

Working Paper No. 10
18 November 1997

Original: ENGLISH ONLY

**STATISTICAL COMMISSION and
ORGANIZATION
ECONOMIC COMMISSION FOR EUROPE**

INTERNATIONAL LABOUR

CONFERENCE OF EUROPEAN STATISTICIANS

Joint ECE/ILO Meeting on consumer price indices
(Geneva, 24-27 November 1997)

Item 9 of the provisional agenda

DATA EDITING WITH ARTIFICIAL NEURAL NETWORKS

Paper submitted by Statistics Norway

Data editing with Artificial Neural Networks

Paper submitted by Statistics Norway

1. The problem

The task of producing price statistics always makes us face the challenge of what to do when we are not supplied with a completely responding sample. One subdivision of the problems posed by this fact is: what should be assumed on behalf of the units not responding? Several methods are available to impute non-response: cold deck, hot deck and functional correction to mention some. Felligi and Holt (1976) made significant contributions to automated editing of survey data. In this paper we will take a different approach to automated editing, attempting to apply Artificial Neural Networks (ANN) to the task of imputing missing observations.

Traditional econometricians have for some time made use of numeric optimization. This knowledge is now being reintroduced to us in a new context; Artificial Neural Networks. As opposed to traditional imputation technics the use of ANN does not necessitate the specification of detailed editing rules or explicit assumptions about the imputation models. In this paper we will attempt to assess the usefulness of this class of algorithms in imputing (estimating) missing values in producing economic statistics.

The present paper is the preliminary results from an investigation into different possible methods for imputing missing observations and extreme observations in the Norwegian Rent Survey.

We shall start by giving a short introduction to the theory of ANN. Next we will give a brief outline of the Rent Survey from which the data used originate. We will then train and apply an ANN in order to impute missing rents. The results are evaluated, and tentative conclusions on the performance of the method are reached in the final sections.

