Computer Assisted Mechanics and Engineering Sciences, 14: 345–352, 2007.

HPC strength prediction using Bayesian neural networks

Marek Słoński
Cracow University of Technology, Institute for Computational Civil Engineering, ul. Warszawska 24, 31-155 Kraków, Poland

(Received in the final form April 23, 2007)

The objective of this paper is to investigate the efficiency of nonlinear Bayesian regression for modelling and predicting strength properties of high–performance concrete (HPC). A multilayer perceptron neural network (MLP) model is used. Two statistical approaches to learning and prediction for MLP based on the likelihood function maximization and Bayesian inference are applied and compared. Results of experimental data sets show that Bayesian approach for MLP offers some advantages over classical one.

Keywords: Bayesian inference, regression, high-performance concrete, neural network

1. Introduction

Concrete is a heterogeneous material the properties of which depend on the properties of its components. Forster defines in [5] a high–performance concrete (HPC) as "a concrete made with appropriate materials (superplasticizer, retarder, fly ash, blast furnace slag and silica fume) combined according to a selected mix design and properly mixed, transported, placed, consolidated, and cured to give excellent performance in some properties of concrete, such as high compressive strength, high density, low permeability, and good resistance to certain forms of attack". Due to the high variability of effects of material properties, mix designs and manufacturing technologies creating HPC properties, concrete mix designing methods are mainly based on laboratory experiments and previous knowledge and technology.

Feed-forward neural networks (FFNNs) are used to perform non-linear functional mappings between a set of input variables and an output scalar variable. FFNNs have successfully been applied to the analysis of many problems in civil and structural engineering [14]. Bayesian techniques have been widely used in prediction problems and can be incorporated into the neural network approach. Bayesian methods for neural networks were introduced, among others mainly by MacKay and Neal [8, 11].

Bayesian neural networks (BNNs) have proved to be an efficient tool for the analysis of regression problems. In paper [7] BNN model was applied to prediction of concrete properties and the Bayesian approach gave better results than alternative non-Bayesian methods in the case problem. A similar problem of predicting concrete fatigue failure with BNN was described in paper [13]. Bayesian neural network was also used to prediction of deformed and annealed microstructures [1], in fault identification in cylinders using vibration data [9] and for prediction of response spectra [15].

In the paper HPC was assumed to be a mixture of six components. Data on about 340 mixes were taken from various publications, and collected and described in [6] by Kasperkiewicz, Racz and Dubrawski.

2. Bayesian neural networks for regression

Nonlinear parametric models such as multilayer perceptron (MLP) and radial basis function (RBF) neural network models are efficient tools for the empirical modelling of relationships for multivariable data sets [2]. The data set consists of a subset of input variables and a subset of corresponding target variables. It is assumed for regression problems that the target variable is related to an unknown model's prediction by the following equation,

$$t_n = y(\mathbf{x}_n;\mathbf{w}) + \epsilon_n\,, \tag{1}$$

where t_n is a scalar target value, $y(\mathbf{x}_n;\mathbf{w})$ is a regression function for an input vector $\mathbf{x}_n$ and a vector of model parameters $\mathbf{w}$. The last term in Eq. (1), ϵ_n , is a noise component.

It is assumed that the unknown function, represented by a regression model, underlies the process which has generated the data set. The task is to infer the function $f(\mathbf{x}_n;\mathbf{w})$ from the given data and make a prediction for a new input vector $\mathbf{x}$.

MLP neural network model with one hidden layer of adaptive units (neurons) is used, which can be expressed as the following equation,

$$y(\mathbf{x};\mathbf{w}) = f\left(\sum_{j=1}^{H} w_j g\left(\sum_{i=1}^{d} w_{ji}x_i + w_{j0}\right) + w_0\right), \tag{2}$$

where d is the number of inputs, H is the number of neurons in the hidden layer. Function $g(\cdot)$ is a nonlinear, sigmoidal activation function of hidden units, function $f(\cdot)$ is a linear activation function of an output neuron and $\mathbf{w}$ is the vector of parameters of the MLP model [2].

In the standard, maximum likelihood (ML) approach, the unknown parameters of MLP model are learned (estimated) by minimizing an error function

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left[t_n - y(\mathbf{x}_n;\mathbf{w})\right]^2, \tag{3}$$

for a given training data set $\mathcal{D} = \{(\mathbf{x}_1,t_1), (\mathbf{x}_2,t_2), \ldots, (\mathbf{x}_n,t_n), \ldots, (\mathbf{x}_N,t_N)\}$.

