V. Conclusion

Interest has focused on using neural networks as empirical models because of their powerful representational capacity and ability to handle multi-input, multi-output problems. However, conventional neural networks do not warn the user when they make predictions without adequate training data, and furthermore they do not provide error estimates on their predictions. We have presented the VI net, an extension of RBFN's that allows the networks to indicate when they are extrapolating and to calculate confidence limits for their predictions. In [12], these measures are demonstrated on both functional estimation and classification problems.

While much previous interest has focused on feedforward nets with sigmoidal nonlinearities, these networks have no inherent ability to indicate when they are functioning outside the domain over which they were trained. This is because the contours of constant hidden node activation are hyperplanes and therefore infinite in length; their activation does not represent proximity to training data. RBFN's, on the other hand, have an intrinsic ability to indicate when they are extrapolating since the activation of the RBF units is directly related to the proximity of the test point to the training data, a property which is exploited in the design of the VI net.

References

[1] P. Medgassy, *Decomposition of Superposition of Distribution Functions.* Budapest: Hungarian Academy of Sciences, 1961.

[2] J. Park and I. W. Sandberg, "Universal approximation using radial-basis function networks," *Neural Computation,* vol. 3, pp. 246–257, 1991.

[3] J. Moody and C. J. Darken, "Fast learning in networks of locally tuned processing units," *Neural Computation,* vol. 1, pp. 281–294, 1989.

[4] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Fifth Berkeley Symp. Math. Stat. and Prob.* (Berkeley, CA), 1967, pp. 281–297.

[5] S. M. Weiss and C. A. Kulikowski, *Computer Systems that Learn.* San Mateo, CA: Morgan Kaufmann, 1991.

[6] R. O. Duda and P. E. Hart, *Pattern Analysis and Scene Classification.* New York: Wiley Interscience, 1973.

[7] D. R. Chand and S. S. Kapur, "An algorithm for convex polytopes," *J. Ass. Comput. Mach.,* vol. 17, pp. 78–86, 1978.

[8] D. F. Specht, "Probabilistic neural networks," *Neural Networks,* vol. 3, pp. 109–118, 1990.

[9] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.,* vol. 33, pp. 1065–1076, 1962.

[10] H. G. C. Tråvén, "A neural network approach to statistical pattern classification by "semiparametric" estimation of probability density functions," *IEEE Trans. Neural Networks,* vol. 2, pp. 366–377, May 1991.

[11] W. Mendenhall, *Introduction to Probability and Statistics,* 6th ed., Boston: Duxbury Press, 1983.

[12] J. A. Leonard, M. A. Kramer and L. H. Ungar, "A neural network architecture that predicts its own reliability," *Comput. Chem. Engng.,* in press, 1992.

Statistically Controlled Activation Weight Initialization (SCAWI)

Gian Paolo Drago and Sandro Ridella

Abstract—**An optimum weight initialization which strongly improves the performance of the BP algorithm is suggested. By statistical analysis, the scale factor, R (which is proportional to the maximum magnitude of the weights), is obtained as a function of the paralyzed neuron percentage (PNP). Also, by computer simulation, the performances on the convergence speed have been related to PNP. An optimum range for R is shown to exist in order to minimize the time needed to reach the minimum of the cost function. The analysis is carried on by properly defining normalization factors, which leads to a distribution of the activations independent of the neurons, and to a single nondimensional quantity, R, whose value may be quickly found by computer simulation.**

I. Introduction

One of the main problems in using the BP algorithm [1] in practical applications is its slow convergence to the global minimum during learning. Many improvements of the BP optimization algorithms have been suggested [2]–[4] and they certainly represent strong improvements with respect to the original algorithm.

Few papers [4]–[6], [8], on the contrary, have been published on the problem of optimum initialization of the weights when considering feedforward neural networks where neurons are of sigmoidal type [1]. The importance of an optimum weight initialization is quite evident both in achieving an optimum training and in achieving very fast learning. Very good and fast results can obviously be obtained when a starting point of the optimization process is very close to an optimum solution.