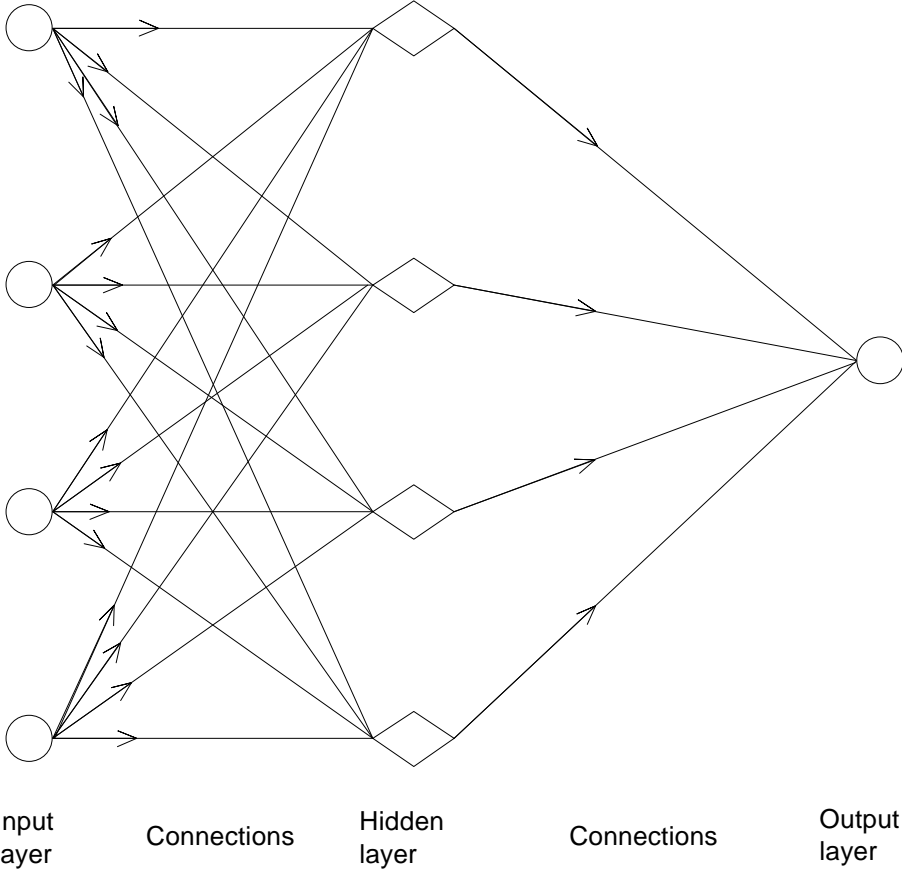
2. Artificial Neural Networks

In this chapter we shall give a brief introduction to the theory of ANN, upon which our exercise on the survey data rests. For the ease of exposition we will concentrate on a single layer network in this chapter, although our application involves a two-layer network.

In this paper we apply an ANN to the problem of imputing missing prices (rents). The ANN we have designed for this purpose is made up of a fully connected network, with two hidden layers of neurons, and the learning algorithm used is known as the Back-Propagation Algorithm. This algorithm belongs to a larger group, known as *feed-forward networks*.

A network with one hidden layer can be visualized as in Fig. 1.

Fig. 1 A single layer artificial neural network.



The network is said to be fully connected because all elements of the input vector is connected to all the neurons, and all these neurons are in turn connected to the next layer. It is possible to manipulate the performance of the network by making it partly connected. We have not investigated these opportunities.

The catagorization *feed-forward* is used for networks not having any output of an element influencing in part, or fully, the input applied to that particular element.

The diamond-shaped boxes in figure 1 represents the neurons, or transfer functions as they are also referred to. These functions take their inputs from the previous layer in the network, and produce an output which again is used as input in the next layer. By numbering the neurons from 1 to k, letting x_1, \dots, x_p represent the input, w_{k1}, \dots, w_{kp} represent the connections or synaptic weights, and u_k be the net input to neuron k, we might express the neuron by two functions:

$$(1) \quad u_k = \sum_{j=1}^p w_{kj} x_j$$

and the input u_k (less a threshold θ_k) is used as argument in the activation- or transfer function f

$$(2) \quad y_k = f(u_k - \theta_k)$$

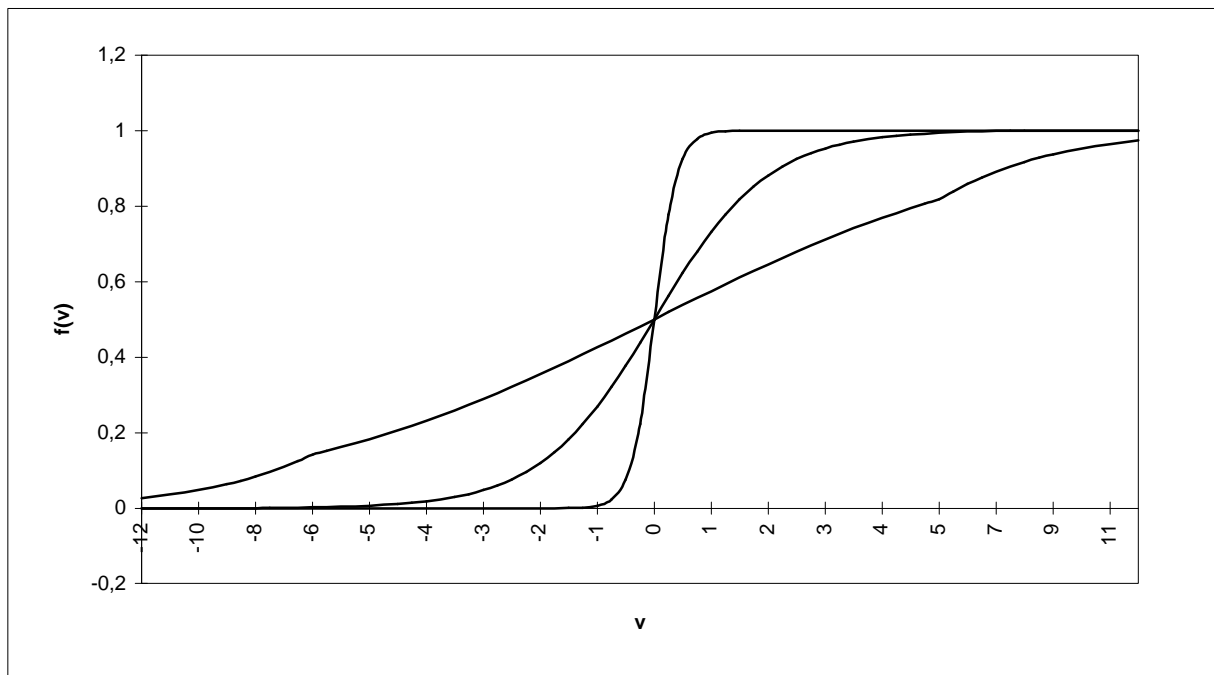
in order to produce the output of the neuron.

