In *Statistics and Neural Networks: Advances at the Interface*, J. W. Kay and D. M. Titterington Eds. Oxford University Press, 1999.

Robust prediction in many-parameter models: Specific control of variance and bias

Nathan Intrator
School of Mathematical Sciences
Tel-Aviv University
http://www.math.tau.ac.il/~nin

March, 1998

Contents

1 Introduction

When parameter estimation is based on a set of training patterns with size that is of the order of the number of free parameters, estimation may become very unreliable. Until recently, estimation in such cases had sounded unrealistic, but it has now been accepted that such estimation is possible under certain conditions. For example, linear discriminant analysis (Fisher, 1936) requires some adjustment when the input dimensionality is large (Buckheit and Donoho, 1995) to account for the added variability of the covariance matrices. In simple terms, innovative ways to reduce the variance portion of the error are requried, as well as methods to impose (reasonable) bias.

One fundamental assumption concerns the 'true' dimensionality of the data. Although the data may be represented in a high-dimensional space, it is often the case that the actual dimensionality is much smaller. Clearly, a dimensionality reduction method which does not 'throw the baby out with the bath water', but retains important information in the data, leads to smaller number of parameters for the required prediction and thus may be more robust.

This chapter reviews several ways to robustify estimation and prediction in many-parameter models. We start with a short review of the bias/variance dilemma and then address separately ways to control the variance portion of the error and the bias. A central issue in variance control methods is that of ensemble averaging. We shall expand on this to present methods for extracting additional information from ensembles of experts by exploiting the variability in responses for estimating the confidence of a certain group of experts. The control or introduction of bias into the model leads naturally to dimensionality reduction. We make this connection and discuss methods for hybrid dimension reduction and classification, and we spend some time on dimension reduction methods. Throughout, we demonstrate the methods with real-world applications.

2 The Variance-Bias Dilemma

The motivation of our approach follows from a key observation regarding the bias/variance decomposition, namely the fact that ensemble averaging does not affect the bias portion of the error, but reduces the variance, when the estimators on which averaging is done are independent.

The classification problem is to estimate a function $f_{\mathcal{D}}(x)$ of observed data characteristics x, for predicting a class label y, based on a given training set $\mathcal{D} = \{(x_1, y_1), \ldots, (x_L, y_L)\}$, using some measure of the estimation error on $\mathcal{D}$. A good estimator will perform well not only on the training set, but also on new *validation* sets which were not used during estimation.

Evaluation of the performance of the estimator is commonly done via the mean squared error distance (MSE) by taking the expectation with respect to the (unknown) probability distribution P of y:

$$E[(y - f_{\mathcal{D}}(x))^2 | x, \mathcal{D}].$$

This can be decomposed into

$$E[(y - f_{\mathcal{D}}(x))^2 | x, \mathcal{D}] = E[(y - E[y|x])^2 | x, \mathcal{D}] + E[(f_{\mathcal{D}}(x) - E[y|x])^2].$$

The first term typically depends on neither the training data $\mathcal{D}$ nor the estimator $f_{\mathcal{D}}(x)$, it measures the amount of noise or variability of y given x. Hence f can be evaluated using

$$E[(f_{\mathcal{D}}(x) - E[y|x])^2].$$

The empirical mean squared error of f is given by

$$E_{\mathcal{D}}[(f_{\mathcal{D}}(x) - E[y|x])^2],$$

where $E_{\mathcal{D}}$ represents expectation with respect to all possible training sets $\mathcal{D}$ of fixed size.

To investigate further the MSE performance we decompose the error into bias and variance components (Geman et al., 1992) to obtain

$$E_{\mathcal{D}}[(f_{\mathcal{D}}(x) - E[y|x])^2] = (E_{\mathcal{D}}[f_{\mathcal{D}}(x)] - E[y|x])^2 + E_{\mathcal{D}}[(f_{\mathcal{D}}(x) - E_{\mathcal{D}}[f_{\mathcal{D}}(x)])^2]. \tag{1}$$

The first term on the right-hand side is called the bias term (strictly, the squared bias) of the estimator and the second term is called the variance term. When training on a fixed training set $\mathcal{D}$, reducing the bias with respect to this set may increase the variance of the estimator and contribute to poor generalisation performance. This is known as the trade-off between variance and bias. Typically, variance is reduced by smoothing, but this may introduce bias since, for example, it may blur sharp peaks. Bias is reduced by incorporating prior knowledge. When prior knowledge is used also for smoothing, it is likely to reduce the overall MSE of the estimator.

When training neural networks such as multilayer perceptrons, the variance arises from two terms. The first term comes from inherent data randomness and the second term arises from the non-identifiability of the model, in that, for a given training dataset, there may be several local minima of the error surface.

Consider the ensemble average $\bar{f}$ of Q predictors, which in our case can be thought of as neural networks with different random initial weights which are trained on data with added Gaussian noise:

$$\bar{f}(x) = \frac{1}{Q}\sum_{i=1}^{Q} f_i(x).$$

These predictors are identically distributed, and thus the variance contribution to equation (1) becomes

$$\begin{aligned} E[(\bar{f} - E[\bar{f}])^2] &= E[(\frac{1}{Q}\sum f_i - E[\frac{1}{Q}\sum f_i])^2] \\ &= E[(\frac{1}{Q}\sum f_i)^2] + \left(E[\frac{1}{Q}\sum f_i]\right)^2 - 2E[\frac{1}{Q}\sum f_i E[\frac{1}{Q}\sum f_i]] \\ &= E[(\frac{1}{Q}\sum f_i)^2] - \left(E[\frac{1}{Q}\sum f_i]\right)^2; \end{aligned} \tag{2}$$

we omit mention of x and $\mathcal{D}$ for clarity. The first term in (2) can be rewritten as

$$E[(\frac{1}{Q}\sum f_i)^2] = \frac{1}{Q^2}\sum E[f_i^2] + \frac{2}{Q^2}\sum_{i<j} E[f_i f_j],$$

and the second term gives

$$\left(E[\frac{1}{Q}\sum f_i]\right)^2 = \frac{1}{Q^2}\sum \left(E[f_i^2]\right)^2 + \frac{2}{Q^2}\sum_{i<j} E[f_i]E[f_j].$$

Plugging these equalities into (2) gives

$$E[(\bar{f} - E[\bar{f}])^2] = \frac{1}{Q^2}\sum\{E[f_i^2] - \left(E[f_i]\right)^2\} + \frac{2}{Q^2}\sum_{i<j}\{E[f_i f_j] - E[f_i]E[f_j]\}. \tag{3}$$

Set

$$\gamma = \mathrm{Var}(f_i) + (Q-1)\mathrm{max}_{i,j}(E[f_i f_j] - E[f_i]E[f_j]).$$

It follows that[1]

$$\frac{1}{Q}\mathrm{Var}(f_i) \le \mathrm{Var}(\bar{f}) \le \frac{1}{Q}\gamma \le \max_i \mathrm{Var}(f_i). \tag{4}$$

This analysis suggests a simple extrapolation to large values of Q by giving an upper bound of $1/Q\gamma$ to the variance behaviour under large network-ensembles from small-size ensembles (Naftaly et al., 1997). Note that

$$E[f_i f_j] - E[f_i]E[f_j] = E\Big(\{f_i - E[f_i]\}\{f_j - E[f_j]\}\Big).$$

Thus, the notion of independence can be understood as independence of the deviations of each predictor from the expected value of the predictor, which can be replaced, because of linearity, by

$$E\Big(\{f_i - E[\bar{f}]\}\{f_j - E[\bar{f}]\}\Big),$$

and is thus interpreted as an indication of the prediction variation around a common mean.

3 Variance Control via Various Ensemble Averaging Methods

The success of ensemble averaging of neural networks in the past (Hansen and Salamon, 1990; Wolpert, 1992; Breiman, 1996; Perrone, 1993) is due to the fact that that error surfaces used to estimate the weights (parameters) of neural networks have in general many local minima, and thus, even with the same training set, different local minima are found when starting from different random initial estimates. These different local minima lead to somewhat independent predictors, and thus the averaging can reduce the variance. When a larger set of independent networks is needed but no more data are available, data reuse methods can be of help. Bootstrapping (Breiman, 1996) has been very helpful, since, by resampling from the training data without replacement, the degree of independence among the training sets, and hence among the resulting sets of estimators, is increased, leading to improved ensemble results. Smoothed bootstrap (Efron and Tibshirani, 1993) is potentially more useful since a wider range of sets of independent training samples can be generated. The smoothed bootstrap approach amounts to generating larger datasets by simulating the *true* noise in the data. The next section demonstrates what can be done with networks trained on the same data, and the following section demonstrates ways to increase the independence between networks and associated considerations.

3.1 Exhaustive training

The analysis presented in Section 2 suggests that training an ensemble of predictors should be done in a different way from training single predictors. This is because an ensemble of predictors has a lower overall bias, and thus the optimal trade-off between variance and bias should correspond to a lower level of bias so as to balance the contribution to error from the lower variance. In neural networks, trained by methods such as iterative gradient descent, this is achieved by stopping the training process at a later point, thus overfitting the individual training dataset somewhat, thereby leading to lower bias but higher variance. Naftaly et al. (1997) have recently demostrated the effect of training on single and ensemble errors, using the well known sun-spots data (Murphy and Aha, 1992). After giving some technical details, we briefly review their results. For full details

[1]We use the fact that $ab \le \frac{a^2+b^2}{2}$, thus, $E[f_i f_j] - E[f_i]E[f_j] = E\Big(\{f_i - E[f_i]\}\{f_j - E[f_j]\}\Big) \le \max_i \mathrm{Var}(f_i)$.

and comparison with other approaches see (Naftaly et al., 1997). As in (Weigend et al., 1990), feed-forward, simple-recurrent (Elman and Zipser, 1988) networks with 4 sigmoidal hidden units were used with data from 12 consecutive time points as inputs. In other words, the neural network was used as a nonlinear predictor of the number of sunspots in a given month, using the data from the 12 previous months as covariates/predictors. The prediction error was measured according to the average relative variance (ARV). Training was done via the error back-propagation algorithm

