Indian Journal of Engineering & Materials Sciences
Vol. 12, April 2005, pp. 117-121

A study of applying nested hyper-rectangle learning model to concrete strength estimation

Li Chen
Department of Civil Engineering, Chung Hua University, 707, Sector 2, WuFu Road, Hsin Chu, Taiwan 300 67, Republic of China

Received 11 November 2003; accepted 3 January 2005

The aim of this paper is to demonstrate an exemplar-based nested hyper-rectangle learning model (NHLM) and apply it to estimate the strength of high performance concrete (HPC). The proposed model is based on the concept of seeding training data in the Euclidean *i*-space (where *i* denotes the number of features) as hyper-rectangles. The well-designed structures and one-shot learning procedures of NHLM could adjust learning abilities efficiently when new examples are added. HPC is a highly complex material, which makes modelling its behaviour a very difficult task. Compared with a standard back-propagation neural network (BPN), the experimental results indicate that NHLM provides a powerful tool for estimating the strength of HPC.

IPC Code: C04B14/00

Concrete has been the most widely used construction material all around the world. Traditionally, concrete is designed in a way following previous experiences. Nowadays, concrete can be made with about four to ten different components. The number of properties to be adjusted has also increased, so that empirical methods are no longer sufficient in concrete mix design[1]. In addition to the four basic ingredients in conventional concrete, i.e., Portland cement, fine and coarse aggregates, and water, the making of high performance concrete (HPC) needs to incorporate supplementary cementitious materials, such as fly ash and blast furnace slag, and chemical admixture, such as super-plasticizer. The earlier studies on this mixing material problems are accomplished by using relationships, established from experimental data, which can guide us to selecting the best combination of ingredients in achieving the desirable properties[2-4]. Several studies[5] independently have shown that concrete strength development is determined not only by the w/c ratio, but that it is also influenced by the content of other ingredients. Chen[6] applied macro-evolutionary genetic programming to concrete strength estimation. These works were focused on building the mathematic functions among concrete ingredients and strength.

The neural network (NN) is a recently popular non-parametric approach to HPC strength estimation. It can obtain the concrete strength via a complex structure, but does not express the relationships using functions. Most neural network applications are based on the back-propagation (BP) paradigm, which uses the gradient-descent method to minimize the error function. The architecture of neural network has been covered widely[7-9]. However, the network's speed of learning is often unacceptably slow, or its generalization capability is often unsatisfactorily low, for solving highly non-linear function mapping problems[10]. The basic drawback of NN is that the optimal configuration of the network is not known *a priori*[11]. Besides, BPN exhibits some serious drawbacks such as slow convergence in learning phase, the potential convergence to local minimum, the common chaotic behaviour, and the inability to detect over-fitting.

In this study, a nested hyper-rectangle learning model (NHLM) is used to process the experimental results of a series of HPC data. The NHLM is based on artificial intelligence, which is an alternative nonparametric method to HPC strength estimation. The NHLM is derived from a learning model that was proposed originally as a human learning model. A trained NHLM can approximate the experimental results of similar data. Due to its special data structures of axis parallel rectangle represented by two points on the diagonal of each rectangle, it requires limited computer memory. Unlike generalization processes using symbolic formulae,

E-mail: lichen@chu.edu.tw

such as the regression approach, the NHLM algorithm modifies hyper-rectangles by growing and reshaping them in a well-defined fashion.

Characteristics of NHLM

The exemplar-based learning (EBL) algorithm is one of the new trends in machine learning[12]. The strategy of the exemplar-based learning (EBL) algorithm is based on storing points (or examples) in Euclidean *i*-space, E^{i}, where *i* is the number of variables or features in an example, then comparing the new example to those, and finding the most similar example[13]. The algorithm is a very powerful tool for prediction tasks. The NHLM allows the points generalized into axis-parallel hyper-rectangles. As the generalizations grow large, there may exist some exceptions, which create "holes" in the hyper-rectangles. These may in turn give holes inside them, resulting in a nested structure of hyper-rectangles. Because of creating the holes, NHLM could have additional exceptions inside the hyper-rectangles, nested as deep as the data require.