For small training data sets, the error function minimizing can lead to over-fitting of the training data and unsatisfactory prediction for a testing data set. The standard approach to the problem of poor generalization is related to the modified procedure by adding a regularization term to the error function (4),

$$E(\mathbf{w}) = E_D(\mathbf{w}) + \lambda\frac{1}{2}\sum_{j=1}^{M} w_j^2\,, \tag{4}$$

where M is the number of parameters and the hyperparameter λ controls the smoothness of the estimated relationship.

The Bayesian approach to neural networks learning and prediction processes is based on the Bayesian inference and integration of the parameters of MLP model instead of searching for a single vector of the parameters [2, 3]. In this approach all weights are treated as random variables. The prior distribution $p(\mathbf{w}\,|\,\alpha)$ over weights is firstly defined. It formalizes beliefs about the true parameters before observing the data. It is generally assumed that the prior distribution is a spherical Gaussian distribution with the zero mean and inverse variance (precision) hyperparameter α. A Gaussian noise model $p(\epsilon_n\,|\,\beta)$ is also adopted with the inverse of variance parameter called precision and defined by $\beta = 1/\sigma^2$. The noise model describes how the output variable is corrupted by the noise process. A detailed description of the Bayesian approach to neural networks can be found in [2].

After observing the data set $\mathcal{D}$, Bayes' theorem is used to update the beliefs and the posterior probability distribution over weights $p(\mathbf{w}\,|\,\mathbf{t},\alpha,\beta)$ is computed

$$p(\mathbf{w}\,|\,\mathbf{t},\alpha,\beta) = \frac{p(\mathbf{t}\,|\,\mathbf{w},\beta)\,p(\mathbf{w}\,|\,\alpha)}{p(\mathbf{t}\,|\,\alpha,\beta)}\,, \tag{5}$$

where $p(\mathbf{t}\,|\,\mathbf{w},\beta)$ is the likelihood function, which for independent and identically distributed (i.i.d.) data set is defined as

$$p(\mathbf{t}\,|\,\mathbf{w},\beta) = \prod_{n=1}^{N} p(t_n\,|\,\mathbf{w},\beta) = \prod_{n=1}^{N} (2\pi\sigma^2)^{-1/2} \exp\left[-\frac{\{t_n - y(\mathbf{x}_n;\mathbf{w})\}^2}{2\sigma^2}\right], \tag{6}$$

where $\sigma^2 = 1/\beta$ and $p(\mathbf{t}\,|\,\alpha,\beta)$ is a normalizing factor of the form

$$p(\mathbf{t}\,|\,\alpha,\beta) = \int p(\mathbf{t}\,|\,\mathbf{w},\beta)\,p(\mathbf{w}\,|\,\alpha)\,\mathrm{d}\mathbf{w}. \tag{7}$$

For regression problems, what is of main interest is making the prediction of t_{N+1} for a new value of $\mathbf{x}_{N+1}$. In the non-Bayesian approach with regularization, a point prediction is given by the model output $y(\mathbf{x}_{N+1}\,;\mathbf{w}_{PLS})$, Eq. (2), using the point estimate of neural network parameters $\mathbf{w}_{PLS}$. In Bayesian approach, instead of point prediction, the predictive distribution over target variable t_{N+1} is computed applying the sum rule of probability and marginalizing out the weight vector $\mathbf{w}$. This leads to the following equation

$$p(t_{N+1}\,|\,\mathbf{x}_{N+1},\mathbf{t},\alpha,\beta) = \int p(t_{N+1}\,|\,\mathbf{x}_{N+1},\mathbf{w},\beta)\,p(\mathbf{w}\,|\,\mathbf{t},\alpha,\beta)\,\mathrm{d}\mathbf{w}, \tag{8}$$

assuming that the hyperparameters α and β are known.

