OPTIMIZING DECISION MAKING WITH NEURAL NETWORKS IN SOFTWARE AGENTS

Arpad Kelemen, Yulan Liang, Robert Kozma,

Stan Franklin, E. Olusegun George

Institute for Intelligent Systems, Department of Mathematical Sciences,

University of Memphis, 373 Dunn Hall, Memphis, TN 38152

Email: kelemena@msci.memphis.edu yuliang@memphis.edu rkozma@memphis.edu franklin@memphis.edu eogeorge@memphis.edu

Key Words:

Decision making, Multilayer perceptron, Structural learning, Autonomous agent, Support vector machine

Abstract

Finding suitable jobs for US Navy sailors from time to time is an important and ever-changing process. An Intelligent Distribution Agent (IDA) and particularly its constraint satisfaction module take up the challenge to automate the process. The constraint satisfaction module's main task is to provide the bulk of the decision making process in assigning sailors to new jobs in order to maximize Navy and sailor happiness. We propose Multilayer Perceptron neural network with structural learning in combination with several statistical criteria to optimize the module's performance and to make decisions in general. Multilayer Perceptron (MLP) with different structures and algorithms, Feedforward Neural Network (FFNN) with logistic regression and Support Vector Machine (SVM) with Radial Basis Function (RBF) as network structure and Adatron learning algorithm are also presented for comparison study. The data were taken from Navy databases and from surveys of Navy experts. The subjective indeterminate nature of the detailer decisions make the optimization problem nonstandard. Multilayer Perceptron neural network with structural learning and Support Vector Machine produced highly accurate classification and encouraging prediction.

1 INTRODUCTION

IDA [1], is a "conscious" [2], [3] software agent [4], [5] being built for the U.S. Navy by the Conscious Software Research Group at the University of Memphis. IDA is designed to play the role of Navy employees, called detailers, who assign sailors to new jobs from time to time. For this purpose IDA is equipped with thirteen large modules, each of which is responsible for one main task. One of them, the constraint satisfaction module, is responsible for satisfying constraints to ensure the adherence to Navy policies, command requirements, and sailor preferences. To better

model human behavior IDA's constraint satisfaction is implemented through a behavior network [6], [7] and "consciousness". The model employs a linear functional approach to assign fitness values to each candidate job for each candidate sailor. The functional yields a value in $[0, 1]$ with higher values representing higher degree of "match" between the sailor and the job. Some of the constraints are soft, while others are hard. Soft constraints can be violated without invalidating the job. Associated with the soft constraints are functions which measure how well the constraints are satisfied for the sailor and the given job at the given time, and coefficients which measure how important the given constraint is relative to the others. The hard constraints cannot be violated and are implemented as Boolean multipliers for the whole functional. A violation of a hard constraint yields 0 value for the functional.

The process of using this method for decision making involves periodic tuning of the coefficients and the functions. A number of alternatives and modifications have been proposed, implemented and tested for some partial but real domain. A genetic algorithm approach is discussed by Kondadadi, Dasgupta, and Franklin [8], and a large-scale network model is developed by Liang, Thompson and Buclatin [9], [10]. Previous study showed that the traditional Gale-Shapley model [11] is not easily applicable to this real life problem if we are to preserve the format of the available data and the way detailers currently make decisions. Our goal in this paper is to use neural networks and statistical methods to enhance decisions made by IDA's constraint satisfaction module and to make better decisions in general. The functions for the soft constraints were set up in consultation with Navy experts. We will assume that they are optimal, though future efforts will be made to verify this assumption.

While human detailers can make judgements about job preferences for sailors, they are not always able to quantify such judgements through functions and coefficients. Using data collected periodically from human detailers, a neural network learns to make human-like decisions for job assignments. It is widely believed that different detailers may attach different importance to constraints, depending on the sailor community (a community is a collection of sailors with similar jobs and trained skills) they handle, and may change from time to time as the environment changes. It is important to set up the functions and the coefficients in IDA to reflect these characteristics of the human decision making process. A neural network gives us more insight on what preferences are important to a detailer and how much. Moreover inevitable changes in the environment will result changes in the detailer's decisions, which could be learned with a neural network although with some delay.

In this paper, we propose several approaches for achieving optimal decisions in software agents. We elaborate on our preliminary results reported in [1], [12]. Feedforward Neural Networks with logistic regression, MLP with structural learning and Support Vector Machine with Radial Basis Function (RBF) as network structure were explored to model decision making. Statistical criteria, like Mean Squared Error, Minimum Description Length, etc. were employed to search for best network structure and optimal performance. We apply sensitivity analysis through choosing different algorithms to assess the stability of the given approaches.

In Section II we describe how the data were attained and formulated into the input of the neural networks. In Section III we discuss FFNNs with Logistic Regression, performance function and statistical criteria of MLP Selection for best performance including learning algorithm selection.

After this we turn our interest to Support Vector Machine since the data involved high level noise. Section IV presents some comparison study and numerical results of all the presented approaches along with the sensitivity analysis.

2 DATA ACQUISITION AND PREPROCESSING

The data was extracted from the Navy's Assignment Policy Management System's job and sailor databases. For the study one particular community, the Aviation Support Equipment Technicians community was chosen. Note that this is the community on which the current IDA prototype is being built [1]. The databases contained 467 sailors and 167 possible jobs for the given community. From the more than 100 attributes in each database only those were selected which are important from the viewpoint of the constraint satisfaction: Eighteen attributes from the sailor database and six from the job database. For this study we chose four hard and four soft constraints. Table 1 shows the four hard constraints, which were applied to these attributes in compliance with Navy policies. Note that some details about the constraints are omitted in both Table 1 and Table 2. The detailed constraints are used in the experiment to construct the data set and to build the models. 1277 matches passed the given hard constraints, which were inserted into a new database.