Helmbold *et al.*[14] proved some very strong optimality results for their algorithm. Salzberg[15] used this model on three different domains: predicting the recurrence of breast cancer, classifying iris flowers, and predicting survival times for heart attack patients. In all cases, he demonstrated that NHLM performs as well as, or better than, other algorithms, which were run with the same data sets.

Learning Procedures

The main procedures including the seeding, matching and learning processes are shown in Fig. 1.

Seeding

For the sake of making predictions, NHLM must have a history of examples as the bases of its predictions.

Matching

Let the new example be *E* and the existing hyper-rectangle be *H*. The match score between *E* and all exemplars *H* stored in memory. It is calculated by measuring the Euclidean distance between the two objects. The distance metric is determined as follows:

$$D_{\mathrm{EH}} = W_{\mathrm{h}} \sqrt{\sum_{i=1}^{m} \left(W_{\mathrm{i}} \frac{E_{\mathrm{i}} - H_{\mathrm{i}}}{\max_{\mathrm{i}} - \min_{\mathrm{i}}} \right)^{2}} \qquad \dots (1)$$

where W_h is the weight of the exemplar *H*, W_i is the weight of the feature *i*, E_i is the value of the *i*th feature in example *E*, H_i is the value of the *i*th feature in exemplar *H*, mini, maxi are the minimum and maximum values of that feature and *m* is the number of features.

Let H_{Li} be the lower end of the range, and H_{Ui} be the upper end, then the distance metric becomes:

$$E_{\mathrm{i}} - H_{\mathrm{i}} = \begin{cases} E_{\mathrm{i}} - H_{\mathrm{Ui}} & \text{when } E_{\mathrm{i}} > H_{\mathrm{Ui}} \\ H_{\mathrm{Li}} - E_{\mathrm{i}} & \text{when } E_{\mathrm{i}} < H_{\mathrm{Li}} \\ 0 & \text{otherwise} \end{cases} \qquad \dots (2)$$

Figure 2 illustrates these different distance measurements.

Learning

There are two weights in the distance metric, W_h and W_i. W_h is a simple measure of how frequently the exemplar, *H*, has been used to make a correct

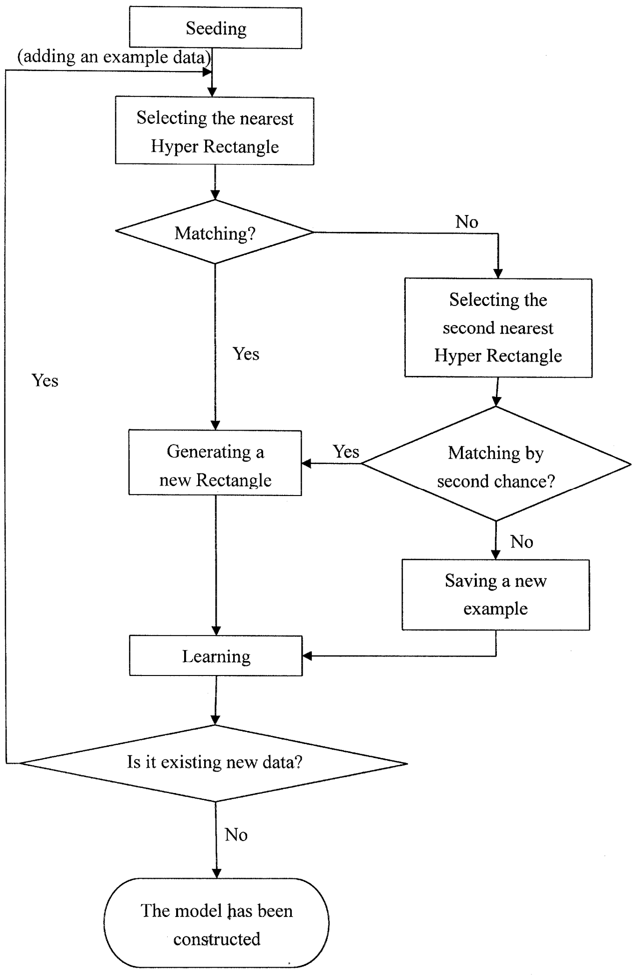
Seeding
(adding an example data)
Selecting the nearest Hyper Rectangle
Matching?
No
Selecting the second nearest Hyper Rectangle
Yes
Yes
Generating a new Rectangle
Yes
Matching by second chance?
No
Saving a new example
Learning
Is it existing new data?
No
The model has been constructed