In the fully Bayesian framework, also the uncertainty over the hyperparameters should be taken into account by defining the prior distributions $p(\alpha)$ and $p(\beta)$ which are called hyperpriors. Then the full posterior distribution is defined by

$$p(\mathbf{w},\alpha,\beta\,|\,\mathbf{t}) = \frac{p(\mathbf{t}\,|\,\mathbf{w},\beta)\,p(\mathbf{w}\,|\,\alpha)\,p(\alpha)\,p(\beta)}{p(\mathbf{t})}, \tag{9}$$

where the denominator (normalizing term) is

$$p(\mathbf{t}) = \int p(\mathbf{t}\,|\,\mathbf{w},\beta)\,p(\mathbf{w}\,|\,\alpha)\,p(\alpha)\,p(\beta)\,\mathrm{d}\mathbf{w}\,\mathrm{d}\alpha\,\mathrm{d}\beta. \tag{10}$$

Finally, the fully Bayesian prediction is computed evaluating the predictive distribution of the form

$$p(t_{N+1}\,|\,\mathbf{x}_{N+1}\,,\mathbf{t}) = \int p(t_{N+1}\,|\,\mathbf{x}_{N+1}\,,\,\mathbf{w},\beta)\,p(\mathbf{w},\alpha,\beta\,|\,\mathbf{t})\,\mathrm{d}\mathbf{w}\,\mathrm{d}\alpha\,\mathrm{d}\beta, \tag{11}$$

using the posterior distribution $p(\mathbf{w},\alpha,\beta\,|\,\mathbf{t})$.

Because the Bayesian prediction is based on integrations in the parameter and hyperparameter space over all parameters and hyperparameters, in general it is analytically intractable and the approximation techniques have to be used. These methods are in general divided into two groups: deterministic and stochastic. The integration can be performed using a local Gaussian approximation to the posterior distribution, which method is known as the Laplace approximation. Also, more general variational methods called variational inference or variational Bayes can be used. They are based on minimizing the Kullback–Leibler divergence which is interpreted as a measure of the dissimilarity of two distributions. These methods are described in the textbooks [3, 8].

In the paper a stochastic method based on hybrid Monte Carlo algorithms developed for Bayesian neural networks by Neal [11, 12] is applied. In the Monte Carlo approach the integrals are approximated by the finite sums of the form

$$\int p(t\,|\,\mathbf{x},\mathbf{w})\,p(\mathbf{w}\,|\,\mathbf{t})\,\mathrm{d}\mathbf{w} \approx \frac{1}{m}\sum_{i=1}^{m} p(t\,|\,\mathbf{x},\mathbf{w}_i), \tag{12}$$

where $\mathbf{w}_i$ are samples of weight vectors generated from distribution $p(\mathbf{w}\,|\,\mathbf{t})$.

3. The HPC data base

Data on about 340 mixes were taken from various publications collected in [6]. The input vector $\boldsymbol{x}$ consists of six variables defining the amount of cement (C), water (W), silica (S), superplasticizer (Su), fine aggregate (FA), coarse aggregate (CA), all in kg/m^3. The output variable t is the 28-day compressive strength f_c', in MPa.

Table 1 shows the input and output variables details (range, median, mean and standard deviation of data set used in modelling the relation). The corresponding Fig. 1 represents graphically statistical properties of the variables with the aid of the box and whisker plot. The relationships between the input variables and output variable f_c' are also shown in scatterplots in Fig. 2.