The transfer function we are using is the logistic function, which belongs to the class of sigmoid functions. This is a strictly increasing and differentiable function, which is important in the procedure of optimizing the weights of the network.

$$(3) \quad f(v) = \frac{1}{1 + e^{-av}}$$

This function yields a continuous range of values as output, in the interval 0 to 1. The parameter a determines the slope of the function, a higher parameter value resulting in a steeper slope. In our application we use $a=1$.

Figure 2. The sigmoid function as defined in the equation (3), $a=0.3$, $a=1$ and $a=5$.

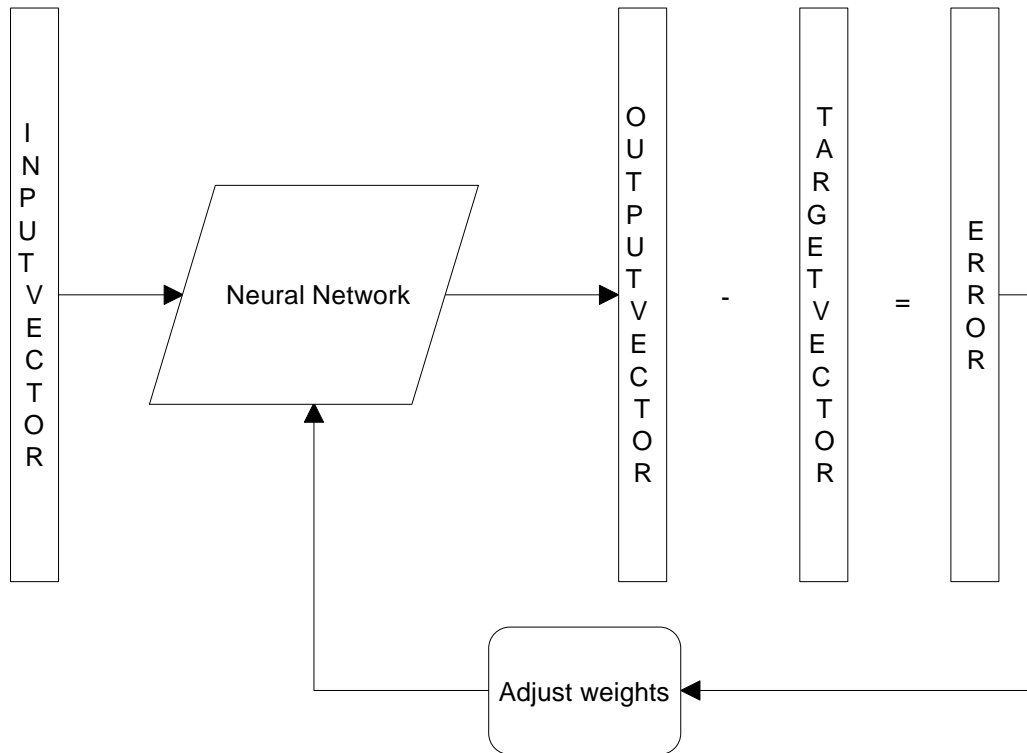


In a multilayer network the output from the first layer of neurons acts as input for the second layer, and so on until the output layer is reached.

