the database, where the larger part was used for training (training set) and the smaller for testing the final result (test set). Each of the 50 models was optimized on a random resampling of the training set: for a set of n_{train} samples we took n_{train} random draws where each data item (an input with its corresponding output) could be drawn more than once. This resampled set was used for the actual optimization (for which we applied a conjugate gradient method), and the undrawn part of the training set was used to implement early stopping. Larger ensembles did not improve performance, smaller ensembles yielded inferior results.

The clustering algorithm was performed for each ensemble. We predicted the outputs in the smaller part of the database (the test set) through Eqs. (9) and (10). The predictions on the Boston housing data were compared to prediction through bagging (taking the average over the outputs of all models in the ensemble on a new input). For the newspaper data, for each split of the data we also trained one network similar to those described previously, with two hidden units, but with one output for each task (outlet). This means that all tasks shared the same input-to-hidden weights of the network, but had independent hidden-to-output weights. See Caruana (1997) for similar work. All prediction results are rated through their sum-squared error on the test set.

Boston housing. We modeled the Boston housing problem by bootstrapping with multilayered perceptrons with their numbers of hidden units varying from 1 to 14. The models within one bootstrap ensemble always had the same numbers of hidden units. Each ensemble was clustered up to the point where no more improvement in sum-squared error was gained by adding more cluster centers. Fig. 1 shows the sum-squared error for the clustered ensemble as a function of the number of cluster centers, for the full ensemble (through bagging) and the average sum-squared error of single models in the ensemble, for models with 2, 8, 12 and 14 hidden units. It can be seen that for more complicated models (more hidden units) less cluster centers are needed to match the performance of the full ensemble. This progression from many cluster centers to one can be understood in terms of bias and variance.

In general, we can say that errors due to (large) variance occur when a model can in fact adequately represent the (hypothetical) data generating function, but still optimizes to the wrong model because of the limited amount of training data. Errors due to bias, on the other hand, occur when the model is not sufficiently sophisticated to match the data generating function, and must settle for the closest approximation. For one simple model it is impossible to give an adequate representation of the hidden (complex)

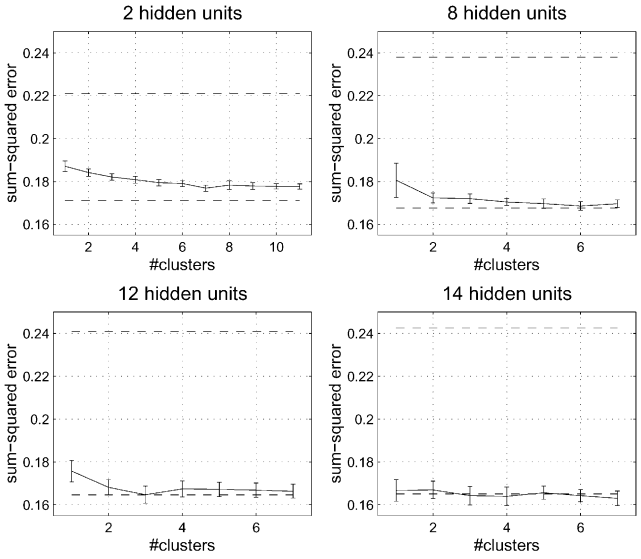

Fig. 1. Boston housing data: prediction error as a function of the number of clusters for models with two (upper left panel), eight (upper right panel), 12 (lower left panel) and 14 hidden units (lower right panel). The lower horizontal lines in each plot indicate the prediction error corresponding to the full bootstrap ensemble. The upper horizontal lines indicate the average error made by the single models in the ensemble. The error bars are calculated based on the difference between the error of the reduced ensembles (cluster centers) and the full ensemble.