Fig. 1— Learning procedures of NHLM

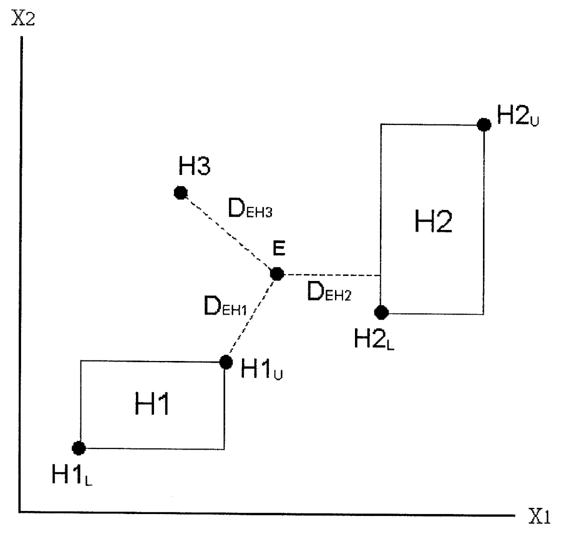

Fig. 2— Distance measurements of NHLM

prediction. W_h is the ratio of the number of times H has been used to the number of times it has resulted in a correct prediction. Consequently, it can indicate the reliability of each exemplar. Thus if a hyper-rectangle H is used many times, but nearly always makes the wrong prediction, the weight W_h will grow very large, and H will tend not to be chosen as the closest match in the future. If H is a noisy point, then it will eventually be ignored as W_h increases. The minimum value of W_h is unity in the case of perfect prediction, shown in Eq. (3).

$$W_h = \frac{N_t}{N_c} \qquad \ldots (3)$$

where N_t is the total number of times an exemplar has been used and N_c is the number of time an exemplar makes the correct prediction.

The other weight measure, W_i, is the weight of the i^{th} feature. These weights are adjusted to reflect the fact that all features do not normally have equal importance in a category decision. NHLM adjusts the weights W_i on the feature i after discovering that it has made the wrong prediction. Weight W_i adjustment is executed in a very simple loop: for each fi if E_{fi} matches H_{fi}, the weight W_i is decreased by setting $W_i = W_i(1-\Delta f)$, where Δf is the global feature adjustment rate. An increase in weight causes the two objects to seem farther apart; if E_{fi} does not match H_{fi}, then W_i is increased by setting $W_i = W_i(1+\Delta f)$. The Δf normally uses as 0.05. With higher values of Δf, accurate rates are lower.

Second chance

If the system makes the wrong prediction, it has one more chance to make the right one. This "second chance" heuristic is used by NHLM in order to avoid creating more memory objects than necessary. The idea is to try very hard to make a generalization and thus keep down the size of memory. So, before creating a new exemplar, NHLM first looks at the second best match in memory. Assume here that $H1$ was the closest exemplar to E (D_{EH1} was smallest) and $H2$ was second closest exemplar; i.e., $H2$ would be the closest if $H1$ were removed. If $H2$ will give the correct prediction, then the system tries to adjust hyper-rectangle shapes to make the second closest exemplar into the closest exemplar. It does this by creating a generalization from $H2$ and E. The goal of this process is to improve the predictive accuracy of the system without increasing the number of exemplars stored in memory. The training algorithms of NHLM are shown in Fig. 3.