The network can be run in two modes, a training mode or a testing mode. When training the network we want it to recognise and learn patterns in the data. When testing the network we supply it with partial observations, leaving it to the network to impute the missing values on basis of the patterns it has learned exists in the data.

In training the network we want to optimize the weights so that the network produces an output wich is

Figure 3. The dynamics of the error back-propagation algorithm.



The error signal at the output of neuron j at observation n is defined by

$$(4) \quad e_j(n) = d_j(n) - y_j(n)$$

where d represents the desired output, and y represents the actual output from neuron j . We assume that j is an output neuron. The squared error for neuron j is $\frac{1}{2} e_j^2(n)$. To obtain an error-measure for the whole network we sum up these over all the neurons in the output layer:

$$(5) \quad e(n) = \frac{1}{2} \sum_C e_j^2(n)$$

The set C includes all the neurons in the output layer. (5) represents the instantaneous sum of squared errors of the network for observation n . Next we generalize this further to incorporate the dimension of the observations. We then achieve an expression for the average squared error:

$$(6) \quad \bar{e} = \frac{1}{N} \sum_{n=1}^N e(n)$$

N is the total number of observations fed to the network in the learning mode. We now optimize the network by minimizing $\bar{\epsilon}$ with respect to the weights. To do this we will apply an algorithm which updates the weights on an observation-by-observation basis. That is: the weights will be adjusted for each observation in accordance with the respective errors computed for each observation as these are fed into the network. In this manner we achieve an estimate of the weight-change we would have obtained by minimizing $\bar{\epsilon}$. Assume we have neuron j being fed by a set of function signals originating from a «previous» layer of neurons, $y_i(n)$. By letting p denote the number of inputs, the magnitude of the input to the nonlinear neuron j can be written as:

$$(7) \quad v_j(n) = \sum_{i=0}^p w_{ij}(n)y_i(n)$$

By feeding this into the activation function we obtain the output of neuron j at iteration n as:

$$(8) \quad y_j(n) = f_j(v_j(n))$$

In optimizing the weights to reduce $\epsilon(n)$ we will apply a change to the weights in the direction determined by the gradient, the size of the change will be determined by a constant η (the learning rate):

$$(9) \quad \Delta w_{ji}(n) = -\eta \frac{\nabla \epsilon(n)}{\nabla w_{ji}(n)}$$

It is possible to use equations (4) - (6) to re-express this as:

$$(10) \quad \begin{aligned} \Delta w_{ji}(n) &= \eta y_i(n) \left[- \frac{\nabla \epsilon(n)}{\nabla e_j(n)} \frac{\nabla e_j(n)}{\nabla y_j(n)} \frac{\nabla y_j(n)}{\nabla v_j(n)} \right] \\ &= \eta y_i(n) d_j(n) \end{aligned}$$

Here $\delta_j(n)$ is the local gradient which can be written as

$$(11) \quad d_j(n) = e_j(n) f_j'(v_j(n))$$

We observe that the change in weights is determined by the learning rate, the neuron output, the error signal, and the derivative of the transfer function. All of these have to be evaluated for each observation presented to the network.

When looking at multi-layer networks this is further complicated by the credit-assignment problem. That is: how to distribute the error at the output node between the hidden neurons. When neuron j is located in a hidden layer there is no specified desired response for that neuron. This is solved by deriving an appropriate local gradient. This will incorporate the error signals from the layers succeeding the hidden neuron and the weights connecting the two layers. In this context we will not complicate things further by going into details on the multilayer networks, but stop the introduction here where we hope the idea of optimizing a feed-forward network is understood. But we shall note that learning algorithms for multilayer ANN's with acceptable properties has been available only in the last 10-12 years.