The interest in investigating an optimum weight initialization originated during our search for improvements of the BP optimization algorithm. The main result we found during computer simulation was the existence of an amplitude window in the maximum magnitude of the initial weights. Small values of the previous quantity lead to local minima, while large ones lead neurons to a saturated state, which with high probability prevents the network from reaching the desired solution. This finding is not new at all: similar behavior has been found by other researchers, e.g. [5] and [6]. Then we were stimulated to try to find a theoretical explanation of the phenomenon or, at least, a practical expression, which lets one start in an optimum way the learning process in a feedforward neural network.

This paper is organized in three sections. First we will give some definitions, which are necessary to understand the theory developed in Section III. In Section IV theory is tested with some examples and results are compared with those obtained by computer simulation. Finally a simple practical solution is derived for one of the most frequently used network topologies.

II. Definitions

In this work neurons of the sigmoidal type [1] are used, and, following our experience and that of other researchers [2], their output

Manuscript received October 1, 1991. This work was supported by the Ministero dell'Universitá e della Ricerca Scientifica e Tecnologica (MURST) of Italy and by the Consiglio Nazionale delle Ricerche (CNR) of Italy.

G. P. Drago is with the Istituto per i Circuiti Elettronici—CNR, via Opera Pia, 11–16145 Genova, Italy.

S. Ridella is with DIBE—Universitá di Genova, via Opera Pia, 11A–16145, Genova, Italy.

IEEE Log Number 9108270.

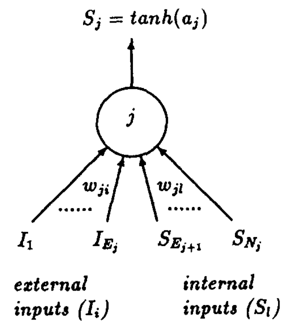

Fig. 1. Neuron input/output variables. Internal inputs coincide with outputs of other neurons. Bias is included within the set of external inputs.

range is chosen between -1 and 1. Consequently we define (Fig. 1)

$$S_j = \tanh(a_j) \tag{1}$$

$$a_j = \sum_{i=1}^{N_j} w_{ji}x_i = \sum_{i=1}^{E_j} w_{ji}I_i + \sum_{i=E_j+1}^{N_j} w_{ji}S_i, \tag{2}$$

where S_j is the output of the jth neuron;
a_j is its total input (activation) of the jth neuron;
x_i is a variable representing the inputs of a neuron, which may be:

$$x_i = \begin{cases} S_i \text{ is the output of the } i\text{th neuron} \\ I_i \text{ is the } i\text{th external input of the network}^1 \end{cases}$$

w_{ji} is the weight between the input of the jth neuron and x_i;
N_j is the number of inputs of the jth neuron;
E_j is the number of external inputs of the network which are connected to the jth neuron, plus one representing its bias.

In the following in fact the bias of a neuron is replaced by an external input whose value is always fixed to 1.

The neural network we consider is a feedforward one where no feedback is allowed, but it is more general than the usual multilayer perceptron since, for example, connections between external inputs and outputs are allowed. We note that this is the topology which was considered by Rumelhart in developing the BP algorithm.

Let us now introduce some concepts and definitions, which will be used in the following sections.

A neuron is in a *saturated* state when the magnitude of its output is greater than a predefined threshold, $\bar{S}_S$, which in our case is $\bar{S}_S = 0.9$. This value corresponds to a value $a_0 = 1.47$ of the total input, using the neuron defined in expression (1). The choice of putting the target values of the network outputs in the range ± 0.9 is quite usual in order to avoid the need of very large values of the weights: we remember that when the magnitude of the output of a neuron is large, the derivative of (1) is very small, back-propagation cannot occur [1], and the training becomes difficult.

A neuron is in a *paralyzed* state when it is saturated and the magnitude of at least one output error (that is, the difference between one output of the network and its target) is greater than or equal to a predefined value, $\bar{S}_P$, which in our case is $\bar{S}_P = 0.9$. This means that the output and the target are markedly different, since they have different signs, and, owing to the saturation state of the neuron, the amplitude of the error signal, which is back-propagated to the neuron input weights, is strongly attenuated.

A parameter, the *paralyzed neuron percentage* (PNP), may be defined by testing how many times a neuron is in a paralyzed state. During computer simulation, PNP is obtained, for a given set of weights, by considering the entire training set and by counting how many neurons (both hidden and output) are paralyzed, and dividing the result by the number of neurons and by the number of patterns.

