Chapter 10

Neural Networks

Chair: Ray Board, Federal Reserve Board

L. H. Roddick

L. H. Roddick

Svein Nordbotten

Chapter 10

Data Editing Using Neural Networks

L. H. Roddick, Statistics Canada

Abstract

Computerized data editing is normally done using rule based systems or programs. Neural Networks provide the ability to create an edit system directly from the data and to allow the edits to evolve over time through periodic retraining. This is an especially useful facility for monthly surveys, where the edits can adapt over time as the external conditions affecting the survey change.

Data Editing Using Neural Networks

L. H. Roddick, Statistics Canada

Introduction

Data editing is a major activity at Statistics Canada and at other statistical agencies. Computerized data editing is normally accomplished either by producing a customized edit program or by using a generalized editing system. A customized approach involves specifying the edits, programming the specifications, testing the program's correctness and then implementing the edit program. A generalized editing system usually requires the user to specify the edits in terms of some complete set of acceptance or rejection rules (Fellegi and Holt, 1976). In both techniques the edits are implemented in terms of rules.

The Neural Network technique described herein addresses editing data based on the actual data, not on rules about the data. Only one facet of statistical data editing is described, but it is thought that non-statistical areas might also make use of this type of Neural Network.

Two types of data are normally edited, discrete valued variables and continuous valued variables. Discrete variables have a limited number of values that are either alphabetically or numerically coded. Continuous variables are numeric variables such as integers.

The Neural Network model presented here addresses discrete variables, however it can be used for continuous variables if they are grouped by value into classes. For example, an income variable can be grouped into a number of income classes such as $0 -- $20,000, $20,001 -- $40,000 and greater than $40,000, thus making the resulting coded income variable discrete.

The Process

Editing of discrete variables can be accomplished using Back-Propagation (B-P) Neural Networks without having to go through the specification, programming and testing phases. This is accomplished by providing the set of correct and incorrect values for the variables being edited to a data generation program that will generate the training set for the Neural Network. The Neural Network is trained on this generated data and the resulting Neural Network module is incorporated into a capture or edit system.

This technique only addresses the inter-field edits within a record. It is not applicable to inter-record edits where a model different from the one proposed would be required.

Although a record may have many fields (numbering even into the hundreds), normally only a few fields (fewer than 15) interact at a time. Thus the inter-field edits can be grouped according to sets of fields to be edited together. The edits for each of these groups is distinct and separable and thus a

separate Neural Network can be trained for each group. This does not mean the proliferation of many programs to edit a record. Once each edit Neural Network is trained, it can become a module of the overall edit system or can be introduced as a module of an interactive capture system to edit the records as they are being keyed.

When this technique was originally tested, it was envisaged that data generation of the complete set of possible values would be the most convenient mechanism for the user to train the Neural Network. It is with this in mind that the example for the paper was generated, however, after discussion of the mechanism it became apparent that there are better ways to construct the training set. Alternative methods of constituting the training set are discussed after the Generated Data example.

Important to data generation is the fact that the set of correct answers for discrete variables, in comparison to the set of all answers, is normally quite small. For example, if three variables interact and each variable has twenty values, there are 8,000 possible answers, although normally only a small number of these is correct. This makes it expedient to specify only correct answers to the training data generation program and allow it to generate the incorrect set. It is however perfectly feasible to specify the incorrect set and generate the correct set.

There are two important points to make about B-P Neural Networks. First, a B-P Neural Network is much better at distinguishing between the values of separate variables than the individual values within a variable (Lawrence, 1991); i.e. , the network will learn far more from nine binary variables than it will from one coded variable using a scale of 1 to 9. This is the principle that the data preparation program uses when it generates the network training data. This technique is also used to transform the data for presentation to the network for editing.