The back-propagation algorithm is run by first doing a forward pass for each observation. This is done simply by calculating the activation functions, neuron by neuron, for each layer, starting by feeding the observation to the first hidden layer, ending with the computation of an error signal for each neuron in the output layer. The weights, which are given small initial random values, are kept constant in this forward pass. Once this pass is completed we know the error signals necessary to alter the weights in an educated direction. This is done in the backward pass by passing the error signals backwards through the network, layer by layer, always computing the local gradient for each neuron. The weights will converge to values which represent the knowledge the model has learned.

This process is run, observation by observation, for the whole training set of observations. The order of the observations is changed by a random mechanism before the next round of learning (iteration) is run with the same set of observations. This will continue until some stopping criterion are fulfilled. One specifies an upper level for what is an acceptable total mean squared error. The learning process will iterate until this condition is satisfied, or until a alternative stopping criterion (maximum number of iterations) is fulfilled. The iterative search for the optimal set of weights will in most applications require a lot of computing time, increasing with the number of observations, the number of layers and the number of neurons in each layer. It is the training mode which requires a lot of computing recourses, the testing mode demands less recourses. Today's powerful processors are more and more making this a non-problem.

What we have described so far in this chapter is the learning mode. The network can also be used in a testing mode. In testing - the network is used to compute output vectors on the basis of input which where not used in training. If there exists a true set of output vectors, the performance of the network can be evaluated by comparing the computed values with the true values. In part 5 of this paper we will attempt to do such an analysis.

The method of applying artificial neural networks for imputing in statistical records has some disadvantages. The theoretical analysis of the network is difficult, mainly because of the distributed non-linearity. The intuition of the estimated model is hard to get a grip on. It is possible to investigate the estimated weights, but a multilayer setup makes it difficult to interpret the model. This is in contrast to conventional regression models.

3. The Norwegian rent survey

We shall here give a brief outline of the rent survey from which the data used in the present experiments originate.

This quarterly survey is, with respect to its share of total household consumption, the most important part of the Norwegian Consumer Price Index. As well as the 5,4 % weight from renters, this survey is also used for imputing the change in cost of shelter for home-owners which includes interest payed on mortgage, renovation and other fees and insurance. These components give the Rent Survey a total weight of 13,9 %.

The sample of respondents used in this survey is established by a two stage sampling procedure. In the first step a sample of municipalities is drawn in order to cover the whole country. In the second stage households are drawn from these geographic areas. A total of 6000 households are included in the sample.

In the first quarter a new sample participates in the survey, the individual households connection to the shelter is established. That is, whether it is a home-owner, a renter or an owner through a housing co-operative. All homeowners are excused from the survey after the first quarters, leaving us with a sample of renters and housing co-operative participants. The first questionnaire also surveys what kind of building the respondent is living in, what year it was build, the size (square meters) of the shelter,

whether the rent includes electricity, heating or garage. And of course we ask for the monthly rent. In the following quarters the questions regarding the shelters' characteristics are removed from the form, as these are not allowed to change. For us to make sure we follow the same shelter in the succeeding quarters, and not necessarily the same household, we include a question asking whether the respondent is still living in the same shelter as he did in the previous round.

The index is calculated by the Laspeyres formulae, using a relative of average rents. The stratification in the computations is done by building year. The weights are obtained from the yearly Norwegian Household Expenditure Survey. The weights, which are updated annually, are calculated as a moving average of the last three available expenditure shares from the Household Expenditure Survey.

On average we obtain approximately 1300 rent observations each quarter. The rent index is published quarterly, as a total for all kinds of dwellings for the entire country.

For the application in this paper we have transformed the data on the categorical questions into binary variables. The quantitative variables have been normalized, so they all have values in the region (0,1). The data used are from the 4. quarter of 1996, including shelter characteristics and rents for December, and 1. quarter of 1997 where only rent information for March is used. In total we have 1059 observations in the sample used in this paper.