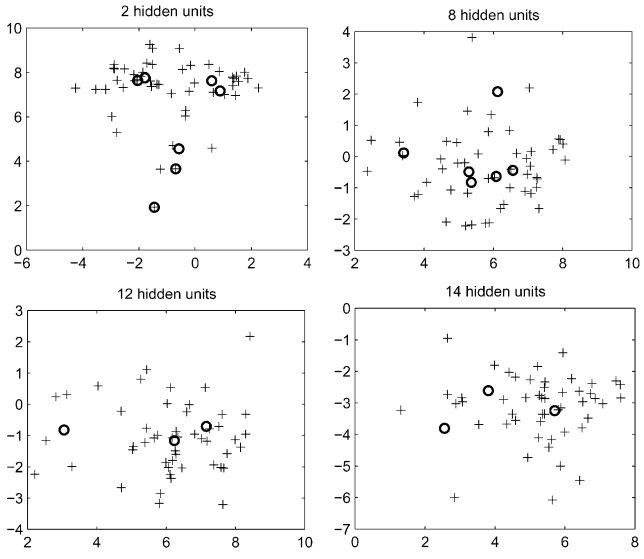

Fig. 2. Clustering of the Boston housing data. Each panel plots the principal components of the outputs of both bootstrap models (crosses) and representative models (circles). For the two hidden units ensemble we show the seven cluster solution, six clusters for the eight hidden units ensemble and three for the remaining ensembles. Note that in the two hidden unit ensemble representative models and bootstrap models start to coincide.

function underlying the data. Bootstrapping in this case serves to find an ensemble of models that, when put together, can closely approximate this function. Since in this case the bootstrapping ensemble contains multiple significantly different models, multiple cluster centers are required to represent the ensemble. Bootstrapping then serves to reduce the bias in the model, since the summation over multiple networks in fact yields a more complex model.

One sufficiently complex model however, may be sufficient to represent the unknown function that generated the data. In this case, bootstrapping serves to reduce the overfitting that may occur when one such model is trained on the full data. Although for complicated models many bootstrap samples may be needed to obtain a low variance estimation, in the end the full ensemble can be represented by just one cluster center. More discussion on the effectiveness of bootstrapping and ensemble learning can be found in Domingos (1997) and Breiman (1998).

One MLP with 14 hidden units appears to be sufficiently complex to describe the Boston housing problem: the one cluster representation performs as well as the full ensemble. Clustering of the two hidden units ensemble appears not to be able to match the performance of the full ensemble. We expect this failure to be due to the fact that in this case, where a large number of representative models is needed, the annealing process needs to reach relatively high values of β. In this regime the algorithm gets too 'greedy', and rather than detecting new, relevant clusters, it creates cluster centers that coincide with models in the ensemble, as can be seen in the lower part of the upper left panel in Fig. 2.

Prediction of newspaper sales. Prediction error for the newspaper data (Fig. 3) decreases strongly until the 3-cluster solution, and then slowly converges to a minimum, which is significantly lower than the error made by the multitask learning network (one output for each outlet, and shared input-to-hidden weights). Averaging over all models in the full ensemble (as in bootstrapping) yields rather poor results in this case. The clustering obtained on the newspaper data does not only improve prediction, but also reveals an interesting structure, which was hidden in the data. Fig. 4 shows the outlets in Holland that are assigned to

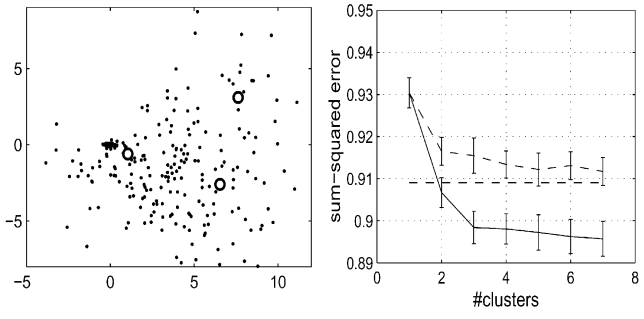

Fig. 3. Clustering of the newspaper data. The left panel plots the principal components of the outputs of both bootstrap (dots) and representative models (circles). The right panel shows the prediction error as a function of the number of clusters. The horizontal line represents the prediction error that is made by the multitask learning network (one output for each outlet), the other dashed line shows the K-means clustering results, the solid line shows the results from the clustering method described in the present article. The error bars are calculated based on the difference between the prediction error corresponding to averaging over the cluster centers and those made by the larger network.