Application for Proportioning Concrete Mixture

Data set

Nine features ($i = 9$) were chosen as input factors to building a multi-variables model, Eq. (4), and using it can predict the strength of HPC. It is a high-dimensional and difficult estimation problem.

$$Y = f(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9) \qquad \ldots (4)$$

Where X_1 is water (kg/m^3), X_2 is cement (kg/m^3), X_3 is coarse aggregate (kg/m^3), X_4 is fine aggregate (kg/m^3), X_5 is age of testing (days), X_6 is fly ash (kg/m^3), X_7 is blast furnace slag (kg/m^3), X_8 is superplasticizer (kg/m^3) and X_9 is water-to-binder ratio,

$$\frac{w}{b} = \frac{X_1 + X_8}{X_2 + X_6 + X_7}$$

Experimental data from different sources were used to check the reliability of the strength model[6,16]. First, a determination was made to ensure that these mixtures were a fairly representative group governing all of the major parameters that influence the strength of HPC and present the complete information required for such an evaluation. In all, 1140 concrete samples from the above investigations were evaluated. Table 1 presents the general details of the concrete evaluated in this study. The NHLM was applied to estimate the HPC strength. The same data were processed using a neural network (NN).

During training, input vectors $(x^{(1)}, y^{(1)}),\ldots.(x^{(n)}, y^{(n)})$; n is the number of input data and x includes m independent variables.

```
1) Seeding
Weights: wi = 1.0, i = 1…m
$H1_L = x^{(1)}$ and $H1_U = x^{(1)}$   //Creating a point rectangle//
$Y_{H1} = y^{(1)}$
$W_{h1} = 1.0$
c = 1                      // c is the current index of rectangle //
2) Matching
For i = 2 to n do
begin
    e1 = nearest of all H to $x^i$   //using Eq. (1) and (2)//
3) Second Chance
    e2 = second of all H to $x^i$   //using Eq. (1) and (2)//
    If (neighbor ($x^i$,e1))//neighbor means under the error tolerance//
    begin
        Expending e1 to e1 $\oplus x^i$
4) Learning
        Adjusting $Wh_{e1}$ by Eq. (3), correct times + 1
    End else
    If (neighbor ($x^i$, e2))
    Begin
        Expending e2 to e2 $\oplus x^i$
        Adjusting $Wh_{e1}$ by Eq. (3), incorrect
        For j = 1…m do
        begin
            If distance(e1j, xj) < distance (e2j, xj), Wj = Wj (1 - $\Delta$f)
            Else
            If distance(e1j, xj) > distance (e2j, xj), Wj = Wj (1 + $\Delta$f)
        end
    End else
    Begin
        c = c + 1
        $Hc_L = x^{(i)}$ and $Hc_U = x^{(i)}$   //Creating a point rectangle//
        $Y_{Hc} = Y^{(i)}$
        $Wh_c = 1.0$
    End
end;
```

Fig. 3—NHLM Training Algorithms

Table 1—Variables of HPC experimental data

Component	Average	Maximum	Minimum
Cement (kg/m^3)	299.2	897	71
Water (kg/m^3)	182.8	314	118
Coarse aggregate (kg/m^3)	964.3	1820	595
Fine aggregate (kg/m^3)	809.9	1300	387
Age of testing (days)	58.3	365	1
Water-to-binder ratio (w/b)	0.5	0.9	0.24
Superplasticizer (kg/m^3)	4.1	32	0
Fly ash (kg/m^3)	30.5	197	0
Blast furnace slag (kg/m^3)	69.1	359	0
Strength (psi)	5600.0	17691	296

NHLM

The error tolerance is set as $0.01\overline{Y}$; i.e., if the difference between the forecasting value and true value is within $0.01\overline{Y}$, it is considered as a "match" and is set in the same hyper-rectangle, where $\overline{Y}$ is the average of HPC strength (5600 psi). The procedure then is run with all the 760 data set to build up the nested hyper-rectangles in the E^9 domain.

After the training task is completed, the performance of NHLM is then estimated. It begins with randomly selected n (n=380) new sets of $(X_1\ldots X_9)$, then uses the set model, without modifying or changing the model structure or parameters, to predict Y values and compare the predicted values with true values of the HPC strength. In order to determine the effect of the number of training data, m, to the error of prediction, in each case, 10 sets of $(X_1\ldots X_9)$ were performed to match their Y values. Then, the root mean square errors (RMSE) were recorded by using the NHLM. The procedures were run 10 times for the purpose of statistical analysis.