Second, a B-P Neural Network must be presented with approximately equal numbers of each output case during the learning phase (Klimasauskas, 1993). This is to ensure that the network learns as well about the rare cases as it does about the more frequent cases. Since the correct cases are normally less numerous that the incorrect cases, the correct cases must be replicated enough times to create equal representation in the training set. Once the network is trained, it's weights are frozen and no longer change, therefore there is no need for this equal representation in the data being edited.

An Example

The B-P Neural Network data editor is presented by way of an example. Assume that there are three variables that interact, Var1, Var2 and Var3, and that these variables have 5, 9 and 9 possible values respectively. The three variables in combination can produce 405 different answers. An arbitrary set of correct answers for the three variables was chosen and are specified in Table 1. These combinations of values produce 17 correct answers and 388 incorrect answers.

Table 1.--Correct Variable/Value Combinations

Var1	Var2	Var3
1	3 or 7	6 or 9
2	1 or 2	2 or 7
3	4 or 5	1 or 3
4	6 or 8	4 or 5
5	9	8

In order to balance the training set the data generation program created 20 copies of the correct answers and 1 copy of each incorrect answer. This produced a training set of 728 records consisting of 340 correct and 388 incorrect records.

The data generation program transformed the codes for each variable into a set of binary variables with one binary variable for each possible data variable answer. This transformation produced 5 binary variables for Var1, 9 binary variables for Var2 and 9 binary variables for Var3. For each value that occurred in Var1, Var2 or Var3, the corresponding binary variable for the value was set to 1, all the other binary variables were set to 0.

For each record, the data generation program created two binary result fields, Correct and Incorrect. When a correct record was to be generated the Correct field was set to 1 and the Incorrect field was set to 0. When an incorrect record was to be generated Correct was set to 0 and Incorrect was set to 1.

The data generation program generated 728 training records, each with 25 binary variables. The list of binary variables, the data variable from which they were generated and the data variable value that produced a 1 in the binary variable are specified in Table 2.

Table 2.--Network Training Record Layout and Inputs

Binary Variable	Generated From
Var1b1	Var1 = 1
Var1b2	Var1 = 2
Var1b3	Var1 = 1
Var1b4	Var1 = 1
Var1b5	Var1 = 2
Var2b1	Var2 = 1
Var2b2	Var2 = 2
Var2b3	Var2 = 3
Var2b4	Var2 = 4
Var2b5	Var2 = 5
Var2b6	Var2 = 6
Var2b7	Var2 = 7
Var2b8	Var2 = 8
Var2b9	Var2 = 9
Var3b1	Var3 = 1
Var3b2	Var3 = 2
Var3b3	Var3 = 3
Var3b4	Var3 = 4
Var3b5	Var3 = 5
Var3b6	Var3 = 6
Var3b7	Var3 = 7
Var3b8	Var3 = 8
Var3b9	Var3 = 9
Correct	Assigned
Incorrect	Assigned

The data generation program also produced a test data set containing 405 records, one for each possible case. These records were generated in the same way as the training data, including the answers (correct or incorrect), except that there was no need to expand the number of correct cases. During the testing phase, the Neural Network did not use the answers, however it wrote these answers to the output file along with its predictions. These answers were compared to the predictions to do the analysis of the Neural Network results.

Several Back-Propagation Neural Networks were constructed and tested using the NeuralWare Inc. NWorks commercial Neural Network to find a reasonable working version. A Neural Network as specified in Table 3 was able to predict the test set answers correctly all of the time. Since the test set had all possible cases, this Neural Network could be used as an editor for the three variables as they were described.