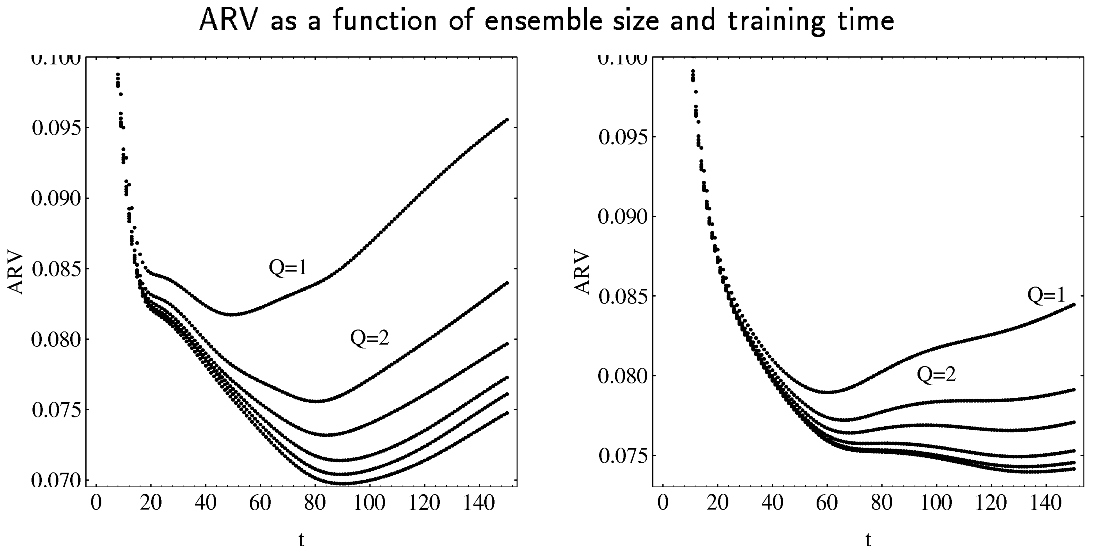

Figure 1: ARV vs. training time in kilo epochs (KE) for the sun-spots data (From (Naftaly et al., 1997)). The curves are shown for different choices of ensemble sizes: $Q = 1, 2, 4, 10, 20$ from top to bottom. The lowest curve is the extrapolation to $Q \to \infty$ (See (Naftaly et al., 1997).) *Left:* Results for a cross-validation set of 35 points. *Right:* Results on a test set. Note the difference in minimum error between a single predictor and an ensemble and the shallower curve as Q, the ensemble size, increases.

(Rumelhart et al., 1986) with a fixed learning rate of 0.003. A validation set containing 35 randomly chosen points was left out during training to serve for performance validation. The only difference between the members of the ensemble of networks was that different sets of weights were used to intialise the training algorithm.

3.2 Estimating the variance reduction for large ensembles

Naftaly et al. (1997) presented a very simple way of estimating what the variance portion of the MSE will be for large ensemble sizes, based on the performance of small ensemble sizes: see Section 2. Figure 1 depicts results from (Naftaly et al., 1997) based on the validation set (Left) and on the test set (Right). Values of ARV are shown as a function of the number of training epochs. The highest curve in each figure corresponds to $Q = 1$, i.e. the case of single networks. Below it appear the curves of $Q = 2, 4, 10, 20$, followed by the extrapolation to $Q \to \infty$. The extrapolation over ensemble size is demonstrated in Fig. 2, where ARV values obtained for training time $t = 70$ and $t = 140$ KE for the test set are depicted as a function of $\frac{1}{Q}$. It is quite clear that a linear extrapolation is very satisfactory.

The amount of variance due to initial conditions can be found by subtracting the $Q \to \infty$ result from that for $Q = 1$ (Fig. 2). Although this variance is due to choice of initialising estimates and not to different training sets, it does increase with time, as would be expected from the variance

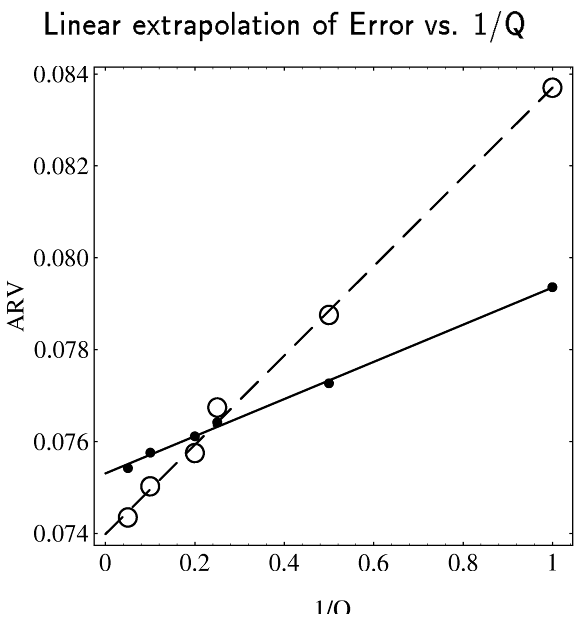

Figure 2: Extrapolation method used for extracting the $Q \to \infty$ prediction (From (Naftaly et al., 1997)). The results for different Q at two different training periods, $t = 70$ (top) and 140KE (bottom), can be extrapolated by a linear regression in $1/Q$.

due to different training sets.

3.3 Noise injection

In the previous section we have emphasised the reduction in the variance portion of the error resulting from ensemble averaging. This reduction was a result of the independence between the errors made by different predictors. While the use of initial random weights may lead to some independence, training on different datasets improves the independence even further. Bootstrapping (Breiman, 1996) is most appropriate for small datasets since, by resampling with replacement from the training data, the independence between the training sets is increased, yet each predictor has more than a $1/Q$ fraction of the training set on which to train. Smoothed bootstrap (Efron and Tibshirani, 1993) is potentially more useful since larger sets of independent training samples can be generated. The smoothed bootstrap approach amounts to generating larger datasets by simulating the *true* noise in the data. In this section, we demonstrate that training with added noise that is more variable than the true data-noise, and which increases the variance of individual predictors, can still be helpful, as it also increases the degree of independence between the training sets and, therefore, the degree of independence between predictors. A simple bootstrap procedure amounts to sampling with replacement from the training data and constructing several training sets, all of the same size as the original training set. Later, the variability between the sets of estimated parameters can be measured, and gives some indication about the true variability of the estimates of model parameters computed from the orginal data. Furthermore, variability or error-bars of the predictions can also be estimated in this way.

One version of bootstrap involves estimation of a model of the form

$$y = f(x) + \epsilon, \tag{5}$$

for some parametric family to which f belongs, and a noise variable ϵ which is assumed to have small variance and zero mean. An estimate $\hat{f}$ of f is obtained from n training samples, leading to fitted residuals $\hat{\epsilon} = (\hat{\epsilon}_1, \ldots, \hat{\epsilon}_n)$. One can then sample n times with replacement from the ϵ_i, giving $\epsilon^* = (\epsilon_1^*, \ldots, \epsilon_n^*)$, and construct new samples of the form (x_i, y_i^*), in which ϵ_i is replaced by ϵ_i^* sampled from the above set. Clearly, this approach can be easily extended to a smoothed bootstrap version (Efron and Tibshirani, 1993). In such a case, one can increase the size of each bootstrap set, since because of the noise the different sets are sufficiently independent. It should be noted that, if $\hat{f}$ is biased, the noise variance may be over estimated.

For classification problems, the form

$$y = f(x + \epsilon), \tag{6}$$

may be more appropriate. In this case, applying noise injection to the inputs during training, one can improve the generalisation properties of the estimator (Sietsma and Dow, 1991). Recently, Bishop has shown that training with small amounts of noise is locally equivalent to regularisation (Bishop, 1995). Here, we give a different interpretation of the addition of noise to the inputs during training, and view it as a regularising parameter that controls, in conjunction with ensemble averaging, the capacity and the smoothness of the estimator. The major role of this noise is to push different estimators to different local minima and thereby produce a more independent set of estimators. Best performance is then achieved by averaging the estimators. For this regularisation, the level of the noise may be larger than the 'true' level which can be indirectly estimated. Since we want to study the effect of bootstrapping with noise on the smoothness of the estimator, separately from the task of input noise estimation, we consider a highly nonlinear, noise-free classification problem, and show that, even in this extreme case, addition of noise during training improves results significantly.

We chose a problem that is very difficult for feed-forward neural networks to deal with. It is difficult because of the highly nonlinear nature of the decision boundaries, and the fact that these nonlinearities are easier to represent in terms of local radially symmetric functions rather than ridge functions such as those given by feed-forward sigmoidal functions. Since the training data are given with no noise, it seems unreasonable to train a network with noise, but we show that, even in this case, training with noise is a very effective approach for smoothing the estimator.

In the bootstrap ensemble with noise (BEN) (Raviv and Intrator, 1996), we push the idea of noise injection further. We observe that adding noise to the inputs increases the first term on the right-hand side of (3), in that it adds variance to each estimator, but it decreases the contribution of the second term as it increases the degree of independence between estimators. Instead of using the 'true' noise, estimated from the data, for bootstrap, we seek an optimal noise level which gives smallest contribution to the error from the sum of the two components of the variance. It is impossible to calculate the optimal variance of the Gaussian noise without knowing f explicitly, and therefore the value of this variance remains a regularisation term, a parameter which has to be estimated so as to minimise the total contribution of the variance term to the total error measure. Furthermore, since the injection of noise increases the degree of independence between different training sets, we can use bootstrap samples that are larger than the original training set. This does not affect the bias, if the noise is symmetric around zero, but it can reduce the variance. Note that the bias contribution to the error is not affected by introducing the ensemble-average estimator, because of the linearity of the expectation operator.

It follows that the BEN approach has the potential to reduce the contribution of the variance term to the total error. We should therefore seek a different level of trade-off between the contribution of variance and bias. In other words, we are able to use large (unbiased) networks without being affected by the large variance associated with such networks. This observation implies that the estimation of optimal noise levels should not be based on the performance of a single estimator, but rather based on the ensemble performance. The large variance of each single network in the ensemble can be tempered with a regularisation term such as weight decay (Krogh and Hertz, 1992; Ripley, 1996), but again the estimation of the optimal regularisation factor should be based on the ensemble-averaged performance. Breiman (Breiman, 1996) and Ripley (Ripley, 1996) present compelling empirical evidence of the importance of weight decay as a single network stabiliser. Our results confirm this fact under the BEN model. When no noise is injected, it was found that the non-regularised ensembles perform better (Taniguchi and Tresp, 1997).

The BEN algorithm

- Let $\{(x_i, y_i)\}$ be a set of training patterns for $i = 1, \ldots, N$.
- Let $\epsilon = \{\epsilon_1, \ldots, \epsilon_J\}$.
- Let $\lambda = \{\lambda_1, \ldots, \lambda_I\}$.
- For a noise level ϵ_j estimate an optimal penalty term for weight decay λ_i:
  - Fix a size K for the bootstrap sample, such that $K \gg N$ (we used $K = 10N$).
  - Let $s_1, s_2, \ldots, s_K$ be a set of indices, each chosen independently at random from $\{1, \ldots, N\}$.
  - Create a noisy bootstrap re-sample of the training set inputs, of the form $\{x_{s_i} + \zeta_i\}_{i=1,\ldots,K}$ and the corresponding re-sampled outputs $\{y_{s_i}\}_{i=1,\ldots,K}$, where ζ_i is a vector whose components are $N(0, \epsilon_j^2)$.
  - Train several networks with the noisy samples using weight decay parameters $\lambda_1, \ldots, \lambda_I$.
  - Generate an ensemble average of the resulting set of networks.
  - Choose the optimal weight decay parameter λ via cross-validation or a test set.
- Repeat the process for a new choice of noise ϵ_j until there is no improvement in prediction.