Table 2 shows the four soft constraints applied to the matches that satisfied the hard constraints and the functions which implement them. These functions measure degrees of satisfaction of matches between sailors and jobs, each subject to one soft constraint. Again, the policy definitions are simplified. All the f_i functions are monotone but not necessarily linear, although it turns out that linear functions are adequate in many cases. Note that monotonicity can be

achieved in cases when we assign values to set elements (such as location codes) by ordering. After preprocessing the function values, which served as inputs to future processing, were defined using information given by Navy detailers. Each of the function's range is $[0, 1]$.

Output data (decisions) were acquired from an actual detailer in the form of Boolean answers for each possible match (1 for jobs to be offered, 0 for the rest). Each sailor together with all his/her possible jobs that satisfy the hard constraints were assigned to a unique group. The numbers of jobs in each group were normalized into $[0, 1]$ by simply dividing them by the maximum value and included in the input as function f_5. This is important because the outputs (decisions given by detailers) were highly correlated: there was typically one job offered to each sailor.

3 DESIGN OF NEURAL NETWORK

One natural way the decision making problem in IDA can be addressed is via the tuning the coefficients for the soft constraints. This will largely simplify the agent's architecture, and it saves on both running time and memory. Decision making can also be viewed as a classification problem, for which neural networks demonstrated to be a very suitable tool. Neural networks can learn to make human-like decisions, and would naturally follow any changes in the data set as the environment changes, eliminating the task of re-tuning the coefficients.

3.1 Feedforward Neural Network with Logistic Regression

We use a logistic regression model to tune the coefficients for the functions $f_1, ..., f_4$ for the soft constraints and evaluate their relative importance. The corresponding conditional probability of

the occurrence of the job to be offered is

$$\hat{y} = P(\text{decision} = 1|w) = g(w^T f) \tag{1}$$

$$g(a) = \frac{e^a}{1 + e^a} \tag{2}$$

where g represents the logistic function evaluated at activation a. Let w denote weight vector and f the column vector of the importance functions: $f^T = [f_1, ..., f_5]$. Then the "decision" is generated according to the logistic regression model.

The weight vector w can be adapted using FFNN topology [13], [14]. In the simplest case there is one input layer and one output logistic layer. This is equivalent to the generalized linear regression model with logistic function. The estimated weights satisfy Eq.(3).

$$\sum_i w_i = 1, \quad 0 \leq w_i \leq 1 \tag{3}$$

The linear combination of weights with inputs $f_1, ..., f_4$ is a monotone function of conditional probability, as shown in Eq.(1) and Eq.(2), so the conditional probability of job to be offered can be monitored through the changing of the combination of weights with inputs $f_1, ..., f_4$. The classification of decision can be achieved through the best threshold with the largest estimated conditional probability from group data. The class prediction of an observation x from group y was determined by

$$C(x) = argmax_k Pr(x|y = k) \tag{4}$$

To find the best threshold we used Receiver Operating Characteristic (ROC) to provide the percentage of detections correctly classified and the non-detections incorrectly classified. To do so we employed different thresholds with range in $[0, 1]$. To improve the generalization performance and achieve the best classification, the MLP with structural learning was employed [15], [16].

3.2 Neural Network Selection and Criteria

Since the data coming from human decisions inevitably include vague and noisy components, efficient regularization techniques are necessary to improve the generalization performance of the FFNN. This involves network complexity adjustment and performance function modification. Network architectures with different degrees of complexity can be obtained through adapting the number of hidden nodes and partitioning the data into different sizes of training, cross-validation and testing sets and using different types of activation functions. A performance function commonly used in regularization, instead of the sum of squared error (SSE) on the training set, is a loss function (mostly SSE) plus a penalty term [17]-[20]:

$$J = SSE + \lambda \sum w^2 \tag{5}$$

From another point of view, for achieving the optimal neural network structure for noisy data, structural learning has better generalization properties and usually use the following modified performance function [15], [16]:

$$J = SSE + \lambda \sum |w| \tag{6}$$

Yet in this paper we propose an alternative cost function, which includes a penalty term as follows:

$$J = SSE + \lambda n/N \tag{7}$$

where SSE is the sum of squared error, λ is a penalty factor, n is the number of parameters in the network decided by the number of hidden nodes and N is the size of the input example set.

For achieving the minimized SSE under the constraint or penalty term in Eq.(7), there is a closed form for the value of N and n. This closed from can be obtained by taking partial derivatives of the Lagrange multiplier equation if n and N are continuous. This yields a relationship between N and n and also among N, n and the SSE. Through structural learning in which the cost function in Eq.(7) is minimized, we would like to find the effective size of training samples included in the network and also the best number of hidden nodes for the one hidden layer case. In our study the value of λ in Eq.(7) is from 0.01 to 1.0. Note that λ=0 represents a case where we don't consider structural learning, and the cost function reduces into the sum of squared error. Normally the size of input samples should be chosen as large as possible in order to keep the residual as small as possible. Due to the cost of the large size samples, the input may not be chosen as large as desired. However, if the sample size is fixed then the penalty factor combined with the number of hidden nodes should be adjusted to minimize Eq.(7).

For achieving the balance between data-fitting and model complexity from the proposed performance function in Eq.(7) and for better generalization performance we designed a two-factorial array to dynamically retrieve the best partition of data into training, cross-validation and testing

sets with adapting the number of hidden nodes given the value of λ. Several statistical criteria were carried out for this model selection in order to find the best FFNN:

- Mean Squared Error (MSE) defined as the Sum of Squared Error divided by the degree of freedom. For this model, the degree of freedom is the sample size minus the number of parameters included in the network.

- Correlation Coefficient (r) can show the agreement between the input and the output or the desired output and the predicted output. In our computation, we use the latter.