The expression we use for weight initialization is

$$w_{ji} = \frac{R \cdot a_0}{\sqrt{\theta_j}} \cdot r_{ji} \tag{3}$$

$$\theta_j = \sum_{i=1}^{N_j} E[x_i^2], \tag{4}$$

where R is a nondimensional scale factor, which will be computed in the following section, $E[y]$ is the expectation of the (random) variable y, and r_{ji} is a random number uniformly distributed in the range $(-1, +1)$. The last definition guarantees that weights are independent random variables with zero mean and

$$E[w_{ji}^2] = \frac{1}{3}\frac{R^2 a_0^2}{\theta_j}. \tag{5}$$

While the motivation underlying expression (3) is discussed in detail in the following section, some preliminary comments are worthwhile. Traditionally, weights are started using a random number generator uniformly distributed in the range $(-r, r)$. Rumelhart [1] clearly explains the benefits of this initialization. It must be stressed that in this case r is not a nondimensional quantity and it is known that its optimum value depends on various factors, such as the number of inputs of every neuron. In our expression (1), the ratio

$$a_0/\sqrt{\theta_j}$$

has the dimension of a weight and it takes into account, through θ_j, for the distribution of every input to the neurons, giving an expression for weight initialization which may be used for every number and type of input signals.

Finally performances of the optimization algorithm have been measured by evaluating the *expected epochs to solution* (EES), i.e., the ratio of total epochs to successful trials. The ratio of total successful epochs to successful trials (RSS) has been also evaluated.

III. Weight Initialization: Theory

In order to develop a theory of weight initialization, two main hypotheses are necessary:

1) The **external inputs** of the network are considered equivalent to random variables characterized by their mean values ($E[I_i]$) and by there variance σ_{Ii}^2:

$$E[I_i^2] = \sigma_{Ii}^2 + E^2[I_i]. \tag{6}$$

2) These quantities may be estimated using the patterns of the training set.
3) The **outputs of the neurons,** which are used as inputs for other neurons, can be considered as a random variable characterized by

$$E[S_j^2] = E[\tanh^2(a_j)], \tag{7}$$

4) where a_j is not known a priori since it may depend on both I_i and S_i.

^{1}Input values should be normalized, as is usually done, within a single range, for example between -1 and 1.

In the following we will make the hypothesis that the number of inputs of a neuron is high enough to allow use of the central limit theorem. This implies that the activation of the neuron has, with a good approximation, a Gaussian probability distribution.

From the previous hypothesis and the use of expression (3) one obtains

$$E[a_j] = 0$$

and

$$\sigma_{a_j}^2 = E[a_j^2] = E\left[\left(\sum_{i=1}^{N_j} w_{ji}x_i\right)^2\right] \tag{8}$$

$$\sigma_{a_j}^2 = \frac{1}{3}\frac{R^2a_0^2}{\theta_j} \cdot \sum_{i=1}^{N_j} E[x_i^2] \tag{9}$$

$$\sigma_{a_j} = \sigma_a = \frac{R \cdot a_0}{\sqrt{3}}. \tag{10}$$

Equation (9) is obtained from (8), since the weights are independent random variables with zero mean and (5) is taken into account. Finally (10) is obtained from (9), remembering (4).

The strong advantage of using weights normalized to the square root of (4) is made clear by (10), where the standard deviation of the activation is the same for *every* neuron. Computational problems may occur, during weight initialization, in the evaluation of θ_j in equation (4) for those neurons which have inputs from the outputs of other neurons. We postpone the description of the procedure for making this computation until the end of the section.

We will now evaluate PNP as a function of R. First, we have evaluated the probability P_j of saturation of the jth neuron. From the computation performed in the first part of the section, this probability is independent of the neuron considered and it may be easily arrived at by standard statistical analysis [7]. By considering that saturation occurs when the magnitude of a_j is greater than a_0, one obtains