In the simplest version, the same noise level is used for each input dimension. This is suitable for problems in which all of the inputs are on the same scale, or, more precisely, when the noise distributions are similar in the different data dimensions. When all covariates have the same interpretation, e.g. similar measurements taken at different time steps, or when dealing with pixel data, such a noise assumption is adequate. However, when the noise is non-homogeneous in space or has a non-diagonal covariance matrix, or when different dimensions represent completely different measurements, it is best to estimate the different noise levels in each dimension separately. When this is too costly, or there are insufficient data for robust estimation, a quick solution is to sphere the data, that is, transform the variables so that the sample covariance matrix is the identity matrix.

3.3.1 The *Two-Spirals* Problem

The following *two-spirals* problem was chosen for the demonstration of the influence of added noise because (a) it is a hard problem for back-propagation networks because of its intrinsic high nonlinearity and radial symmetry, (b) it is a noise-free problem, and (c) the generalisation performance of

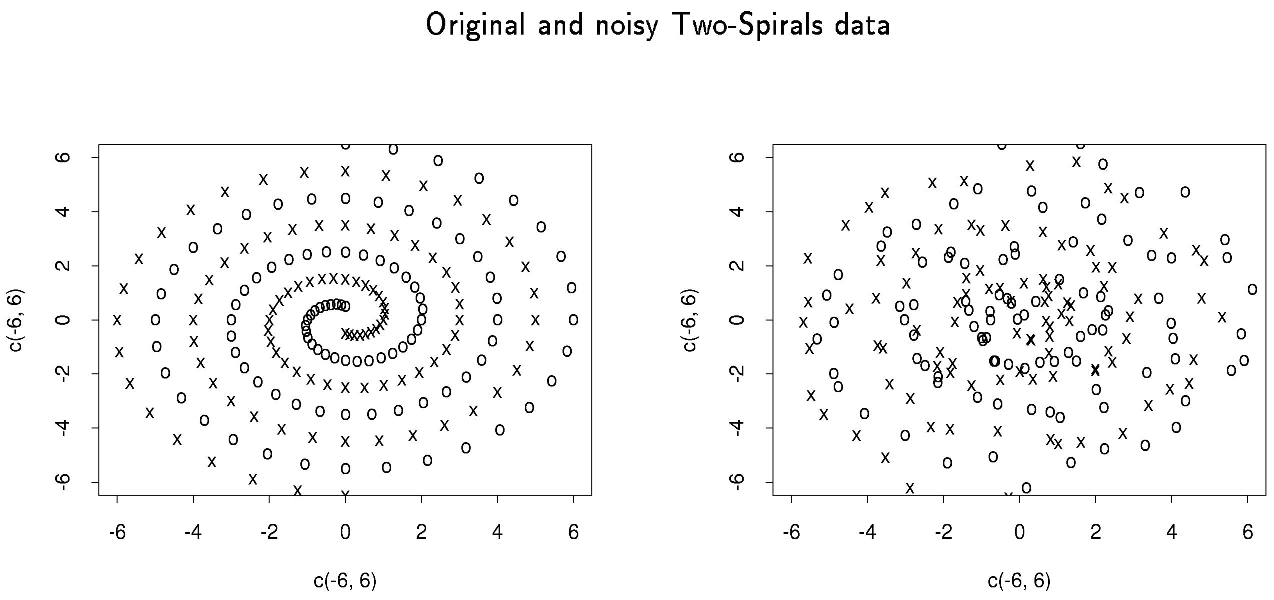

Figure 3: The *Two-Spirals* Training data (left). Training Points with noise - SD=0.3 (right). As can be seen, the noise level that contaminates the data, causes objects to *cross* the virtual boundary defined by the data, namely the noise leads to wrong class labeling for the training data. This reduces performance of single predictors, but the added independence between the predictors leads to improved ensemble performance.

different predictors can be easily visualied in two dimensions. This problem consists of a training set of 195 points, half of which are to produce an output value of 1 and half an output of 0. These training points are arranged in two interlocking spirals that go around the origin three times, as shown in Fig. 3. The example was first proposed by Alexis Wieland of MITRE Corp. It is easy to see that the two sets of points in the spirals cannot be separated by a combination of a small number of linear separators. Lang and Witbrock (Lang and Witbrock, 1988) proposed a $2-5-5-5-1$ multilayer network with logisitc sigmoid activation functions and 'short-cuts' using 138 weights. They used a variant of the quick-prop learning algorithm (Fahlman, 1988) with weight decay. They claimed that the problem could not be solved with simpler architecture involving fewer layers or without short-cuts. Their resulting predictor applied to the original dataset seems to give poor generalisation results. Baum and Lang (Baum and Lang, 1991) demonstrated that there are many sets of weights that would cause a $2-50-1$ network to be consistent with the training set, but when such a network was trained with error back-propagation, starting with random initial weights, none of these desirable solutions was found.

Fahlman (Fahlman and Lebiere, 1990) used the Cascade-Correlation architecture for this problem, obtaining better results, but still little 'spiralness'. Recently, Deffuant (Deffuant, 1995) suggested the 'Perceptron Membrane' method that uses piecewise linear surfaces as discriminators, and applied it to the spiral problem. He used 29 Perceptrons but had difficulties capturing the structure of the spirals because of the piecewise linearity of his decision boundaries.

The minimisation criterion dictating the training process is mean squared error with weight decay regularisation:

$$E = \sum_p |t_p - y_p|^2 + \lambda \cdot \sum_{i,j} w_{i,j}^2, \tag{7}$$

where t_p is the target and y_p the output for the pth example pattern. The $w_{i,j}$ are the weights and λ is a parameter that controls the amount of weight decay regularisation. This minimisation was achieved using Ripley's S-Plus 'nnet' package (Ripley, 1996), which implements back-propagation, along with our code for boosting, noise injection and ensemble averaging.

The network had two inputs, 30 hidden units and one output. The first and last layers were fully connected to the hidden layer, giving a total of 121 weights. The activation function at the hidden and output units was the logistic sigmoidal function. The initialising weights were chosen randomly from $U(-0.7, 0.7)$. It should be noted here that, although we are training 5–40 networks, the effective number of parameters is not more, and is probably even less, than the number of parameters for a single network. This is because we do not have the flexibility to estimate an optimal combination of predictors, but rather take the simple average of them.

Baseline results were obtained by training 40 networks without any regularisation. We derived then an average predictor whose output is the mean of the ouputs of the 40 nets (Figure 5, top left). The predictor had no smoothness constraint and therefore found relatively linear boundaries.

3.3.2 Applying bootstrap to networks with weight decay

The best results (Raviv and Intrator, 1996) were obtained when applying the bootstrap ensemble with noise (BEN) method to networks with optimal weight-decay regularisation ($\lambda = 3e^{-4}$). Figure 4 demonstrates the effect of bootstrap with noise on the performance of a 5-net ensemble trained with optimal weight decay. Each image is a thresholded output of a 5-nets ensemble average predictor. Noise level goes from $\epsilon = 0$ in the upper left image through $\epsilon = 0.8$ in the lower right. The classification results are drawn on a uniform grid of 100×100 points (representing therefore a much larger test set) so as to get a clear view of the classification boundaries defined by the classifier. The effect of ensemble averaging over networks that were trained with different random initial conditions only is demonstrated in the top left image, which represents the case of adding no noise during training. It can be seen that, for small noise levels ϵ, the ensemble average predictor is unable to find any smooth structure in the data and merely over-fits the training data. For moderate levels of the noise, better structure can be found, and, for large levels of the noise, the data are so badly corrupted that again no structure can be found. The optimal noise standard deviation, σ, was around $\sigma = 0.35$. Optimal noise values are similar to those obtained when training with no weight decay, and are surprisingly high (see Fig. 3, (right) for the result of adding noise to the data). Although the results look better than those obtained with no weight decay, in the sense that the boundaries look smoother, they can be further improved by averaging over a larger ensemble of 40 networks. This is demonstrated in Fig. 5 which summarises the effect of averaging. There was a clear improvement in all results with the large ensemble size as well as the addition of either noise or weight decay regularisation. Finally, the combination of weight decay, noise and a 40-net ensemble clearly gives the best results (Fig. 5, bottom right). Thus, while earlier work suggested that a single-layer feed-forward network is not capable of capturing the structure in the spiral data, it is evident that a network ensemble with strong control over its capacity (via weight decay) which is trained with heavy noise can discover the highly nonlinear structure of the problem.

3.3.3 Properties of variance regularisation via noise

The motivation of our approach comes from a key observation regarding the bias/variance decomposition of prediction error, namely the fact that ensemble averaging does not affect the bias portion of the error but reduces the variance, when the estimators on which averaging is done are independent. The level of noise affects the level of independence between the training sets, and