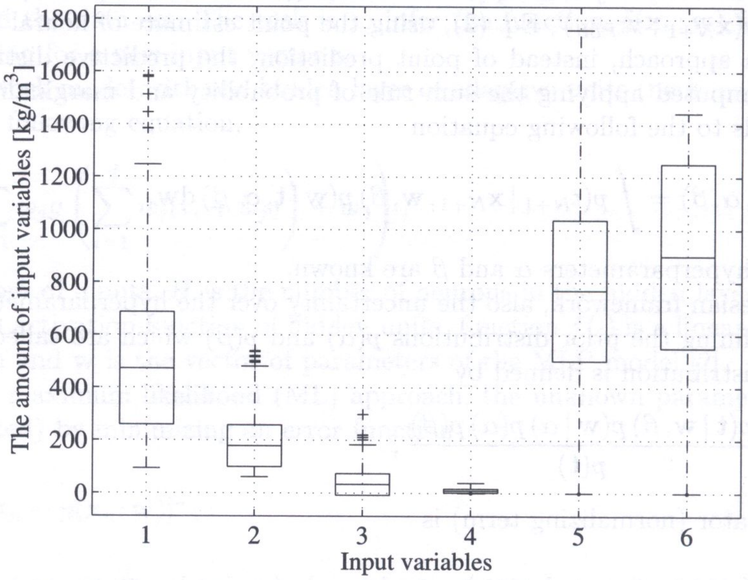

Fig. 1. Box and whisker plot for HPC data set

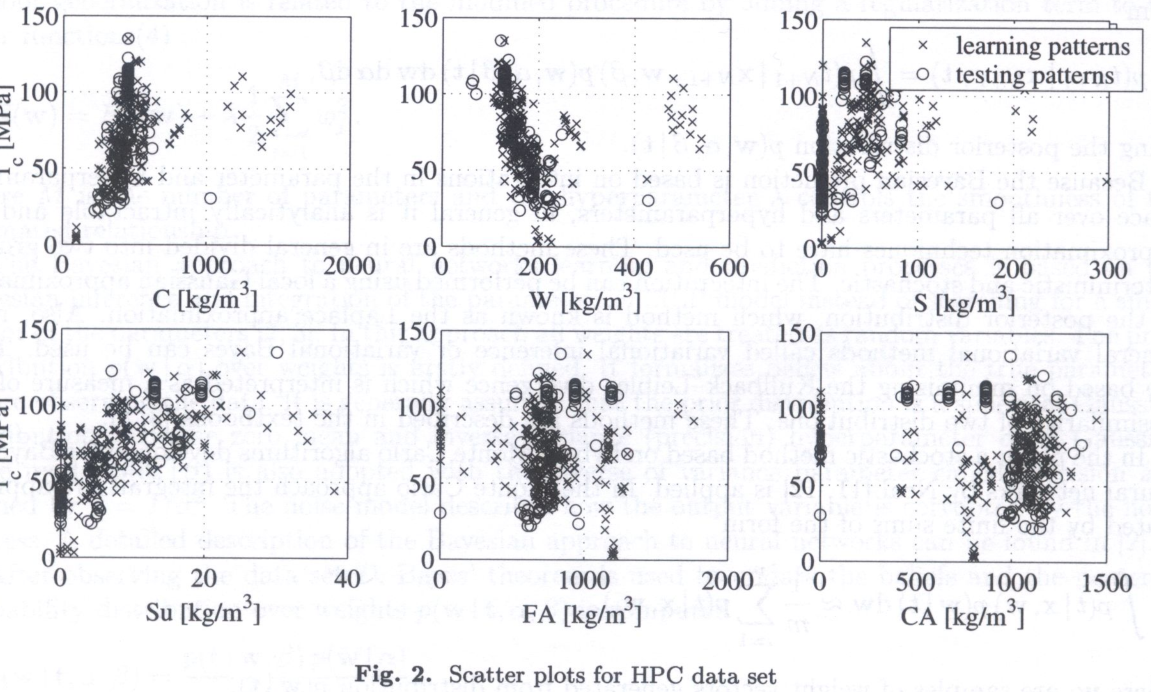

Fig. 2. Scatter plots for HPC data set

Table 1. Statistical properties of input and output variables

Number	Variable	Range	Median	Mean	St.Dev.
1	C [kg/m^3]	94–1585	449	473	214
2	W [kg/m^3]	61–540	160	178	79
3	S [kg/m^3]	0–298	22.5	32.8	41.2
4	Su [kg/m^3]	0–38.1	7.6	9.1	7.6
5	FA [kg/m^3]	0–1760	750	769	267
6	CA [kg/m^3]	0–1443	1038	900	350
7	f'_c [MPa]	2.8–135	71.9	72.2	26.8

4. HPC COMPRESSIVE STRENGTH PREDICTION

4.1. Empirical formula

The 28-day HPC compressive strength f'_c can be computed using de Larrard's empirical formula [4]

$$f'_c = \frac{K_k R_c}{\left(\frac{1+3.1W/C}{1.4-0.4\exp(-11S/C)}\right)^2} \tag{13}$$

where K_k is the coefficient which takes into account the effects of aggregate characteristics on concrete strength (the commonly assumed value is $K_k = 4.9$) and R_c is the actual 28-day compressive strength of cement in MPa. In Fig. 3 the 28-day HPC compressive strength f'_c as a function of W/C and S/C based on Larrard's formula is plotted.