Fig. 4. Geographical clustering of the newspaper outlets. Circles mark outlets assigned with weight larger than 0.9 to either the 'seasonal' cluster (left panel), the 'short term' cluster (middle panel) or the 'undetermined' cluster (right panel).

clusters 1, 2 and 3 with probability $p_{i\alpha} > 0.9$. It can be seen that the outlets in the first cluster tend to be near the beach, and in the eastern part of Holland, touristic spots without many large cities. The outlets in cluster 2 center around the 'Randstad' (Amsterdam and other relatively large Dutch cities). The outlets in cluster 3 are spread all around, and can be considered 'undetermined'. Fig. 5 (a Hinton diagram (Bishop, 1995)) plots the 'sensitivity' (the derivative of the model output with respect to a model input, averaged over a set of training samples) of the representative models for each cluster. The figure clearly shows that the first of these models (corresponding to the 'touristic' cluster) is especially sensitive to the inputs corresponding to last year's sales and season, whereas for the second model ('city cluster') short term sales are weighed more heavily. The third, 'undetermined' model features much less pronounced sensitivities. This clustering, which makes intuitive sense, was obtained without any information with respect to city size or level of tourism.

Other methods. Model clustering can in many cases be implemented through less involved clustering algorithms, such as K-means or nearest-neighbor clustering. We repeated the model clustering process and predictions described in the previous paragraphs, where this time we applied K-means clustering to the model outputs, instead of deterministic annealing. The performance of the simpler K-means clustering method on the Boston housing database was not significantly worse than that of the more involved method. Note however, that in this case only 50 data vectors were clustered; more complex databases may still benefit from the more involved algorithm described in Section 3. See Miller and Rose (1996) for examples.

The newspaper example, however, does benefit from the more involved algorithm. The prediction error, shown in Fig. 3, is significantly higher after K-means clustering. This effect is partly due to the fact that deterministic annealing yields a 'soft' clustering, where in the case of multitask learning for each task we get a distribution over the full set of clusters, instead of a hard assignment to one cluster; when for the deterministic annealing method we assign each task to its 'most likely' cluster (i.e. task t is assigned to the cluster α corresponding to the highest $p_{t\alpha}$), its performance becomes significantly worse (but is still better than the performance of K-means). The other part of the improvement most probably is due to the larger number of samples, which makes a more complex clustering algorithm more effective.

We also looked at a selection method, as an alternative to clustering. This alternative would use a (small) selection of models from the existing ensemble instead of cluster centers. We made a random 2:1 split of the original training data into a training set and a validation set, and trained an ensemble on the training set. We selected models from this ensemble through a method much like the 'Tabu method' (Roli, Giacinto, & Vernazzo, 2001). Here, we start out with a random selection of n models from the ensemble (n being the number of models we eventually want to have in our selection; in our case $1 \leq n \leq 7$). In each step we consider all possible subsets created through removing one model from the selection and adding another (out of the ensemble) to it, and accept the subset with the lowest sum-squared error on the validation set. This process is continued for a fixed number of steps, and continues even when the validation error rises from one set to the other. Eventually, we accept that selection that over the course of the algorithm showed the lowest validation error. Although this method may often be less time consuming than the clustering method, the results in terms of prediction

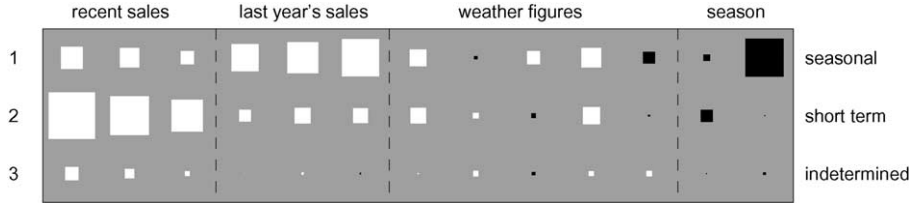

Fig. 5. Sensitivity of the representative models to the model inputs. White squares represent positive dependencies, black squares negative dependencies. Larger squares represent stronger effects.

Table 1
Average sum-squared error on the test set and required computation time for different moments of retraining. The models in the clustered ensembles concerned the newspaper data. Clustering continued until three clusters were found

Method	Sum-squared error	Required time (min)
Retrain afterwards	0.897 ± 0.011	9.0 ± 1.6
Retrain per cluster	0.897 ± 0.005	11.7 ± 2.2
Retrain per β	0.897 ± 0.005	28 ± 4
Cluster on parameters	0.897 ± 0.005	116 ± 19

error were significantly worse than those of the method described in the present article in each instance except for the Boston housing example with two hidden units, where they were equivalent.