- Akaike Information Criteria (AIC) [21] defined as

$$AIC(K_a) = -2\log(L_{ml}) + 2K_a. \tag{8}$$

- Minimum Description Length (MDL) [22] defined as

$$MDL(K_a) = -\log(L_{ml}) + 0.5K_a \log N, \tag{9}$$

  where L_{ml} is the maximum value of the likelihood function and K_a is the number of adjustable parameters. N is the size of the input examples' set.

The MSE can be used to determine how well the predicted output fits the desired output. More epochs generally provide higher correlation coefficient and smaller MSE for training in our study. To avoid overfitting and to improve generalization performance, training was stopped when the MSE of the cross-validation set started to increase significantly. Sensitivity analysis

were performed through multiple test runs from random starting points to decrease the chance of getting trapped in a local minimum and to find stable results.

The network with the lowest AIC or MDL is considered to be the preferred network structure. An advantage of using AIC is that we can avoid a sequence of hypothesis testing when selecting the network. Note that the difference between AIC and MDL is that MDL includes the size of the input examples which can guide us to choose appropriate partition of the data into training and testing sets. Another merit of using MDL/AIC versus MSE/Correlation Coefficient is that MDL/AIC use likelihood which has a probability basis. The choice of the best network structure is based on the maximization of predictive capability, which is defined as the correct classification rate and the lowest cost given in Eq.(7).

3.3 Learning Algorithms for FFNN

Various learning algorithms have been tested for comparison study [17], [20]:

- Back propagation with momentum
- Conjugate gradient
- Quickprop
- Delta-delta

The back-propagation with momentum algorithm has the major advantage of speed and is less susceptible to trapping in a local minimum. Back-propagation adjusts the weights in the steepest descent direction in which the performance function is decreasing most rapidly but it

does not necessarily produce the fastest convergence. The search of the conjugate gradient is performed along conjugate directions, which produces generally faster convergence than steepest descent directions. The Quickprop algorithm uses information about the second order derivative of the performance surface to accelerate the search. Delta-delta is an adaptive step-size procedure for searching a performance surface [20]. The performance of best MLP with one hidden layer network obtained from above was compared with popular classification method Support Vector Machine and FFNN with logistic regression.

3.4 Support Vector Machine

Support Vector Machine is a method for finding a hyperplane in a high dimensional space that separates training samples of each class while maximizes the minimum distance between the hyperplane and any training samples [23]-[26]. SVM has properties to deal with high noise level and flexibly applies different network architectures and optimization functions. Our used data involves relatively high level noise. To deal with this the interpolating function for mapping the input vector with the target vector should be modified such a way that it averages over the noise on the data. This motivates using Radial Basis Function neural network structure in SVM. RBF neural network provides a smooth interpolating function, in which the number of basis functions are decided by the complexity of mapping to be represented rather than the size of data. RBF can be considered as an extension of finite mixture models. The advantage of RBF is that it can model each data sample with Gaussian distribution so as to transform the complex decision surface into a simpler surface and then use linear discriminant functions. RBF has good properties for function approximation but poor generalization performance. To improve this we employed

Adatron learning algorithm [27], [28]. Adatron replaces the inner product of patterns in the input space by the kernel function of the RBF network. It uses only those inputs for training that are near the decision surface since they provide the most information about the classification. It is robust to noise and generally yields no overfitting problems, so we do not need to cross-validate to stop training early. The used performance function is the following:

$$J(x_i) = \lambda_i(\sum_{j=1}^{N} \lambda_j w_j G(x_i - x_j, 2\sigma^2) + b), \tag{10}$$

$$M = \min_i J(x_i), \tag{11}$$

where λ_i is multiplier, w_j is weight, G is Gaussian distribution and b is bias.

We chose a common starting multiplier (0.15), learning rate (0.70), and a small threshold (0.01). While M is greater than the threshold, we choose a pattern x_i to perform the update. After update only a few of the weights are different from zero (called the support vectors), they correspond to the samples that are closest to the boundary between classes. Adatron algorithm can prune the RBF network so that its output for testing is given by

$$g(x) = sgn(\sum_{i \in \text{Support Vectors}} \lambda_i w_i G(x - x_i, 2\sigma^2) - b), \tag{12}$$

so it can adapt an RBF to have an optimal margin. Various versions of RBF networks (spread, error rate, etc.) were also applied but the results were far less encouraging for generalization than with SVM with the above method.

4 DATA ANALYSIS AND RESULTS

For implementation we used a Matlab 6.1 [29] environment with at least a 500 MHz Pentium III processor. For data acquisition and preprocessing we used SQL queries with SAS 8.0.

4.1 Estimation of Coefficients for Soft Constraints

FFNN with back-propagation with momentum with logistic regression gives the weight estimation for the four coefficients as reported in Table 3. Simultaneously, we got the conditional probability for decisions of each observation from Eq.(1). We chose the largest estimated logistic probability from each group as predicted value for decisions equal to 1 (job to be offered) if it was over threshold. The threshold was chosen to maximize performance and its value was 0.65. The corresponding correct classification rate was 91.22% for the testing set. This indicates a good performance. This result still can be further improved as it is shown in the forthcoming discussion.

4.2 Neural Network for Decision Making

Multilayer Perceptron with one hidden layer was tested using tansig and logsig activation functions for hidden and output layers respectively. Other activation functions were also used but did not perform as well. MLP with two hidden layers were also tested but no significant improvement was observed. Four different learning algorithms were applied for sensitivity analysis. For reliable results and to better approximate the generalization performance for prediction, each experiment was repeated 10 times with 10 different initial weights. The reported values were averaged over the 10 independent runs. Training was confined to 5000 epochs, but in most cases there were no

significant improvement in the MSE after 1000 epochs. The best MLP was obtained through structural learning where the number of hidden nodes ranged from 2 to 20, while the training set size was setup as 50%, 60%, 70%, 80% and 90% of the sample set. The cross-validation and testing sets each took the half of the rest. We used 0.1 for the penalty factor λ, which gave better generalization performance then other values for our data set. Using MDL criteria we can find out the best match of percentage of training with the number of hidden nodes in a factorial array.