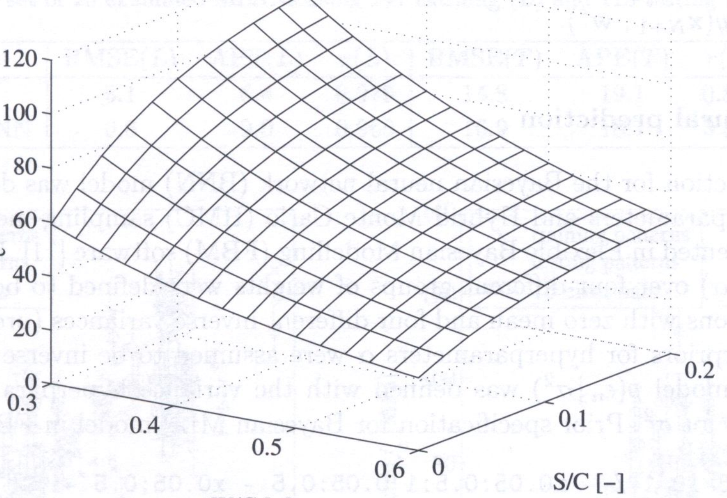

Fig. 3. The 28-day HPC compressive strength f'_c as a function of W/C and S/C based on Larrard's formula computed for $K_k = 4.9$ and $R_c = 45$ MPa

4.2. Neural prediction

The 28-day compressive strength f'_c was defined as a function of 6 variables i.e. amount of cement (C), water (W), silica (S), superplasticizer (Su), fine aggregate (FA) and coarse aggregate (CA). The problem of predicting the 28-day compressive strength f'_c was formulated as a mapping from the input vector $\boldsymbol{x}$ to the scalar output t.

After the preliminary experiments, a multilayer perceptron neural network (MLP) model with a single hidden layer of ten hyperbolic tangent units (neurons) and linear output unit was used to model the relationship between the inputs and the output,

$$y(\mathbf{x};\mathbf{w}) = \sum_{j=1}^{10} w_j g\left(\sum_{i=1}^{6} w_{ji} x_i + w_{j0}\right) + w_0 \,. \tag{14}$$

The experiments were performed using 346 examples (231 for training and 115 for testing) which are depicted in Fig. 2. Both the input and output variables were first standardized to zero mean and unit standard deviation by the following transformation,

$$\tilde{x}_i^n = \frac{x_i^n - \bar{x}_i}{s_i} \,, \tag{15}$$

where $\bar{x}_i$ is an average value and s_i is the standard deviation

$$\bar{x}_i = \frac{1}{N}\sum_{n=1}^{N} x_i^n \,, \qquad s_i = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x_i^n - \bar{x}_i)^2} \,. \tag{16}$$

Two statistical approaches to learning and prediction for MLP were used. The first approach was maximum likelihood (ML) and the second one was the Bayesian inference with stochastic approximation based on Hybrid Monte Carlo (HMC).

Maximum Likelihood neural network (MLNN) model was trained using scaled conjugate gradient method implemented in MATLAB Netlab Toolbox [10]. The network weights were initially randomized from a Gaussian distribution with the zero mean and unit variance. The total number of 100 iterations of the optimization algorithm (epochs) were used. After learning the "optimal" weights vector $\mathbf{w}^*$ was applied to the prediction for a new input vector $\mathbf{x}_{N+1}$ which for MLNN is a point prediction $y(\mathbf{x}_{N+1}\,,\,\mathbf{w}^*)$.

4.3. Bayesian neural prediction

Learning and prediction for the Bayesian neural network (BNN) model was done using Gibbs sampling for the hyperparameters and Hybrid Monte Carlo (HMC) sampling method for the weights, which were implemented in Flexible Bayesian Modelling (FBM) software [11]. In the paper four prior distributions $p(\mathbf{w}\,|\,\alpha)$ over four different groups of weights were defined to be spherical (isotropic) Gaussian distributions with zero mean and four different inverse variances (precisions) hyperparameters α. Also hyperpriors for hyperparameters α were assumed to be inverse Gamma hyperpriors. The normal noise model $p(\epsilon_n\,|\,\sigma^2)$ was defined with the variance hyperparameter σ^2 and inverse Gamma hyperprior for σ^2. Prior specification for Bayesian MLP model in FBM notation was

```
net-spec log 6 10 1 / - x0.05:0.5:1 0.05:0.5 - x0.05:0.5 -1
model-spec log real 0.05:0.5
```

The main HMC parameters had the following values: length of individual chains was 50, step size was 0.5 and persistence parameter was 0.9. The burn-in stage contained 30 iterations and the actual sampling 240 iterations from which 20 samples of weights were stored for the final HPC compressive strength predictions.