Table 3.--Back-Propagation Neural Network Parameters

Control Strategy:	Back-Propagation
Type:	Hetero-Associative
Input PEs:	23
Hidden Layer PEs:	23 in 1 hidden layer
Output PEs:	2
Input Ranges:	0 -- 1, using a min./max. table
Output Ranges:	0 -- 1, using a min./max. table
Transfer Function:	Input & Bias layers -- Linear Hidden & Output layers -- TanH
Learning Rule:	Input & Bias layers -- None Hidden & Output layers -- Norm-Cum_Delta
1st Learning Coefficient:	Hidden layer -- 0.3 Output Layer -- 0.15
2nd Learning Coefficient:	Hidden layer -- 0.4 Output layer -- 0.4
Learn Data Presentation:	Shuffle & Deal Randomization without replacement 50,000 presentations

Discussion

A number of points should be made about this data editing technique. First, the specification of the correct data required to generate the training and test sets need not be long and cumbersome. The data generation program input could be in a tabular format with the user specifying only the axis points of interest and allowing the program to generate the correct combinations. This would simplify and minimize the data specification process. It should be noted that if the edit were done conventionally, the user would have to specify all of the correct cases anyway.

Second, the outcome of the Neural Network, i.e., the set of edits is not limited to a correct/incorrect scenario. The outputs could be numerous, reflecting a much more complex relationship in the data being edited. As an example, the desired result could be the stratification of the data based on its inputs, in which case the outputs could be as many as there are strata.

Third, the implementation of these Neural Networks has been made much simpler by the appearance of commercial Neural Network packages on the market. While more complex in nature than a spreadsheet, Neural Networks can be generated and modified with this same spreadsheet type of ease using the new commercial packages.

Although a Neural Network can be generated quickly using commercial packages, this should not be interpreted to mean that Neural Networks are easy to use. The Neural Network concepts and packages have a long learning curve to master and every implementation requires its own analysis and tuning. However, once the expertise is there, the time to produce a Neural Network is very fast. The Neural Network presented in this paper underwent three iterations where it was specified, run and tested each time, this took two person-days to complete.

Finally, the advantage of a data generation technique from the Neural Network point of view is that it is not vulnerable to memorization, a major problem with Neural Networks. When memorization occurs instead of learning, it causes poor generalization of the Neural Network. However since the data generation program has generated all possible cases, it does not matter whether the Neural Network has memorized them or learned them as long as it always gets its prediction right.

Other Applications

An effective use of the Neural Network's ability to learn would be its application to monthly surveys. The conditions under which a monthly survey occurs change over time, for example the edit limits for an economic survey might be much looser during good economic times and much more severe during a recession. It would be desirable for the edits to modify themselves over time as the conditions changed.

The Neural Network could be trained initially on historical data and run on the first month's data. After each month has been run the training set could be augmented with selected data from the previous month and the network retrained. This new set of data could include data that the network found in error, but that follow-up procedures found to be correct due to changing conditions. Depending on the extent of the survey follow-up, data could also be added for records that the network predicted as correct, but that conditions now invalidate.

Another technique to assist in the building of monthly survey training sets is the thermometric encoding of the outputs (Guiver and Klimasauskas, 1991). The output of the training and the test set are predictions of the correctness of the records. The training set answers could be provided as a set of variables which represent the "hotness" (correctness) of the records. The more correct the record, the hotter the answer, thus the network would learn to predict correctness on a gradient scale. This could be very useful in surveys with critical and non-critical populations, where the critical population must all be edited to the "hottest" (most correct) level, but the non-critical population could be edited to a lower temperature. This is one of a number of output encoding techniques used to facilitate different types of learning in Neural Networks.

Acknowledgment

The author would like to acknowledge the Research Fellowship Program at Statistics Canada, without which none of this work would have been accomplished.

References

Fellegi, I. P. and Holt, D. (1976). A Systematic Approach to Automatic Edit and Imputation, *Journal of the American Statistical Association*, 71, 353, Applications Section, 17-35.

Guiver, J. P. and Klimasauskas, C. C. (1991). Applying Neural Networks Part IV: Improving Performance, PC AI, July/August, 34-41.

Klimasauskas, C. C. (1993). Neural Network Development, A Methodology, Course Notes, NeuralWare Inc., Revision 4.2, 173-197.