Table 4 reports MDL/AIC values for given number of hidden nodes and given testing set sizes. As shown in the table, for 2, 5 and 7 nodes, 5% for testing, 5% for cross validation, and 90% for training provides the lowest MDL. For 9 nodes the lowest MDL was found for 10% testing, 10% cross validation, and 80% training set sizes. For 10-11 nodes the best MDL was reported for 20% cross-validation, 20% testing and 60% training set sizes. For 12-20 nodes the best size for testing set was 25%. We observe that by increasing the number of hidden nodes the size of the training set should be increased in order to lower the MDL and the AIC. Since MDL includes the size of the input examples, which can guide us to the best partition of the data for cases when the MDL and AIC values do not agree we prefer MDL.

Table 5 provides the correlation coefficients between inputs and outputs for best splitting of the data with given number of hidden nodes. 12-20 hidden nodes with 50% training set provides higher values of the correlation coefficient than other cases. Fig. 1 gives the average of the correct classification rates of 10 runs, given different numbers of hidden nodes assuming the best splitting of data. The results were consistent with Tables 4 and 5. The 0.81 value of the correlation coefficient shows that the network is reasonably good.

4.3 Comparison of Estimation Tools

In this section we compare results obtained by FFNN with logistic regression, MLP with structural learning and SVM with RBF as network and Adatron as learning algorithm. Fig. 2 gives the errorbar plots of MLP with 15 hidden nodes (best case of MLP), FFNN with logistic regression, and SVM to display the means with unit standard deviation and medians for different size of testing samples. It shows how the size of the testing set affects the correct classification rates for three different methods. As shown in the figure the standard deviations are small for 5%-25% testing set sizes for the MLP. The median and the mean are close to one another for 25% testing set size for all the three methods, so taking the mean as the measurement of simulation error for these cases is as robust as the median. Therefore the classification rates given in Fig. 1 taking the average of different runs as measurement is reasonable for our data. For cases when the median is far from the mean, the median could be more robust statistical measurement than the mean. The best MLP network from structural learning as it can be seen in Fig. 1 and Fig. 2 is 15 nodes in the hidden layer and 25% testing set size.

Early stopping techniques were employed to avoid overfitting and to better the generalization performance. Fig. 3 shows the MSE of the training and the cross-validation data with the best MLP with 15 hidden nodes and 50% training set size. The MSE of the training data goes down below 0.09 and the cross validation data started to significantly increase after 700 epochs, therefore we use 700 for future models. Fig. 4 shows the sensitivity analysis and the performance comparison of back-propagation with momentum, conjugate gradient descent, quickprop, and delta-delta learning algorithms for MLP with different number of hidden nodes and best cutting

of the sample set. As it can be seen their performance were relatively close for our data set, and delta-delta performed the best. MLP with back-propagation with momentum also performed well around 15 hidden nodes. MLP with 15 hidden nodes and 25% testing set size gave approximately 6% error rate, which is a very good generalization performance for predicting jobs to be offered for sailors. Even though SVM provided slightly higher correct classification rate than MLP, it has a significant time complexity.

Some noise is naturally present when humans make decisions in a limited time frame. According to one detailer's estimation a 20% difference would occur in the decisions even if the same data would be presented to the same detailer at a different time. Also, it is widely believed that different detailers are likely to make different decisions even under the same circumstances. Moreover environmental changes might further bias decisions.

5 CONCLUSION

High-quality decision making using optimum constraint satisfaction is an important goal of IDA, to aid the NAVY to achieve the best possible sailor and NAVY satisfaction performance. A number of neural networks with statistical criteria were applied to either improve the performance of the current way IDA handles constraint satisfaction or to come up with alternatives. IDA's constraint satisfaction module, neural networks and traditional statistical methods are complementary with one another. In this work we combined MLP with structural learning and statistical criteria, which provided us with the best MLP with one hidden layer. 15 hidden nodes and 25% testing set size using back-propagation with momentum and delta-delta learning algorithms provided good

generalization performance for our data set. SVM with RBF network architecture and Adatron learning algorithm gave the best prediction performance for decision classification with an error rate below 6%, although with significant time cost. In comparison to human detailers such a performance is remarkable. Coefficients for the existing IDA constraint satisfaction module were adapted via FFNN with logistic regression. It is important to keep in mind that the coefficients have to be updated from time to time as well as online neural network trainings are necessary to comply with changing Navy policies and other environmental challenges.

ACKNOWLEDGEMENTS

This work is supported in part by ONR grant (no. N00014-98-1-0332). The authors thank the "Conscious" Software Research group for its contribution to this paper.

References

[1] S. Franklin, A. Kelemen, and L. McCauley, "IDA: A cognitive agent architecture," in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics '98*, IEEE Press, pp. 2646, 1998.

[2] B. J. Baars, *A Cognitive Theory of Consciousness*, Cambridge University Press, Cambridge, 1988.

[3] B. J. Baars, *In the Theater of Consciousness*, Oxford University Press, Oxford, 1997.

[4] S. Franklin and A. Graesser, "Intelligent Agents III: Is it an Agent or just a program?: A Taxonomy for Autonomous Agents," in *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, Springer-Verlag, pp. 21-35, 1997.

[5] S. Franklin, *Artificial Minds*, Cambridge, Mass, MIT Press, 1995.

[6] P. Maes, "How to Do the Right Thing," *Connection Science*, 1:3, 1990.

[7] H. Song and S. Franklin, "A Behavior Instantiation Agent Architecture," *Connection Science*, Vol. 12, pp. 21-44, 2000.

[8] R. Kondadadi, D. Dasgupta, and S. Franklin, "An Evolutionary Approach For Job Assignment," in *Proceedings of International Conference on Intelligent Systems-2000*, Louisville, Kentucky, 2000.