$$P_j = 1 - 2\text{erf}\left(\frac{a_o}{\sigma_{a_j}}\right) \tag{11}$$

$$P_j = P = 1 - 2\text{erf}\left(\frac{\sqrt{3}}{R}\right). \tag{12}$$

From the definition of PNP given in the first section, one obtains

$$PNP = (1/\beta) \cdot P, \tag{13}$$

where

$$1/\beta = \left(1 - \frac{1}{2^{n_o}}\right) \tag{14}$$

where n_o is the number of outputs of the network. In order to obtain (13), digital targets have been supposed, with a 50% probability of correct output. Then $1/\beta$ is the probability that at least one output of the neural network is incorrect. By inspection one obtains that $1 < \beta \leq 2$. Thanks to expressions (12) and (13), one may obtain PNP from R, or, inverting expression (12), R from PNP:

$$R = \frac{\sqrt{3}}{\Psi(PNP)}, \tag{15}$$

where

$$\Psi = \text{erf}^{-1}\left(\frac{1}{2} - \frac{\beta}{2} \cdot PNP\right). \tag{16}$$

In the next section we will make computer simulations in order to find the dependence of EES from R: this is unavoidable, since no theory is at the moment available to correlate EES and PNP. As a by-product the validity of (15) will be confirmed.

These empirical results suggest the formulation of the following PNP rule:

The optimum value of the scale factor, R, of the initial weights corresponds to a value of PNP which is called PNP_{opt}.: experience suggests a value of PNP_{opt} near 5%.

The value of R may be found by using the PNP rule and (15): optimum weight initialization may be now performed through (3).

We conclude the section by showing how it is possible to evaluate θ_j (eq. (4)) for those neurons which have inputs from the outputs of other neurons. From the results obtained at the beginning of the section (especially (10)), (7) becomes

$$E[S^2] = \int_{-\infty}^{\infty} \frac{1}{\sigma_a\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma_a^2}} \tanh^2(x)dx \tag{17}$$

and, consequently, θ_j becomes

$$\theta_j = \sum_{i=1}^{N_j} E[I_i^2] + (N_j - E_j) \cdot E[S^2]. \tag{18}$$

The integral in (17) is easily evaluated by standard numerical methods; σ_a is obtained from PNP through

$$\sigma_a = \frac{a_0}{\Psi(PNP)}, \tag{19}$$

which is easily derived from (10) and (15). The trick lies in using the weight normalization of (4), which leads to a distribution of the activations independent of the neurons. The standard deviation of this activation may be easily computed from the value of PNP. This completes the details of the optimum starting weights we suggest. Computer simulations in the following section will be used both for assessing the PNP rule and for showing the validity of the initialization procedure.

IV. Weight Initialization: Computer Simulation

In this section we will carry out computer simulation in order to

1) show the performances on the convergence speed (EES) using the initialization procedure previously described;
2) find the PNP rule, which makes it easy to compute the optimum value of the scale factor, R,
3) derive from (3) a closed-form expression for the frequently used topology of a *two-layer* network without direct connections between inputs and outputs.

Many tests have been suggested in order to estimate the performances of supervised networks [1], [2]. We have chosen parity and symmetry, which are considered to show a long convergence, and the encoder, which allow us to test networks with more than one output.

In order to train an MLP, we need an optimization routine: we use the adaptive momentum back-propagation (AMBP) algorithm, which is a modified version of the BP algorithm characterized by adaptive learning parameters [3], [4]. The quality of the results we show in this section refers then to the combination of SCAWI and AMBP.

It is not possible to theoretically guarantee that our results will be confirmed if AMBP is superseded by another optimization technique. In the latter case, and also if other tests are considered, a different value of PNP_{opt} in the PNP rule could occur. By computer simulation using its own learning routine, one can find the value of PNP_{opt} matched to its specific task. This metaoptimization may be expensive; in our experience, the value we suggested for PNP_{opt} is quite satisfactory, since in any case values in the range 2%–10% produce good performances in practical applications. As a

TABLE I
COMPUTATIONAL COSTS AS A FUNCTION OF THE WEIGHTS NUMBER FOR $n_i - n_i - 1$ PARITY TASK

n_i	n_w Weights Number	Trials	*RSS* Epochs	*EES* Epochs
2	9	100	16.56	33.13
3	16	100	20.60	22.52
4	25	100	73.45	247.48
5	36	100	70.01	122.20
6	49	100	169.19	635.13
7	64	100	181.35	500.23
8	81	100	257.33	1815.50