Lawrence, J. (1991). Data Preparation for a Neural Network, *AI Expert*, November, 34-41. ■

10 Chapter

Editing Monthly Survey Data Using Neural Networks

L. H. Roddick, Statistics Canada

Abstract

Systems which are used to edit survey data should be able to adjust automatically for the changing conditions under which the survey is taken. For example, a set of rules developed to edit respondent data from an economic survey may need to be adjusted as the underlying economy moves from a period of recession to a period of economic expansion. Since the progression from recession to expansion is generally a gradual process, the system should be capable of evolving a set of edits, as changes are observed in the characteristics of the input respondent data set. This requirement is especially relevant in the case of monthly surveys, where time constraints do not normally permit the re-specification, re-programming, and re-testing of the edits. An edit program, based on Neural Network technology, can be developed which is able to evolve its set of edits automatically, as the characteristics of the input survey data set change. Unlike a conventional edit system, it is able to produce automatically a new instance of itself, tailored to the changed survey environment, each month. The paper presents an implementation of an edit system using an evolving artificial neural network.

Editing Monthly Survey Data Using Neural Networks

L. H. Roddick, Statistics Canada

Introduction

An artificial neural network (ANN) can be created to perform an edit of several related variables (Nordbotten, 1993); (Roddick, 1994); and a number of ANN's can be connected together to create an edit system. This editing system can be implemented as a stand alone edit system or as modules that are invoked as part of a data capture system.

An editing ANN must be able to recognize three types of records: correct (CR), incorrect (IC) and don't know (DK) records. An edit will normally involve the interaction of several variables, and when these are discrete variables, the cross product of these variables can be listed empirically. For example, an edit of the variables Mother Tongue versus Ethnicity will produce a large set of possible combinations; however, relatively few of the combinations are CR, a few are known to be IC, and the rest are initially DK. An edit system can be built that will allow the subject matter expert to analyze DK cases as they occur and assign them to either the CR or the IC set. If an occasional unique DK record appears, then there is a likelihood that it is IC, however if a number of records of the same DK appear in a survey occasion, then maybe a new, valid Mother Tongue/Ethnicity combination has been identified. This approach uses data analysis to produce the edit program rather than the traditional specification, programming and testing approach.

Continuous variables can be edited with this form of ANN by creating classes of values for each continuous variable to be edited. For example, an income variable can be grouped into 3 classes: 0-20,000; 20,001-50,000; and > 50,001, and thus the continuous variable can be treated as a 3-value discrete variable.

There are three issues involved in creating an ANN which can edit survey responses and in which the edits can evolve. The first issue is the creation and ongoing maintenance of the ANN training set. The second issue is the training of the ANN, both initially and the monthly retraining as conditions change. The third issue is the assignment of the ANN DK cases to either the CR or IC sets and the periodic re-analysis of the whole training set.

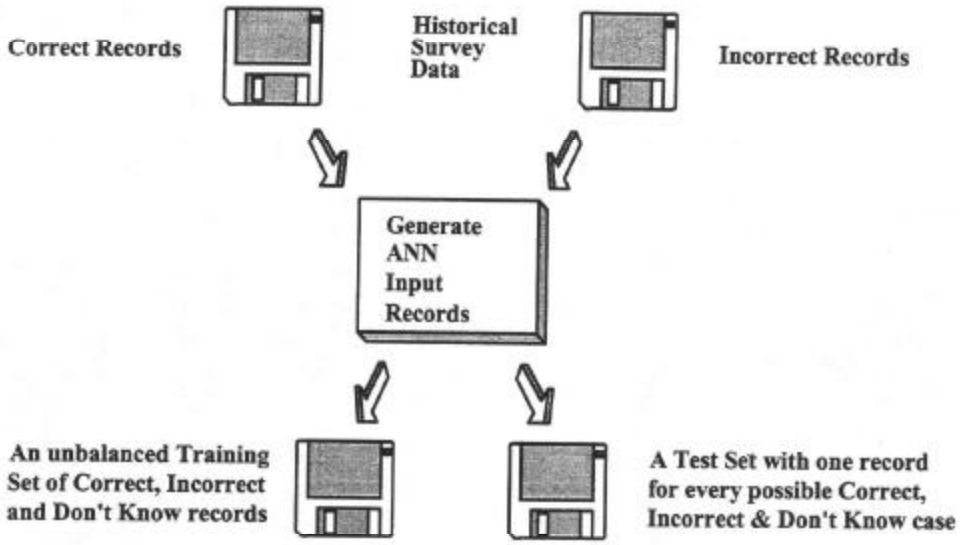
ANN Training Set Creation and Maintenance