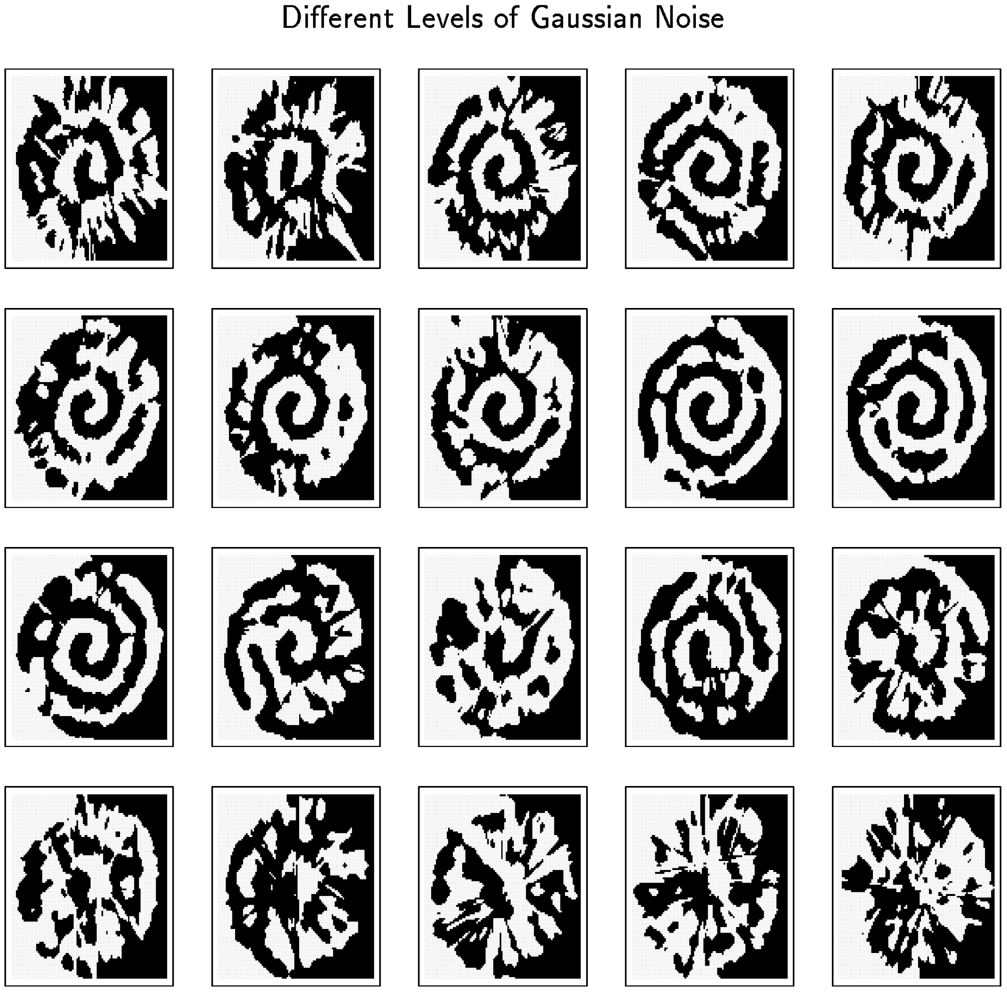

Figure 4: Ensembles of 5 networks with fixed weight decay and a varying degree of noise (top left is zero noise, bottom right is noise with SD=0.8.) The classification threshold is 0.5.

thus the relative improvement of ensemble averaging. However, the level of noise also affects the quality of each predictor separately, increasing its variance by increasing the variability in the data. Thus, there should be an optimal level of the noise, which may not correspond to the true noise level and which leads to optimal ensemble performance. This performance can be further improved if the variance of individual networks can be tempered, e.g. with weight decay.

Raviv and Intrator (1996) demonstrated the effect of noise injection on prediction in three different cases. (i) The highly nonlinear (spiral) data, using a suboptimal model; the data are almost radially symmetric and the neural net is not. This required the use of an ensemble of high capacity single predictors and thus made the regularisation task challenging. It was shown that the excess variance of high capacity models could be effectively trimmed only by a combination of all three of weight decay, noise injection and ensemble averaging. (ii) Highly nonlinear (spiral) data with essentially the perfect model for it, namely a Generalised Additive Model (GAM) with locally linear units (Hastie and Tibshirani, 1986). Even in this case, where regularisation provides the perfect degree of bias to the model, performance could be improved. (iii) A highly linear problem, where practically any network has excess capacity, that is, is overparameterised. This case is a

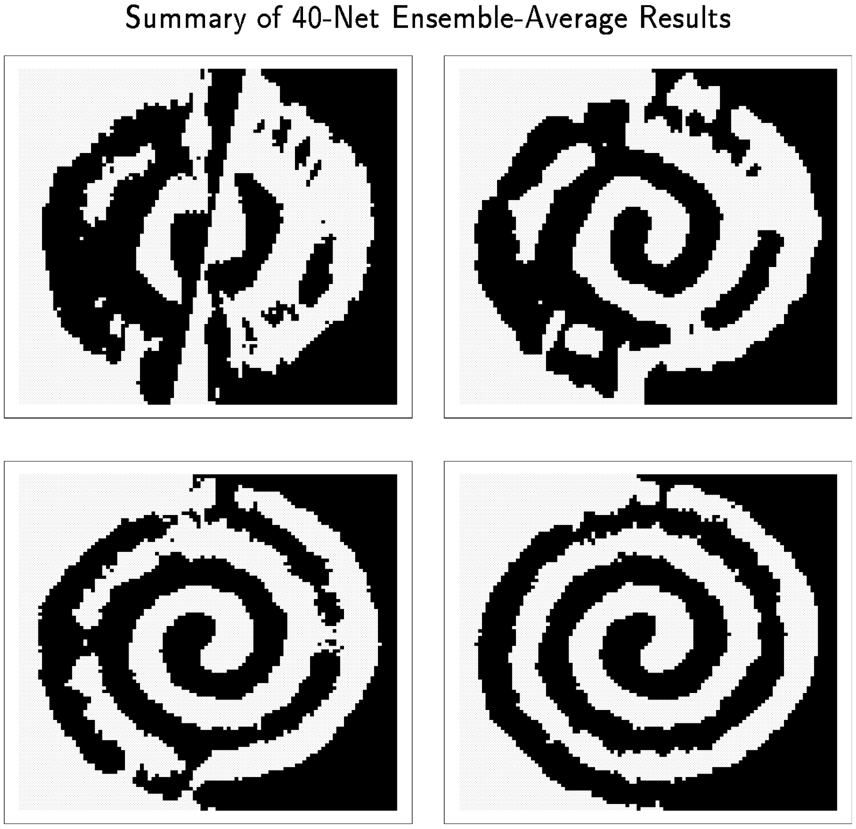

Figure 5: *Top left:* No constrains (no weight decay or noise). *Top right:* Optimal weight decay ($\lambda = 3e^{-4}$) and no noise. *Bottom left:* Optimal noise (Gaussian SD=0.35) and zero weight decay. *Bottom right:* Optimal noise and optimal weight decay.

representative of a family of clinical datasets, in which (linear) variable selection was applied to a high-dimensional dataset and resulted in a highly linear low-dimensional data structure. It was thus challenging to be able to show that the BEN algorithm is useful in this case, and can lead to improved classification results. Performance was also evaluated using the Receiver Operating Characteristic (ROC) measure (Hanley and BJ, 1982), which is a standard model comparison tool for clinical data analysis.

Theoretical analysis suggests that it is best to start with a very flexible function approximation technique (e.g. a feed-forward network with a large number of hidden units) and then control its capacity and smoothness using noise and averaging. These conclusions hold also for other models such as generalised additive models.

4 Integrating Ensembles of Estimators

After having established that ensembles of estimators are very useful for variance control and having discussed ways to increase the level of independence between ensemble members, as well as optimal stopping criteria for ensembles, we study the possible additional benefits of ensembles of experts. In particular, we would like to assign different ensembles to different regions in input space, or different representations of the input space, constructed by various types of preprocessing, in an attempt to exploit the independence between different ensembles. We start with a review of current related methods and demonstrate our ideas on a specific seismic classification problem. Full details of the algorithms and application to this problem are given in (Shimshoni, 1995; Shimshoni and Intrator, 1998).

The lack of a-priori knowledge about the true underlying model of the data in the seismic classification problem, as in many other real-life problems, leads practitioners to examine various sub-optimal classifiers. Different classifiers can exploit various types and sets of features, and hence combining multiple estimated classifiers might yield better performance than the best single candidate. In the Neural Network and Machine Learning literature there are several methods for combining estimators, and important questions such as *what* types of estimator to combine and *how* to do it are currently getting considerable attention.

A well known class of combined models contains methods such as the Adaptive Mixture of Experts (AME) (Jacobs et al., 1991) and Hierarchical Mixture of Experts (HME) (Jordan and Jacobs, 1994), which are both based on the *divide and conquer* approach where a mixture of experts compete to gain responsibility in modelling the output in a portion of the input space. The system's output is obtained as a linear combination of the experts' outputs, where the weights are computed as a parametric function of the inputs by a gating module. The underlying probabilistic model is based on the assumption of mutual exclusivity, i.e. a single expert is responsible for each data point. In mixture models like AME and HME the different experts are usually trained on a single dataset simultaneously by minimising a combined cost function. The final combination of the experts is determined by the gating module, which is constructed during the same training session. When training all the experts on the same dataset with the same data representation, the dependence of errors among experts is high, thus diminishing their collective contribution.

A different aspect of the multiple model concept is introduced by using a committee of classifiers, also called an *ensemble* (Hansen and Salamon, 1990; Perrone and Cooper, 1993). Consider a trained classifier as a realisation of a generic model trained on the given data. Thus, different datasets would yield different realisations, all of which are members of the *post training* distribution of the possible solutions. As the solution space is generally highly degenerate and includes many local minima, it is more robust to use a sample from this solution space (i.e. ensemble of estimators) rather than a single representative (Hansen et al., 1994). The ensemble of classifiers can be averaged to produce an aggregated classifier, or any linear combination of the realisations can be used.

When all the classifiers give similar results, the accuracy of their combined classification depends largely on their bias, because of the bias-variance tradeoff (Geman et al., 1992). Whenever the bias of a generic model is high, the multiple classifications of its ensemble might all be wrong even if the variance is low, in which case no combination operation will help. On the other hand, when the bias is small and the variance is high, one could expect the ensemble to disagree whenever the input signals are ambiguous. In such cases the ensemble's result, i.e. the aggregated classification, will have the same bias but a reduced variance. Hence, combining multiple classifiers can eliminate the need to regularise over-fitted models with high variance (Sollich and Krogh, 1996). Moreover, the classification confidence can be evaluated from the variance which represents the agreement within the ensemble, regarding signals with high classification variance as being treated with suspicion.

Estimation of the optimal combination of the experts should be done in a robust way, i.e. by averaging or cross-validation techniques rather than by parametric estimation based on the same training data (LeBlanc and Tibshirani, 1993). It has been shown that, in order for the combination of experts to be optimal, the experts should be made as independent as possible (Krogh and Vedelsby, 1995; Meir, 1994; Jacobs, 1995; Raviv and Intrator, 1996). Another method which uses multiple classifiers in a sequential training scheme is that of 'Boosting' (Schapire, 1990; Drucker et al., 1994). In this method, which is typically suitable for very large training sets as in optical character recognition (OCR) problems, each classifier is trained on patterns that have been filtered by the previous classifier. Thus, the result is a combination of classifiers that have been trained on statistically different datasets in a sequential process.

A more general framework for combining multiple estimators is that of 'Stacked Generalization' (Wolpert, 1992), in which each estimator is trained with a different subset of the data and the optimal combination is estimated using cross validation methods. A formulation of this method for regression estimators was presented by (Breiman, 1993) and was compared with other methods by (LeBlanc and Tibshirani, 1993). 'Bagging' is a method which produces an aggregated estimator using bootstrap replicas of the training data (Breiman, 1996). It is reported to be useful whenever the estimator is unstable, i.e. when perturbing the training set can cause significant changes in the constructed classifier. Notice that this condition corresponds to the requirement mentioned earlier of maximum independence among the experts. Several ways have been suggested for making the experts less dependent, for example injecting noise during training, as in smoothed bootstrap (Raviv and Intrator, 1996).