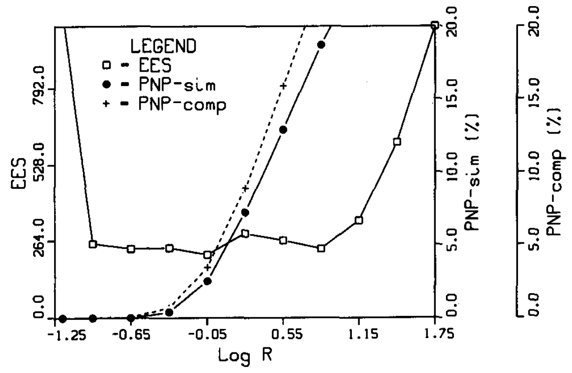

Fig. 2. EES, PNP_{sim}, and PNP_{opt} against R for the $n_i = 4$ parity task.

consequence, an optimum range for R exists, which has been also found in other papers [5], [6].

One possible reason for this behavior is the long time needed to leave the initial state owing to the small values of the error back-propagated signals in the network. This is caused by the small values of the weights and of the derivatives of the neuron transfer function for small and large values of R, respectively.

The parity problem has been tested with the number of inputs varying from $n_i = 2$ to $n_i = 8$, a number of hidden neurons $n_h = n_i$, one output neuron ($n_o = 1$), and a complete interconnection network between the inputs and the hidden neurons and between these and the output neuron. The results are shown in Table I.

Fig. 2 shows, as a function of R, EES, the value of PNP_{sim} (which is found with the procedure described in Section II), and its computed value, PNP_{comp} (evaluated using (13), (14), and (12)). The figure refers to the $n_i = 4$ parity problem: similar results are obtained in the other cases we tested.

Comparison between PNP_{sim} and PNP_{comp} is satisfactory and supports the theory described in Section III.

From Fig. 2, one can see that EES shows a minimum for $R = 0.95$; this value corresponds to a PNP of about 5%. The figure shows that the value of R (and of PNP) is not critical, provided that the value of R is inside a window, in order to have a satisfactory value of EES. This EES window justifies the previously stated PNP rule.

Similar results have been obtained for symmetry and encoder. The tests on symmetry (Table II) have been performed with the number of inputs varying from $n_i = 2$ to $n_i = 8$, two hidden neurons ($n_h = 2$), and one output neuron ($n_o = 1$).

The tests on encoder (Table III) have been performed with $n_i = n_o = 8$ and the number of hidden neurons varying from $n_h = 2$ to $n_h = 16$.

TABLE II
COMPUTATIONAL COSTS AS A FUNCTION OF THE WEIGHTS NUMBER FOR $n_i - 2 - 1$ SYMMETRY TASK

n_i	n_w Weights Number	Trials	*RSS* Epochs	*EES* Epochs
2	9	100	16.56	33.13
3	11	100	20.70	33.40
4	13	100	40.71	961.86
5	15	100	50.00	445.90
6	17	100	74.61	1111.00
7	19	100	75.76	722.96
8	21	100	104.71	947.00

TABLE III
COMPUTATIONAL COSTS AS A FUNCTION OF THE WEIGHT NUMBER FOR ENCODER TASK (EES VALUES ARE EQUAL TO RSS ONES OWING TO THE SUCCESS PERCENTAGE OF 100)

n_i	Problem	n_w Weights Number	Trials	*RSS* Epochs	*EES* Epochs
2	8-2-8	42	100	92.64	92.64
3	8-3-8	59	100	25.06	25.06
4	8-4-8	76	100	15.44	15.44
5	8-5-8	93	100	11.56	11.56
6	8-6-8	110	100	9.84	9.84
7	8-8-8	144	100	8.26	8.26
8	8-16-8	280	100	5.66	5.66

As a general comment about the satisfactory performance of our learning algorithm, we point out the importance of the previously quoted normalization of those inputs which corresponds to analogical training data. This is confirmed by preliminary results on the *two-spirals* benchmark [9]: during 30 trials, using $n_h = 30$ hidden units (weight number $n_w = 121$), only 3000 epochs (EES) were required, comparing very favorably with Lang and Witbrock's results (60 000 epochs, $n_w = 68$) presented in [9] and stressing the ability of SCAWI to work with a very complicated data structure, which is very difficult to learn. Nevertheless, the statistical property of the training set will surely influence the convergence speed, since one has a very low probability to start correctly when a very deep global minimum, with a very small attraction basin, is present. Our experience, however, is that the previous situation occurs very infrequently: SCAWI is extensively used for artificial music composition, with outstanding results with respect to both speed and compression ratio [10].