The ANN training set can be created initially by either data analysis or by specifying all of the CR and IC cases. For an ongoing survey, the CR cases can be generated from historical edited data, as all the responses on the edited data set should be correct. The IC cases can be generated by comparing the unedited historical data with the edited data and assigning all the cases that exist in the unedited data, but not in the edited data, to the IC set. The DK cases are all of the remaining possibilities and can be

generated automatically. For a new survey, the expert can specify all of the known CR and IC cases, allow the system to assign the remainder to DK and let the first survey occasion generate DK records, which the analyst will then assign to either CR or IC. This may produce a large workload of DK cases on the first survey occasion, but the volume of DK cases should diminish quickly thereafter

The ANN training sets are maintained on a relational database, with a table for each edit, containing all of the training records for that edit. The edit tables contain a column for each variable used in the edit and one column containing the answer for each case: CR, IC, or DK. These edit tables are uniquely keyed by all of the edited variables to ensure that there are no duplicate cases. To ensure that no invalid values are introduced, each variable in the edit table has a codeset of the acceptable values for the variable. This database approach provides access to all of the most current tools for the analysis and maintenance of the edit sets and ensures the completeness and the correctness of the training data.

Figure 1.--Initial ANN Training Set Creation



The question that arises from this approach is: If all of the edit answers are contained in a database table, why not just look up the answers directly instead of using an ANN? The answer is a performance issue. If the edit system is only applying a few edits to a limited number of records and has only a few possible cases, then a lookup table is very feasible. Often however there are many edits and hundreds of possible cases for each edit. An ANN reduces the edit to the execution of a series of mathematical equations with no I/O involved, and is therefore orders of magnitude faster than a lookup approach.