4. ANN applied on real survey data

4.1 The experiment

The dataset to be used in this analysis is expert-edited data from the Rent Survey covering 4th quarter of 1996 and 1st quarter of 1997. In total we have 1059 observations, when we have removed observations not complete enough for our experiment. This data material is then «inflicted» with a partial non-response for the rent for March 1997.

By means of a random device we divide the data in two separate files: This separation is done in order to simulate a real production of the consumer price index. Approximately 90 % (969 observations) of the manually expert revised observations are assigned to training the ANN. The remaining 10 % (90 observations) of the observations are subject to a partial non-response in the variable «rent for March».

We are then actually simulating total non-response in the 1st quarter of 1997 of 90 units. This is quite a realistic simulation, where the collected and revised data form the basis for imputing the non-response.

This procedure will supply the material with automated imputations based on manually revised data. By starting out with a complete dataset, and impose errors (non-response) we will be able to form an opinion on the success of the methods applied. By comparing the imputed rents with the «true» rents originally on the dataset, we will be able to decide whether there are *significant differences*. If not, this points in the direction that we can assume that the use of ANN is a method of imputation that we recommend for this survey.

In designing the network, we have chosen to use two hidden layers of neurons because of the superior learning ability in contrast to a network with only one hidden layer. The multilayer network has proved to have a greater ability to extract higher order patterns from the data material. When electing between a network with two layers, and networks of higher orders, empirical evidence indicates that the improvement in performance when increasing beyond two layers is negligible in many applications. On this basis we have limited this investigation to networks with two hidden layers of neurons.

At present there exists little theoretical support in how to set up an ANN. The specifications we have made are heavily based on empirical evidence. The learning rate is set to 0.25, the hidden layers both consists of 20 neurons. The weights are initially given random weights in the area -0.5, 0.5. As a stopping criteria we use a maximum number of iterations of 10000, or a total mean squared error of 0.002.

The training was done surprisingly fast, using a Unix machine. We believe this is caused by the fact that our network had one variable only in the output vector. The 46 variables of each input vector (of which 44 were binary) then only has one value to aim at. This will speed up the convergence to a small error considerably compared with an experiment with more variables on the output vector.

4.2 Evaluation of the ANN-edits

We shall here try to assess the performance of the automated edits. First we will investigate whether the ANN-edits are significantly different from the target values.

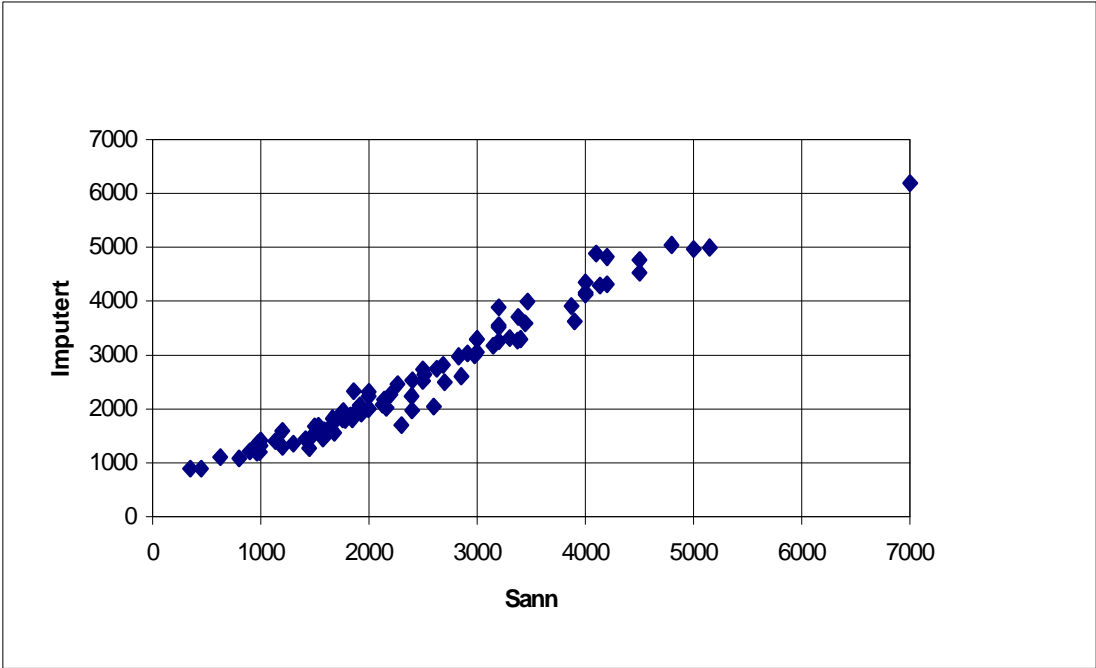
Our rentindex is calculated as an (arithmetic) relative of means of prices:

$$(12) \quad I_t = \frac{\sum_i P_{i,t}}{\sum_i P_{i,t-1}} \quad , i=1,2,\dots,n$$

here p represents price observation i in the time period t . This implies that the interpretation of *do not differ significantly* should be *does not produce an average that differs significantly*. By assuming that the two sets a) the imputed prices, and b) the «true» survey observations, are independently distributed we are able to construct a 98% confidence interval for the differences in the mean price in the respective populations. Our calculations produce a 98% confidence interval for the difference in means between the two populations with a lower limit of -525.15 and an upper limit of 305,88. On this basis we can reject the hypothesis that there are differences between the averages of the two populations.

Below we also include a graph where the prices as imputed by the ANN are plotted against the «true» target values. Visual inspection of the graph reveals no systematic deviations between the two series'. There might be some indications that the network underestimates prices in the area of 2500 kroner, and overestimates rents in the area 4200 kroner.

Figure 3. Imputed values vs «true» target values.



We will also calculate three different indices for the rent change: A) for the complete material (assuming no non-response), B) for the 90 % of the observations not allocated to non-response (implicitly imputing missing rents with the mean of the remaining observations), and C) using the material obtained by using the ANN-algorithm. All observations are equally weighted. This yields the following results:

$$I^A = 1,00788$$

$$I^B = 1,00768$$

$$I^C = 1,01188$$

The results indicate that giving no explicit treatment of non-response, gives a result closer to the «true» index than using our ANN to impute the missing values.

This might be altered if several quarters data were pooled to give the network a larger sample to learn from. One should also bear in mind that our network was set up to impute rent - not rent *change*. An alternative formulation where the ANN was trained in patterns determining the rate of change would be an interesting further development of our study.

5. Conclusions

The specification of a ANN for automated editing procedures has to be based on trial and error to an extensive degree. This makes the setup somewhat arbitrary, making it difficult to conclude on the performance of ANN's in general. Our experiences therefore have to be limited to the setup we have chosen.

The training of the network was a lot faster than we had expected. This should make the method more available as CPU-resources are not big bottlenecks.

The performance of the ANN is not overwhelming. It seems that it imputes levels better than change. This might be overcome by an alternative formulation of the problem, setting up the net to learn the rate of changes and not the prices themselves. Supplying the ANN with a larger number of observations to learn from the patterns between price and the characteristics, will also possibly improve the performance in imputing for the calculation of index numbers.

6. References

Fellegi, I.P. and Holt, D. (1976) A Systematic approach to Automatic Editing and Imputation. *Journal of the American Statistical Association*, 71.

Fletcher, R. (1987) *Practical Methods of Optimization*, 2nd ed., Wiley

Haykin, S. (1944), *Neural Networks - A Comprehensive Foundation*, Prentice Hall.

Nordbotten, S. (1996), Editing Statistical Records by Neural Networks. *Journal of Official Statistics*, No 4.