Since the search for an optimal classifier is tied to the search for an optimal data representation (i.e. an optimal transformation of the input signals with respect to the classification task at hand), it is advisable to examine and possibly use more than one signal representation. In order to maximise both the information extracted from the data and the gain of combining multiple classifiers, we suggest constructing a 'Redundant Classification Environment', which allows for different signal representations to supply a wide coverage of the effective feature space.

4.1 Creating a Redundant Classification Environment

The hierarchical scheme proposed by Shimshoni and Intrator is based on experts which are ensembles of Artificial Neural Networks (ANN), each of which is associated with a specific data representation and network architecture. The redundancy is pronounced within these ensembles, which are collections of ANN realisations trained on different subsets of the data. Each ensemble is trained using the same data representation i.e. a unique Time-Frequency decomposition and smoothing level of the input signals. The network architecture (number of hidden units) is also fixed for all members in an ensemble. The redundancy is even more pronounced, as we use various combinations of Time-Frequency decompositions, smoothing levels and network architectures to create and train several such ensembles. The different training conditions yield relatively independent classifiers, that might produce different classification results given an ambiguous signal. The *Integrated Classification Machine* (ICM) that is formulated next integrates the different ensembles in this Classification Environment and produces a final classification which achieves better generalisation performance than the single classifying components; see the results in Section 4.4.

The Integrated Classification Machine is constructed from a hierarchy of classifiers as shown in Figure 6. The smallest building block of the ICM is a neural network (feed-forward multilayer perceptron with logistic sigmoidal activation functions). All networks are trained to predict the class label of a given signal. As mentioned, we use several representations for the input signals, so that the input layers of the neural networks in the ICM have different dimensionalities *(N)* according to the input representations used. The hidden layer can contain various numbers *(H)* of sigmoidal units and the output layer contains two sigmoidal units.

4.1.1 The Network's Prediction Value

The desired output of the networks for a given signal can be either {1,0} for *Natural* events or {0,1} for *Artificial* events. The sigmoidal output units of the trained networks will produce continuous values in the range of 0 to 1 according to the network weights:

$$O^l = \sigma\left(\sum_{i=1}^{H} W_{li}\ \sigma(\sum_{j=1}^{N} w_{ij}x_j + w_{i0})\ + W_{l0}\right) \qquad l = 1, 2. \tag{8}$$

Let us define the (signed) difference between the two output units, $y = (O^1 - O^2)$, to be the *prediction value* of the network. Hence, $y \in [-1, 1]$ and the predicted class label is given by thresholding y at zero, assigning positive values to the class of *Natural* events and negative values to the class of *Artificial* events. Each network is trained on T repeated trials, changing only the initial random weights; in our implementation $T = 5$. We define the *prediction value* of the network component in the ICM with respect to a signal x as the average prediction value $y(x)$ from the T training trials: $y^{NET}(x) = \frac{1}{T}\sum_{t=1}^{T} y_t(x)$.

4.1.2 The Ensemble's Prediction Value

Each Ensemble is a collection of B networks, where each network is trained on one of the B replicas, $D_r^b\ \ b = 1, \ldots, B$, of the original dataset for a specific data representation r; we used 30 bootstrap sets, which is fewer than suggested in (Efron and Tibshirani, 1993) (≈ 200), but was found to be sufficient. Details of the replication of the data into bootstrap sample sets (Efron and Tibshirani, 1993) are given in (Shimshoni and Intrator, 1998).

All the networks in an ensemble share the same data representation and the same network architecture. The *prediction value* of an ensemble with respect to a signal x is defined as the average over all the prediction values of the participating networks, as in the method of 'Bagging' (Breiman, 1996):

$$y^{ENS}(x|D_r) = \frac{1}{B}\sum_{b=1}^{B} y_b^{NET}(x|D_r^b) \tag{9}$$

4.1.3 The Integrated Prediction Value

A collection ($\mathcal{K}$) of ensembles, which use different input representations, i.e. Time-Frequency decompositions or smoothing levels, and different network architectures, i.e. number of hidden units, form the Integrated Classification Machine (ICM) shown in Figure 6. The *integrated prediction value* of the ICM with respect to a signal x is defined by

$$y^\star(x) = \sum_{k\in\mathcal{K}} \alpha_k\ \beta_k(x)\ y_k^{ENS}(x). \tag{10}$$

Here α_k is a prior reliability measure of the kth ensemble, which can be determined from the training data or from prior knowledge, with $\alpha_k = \frac{1}{K}$ as a default value. The quantity $\beta_k(x)$ is a posterior classification confidence measure that is specific to each signal and is discussed in the next section. Both measures are normalised and together determine the strength of the 'vote' of ensemble k in the 'classification committee' for signal x.

In order to detect signals with ambiguous class membership and to rank the different ensembles in terms of their accuracy of classification, we have constructed a confidence measure for the ensemble's classification. This posterior confidence is based on the variance of the networks' prediction values,

$$CONF^{ENS}(x) = [\ \mathrm{Var}(y^{NET}(x))\ ]^{-1}, \tag{11}$$

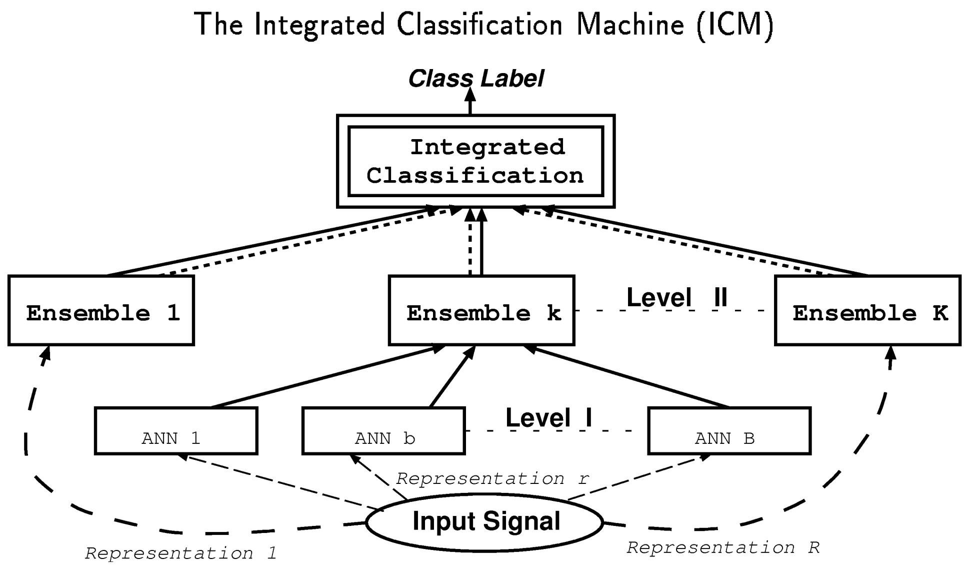

Figure 6: Several representations of the waveform are fed into different Ensembles, then integrated to produce the final classification. (In level II, the regular arrows are the Ensembles' prediction values and the dashed arrows are the attached confidence values).

where $y^{NET}(x)$ is the network's prediction value. The $CONF$ score represents the amount of 'agreement' among all the participating networks in the ensemble (Krogh and Vedelsby, 1995; Hansen et al., 1994). It should be remembered that, when the bias of the networks is high, such a measure will not convey confidence (i.e. when all members agree on the same wrong class). Therefore, one should examine the classification results carefully to check whether or not there is a correlation between the confidence scores and the errors of the combined classification.

4.2 Combining the Hierarchy of Classifiers

The *ICM* shown in Figure 6 is a hierarchy of classifiers with two levels at which multiple classifiers are combined. At the first level, for each ensemble, B networks are combined using simple averaging to construct an aggregated ensemble classification (Eq. 9). These networks are in fact multiple realisations of the same classifier trained on different Bootstrap samples.

The second level integrates ensembles into the final classification. Integrating ensembles involves an estimation of $\beta_k(x)$, namely the strength of the 'vote' of the kth ensemble (Eq. 10). This is done by applying a Least-Squares calculation with non-negativity constraints (Breiman, 1993). Finally, prior weights can also be used to combine the ensembles.

Unlike the integration of networks into ensembles at the first level, where each network is the same apart from being trained on a different subset of the data, in the second level it is less likely that a fixed weighting will produce better classification results than those of *all* single ensembles. There might be some ensembles which use inferior data representations or less suitable model architectures, which will have disturbing effects on the weighted result. In practice, we have noticed that, for fixed weighting methods like those mentioned above, the integrated classification results were often worse than the best participating single ensemble.

Given a low signal-to-noise ratio and non-stationarity of the signal space, along with a shortage of training data, it is furthermore desirable not to base the choice of averaging weights on the characteristics of the currently available dataset. Therefore, we suggest using the signal x to find the optimal set of weights for its own 'classification committee', namely to apply a non-fixed weighting of the classifiers. One can use the ensemble's prediction value $y^{ENS}(x)$ as the basis for the weighting, or decide how to weight the votes using the classification confidence $CONF^{ENS}(x)$ (Tresp and Taniguchi, 1995). Application of the maximum entropy principle (Jaynes, 1982) suggests that an optimal combination of ensembles should set

$$\beta(x) = \exp[-\mathrm{Var}(y^{NET}(x))]. \tag{12}$$

We have used a non-fixed weighting strategy which is a dynamic *Winner Takes All* procedure. In order to integrate the different ensembles in the *ICM* for a *specific* signal x, all classifiers (ensembles) are ranked and the optimal *one* is selected. The ranking is governed by the overall reliability of the classifiers and the selection is based on the classification confidence $CONF^{ENS}(x)$ supplied by the ensembles along with their classification (see Figure 6). This approach, which will be elaborated in the next section, is different from the linear non-fixed weighting that was suggested by (Tresp and Taniguchi, 1995) for combining ANN's, where all classifiers participate in the committee with weights inversely proportional to the variance; this is similar to the use of $CONF$ values here, but the weights are estimated in a different way as they use single ANN classifiers rather than ensembles. The next section discusses a similar approach, but exploits further the variability between members of each ensemble.

4.3 Competing Rejection Algorithm (CRA)

Generally, a signal is said to be rejected by a classifier if some measure representing the quality of its classification falls below a pre-defined threshold. Obviously, the higher the threshold, the more signals will be rejected and the smaller will be the misclassification rate over the remaining un-rejected signals. Shimshoni and Intrator (1997) have presented an algorithm which performs a sequence of classifications by polling the group of K classifiers with respect to the signal at hand. Each classifier (ensemble) in turn can either classify or reject the signal. A key motivating observation is that some classifiers perform globally better than others. Nevertheless, classifiers can occasionally outperform 'superior' classifiers, and should therefore be given the opportunity to compete and possibly to 'steal' a classification even if signal is rejected by those 'superior' classifiers.