[9] T. T. Liang and T. J. Thompson, "Applications and Implementation - A large-scale personnel assignment model for the Navy," *The Journal For The Decisions Sciences Institute*, Volume 18, No. 2 Spring, 1987.

[10] T. T. Liang and B. B. Buclatin, "Improving the utilization of training resources through optimal personnel assignment in the U.S. Navy," *European Journal of Operational Research*, 33, pp. 183-190, North-Holland, 1988.

[11] D. Gale and L. S. Shapley, "College Admissions and stability of marriage," *The American Mathematical monthly*, Vol 60, No 1, pp. 9-15, 1962.

[12] A. Kelemen, Y. Liang, R. Kozma, and S. Franklin, "Optimizing Intelligent Agent's Constraint Satisfaction with Neural Networks," in *"Innovations in Intelligent Systems"* (A. Abraham, B. Nath, Eds.), in the Series "Studies in Fuzziness and Soft Computing", Springer-Verlag, Heidelberg, Germany, 2002 (in press).

[13] M. Schumacher, R. Rossner, and W. Vach, "Neural networks and logistic regression: Part I'," *Computational Statistics and Data Analysis*, 21, pp. 661-682, 1996.

[14] E. Biganzoli, P. Boracchi, L. Mariani, and E. Marubini, "Feed Forward Neural Networks for the Analysis of Censored Survival Data: A Partial Logistic Regression Approach," *Statistics in Medicine*, 17, pp. 1169-1186, 1998.

[15] R. Kozma, M. Sakuma, Y. Yokoyama, and M. Kitamura, "On the Accuracy of Mapping by Neural Networks Trained by Backporpagation with Forgetting," *Neurocomputing*, Vol. 13, No. 2-4, pp. 295-311, 1996.

[16] M. Ishikawa, "Structural learning with forgetting," *Neural Networks*, Vol. 9, pp. 509-521, 1996.

[17] S. Haykin, *Neural Networks*, Prentice Hall Upper Saddle River, New Jersey, 1999.

[18] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, 7:219–269, 1995.

[19] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *D. S. Toureczky, ed. Adavnces in Neural Information Processing Systems 2 (Morgan Kaufmann)*, pp. 598-606, 1990.

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

[21] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Automatic Control*, Vol. 19, No. 6, pp. 716-723, 1974.

[22] J. Rissanen, "Modeling by shortest data description," *Automat.*, Vol. 14, pp. 465-471, 1978.

[23] C. Cortes and V. Vapnik, "Support vector machines," *Machine Learning*, 20, pp. 273-297, 1995.

[24] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines (and other kernel-based learning methods)*, Cambridge University Press, 2000.

[25] B. Scholkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik, "Comparing support vector machines with gaussian kernels to radial basis function classifiers," *IEEE Trans. Sign. Processing*, 45:2758 – 2765, AI Memo No. 1599, MIT, Cambridge, 1997.

[26] K.-R. Muller, S. Mika, G. Ratsch, and K. Tsuda. "An introduction to kernel-based learning algorithms," *IEEE Transactions on Neural Networks*, 12(2):181-201, 2001.

[27] T. T. Friess, N. Cristianini, and C. Campbell, "The kernel adatron algorithm: a fast and simple learning procedure for support vector machine," in *Proc. 15th International Conference on Machine Learning*, Morgan Kaufman Publishers, 1998.

[28] J. K. Anlauf and M. Biehl, "The Adatron: an adaptive perceptron algorithm," *Europhysics Letters*, 10(7), pp. 687–692, 1989.

[29] *Matlab User Manual*, Release 6.0, Natick, MA: MathWorks, Inc, 2001.

Figure Legends:

Figure 1: Correct classification rates for MLP with one hidden layer. The dotted line shows results with different number of hidden nodes using structural learning. The solid line shows results of Logistic regression projected out for comparison. Both lines assume the best splitting of data for each node as reported in Tables 4 and 5.

Figure 2: Errorbar plots with means (circle) with unit standard deviations and medians (star) of the correct classification rates for MLP with one hidden layer (H=15), Logistic Regression (LR), and SVM.

Figure 3: Typical MSE of training (dotted line) and cross-validation (solid line) with the best MLP with one hidden layer (H=15, training set size=50%).

Figure 4: Correct classification rates for MLP with different number of hidden nodes using 50% training set size for four different algorithms. Dotted line: Back-propagation with Momentum algorithm. Dash-dotted line: Conjugate Gradient Descent algorithm. Dashed line: Quickprop algorithm. Solid line: Delta-Delta algorithm.