6.3. The influence of frequent retraining

The results from Section 6.2 were obtained by retraining the models at the end of the clustering procedure. We mentioned in Section 4 that the moment and frequency of retraining the representative models may have its influence on the final solutions. Therefore, we repeated the above simulations on newspaper sales for clustering with more frequent retraining, where we tried retraining either for each new β or after addition of each new cluster. Table 1 shows the simulation time and prediction error for each of these approaches. The values are averages over 10 independent runs, as in Section 6.2. Simulations on the Boston housing data yielded similar results. It can be seen that, at least for the databases used in the present article, retraining at the end of the clustering procedure has no adverse effect on the quality of the method's predictions, and takes significantly less computation time. Still, other databases might produce other results.

As a benchmark we also applied the original EM algorithm outlined in Section 3, i.e. where the maximization step itself involves fitting the model parameters of the cluster centers. Even here, where we do not loose any accuracy due to retraining, the results are still not significantly better, whereas the computation time is considerably longer than for any of the retraining methods.

7. Discussion

In the present article (which elaborates on our earlier work in Bakker and Heskes (1999)) we have presented a method to summarize large ensembles of models to a small number of representative models. We have shown that predictions based on a weighted average of these representatives can be as good as, and sometimes even better than, predictions based on the full ensemble. We believe that this method provides an extremely useful addition to any method featuring an ensemble of models, such as bootstrapping, sampling of Bayesian posterior distributions or multitask learning. The method is not only valuable in terms of predictive quality, but also on a more abstract level. This improvement was apparent on the newspaper data, where different clusters of models brought out different aspects of the data.

A considerable body of literature exists on the subject of the 'overproduce and choose' paradigm, where in the first step a (too) large ensemble of models is trained, and in the second a selection or combination of these models is made to optimize performance (Perrone & Cooper, 1993; Partridge & Yates, 1996; Hashem, 1996; Sharkey, Sharkey, Gerecke, & Chandroth, 2000; Roli et al., 2001). Roli et al. (2001) have implemented a clustering algorithm for classifiers, where distance between two classifiers depends on the probability that both misclassify the same pattern. This concept of 'methodological diversity' plays an important role in most methods in this field; in the present article we have implemented this concept in the entropy term (Eq. (3)), which makes sure cluster centers are optimally diverse. Note that our method is in fact unsupervised, which makes the clustering part less dependent on the availability of supervised data; if model inputs can be generated artificially, the data set used for clustering can be made arbitrarily large.

In the present article we have chosen the clustering procedure outlined by Rose et al. (1990) to perform model clustering. Alternative clustering procedures can of course be considered and, as shown in Section 6.2, may sometimes yield equivalent results. We do, however, stress the importance of using a 'natural' distance function based on model outputs rather than a more arbitrary distance based on model parameters.

At this point we wish to underline that in the present article we do not claim to outperform the above mentioned authors in terms of prediction error. The present article is meant to show that model clustering, through any desired clustering algorithm, is able to produce a fitting 'summary' of any large ensemble of models. In the present article we have taken bootstrapping ensembles for an example, but ensembles obtained from sampling a Bayesian posterior distribution, or indeed from any source, can in theory be summarized just as well. The strength of such summaries lies mainly in the fact that they make qualitative analysis easier than it would be on the full ensemble. We demonstrated this through two examples: in the Boston housing example we used the summaries to analyze the bootstrapping process in terms of bias/variance, in the newspaper example we discovered a division of outlets into 'seasonal' outlets and 'short term' outlets. We feel therefore, that the model clustering method described in this article can make a valuable contribution in the field of qualitative data analysis (data mining).

Cadez et al. (2000) have done related work on model clustering. An important difference between our work and (Cadez et al., 2000) is that we cluster trained models, slowly increasing the number of clusters, where in Cadez et al.

(2000) clustering and training are combined into a generative model with a fixed number of clusters.

We addressed the problem of retraining a model with a desired architecture from a cluster center expressed in outputs on a data set. Although we recognize that a risk exists in letting the cluster centers run free in output space, we showed that at least for the databases considered in the present model retraining at the end of the clustering procedure does not lead to the wrong representative models. For cases where this method may not be correct, we have suggested saver, yet more time consuming methods where the models are retrained at clearly marked points in the process.