Implementation of this idea requires a prior reliability ranking of the classifiers and a definition of a rejection criterion. The rejection is done by thresholding the confidence measure $CONF^{ENS}(x)$, defined in (11), so that a classification is rejected when its confidence value is lower than the it 'reject' threshold. Each ensemble k, has its own threshold, Θ_k, depending on its accuracy and variability, that fixes the minimum level of confidence 'allowed' for its classifications. Θ_k is calculated as a certain upper percentile of the confidence scores $CONF^{ENS}(x)$, of an unlabelled dataset $D^\star$ as follows:

$$\Theta_k = \mathrm{percentile}\{CONF^k(x), \text{Reject-Rate}_k\}. \tag{13}$$

The 'Reject-Rate' of a classifier represents its global credibility and can be determined either from its performance on training data or by using subjective information. One simple approach is for all ensembles to have the same credibility, and thus a uniform Reject-Rate is set, for example based on the upper 20th percentile. When all classifiers reject a signal, it can be either 'globally rejected' or classified by the *Ultimate Classifier*, which is optionally pre-defined by the user. The global rejection rate cannot be determined by the user and is usually much smaller than the individual Reject-Rates.

This selection algorithm,along with the whole ICM structure, is easily scalable, and no re-training is required when new classifiers are added. It is also flexible, in the sense that it is straightforward to incorporate other types of experts, including human ones, as long as they produce suitable prediction-values and $CONF$ values.

4.4 Results on seismic data classification

The Integrated Classification Machine was applied to a dataset consisting of 380 seismic events, including all the natural local earthquakes that occurred between January, 1990, and June, 1993, inside an area of 22500 km^2 in the northern part of Israel. The seismic dataset is available from `ftp://www.math.tau.ac.il/$\sim$shimsh/pub/seismic-data/`. A similar number of artificial explosion events were randomly sampled from the same spatio-temporal window. All events have magnitude $M_L < 2.7$, while 77% of the events are below 2.0 and the mean magnitude is 1.53. Results from these data (Shimshoni, 1995; Shimshoni and Intrator, 1998) indicate that the ICM could further improve results of the best ensemble by dynamically combining results from inferior ensembles using the algorithms discussed above. Comparison to other ensemble methods and more classical statistical methods further demonstrates the strength of this method as a general mixture of experts machine.

5 Bias control: Imposing prior knowledge

Smoothness constraints (Wahba, 1990; Poggio and Girosi, 1990) are often used as variance constraints. Similarly, in training neural networks, weight decay, which attempts to shrink the weights to zero or to a single value (Hinton, 1986; Krogh and Hertz, 1992) is often used. In other statistical contexts, such as regression or generalised linear models, this method is often called shrinkage (James and Stein, 1960). The resulting constraint on the weights effectively reduces the number of free parameters in the model. A brute force method for reducing the number of model parameters is 'optimal brain damage' (Le Cun et al., 1990), in which weights which become smaller than a predefined threshold are set to zero. Other related methods include weight sharing, in which a single weight is shared among many connections in the network (Waibel et al., 1989; Le Cun et al., 1989). An extension of this idea is 'soft weight sharing' which favours irregularities in the weight distribution in the form of multi-modality (Nowlan and Hinton, 1992). This approach may improve upon generalisation results obtained by weight elimination. Both these methods make an explicit assumption about the structure of the weight space, independently of the structure of the input space.

We now turn our attention to the second part of prediction error, namely the bias. It may sometimes appear difficult to distinguish between bias constraints and variance constraints. For example, how do we treat smoothness constraints? While it reduces the variance, it clearly enforces a bias towards smooth models. Since we are dealing with additive constraints, as will be clear from the way that parameter estimates are calculated (see equation (18)), we can offer a simple distinction. If the update rule leads to no meaningful result when only the additional (bias/variance) constraint is effective, we regard it as a *variance* constraint. When some meaningful result is obtained via this unsupervised constraint only, we regard it as a *bias* constraint. Thus, classical constraints such as smoothness, as well as assumptions about the distribution of the parameters, e.g. favouring small weights via a weight decay or favouring particular distributions such as mixtures of Gaussians (Nowlan and Hinton, 1992), are actually variance constraints. The next section discusses a general framework for imposing bias constraints and describes some specific examples of such constraints.

5.1 Projection Pursuit

A general framework for additive and semi-linear dimensionality reduction is projection pursuit. It is useful when a low dimensional representation is embedded in high-dimensional data. The supervised version, called Projection Pursuit Regression (PPR) (Friedman and Stuetzle, 1981), is capable of performing dimensionality reduction by composition, in that it constructs an approximation to the desired response function using a composition of lower-dimensional smooth functions. These functions depend on linear-dimensional projections through the data.

When the dimensionality of the problem is in the thousands, even projection pursuit regression methods are almost always over-parameterised and lead to data over-fitting. At that stage we must impose prior knowledge on the desired low dimensional representation. The unsupervised version, called Exploratory Projection Pursuit (EPP) (Friedman and Tukey, 1974; Friedman, 1987) becomes relevant. It searches in a high-dimensional space for structure in the form of (semi-)linear projections with constraints characterised by a *projection index*, which measures the goodness of a projection. The projection index may be considered as a universal prior for a large class of problems, or may be tailored to a specific problem based on prior knowledge.

5.2 Brief Description of Projection Pursuit Regression

Let (X, Y) be a pair of random variables, with $X \in R^d$ and $Y \in R$. The problem is to approximate the d dimensional regression surface,

$$f(x) = E[Y|X = x], \tag{14}$$

on the basis of n observations $(x_1, y_1), \ldots, (x_n, y_n)$.

Projection Pursuit Regression tries to approximate f by a sum of ridge functions, i.e. functions that are constant along lines:

$$f(x) \simeq \sum_{j=1}^{m} g_j(a_j^T x). \tag{15}$$

The fitting procedure alternates between estimation of directions by $\hat{a}_j$ and estimation of smooth functions by $\hat{g}_j$. At stage j, the sum of squares of the residuals

$$r_{ij}(x_i) = r_{i,j-1}(x_i) - \hat{g}_j(\hat{a}_j^T x_i) \tag{16}$$

is minimised. The process is initialised by setting $r_{i0}(x_i) = y_i$, for each i. Usually, the initial values of $\hat{a}_j$ are taken to be the first few principal components of the data.

Estimation of the ridge functions can be achieved by various nonparametric smoothing devices such as locally linear functions (Friedman and Stuetzle, 1981), k-nearest neighbour methods (Hall, 1989), splines or variable degree polynomials. Imposition of a smoothness constraint on $\hat{g}_j$ implies that the actual projection pursuit is achieved by minimising, at iteration j, the sum

$$\sum_{i=1}^{n} r_{ij}^2(x_i) + C(\hat{g}_j), \tag{17}$$

for some smoothness measure C.

Since the estimation of the nonparametric ridge functions is not decoupled from the estimation of the projections, over-fitting is very likely to occur in one of the low-order $\hat{g}_j$, thereby invalidating subsequent estimation stages. Obviously, if $\hat{g}_j$ is not well estimated, the search for an optimal projection direction will not yield good results.

Artificial neural network architectures are closely related and offer some properties which improve their variance control in high dimensional cases. In feed-forward neural nets, the family of ridge functions is limited to sigmoidal functions with variable threshold. This avoids the non-parametric or semi-parametric estimation of the ridge functions, but may require a large number of projections, as in the example in Section 3.3.1. Secondly, the estimation of several projections is performed concurrently. This allows one to find a low dimensional representation which cannot be found if the search is done sequentially (Huber, 1985).

5.3 Hybrid attempts at dimensionality reduction

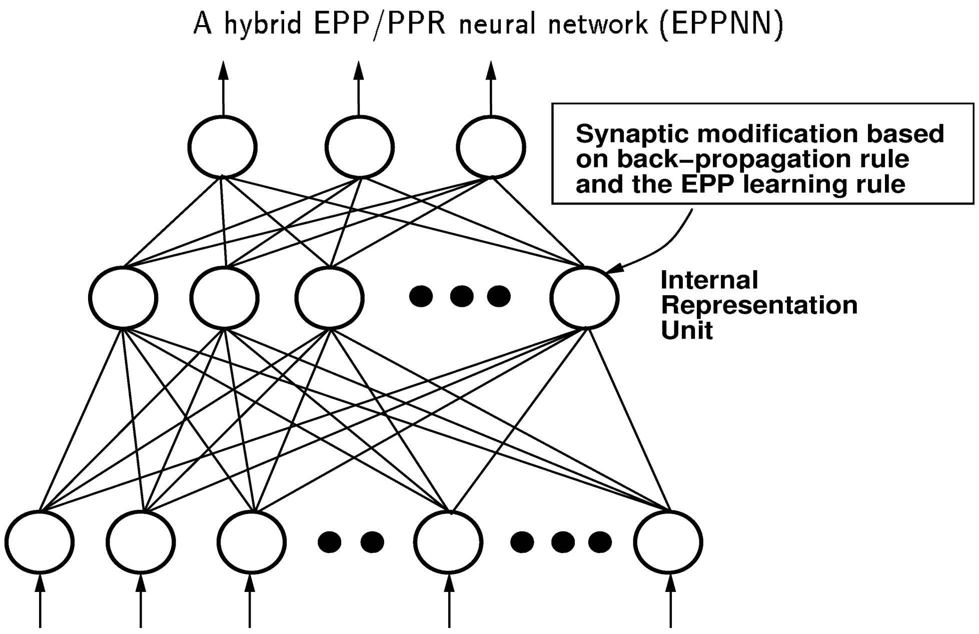
A hybrid EPP/PPR neural network (EPPNN)
Synaptic modification based on back–propagation rule and the EPP learning rule
Internal Representation Unit

Figure 7: The modification of the hidden units' weights is achieved by back-propagation of the error from the output layer (via the chain rule) and by the gradient of the projection index.

There have been various attempts to combine unsupervised learning with supervised learning (Yamac, 1969; Gutfinger and Sklansky, 1991; Bridle and MacKay, 1992). The formulation discussed below is based on projection pursuit ideas, which generalise many of the classical statistical methods, and, in our case, suggest a well-defined statistical framework that allows formulation and comparison between various methods. Consider the artificial neural network architecture presented in Fig. 7. The only difference from a classical feed-forward architecture (Rumelhart et al., 1986) is the additional modification term among the hidden units. We consider the hidden unit representation as a new, reduced-dimensionality representation of the data. As described in the context of projection pursuit regression (Intrator, 1993), we add a penalty term to the energy functional minimized by error back-propagation, for the purpose of measuring directly the goodness of the projections sought by the network. This puts the emphasis on choosing the right prior, as a means to improving the bias/variance tradeoff.