5. Results

In this section the Bayesian neural network (BNN) and Maximum Likelihood (MLNN) neural network models are compared on the base of the following error formulae:

– root-mean-squared (RMS) error

$$RMS = \sqrt{\frac{1}{V}\sum_{n=1}^{V}(t_n - y_n)^2}\,, \tag{17}$$

– average percentage (AP) error

$$AP = \frac{1}{V}\sum_{n=1}^{V}\left|\frac{t_n - y_n}{t_n}\right| \cdot 100\%. \tag{18}$$

Also the coefficient of correlation r was computed,

$$r = \frac{\sum_{n=1}^{V}(t_n - \bar{t})(y_n - \bar{y})}{\sum_{n=1}^{V}(t_n - \bar{t})\ \sum_{n=1}^{V}(y_n - \bar{y})}\,, \tag{19}$$

where $\bar{t}$ and $\bar{y}$ are mean values of targets t_n and predicted values y_n, respectively.

The computed errors and the coefficient of correlation for learning (L) and testing (T) patterns are presented in Table 2. For BNN model the errors are evaluated using the predictions of 20 NNs samples generated from the posterior distribution approximated using Hybrid Monte Carlo method. For MLNN model, the errors presented in Table 2 are computed using the best Maximum Likelihood neural network (bMLNN) taken from the set of 20 MLNNs with different initial weights values generated from the Gaussian distribution with zero mean and unit variance. The best MLNN was taken with respect to the minimum testing error.

Table 2. Comparison of generalization performance in predicting HPC compressive strength for Bayesian neural network (BNN) based on 20 NNs samples and the best Maximum Likelihood neural network (bMLNN) taken from the set of 20 examined MLNNs using 241 learning (L) and 115 testing (T) patterns

Model	RMSE(L)	APE(L)	$r(L)$	RMSE(T)	APE(T)	$r(T)$
BNN	5.1	6.4	0.976	14.8	19.1	0.887
bMLNN	6.6	9.0	0.960	16.9	18.1	0.848