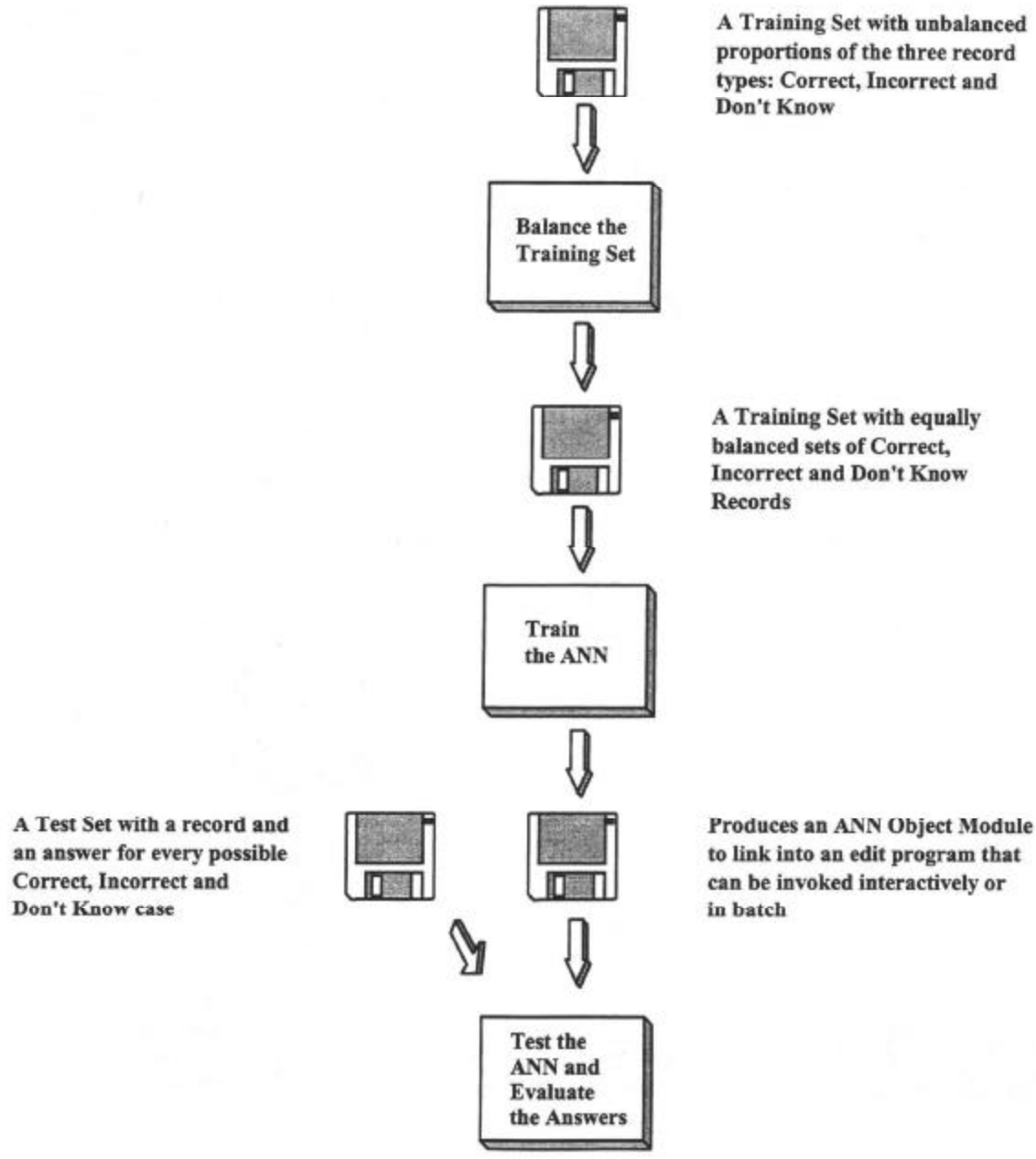
Training and Deploying the ANN

The ANN must be trained initially and then re-trained every time the training set is modified. The training set is generated from the database edit table by transforming the records into a form that the ANN can use. Two functions must be performed on the records; the distance must be established between the values of each variable (Roddick, 1994) and the training set must be balanced to have equal representation of all record types in the training set (Roddick, 1994).

Distance is achieved by mapping each of the variable values to a binary variable. The ANN does not distinguish well between a value of 5 versus a value of 6 in a single variable. However if that single variable is converted to a set of binary variables, with one for each possible value, the ANN distinguishes very well between the binary variables Var5 = 1, Var6 = 0, versus Var5 = 0, Var6 = 1.

Balancing the training set forces the ANN to learn about all of the cases. It involves replicating the rare cases enough times to ensure that the network sees them as often during training as the frequent cases. If 95 percent of the training set were DK cases, the ANN would learn very quickly to always choose DK, and it would be right 95 percent of the time. By making the CR and IC cases occur as often as DK, this forces the ANN to learn what distinguishes CR, IC, and DK.