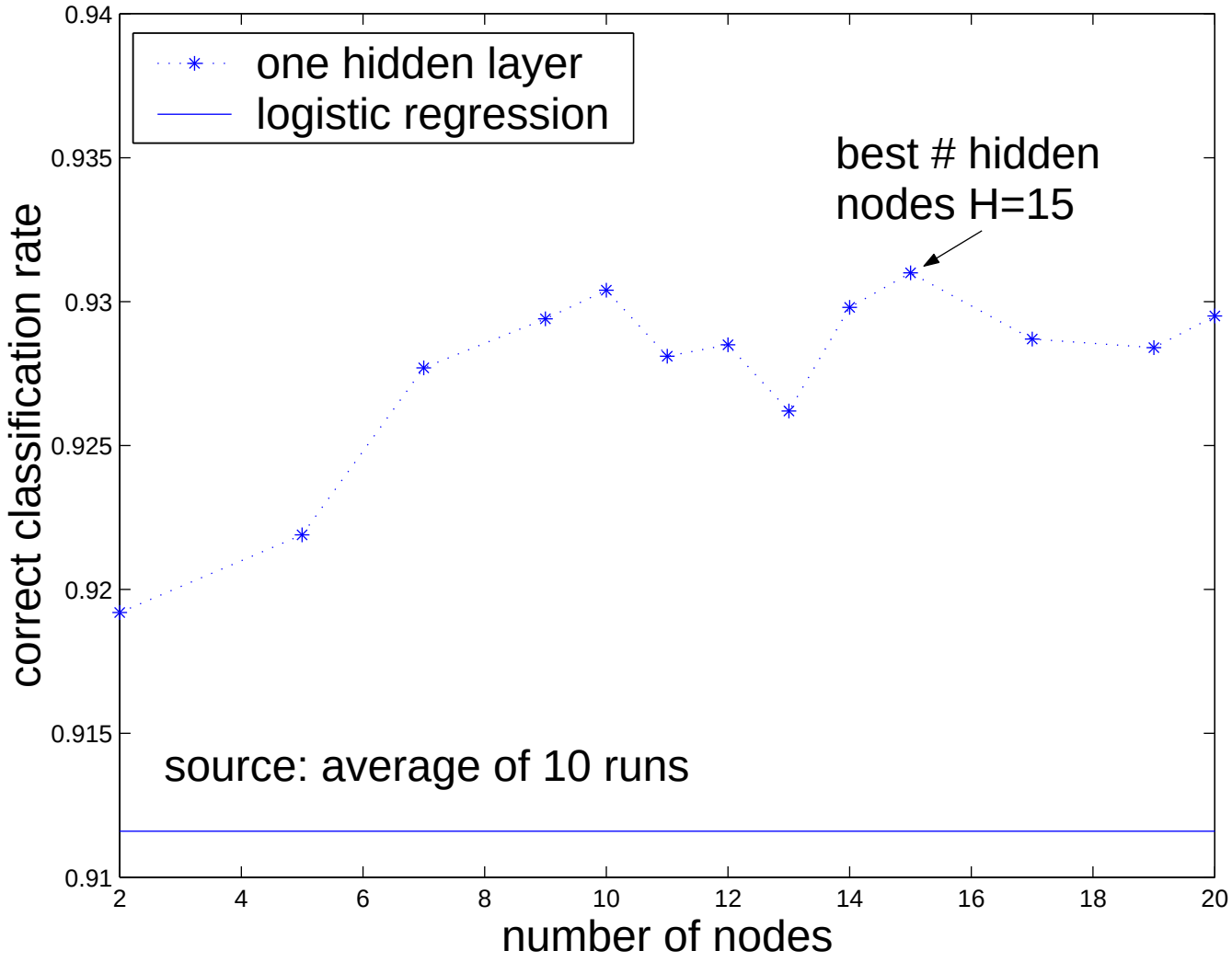

Figure 1: Dotted line: correct classification rates for MLP with one hidden layer. Solid line: Logistic regression (LR).

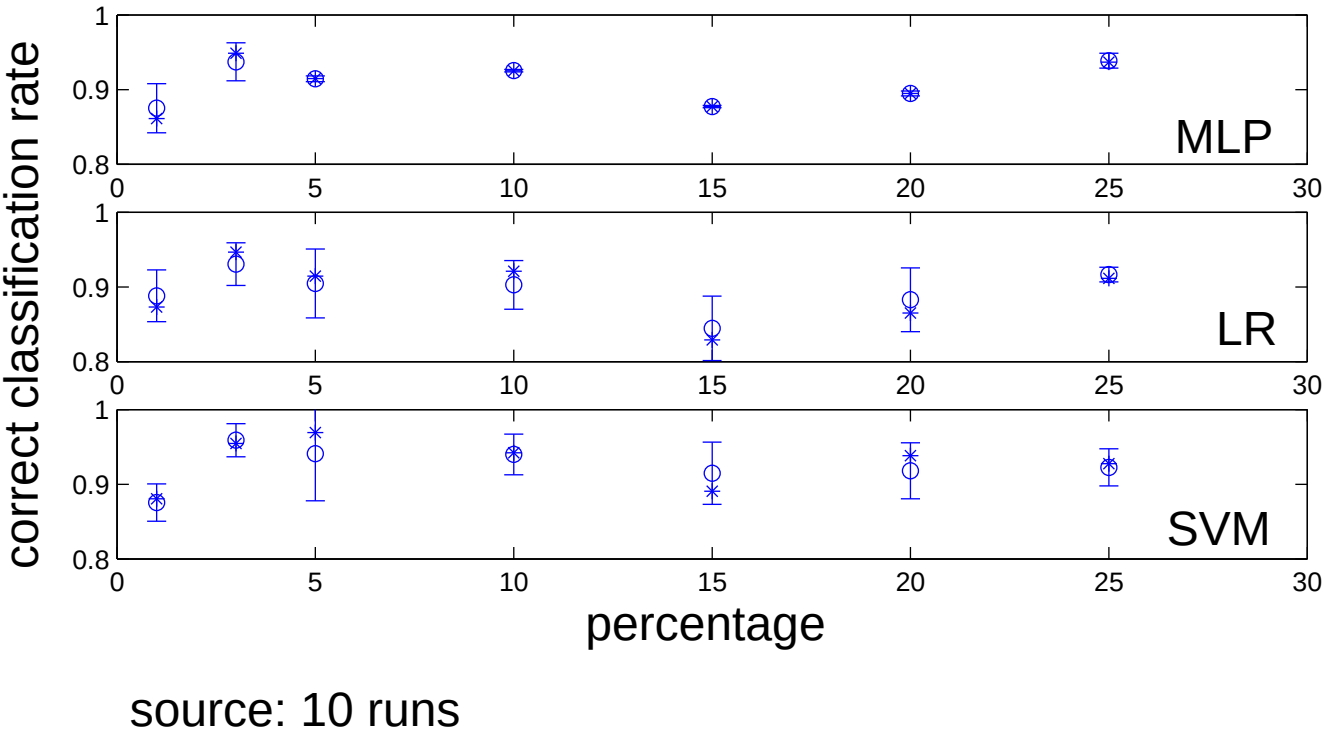

Figure 2: Errorbar plots with means (circle) with unit standard deviations and medians (star) of the correct classification rates for MLP with one hidden layer (H=15), Logistic Regression (LR), and SVM