In the present article we have applied model clustering to gain better predictions and, for the newspaper project, to detect hidden structure in the data. Other applications can be found e.g. in the comparison of networks which do not have the same structure, but are trained to perform the same task. If the network outputs depend strongly on the network's structure, different models are likely to be assigned to different cluster centers. If, however, two differently structured networks produce similar outputs, there will be clusters inhabited by both types of networks.

Our model clustering method may also have useful applications for (Bayesian) sampling: in this case, an ensemble of models is obtained through sampling from a probability density function which cannot (easily) be expressed analytically. Model clustering can be used to find clusters in this ensemble of samples, and make subsequent analysis easier.

Another application would be the detection of symmetries in a network (Rüger & Ossen, 1996) through study of the differences between clustering based on a distance function dependent on the outputs of the networks and clustering in weight space directly (e.g. with a Euclidean distance function).

Acknowledgements

This research was supported by the Technology Foundation STW, applied science division of NWO and the technology program of the Ministry of Economic Affairs.

References

Bakker, B., & Heskes, T. (1999). Model clustering by deterministic annealing. In M. Verleysen (Ed.), (pp. 87–92). *Proceedings of the European Symposium on Artificial Neural Networks'99*.

Bishop, C. (1995). *Neural networks for pattern recognition*. Oxford: Clarendon Press.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

Breiman, L. (1998). Arcing classiers. *The Annals of Statistics*, *26*.

Buhmann, J., & Kühnel, H. (1993). Vector quantization with complexity costs. *IEEE Transactions on Information Theory*, *39*(4), 1133–1145.

Cadez, I., Gaffney, S., & Smyth, P (2000). A general probabilistic framework for clustering individuals and objects. *Proceedings of the ACM SIGKDD Conference* (pp. 140–149).

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41–75.

Domingos, P (1997). Why does bagging work? A Bayesian account and its implications. *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (pp. 155–158).

Efron, B., & Tibshirani, R. (1993). *An introduction to the bootstrap*. London: Chapman and Hall.

Harrison, D., & Rubinfeld, D. (1978). Hedonic prices and the demand for clean air. *Journal of Environmental Economics and Management*, *5*, 81–102.

Hashem, S. (1996). Optimal linear combinations of neural networks. *Neural Networks*, *10*(4), 599–614.

Heskes, T. (1998). *Solving a huge number of similar tasks: a combination of multi-task learning and a hierarchical Bayesian approach. Proceedings of the International Conference on Machine Learning*, San Mateo: Morgan Kaufmann.

Jaynes, E. (1957). Information theory and statistical mechanics. *Physical Review*, *106*, 620–630.

Miller, D., & Rose, K. (1996). Hierarchical, unsupervised learning with growing via phase transitions. *Neural Computation*, *8*, 425–450.

Neal, R. (1996). *Bayesian learning for neural networks*. New York: Springer.

Partridge, D., & Yates, W. (1996). Engineering multiversion neural-net systems. *Neural Computation*, *8*(4), 869–893.

Perrone, M., & Cooper, L. (1993). When networks disagree: ensemble methods for hybrid neural networks. In R. Mammone (Ed.), *Neural networks for speech and image processing* (pp. 126–142). London: Chapman and Hall, Chapter 10.

Roli, F., Giacinto, G., & Vernazza, G. (2001). Methods for designing multiple classifier systems. *Multiple Classifier Systems*, 78–87.

Rose, K., Gurewitz, E., & Fox, G. (1990). Statistical mechanics of phase transitions in clustering. *Physical Review Letters*, *65*, 945–948.

Rubin, D. B. (1991). EM and beyond. *Psychometrika*, *56*(2), 241–254.

Rüger, S., & Ossen, A. (1996). Clustering in weight space of feedforward nets. In C. Von der Malsburg (Ed.), *ICANN'96* (pp. 83–88). Berlin: Springer.

Sharkey, A., Sharkey, N., Gerecke, U., & Chandroth, G. (2000). The test and select approach to ensemble combination. *Multiple Classifier Systems*, 30–44.

Vlassis, N., & Likas, A. (2002). A greedy EM algorithm for Gaussian mixture learning. *Neural Processing Letters*, *15* in press.