Therefore, the 1140 data were thus randomly classified as follows. (i) Training data: several training data set's values of m = 100, 300, 500, 700 and 760. (ii) Testing data: n = 380 (had never used these output values in the training state).

Neural network (NN)

The values of parameters considered in the neural network approach are as follows: (i) number of hidden layers = 1; (ii) number of hidden units = 9; (iii) learning rate = 1.0; (iv) momentum factor = 0.5 ; and (v) learning cycles = 1000.

Results and Discussion

The RMSE of these two models at testing stage are demonstrated in Table 2. We found that as "m" increases, the mean value and the standard deviation

Table 2—Comparison of root mean square errors (RMSE) for 10 runs

The number of training data	Mean of RMSE using NHLM (psi)	Mean of RMSE using BPN (psi)
100	1509.4	2247.8
300	1237.6	1935.2
500	998.1	1771.6
700	882.7	1586.3
760	876.3	1492.5

of the RMSE will decrease by using NHLM. The results support the concept that when we have more experiences, we are more sophisticated. Obviously, the NHLM model is better than the BPN. The performance of HPC strength estimation using NHLM is more efficient and accurate than when using BPN. Moreover, the learning time of NHLM is significantly less than that of BPN, because NHLM is based on a one-shot learning process. In contrast, BPN requires considerable iteration. NHLM requires more memory than BPN, although the difference is minimal.

Conclusions

The nested hyper-rectangle learning model (NHLM) is applied to estimate the strength of high performance concrete (HPC) using nine input variables. NHLM performance improves with increasing training data and outperforms back-propagation network in estimating the concrete strength estimation with a lower root mean square error. The results show that this model is a powerful and efficient tool for HPC strength estimation. Another important feature of NHLM is that humans can easily interpret the hyper-rectangles as a decision making tool.

References

1 Yeh I C, *J Comput Civil Eng, ASCE*, 13(1) (1999) 36-42.
2 Abrams D, *Design of Concrete Mixtures*, Bull Structural Materials Research Laboratory, Lewis Institute, Chicago, 20 (1918).
3 Jerath S & Kabbani I A, *ACI J Proc*, 80(4) (1983) 312-317.
4 Gambhir M L & Bansal S P, *Indian Concr J*, 63(10) (1989) 495-501.
5 Oluokun F A, *ACI Mater J*, 91 (1994) 362.
6 Chen L, *J Comput Civil Eng, ASCE*, 17 (4) (2003) 290-294.
7 Rumelhart D E, Hinton G E & Wiliams R J, *Parallel distributed processing, 1*, edited by Rumelhart D E & Mc Clelland, J L, (MIT Press, Cambridge), 1986, 318-362.
8 Yeh I C, Kuo Y H & Hsu D S, *Expert Syst*, 5 (1992) 59-70.
9 Yeh I C, Kuo Y H & Hsu D S, *J Comput Civil Eng, ASCE*, 7(1) (1993) 71-93.
10 Yeh I C, *J Mater Civil Eng*, 10 (4) (1998) 263-268.
11 Kishore J K & Patnaik L M, *IEEE Trans Evol Comput*, 4(3) (2000) 242-257.
12 Michalski R, Carbonell J & Mitchel, T, *Machine Learning*, (Tioga Publishing Co.), 1983.
13 Medin D & Schaffer M, *Psychol Rev*, 85(3) (1978) 207-238.
14 Helmbold D, Sloan R & Warmuth M, *Learning nested differences of intersection closed concept classes*, Proc Workshop on Computational Learning Theory, San Mateo, CA: Morgan Kaufmann, 1989.
15 Salzberg, S, in *Lecture Notes in Artificial Intelligence*, edited by J Siekmann, (Springer-Verlag Berlin Heidelberg), 1989, 184-201.
16 Tsai C S, *A study of applying grammar evolution for high performance concrete strength estimation*, Master’s thesis, Department of Civil Engineering, Chung Hua University, Hsin Chu, Taiwan, 2003.