Since our main interest is in reducing over-fitting for high dimensional problems, our underlying assumption is that the surface function to be estimated can be faithfully represented using a low-dimensional composition of sigmoidal functions, namely, using a feed-forward architecture in which the number of hidden units is *much smaller* than the number of input units. Therefore, the penalty term may be added to the hidden layer only; see Fig. 7. The synaptic modification equations for the hidden units' weights become

$$\begin{aligned}\frac{\partial w_{ij}}{\partial t} &= -\epsilon\,[\frac{\partial \mathcal{E}(w,x)}{\partial w_{ij}} \\ &\quad + \frac{\partial \rho(w_1,\ldots,w_n)}{\partial w_{ij}} \\ &\quad + (\text{Contribution of cost/complexity terms})],\end{aligned} \tag{18}$$

where $\mathcal{E}$ is the error function and ρ is the explicit measure of goodness of projections (bias constraints), while the contribution of the cost/complexity terms (variance constraints) are also additively imposed.

5.4 Specific bias constraints

Exploratory projection pursuit is based on seeking *interesting* projections of high-dimensional data (Kruskal, 1969; Switzer, 1970; Kruskal, 1972; Friedman and Tukey, 1974; Friedman, 1987; Jones and Sibson, 1987; Hall, 1988; Huber, 1985). The notion of interesting projections is motivated by an observation that, for most high-dimensional data clouds, most low-dimensional projections are approximately normal (Diaconis and Freedman, 1984). This finding suggests that the important information in the data is conveyed in those directions whose univariate projected distribution is far from Gaussian. Various projection indices differ in the assumptions about the nature of the deviation from normality, and in their computational efficiency. They can be considered as different priors motivated by specific assumptions on the underlying model.

Since the Gaussian distribution maximizes entropy, subject to specified mean and variance, it is possible to test deviation from Gaussianity by the difference between the entropy of a Gaussian, with the same variance, and the given distribution. This measure is non-negative and provides a measure of the information content of the distribution relative to the maximal content of a Gaussian with the same variance. The index is given by

$$\begin{aligned}J_I(p) &= H(p_G) - H(p) \\ &= \frac{1}{2}\log(2\pi e) + \log(\sigma) + \int p(x)\log p(x)dx,\end{aligned} \tag{19}$$

where p is the probability density function and σ is the standard deviation of the distribution of interest. This index also leads to redundancy reduction and independent component analysis; see below.

As the density $p(x)$ is unknown, it has to be estimated from the data. This is computationally expensive and not very robust. Although it is possible to use the data to estimate the density nonparametrically, for example using the kernel method (Wand, 1994; Viola and Wells, 1995), it is usually preferred to estimate the integral in (19) by some other approximation to the density. When the third and fourth cumulants (Kendall and Stuart, 1977),

$$\begin{aligned}\kappa_3 &= \frac{E[(x-\bar{x})^3]}{\sigma^3}, \\ \kappa_4 &= \frac{E[(x-\bar{x})^4]}{\sigma^4} - 3,\end{aligned} \tag{20}$$

of the distribution are known, it is possible to estimate the integral using some approximation to the probability density function. Comon(1994) proposed using the Edgeworth expansion (Stuart and Ord, 1994), but recently the Gram-Charlier expansion (Stuart and Ord, 1994) has been proposed

(Amari et al., 1996) as it explicitly depends on the third and fourth cumulants of the distribution. This Gram-Charlier approximation has the form

$$p(x) \simeq \alpha(x)\{1 + \frac{\kappa_3}{3!}H_3(x) + \frac{\kappa_4}{4!}H_4(x)\}, \tag{21}$$

where $\alpha(x) = \frac{1}{\sqrt{2\pi}}\exp(-x^2/2)$ and $H_k(x)$ are Chebyshev-Hermite polynomials, given in our case by

$$\begin{aligned} H_3(x) &= 4x^3 - 3x, \\ H_4(x) &= 8x^4 - 8x^2 + 1. \end{aligned} \tag{22}$$

The exact measure of deviation from a Gaussian distribution is clearly expressed in approximation (21) in terms of the skewness and kurtosis of the distribution. If we substitue this approximation into (19) we obtain

$$\hat{J}_I(p) = \sigma - \frac{(\kappa_3)^2}{2 \cdot 3!} - \frac{(\kappa_4)^2}{2 \cdot 4!} + \frac{5}{8}(\kappa_3)^2\kappa_4 + \frac{1}{16}(\kappa_4)^3. \tag{23}$$

This expansion suggests natural bias constraints on projected distributions.

Recently, Intrator (1990) has shown that a BCM neuron can find structure in the input distribution that exhibits deviation from a Gaussian distribution in the form of multi-modality in the projected distributions. (BCM stands for Bienenstock, Cooper and Munro (Bienenstock et al., 1982). It is a learning rule that was constructed to model early visual cortical plasticity. Current versions of this rule, including mathematical properties, statistical motivation and network extensions, are discussed in (Intrator and Cooper, 1992).) Since clusters cannot be found in the data directly, because of its sparsity, this type of deviation, which is measured by the first three moments of the distribution, is particularly useful for finding clusters in high-dimensional data, and is thus useful for classification or recognition tasks. It thus renders this feature extraction technique also appropriate for bias constraints. We present some of the constraints here in a form that exhibits their relation to the BCM feature extraction rule. Connections with independent components analysis and results on natural scene feature extraction are described in (Blais et al., 1998). Results on classification of faces are described in (Intrator et al., 1996; Stainvas et al., 1997).

We are interested in finding projections that indicate interesting structure. Furthermore, we are not interested in deviation in the form of asymmetry, but more in deviations in the tails or in manifestations of multimodality. We also want to avoid sensitivity to outliers and therefore we use a rectified activation function (this rectification has little effect on the ability of kurtosis rules to find interesting projections (Blais et al., 1998)) denoted by $c = \sigma(\mathbf{d} \cdot \mathbf{m})$ and assume that the sigmoid σ is a smooth monotone function with a positive output, although a slight negative output is also allowed; σ' denotes the derivative of the sigmoid. The rectification is required for all rules that depend on odd moments, because these vanish in a symmetric distribution such as is commonly used in contexts involving natural scenes (Blais et al., 1998).

Skewness 1 This measures deviation from symmetry (Kendall and Stuart, 1977) and is of the form

$$S_1 = E[c^3]/(E[c^2])^{3/2}. \tag{24}$$

Maximisation of this measure via gradient ascent uses

$$\nabla S_1 = \frac{1}{{\Theta_M}^{1.5}} E\left[c\left(c - E[c^3]/E[c^2]\right)\sigma'\mathbf{d}\right], \tag{25}$$

where Θ_M is defined as $E[c^2]$.

Skewness 2 A similar measure which requires some stabilisation mechanism is given by

$$S_2 = E[c^3] - E^{3/2}[c^2]. \tag{26}$$

This measure has a gradient of the form

$$\nabla S_2 = 3E\left[c\left(c - \sqrt{\Theta_M}\right)\sigma'\mathbf{d}\right], \tag{27}$$

Kurtosis 1 Kurtosis measures deviation from a Gaussian distribution mainly in the tails of the distribution. It has the form

$$K_1 = E[c^4]/E^2[c^2] - 3. \tag{28}$$

This measure has a gradient of the form

$$\nabla K_1 = \frac{1}{{\Theta_M}^2} E\left[c\left(c^2 - E[c^4]/E[c^2]\right)\sigma'\mathbf{d}\right]. \tag{29}$$

Kurtosis 2 As before, there is a similar form which requires some stabilisation:

$$K_2 = E[c^4] - 3E^2[c^2]. \tag{30}$$

This measure has a gradient of the form

$$\nabla K_2 = 4E\left[c(c^2 - 3\Theta_M)]\sigma'\mathbf{d}\right]. \tag{31}$$

With all the above, maximisation of the measure can be used as a goal for projection seeking, so the variable c can be thought of as a (nonlinear) projection of the input distribution on to a certain vector of weights, and the maximization then defines a learning rule for this vector of weights. Under this framework, it is easy to stabilise the above learning rules by requiring for example that the vector of weights, which we denote by m, has a fixed norm, $\| m \| = 1$, say. The multiplicative forms of both kurtosis and skewness do not require this type of stabilisation, because of the normalising factor $1/{\Theta_M}^p$ in each rule.

Quadratic BCM The Quadratic BCM (QBCM) measure as given in (Intrator and Cooper, 1992) is of the form

$$\mathrm{QBCM} = \frac{1}{3}E[c^3] - \frac{1}{4}E^2[c^2]. \tag{32}$$

Maximising this form using gradient ascent uses the gradient function

$$\nabla\mathrm{QBCM} = E[c(c - \Theta_M)\sigma'\mathbf{d}]. \tag{33}$$

The Quadratic BCM rule does not require any additional stabilisation. This turns out to be an important property, since additional information can then be transmitted using the resulting norm of the weight vector m (Intrator, 1996).

Applicability of such bias constraints has been demonstrated in various real-world applications. Entropy constraints have been used in image compression, with a penalty aimed at minimizing the entropy of the projected distributions (Bichsel and Seitz, 1989). BCM constraints have been used in face recognition (Intrator et al., 1996) and more recently for partially occluded and blurred face recognition (Stainvas et al., 1997). These constraints are also useful for acoustic signal classification from wavelet representations (Huynh et al., 1998). Kurtosis constraints have been used in conjunction with reconstruction to find sparse representations (Olshausen and Field, 1996), and recently kurtosis and skewness have been found to be useful for neg-entropy calculations and independent components analysis; see (Yang and Amari, 1997) for a review.

6 Summary

Our aim in this chapter has been to suggest several ingredients which may be crucial for multi-parameter model estimation. As computational power and database availablity increase, real-world problems like these will be encountered increasingly often, and methods such as these can lead to significant improvement in their treatment.

References

Amari, S., Cichocki, A., and Yang, H. H. (1996). A new learning algorithm for blind signal separation. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 757–763. MIT Press.

Baum, E. and Lang, K. (1991). Constructing hidden units using examples and queries. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems*, volume 3, pages 904–910. Morgan Kaufmann, San Mateo, CA.

Bichsel, M. and Seitz, P. (1989). Minimum class entropy: A maximum information approach to layered netowrks. *Neural Networks*, 2:133–141.

Bienenstock, E. L., Cooper, L. N., and Munro, P. W. (1982). Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *Journal Neuroscience*, 2:32–48.