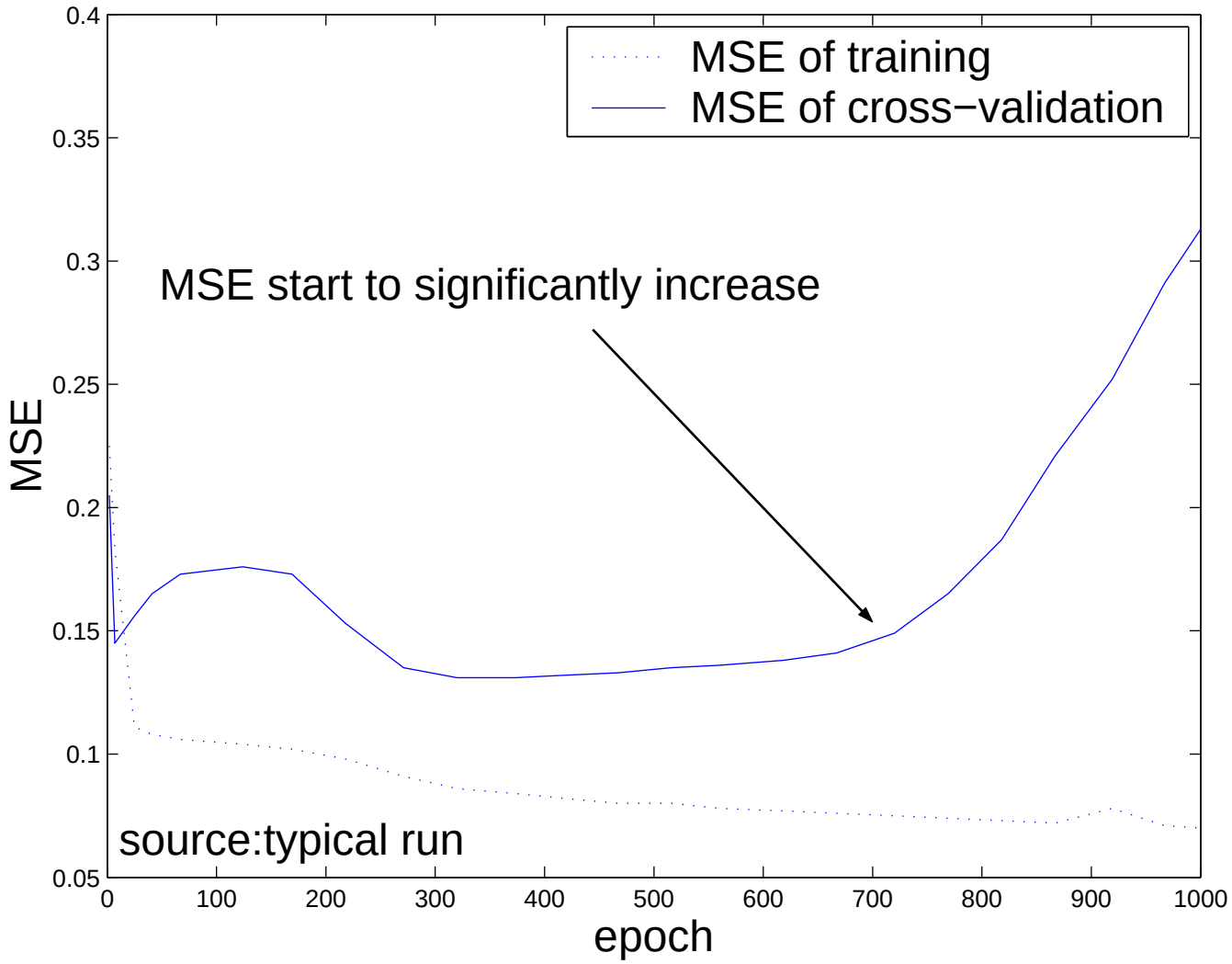

Figure 3: Typical MSE of training (dotted line) and cross-validation (solid line) with the best MLP with one hidden layer