We conclude the paper by deriving from (3) an expression for the frequently used topology of a two-layer network without direct connections between inputs and outputs.

Let us define

$$v^2 = \frac{1}{n_i} \sum_{i=1}^{n_i} E[I_i^2]. \tag{20}$$

That is, v^2 is the mean of the expectation of the quadratic values of the inputs defined in (6). The values of the weights are denoted w_{ji}^w and w_{ji}^z, for those connecting the inputs with the hidden units and those connecting the hidden units and the outputs, respectively. We can now compute the factor θ_j defined in (18); its values are

$$\theta^w = 1 + n_i \cdot v^2 \tag{21}$$

TABLE IV
THE OPTIMUM VALUE OF R AND THE CORRESPONDING $E[S^2]_{\text{opt}}$ AS A FUNCTION OF THE NUMBER OF OUTPUTS, n_o

n_o	R_{opt}	$E[S^2]_{\text{opt}}$
1	1.053	0.354
2	0.945	0.316
3	0.911	0.303
4	0.896	0.298
5	0.890	0.295
10	0.884	0.293

and

$$\theta^z = 1 + n_h \cdot E[S^2], \tag{22}$$

where θ^w and θ^z are referred to w_w and w_z, respectively, through (3). Table IV relates n_o, the values of R and $E[S^2]$, corresponding to $PNP_{\text{opt}} = 5\%$ (R_{opt} and $E[S^2]_{\text{opt}}$).

In Table IV R_{opt} is numerically obtained from (15) and $E[S^2]_{\text{opt}}$ from (17).

Remembering that $PNP_{\text{opt}} = 5\%$ is not a critical value, we can estimate, with a sufficient accuracy, $R_{\text{opt}} = 0.9$ and $E[S^2]_{\text{opt}} = 0.3$. The weights can then be initialized using

$$w_{ji}^w = \frac{1.3}{\sqrt{1 + n_i \cdot v^2}} \cdot r_{ji} \tag{23}$$

$$w_{ji}^z = \frac{1.3}{\sqrt{1 + 0.3 \cdot n_h}} \cdot r_{ji}. \tag{24}$$

These expressions may be easily used in any BP software.

REFERENCES

[1] D. E. Rumelhart and J. L. McClelland, Eds., *Parallel distributed processing*, vol. 1. Cambridge, MA: MIT Press, 1988.

[2] J. Hertz, A. Krogh, and G. R. Palmer, *Introduction to the Theory of Neural Computation.* Reading, MA: Addison-Wesley, 1991.

[3] G. P. Drago and S. Ridella, "Decreasing backpropagation computational cost by adapting learning parameters," in *Proc. Ninth IASTED Int. Symp. Appl. Informatics* (Innsbruck), Feb. 18–21, 1991, pp. 137–140.

[4] G. P. Drago and S. Ridella, "An optimum weights initialization for improving scaling relationships in BP learning," in *Proc. 1991 Int. Conf. Artificial Neural Networks* (Espoo, Finland), 24–28 June, 1991, pp. 1519–1522.

[5] Y. Hirose *et al.*, "BP algorithm which varies the number of hidden units," *Neural Networks*, vol. 4, pp. 61–66, 1991.

[6] P. J. G. Lisboa and S. J. Peratonis, "Complete solution of the local minima in the XOR problem," *Network: Computation in Neural Systems*, vol. 2, no. 1, pp. 119–124, Feb. 1991.

[7] A. Papoulis, *Probability, Random Variables and Stochastic Processes.* New York: McGraw-Hill, 1965.

[8] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of adaptive weights," in *Proc. Int. Joint Conf. Neural Networks,* June 1989, pp. III-21–26.

[9] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," CMU-CS-90-100, Feb. 14, 1990, p. 6.

[10] G. P. Drago, C. Martini, M. Morando, and S. Ridella, "A neural network for music composition," in *Proc. Tenth IASTED Int. Symp. Appl. Informatics* (Innsbruck), Feb. 10–12, 1992.