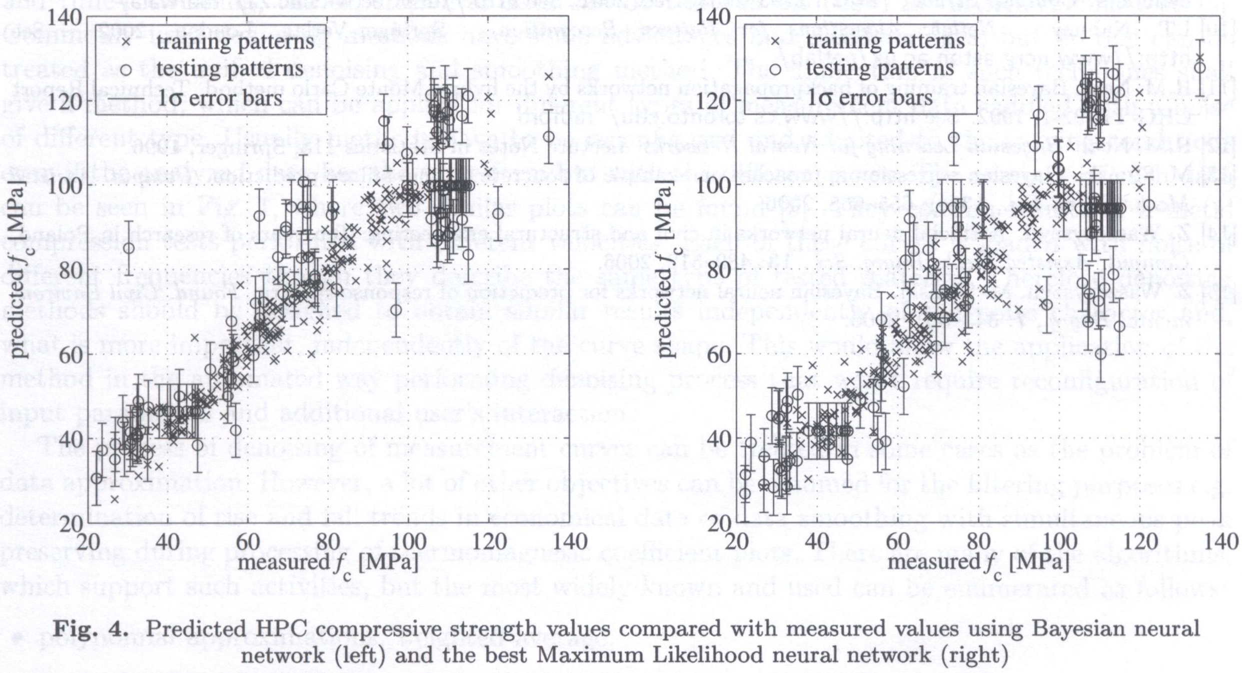

Fig. 4. Predicted HPC compressive strength values compared with measured values using Bayesian neural network (left) and the best Maximum Likelihood neural network (right)

In Fig. 4 the measured HPC compressive strength values with the predicted values are plotted for both learning and testing patterns. For testing patterns also 1σ error bars are depicted, corresponding to the estimated standard deviation of the assumed Gaussian noise model, which are $\sigma_{\mathrm{BNN}} = 6.4\,\mathrm{MPa}$ and $\sigma_{\mathrm{bMLNN}} = 6.6\,\mathrm{MPa}$. For MLNN model this estimated standard deviation is equal to the learning RMS error.

6. Conclusions

Some conclusions from the results of the experiments can be drawn. The Bayesian approach applied to learning and prediction for MLP neural network has better generalization performance than the classical approach based on the Maximum Likelihood (ML) principle. The ML approach is not able to control sufficiently the complexity of the neural model compared with the BNN model.

The results have confirmed the feasibility of using BNN to model the unknown relationship between the input and output variables exploring only experimental data. The great uncertainty in the prediction is due to the intrinsic noise in the training data. Smaller uncertainties can be achieved through the use of a larger and more accurate data set.

References

[1] C.A.L. Bailer-Jones, T.J. Sabin, D.J.C. MacKay, P.J. Withers. Prediction of deformed and annealed microstructures using Bayesian neural networks and Gaussian processes. In: *Proc. of the Australia-Pacific Forum on Intelligent Processing and Manufacturing of Materials*. 1997. See http://www.mpia-hd.mpg.de/homes/calj/

[2] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1995.

[3] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[4] F. de Larrard. *Concrete Mixture Proportioning — A scientific approach*. E&FN SPON, London, 1999.

[5] S.W. Forster. High-performance concrete-stretching the paradigm. *Concrete International*, **16**(10): 33–34, Oct. 1994.

[6] J. Kasperkiewicz, J. Racz, A. Dubrawski. HPC strength prediction using artificial neural network. *J. Comp. Civil Engrg.*, **9**(4): 1–6, Oct. 1995.

[7] J. Lampinen, A. Vehtari. Bayesian approach for neural networks — review and case studies. *Neural Networks*, **14**(3): 7–24, Apr. 2001. See http://www.lce.hut.fi/~ave/

[8] D.J.C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[9] T. Marwala. Scaled conjugate gradient and Bayesian training of neural networks for fault identification in cylinders. *Comput. Struct.*, **79**(32): 2793–2803, Dec. 2001. See http://dept.ee.wits.ac.za/~marwala/

[10] I.T. Nabney. *Netlab: Algorithms for Pattern Recognition*. Springer-Verlag, London, 2002. See http://www.ncrg.aston.ac.uk/netlab/

[11] R.M. Neal. Bayesian training of backpropagation networks by the hybrid Monte Carlo method. Technical Report CRG-TR-92-1, 1992. See http://www.cs.toronto.edu/~radford

[12] R.M. Neal. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics 118, Springer, 1996.

[13] M. Słoński. Bayesian regression approaches on example of concrete fatigue failure prediction. *Comput. Assisted Mech. Engrg. Sci.*, **13**(4): 655–668, 2006.

[14] Z. Waszczyszyn. Artificial neural networks in civil and structural engineering: Ten years of research in Poland. *Comput. Assisted Mech. Engrg. Sci.*, **13**: 489–512, 2006.

[15] Z. Waszczyszyn, M. Słoński. Bayesian neural networks for prediction of response spectra. *Found. Civil Environmental Engrg.*, **7**: 343–361, 2006.