Bishop, C. M. (1995). Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1):108–116.

Blais, B. S., Intrator, N., Shouval, H., and Cooper, L. N. (1998). Receptive field formation in natural scene environments: comparison of single cell learning rules. *Neural Computation*, 10(7):1797–1813.

Breiman, L. (1993). Stacked regression. Technical Report TR-367, Department of Statistics, University of California, Berkeley.

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

Bridle, J. S. and MacKay, D. J. C. (1992). Unsupervised classifiers, mutual information and 'Phantom Targets'. In Moody, J., Hanson, S., and Lippmann, R., editors, *Advances in Neural Information Processing Systems*, volume 4, pages 1096–1101. Morgan Kaufmann, San Mateo, CA.

Buckheit, J. and Donoho, D. L. (1995). Improved linear discrimination using time-frequency dictionaries. Technical Report, Stanford University.

Deffuant, G. (1995). An algorithm for building regularized piecewise linear discrimination surfaces: The perceptron membrane. *Neural Computation*, 7(2):380–398.

Diaconis, P. and Freedman, D. (1984). Asymptotics of graphical projection pursuit. *Annals of Statistics*, 12:793–815.

Drucker, H., Cortes, C., Jackel, L., LeCun, Y., and Vapnik, V. (1994). Boosting and other ensemble methods. *Neural Computation*, 6:1289–1301.

Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap.* Chapman and Hall, New York.

Elman, J. L. and Zipser, D. (1988). Learning the hidden structure of speech. *Journal of the Acoustical Society of America*, 4(83):1615–1626.

Fahlman, S. E. (1988). Faster-learning variations on back-propagation: An empirical study. In Sejnowski, T. J., Hinton, G. E., and Touretzky, D. S., editors, *Connectionist Models Summer School.* Morgan Kaufmann, San Mateo, CA.

Fahlman, S. E. and Lebiere, C. (1990). The cascade–correlation learning architecture. CMU-CS-90-100, Carnegie Mellon University.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188.

Friedman, J. H. (1987). Exploratory projection pursuit. *Journal of the American Statistical Association*, 82:249–266.

Friedman, J. H. and Stuetzle, W. (1981). Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823.

Friedman, J. H. and Tukey, J. W. (1974). A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C(23):881–889.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias-variance dilemma. *Neural Computation*, 4:1–58.

Gutfinger, D. and Sklansky, J. (1991). Robust classifiers by mixed adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:552–567.

Hall, P. (1988). Estimating the direction in which data set is most interesting. *Probab. Theory Rel. Fields*, 80:51–78.

Hall, P. (1989). On projection pursuit regression. *The Annals of Statistics*, 17:573–588.

Hanley, J. A. and BJ, B. J. M. (1982). The meaning and use of the area under a reciever operating characteristic (ROC) curve. *Radiology*, 143:29–36.

Hansen, L. K., Liisberg, C., and Salamon, P. (1994). The error-reject tradeoff. *Can be obtained by FTP fron NeuroProse* (`archive.cis.ohio-state.edu`).

Hansen, L. K. and Salamon, P. (1990). Neural networks ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:993–1001.

Hastie, T. and Tibshirani, R. (1986). Generalized additive models. *Statistical Science*, 1:297–318.

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, pages 1–12. Hillsdale: Erlbaum.

Huber, P. J. (1985). Projection pursuit. (with discussion). *The Annals of Statistics*, 13:435–475.

Huynh, Q., Cooper, L. N., Intrator, N., and Shouval, H. (1998). Classification of underwater mammals using feature extraction based on time-frequency analysis and bcm theory. *IEEE-Signal Processing*, 46(5):1202–1207.

Intrator, N. (1990). A neural network for feature extraction. In Touretzky, D. S. and Lippmann, R. P., editors, *Advances in Neural Information Processing Systems*, volume 2, pages 719–726. Morgan Kaufmann, San Mateo, CA.

Intrator, N. (1993). Combining exploratory projection pursuit and projection pursuit regression with application to neural networks. *Neural Computation*, 5(3):443–455.

Intrator, N. (1996). Neuronal goals: Efficient coding and coincidence detection. In Amari, S., Xu, L., Chan, L. W., King, I., and Leung, K. S., editors, *Proceedings of ICONIP Hong Kong. Progress in Neural Information Processing*, volume 1, pages 29–34. Springer.

Intrator, N. and Cooper, L. N. (1992). Objective function formulation of the BCM theory of visual cortical plasticity: Statistical connections, stability conditions. *Neural Networks*, 5:3–17.

Intrator, N., Reisfeld, D., and Yeshurun, Y. (1996). Face recognition using a hybrid supervised/unsupervised neural network. *Pattern Recognition Letters*, 17:67–76.

Jacobs, R. A. (1995). Methods for combining experts' probability assessments. *Neural Computation*, 7:867–888.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

James, W. and Stein, C. (1960). Estimation with quadratic loss. In *Proc. Fourth Berkeley Symp. Math. Stat. Probab.*, volume 1, pages 361–380, Berkeley. University of California Press.

Jaynes, E. T. (1982). On the rationale of maximum entropy methods. *Proc. IEEE*, 70:939–952.

Jones, M. C. and Sibson, R. (1987). What is projection pursuit? (with discussion). *J. Roy. Statist. Soc.*, Ser. A(150):1–36.

Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214.

Kendall, M. and Stuart, A. (1977). *The Advanced Theory of Statistics*, volume 1. MacMillan Publishing, New York.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In Moody, J., Hanson, S., and Lippmann, R., editors, *Advances in Neural Information Processing Systems*, volume 4, pages 950–957. Morgan Kaufmann, San Mateo, CA.

Krogh, A. and Vedelsby, J. (1995). Neural network ensembles, cross validation, and active learning. *Advances in Neural Information Processing Systems 7.*

Kruskal, J. B. (1969). Toward a practical method which helps uncover the structure of the set of multivariate observations by finding the linear transformation which optimizes a new 'index of condensation'. In Milton, R. C. and Nelder, J. A., editors, *Statistical Computation*, pages 427–440. Academic Press, New York.

Kruskal, J. B. (1972). Linear transformation of multivariate data to reveal clustering. In Shepard, R. N., Romney, A. K., and Nerlove, S. B., editors, *Multidimensional Scaling: Theory and Application in the Behavioral Sciences, I, Theory*, pages 179–191. Seminar Press, New York and London.

Lang, K. J. and Witbrock, M. J. (1988). Learning to tell two spirals apart. In Touretzky, D. S., Ellman, J. L., Sejnowski, T. J., and Hinton, G. E., editors, *Proceedings of the 1988 Connectionists Models*, pages 52–59.

Le Cun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989). Back-propagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551.

Le Cun, Y., Denker, J., and Solla, S. (1990). Optimal brain damage. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 2, pages 598–605, San Mateo. (Denver 1989), Morgan Kaufmann.

LeBlanc, M. and Tibshirani, R. (1993). Combining estimates in regression and classification. *Can be obtained by FTP fron NeuroProse* (`archive.cis.ohio-state.edu`).

Meir, R. (1994). Bias, variance and the combination of estimators: The case of linear least squares. ftp://archive.cis.ohio-state.edu/pub/neuroprose/meir.bias-variance.ps.Z.

Murphy, P. M. and Aha, D. W. (1992). UCI Repository of machine learning databases. Department of Information and Computer Science. University of California at Irvine. `anonymous ftp from ics.uci.edu:/usr2/spool/ftp/pub/machine-learning-databases`.

Naftaly, U., Intrator, N., and Horn, D. (1997). Optimal ensemble averaging of neural networks. *Network*, 8(3):283–296.

Nowlan, S. J. and Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4:473–493.

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609.

Perrone, M. P. (1993). *Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization.* PhD thesis, Brown University, Institute for Brain and Neural Systems.

Perrone, M. P. and Cooper, L. N. (1993). When networks disagree: Ensemble method for neural networks. In Mammone, R. J., editor, *Neural Networks for Speech and Image processing*. Chapman-Hall.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *IEEE Proceedings*, 78(9):1481–1497.

Raviv, Y. and Intrator, N. (1996). Bootstrapping with noise: An effective regularization technique. *Connection Science, Special issue on Combining Estimators*, 8:356–372.

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Oxford Press.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press, Cambridge, MA.

Schapire, R. (1990). The strength of weak learnability. *Machine Learning*, 5:197–227.

Shimshoni, Y. (1995). Classification of seismic signals using ensembles of neural networks. M.Sc. thesis, Tel-Aviv University, School of Mathematical Sciences. `anonymous ftp from: ftp.math.tau.ac.il (pub/shimsh/thesis.ps.Z)`.

Shimshoni, Y. and Intrator, N. (1998). Classifying seismic signals by integrating ensembles of neural networks. *IEEE-Signal Processing*, 46(5):1194–1201. `ftp://cns.brown.edu/nin/papers/p.ps.Z`.

Sietsma, J. and Dow, R. J. F. (1991). Creating artificial neural networks that generalize. *Neural Networks*, 4:67–79.

Sollich, P. and Krogh, A. (1996). Learning with ensembles: How over-fitting can be useful. *Advances in Neural Information Processing Systems 8*. To Appear.

Stainvas, I., Intrator, N., and Moshaiov, A. (1997). Improving recognition via reconstruction. Preprint (ftp://cns.brown.edu/nin/papers/tradenips.ps.Z).

Stuart, A. and Ord, J. K. (1994). *Kendall's Advanced Theory of Statistics*. Edward Arnold.

Switzer, P. (1970). Numerical classification. In Barnett, V., editor, *Geostatistics*. Plenum Press, New York.

Taniguchi, M. and Tresp, V. (1997). Averaging regularized estimators. *Neural Computation*, 9:1163–1178.

Tresp, V. and Taniguchi, M. (1995). Combining estimators using non-constant weighting functions. *Advances in Neural Information Processing Systems 7*.

Viola, P. and Wells, W. M. (1995). Alignment by maximization of mutual information. In *Proceedings of the 1st International Conference on Computer Vision*.

Wahba, G. (1990). *Splines Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on ASSP*, 37:328–339.

Wand, M. P. (1994). Fast computation of multivariate kernel estimators. *Journal of Computational and Graphical Statistics*, 3:433–445.

Weigend, A. S., Huberman, B. A., and Rumelhart, D. (1990). Predicting the future: A connectionist approach. *Int. J. Neural Syst*, 1:193–209.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5:241–259.

Yamac, M. (1969). Can we do better by combining 'supervised' and 'nonsupervised' machine learning for pattern analysis. Ph.D. dissertation, Brown University.

Yang, H. and Amari, S. (1997). Adaptive on-line learning algorithms for blind separation – maximum entropy and minimum mutual information. *To appear in Neural Computation*.