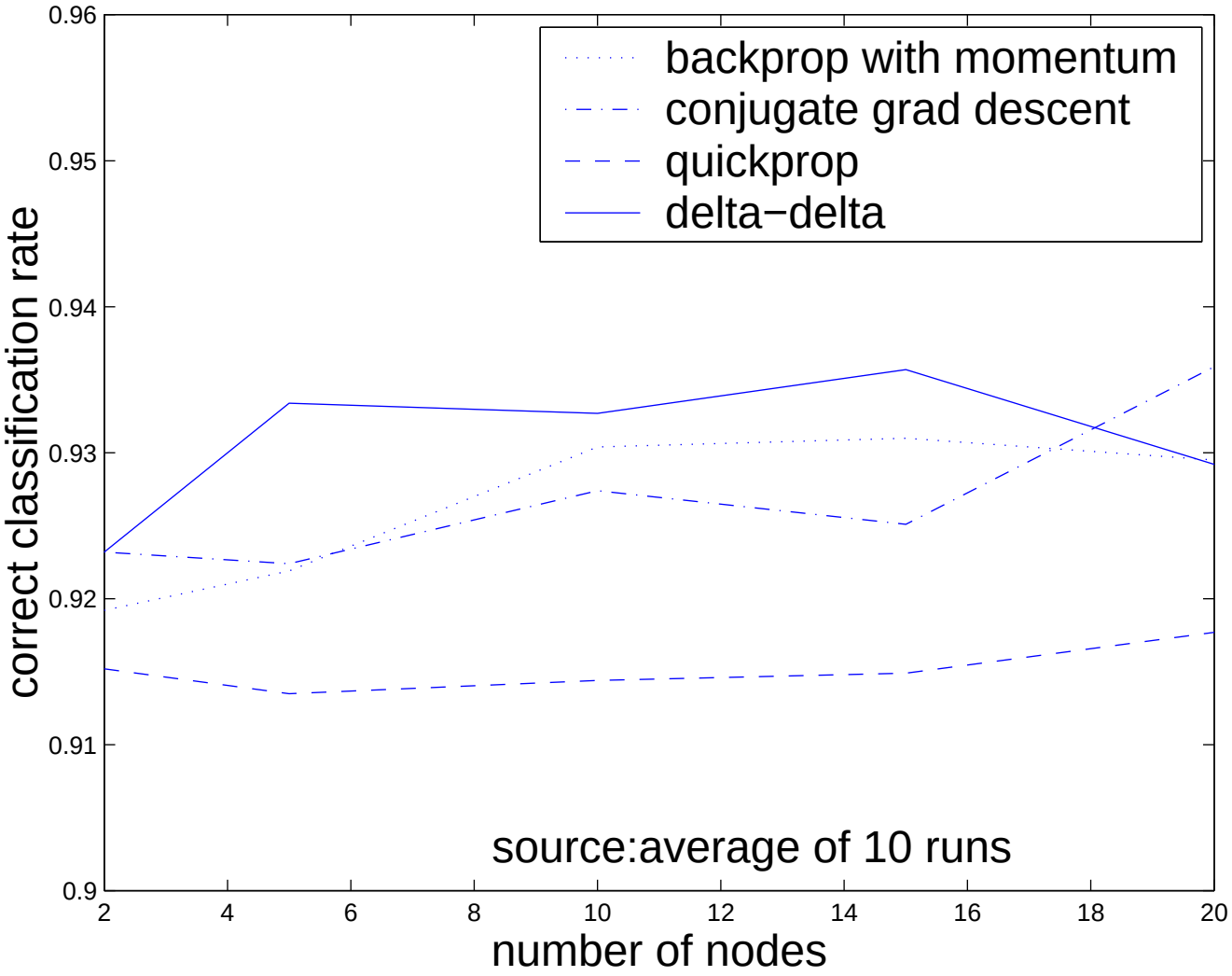

Figure 4: Dotted line: Back-propagation with Momentum, Dash-dotted line: Conjugate Gradient Descent, Dashed line: Quickprop, Solid line: Delta-Delta algorithm

Table 1: Hard constraints

Function	Policy Name	Policy
c_1	Sea/Shore Rotation	If a sailor's previous job was on shore then he/she is only eligible for jobs at sea and vice versa
c_2	Dependents Match	A sailor with more than 3 dependents is not eligible for overseas jobs
c_3	Navy Enlisted Classification (NEC) Match	The sailor must have an NEC (trained skill) that is required by the job
c_4	Paygrade Match (hard)	The sailor's paygrade cannot be off by more than one from the job's required paygrade

Table 2: Soft constraints

Function	Policy Name	Policy
f_1	Job Priority Match	The higher the job priority, the more important to fill the job
f_2	Sailor Location Preference Match	It is better to send a sailor to a place he/she wants to go
f_3	Paygrade Match (soft)	The sailor's paygrade should exactly match the job's required paygrade
f_4	Geographic Location Match	Based on locations certain moves are more preferable than others

Table 3: Estimated coefficients for the soft constraints

Coefficient	Policy Name	Estimated Value
w_1	Job Priority Match	0.316
w_2	Sailor Location Preference Match	0.064
w_3	Paygrade Match	0.358
w_4	Geographic Location Match	0.262

Table 4: Factorial array for model selection for MLP with structural learning with correlated group data: values of MDL and AIC up to 1000 epochs, according to Eqs. (7) and (8)

hidden nodes	Size of testing set				
H	5%	10%	15%	20%	25%
2	**-57.2/-58.3**	-89.9/-96.3	-172.3/181.7	-159.6/-171.1	-245.3/-258.5
5	**-12.7/-15.3**	-59.5/-74.7	-116.4/-138.9	-96.5/-124.3	-188.0/-219.7
7	**13.9/10.3**	-48.9/-27.8	-93.5/-62.2	-62.3/-100.9	-161.0/-116.9
9	46.0/41.5	**7.4/-21.6**	-22.1/-62.2	-15.1/-64.4	-91.7/-148.1
10	53.7/48.6	-15.1/-64.4	63.4/19.0	**10.3/-41.9**	-64.9/-127.6
11	70.1/64.5	41.1/8.3	152.4/103.6	**29.2/-30.8**	-52.4/-121.2
12	85.6/79.5	60.5/24.6	39.9/-13.2	44.5/-21.0	**-27.7/-102.7**
13	99.7/93.1	73.4/34.5	66.0/8.4	80.8/9.9	**-14.2/-95.4**
14	120.4/113.3	90.8/49.0	86.6/24.6	101.4/25.1	**20.8/-67.6**
15	131.5/123.9	107.0/62.7	95.6/29.2	113.9/32.2	**38.5/-58.4**
17	166.6/158.0	138.0/87.4	182.4/107.2	149.4/56.9	**62.2/-26.6**
19	191.2/181.7	181.5/124.9	166.1/109.5	185.8/82.6	**124.0/5.7**
20	201.2/191.1	186.6/127.1	231.3/143.0	193.2/84.5	**137.2/12.8**

Table 5: Correlation Coefficients of inputs with outputs for MLP

Number of Hidden Nodes	Correlation Coefficient	Size of training set
2	0.7017	90%
5	0.7016	90%
7	0.7126	90%
9	0.7399	80%
10	0.7973	60%
11	0.8010	60%
12	0.8093	50%
13	0.8088	50%
14	0.8107	50%
15	0.8133	50%
17	0.8148	50%
19	0.8150	50%
20	0.8148	50%