Figure 2.--ANN Training and Deployment

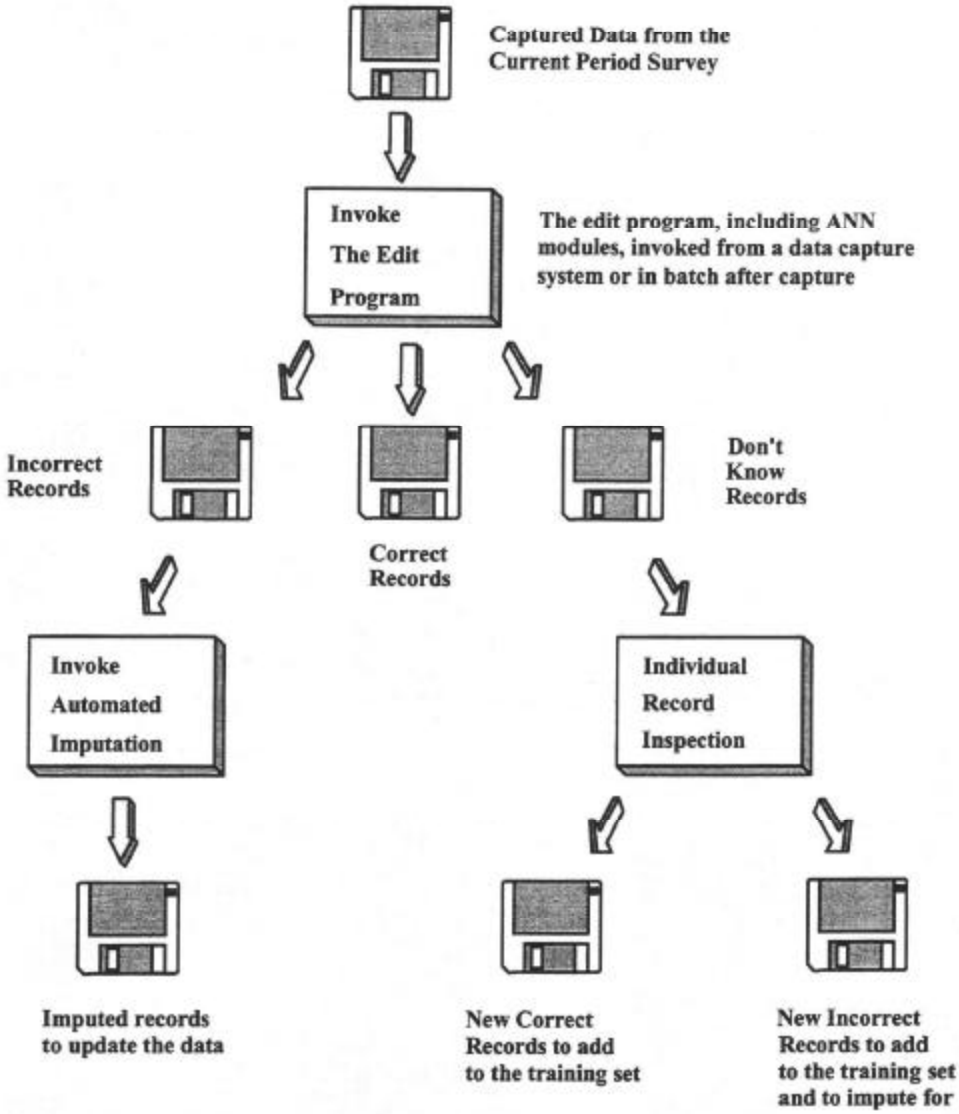


When all of the possible cases are available to train the ANN, it can be trained to be 100 percent correct, if the training set is presented properly. Having all of the possible cases for training produces a classifier ANN, whereas having a subset of the cases for training produces a predictor ANN.

The result of the ANN training is an executable subroutine that can be called from a data capture system or an edit system. The ANN is tested to ensure that it is correctly trained by passing all of the cases on the database edit table through the ANN. The ANN is successfully trained if it correctly classifies all of the cases.

The ANN is invoked by a capture system (or an edit system) via a control subroutine that the capture system calls. The capture system calls the control subroutine with the name of the edit being invoked and the data from the fields being edited. The control subroutine converts the discrete variable values to

Figure 3.--Edit ANN Invocation



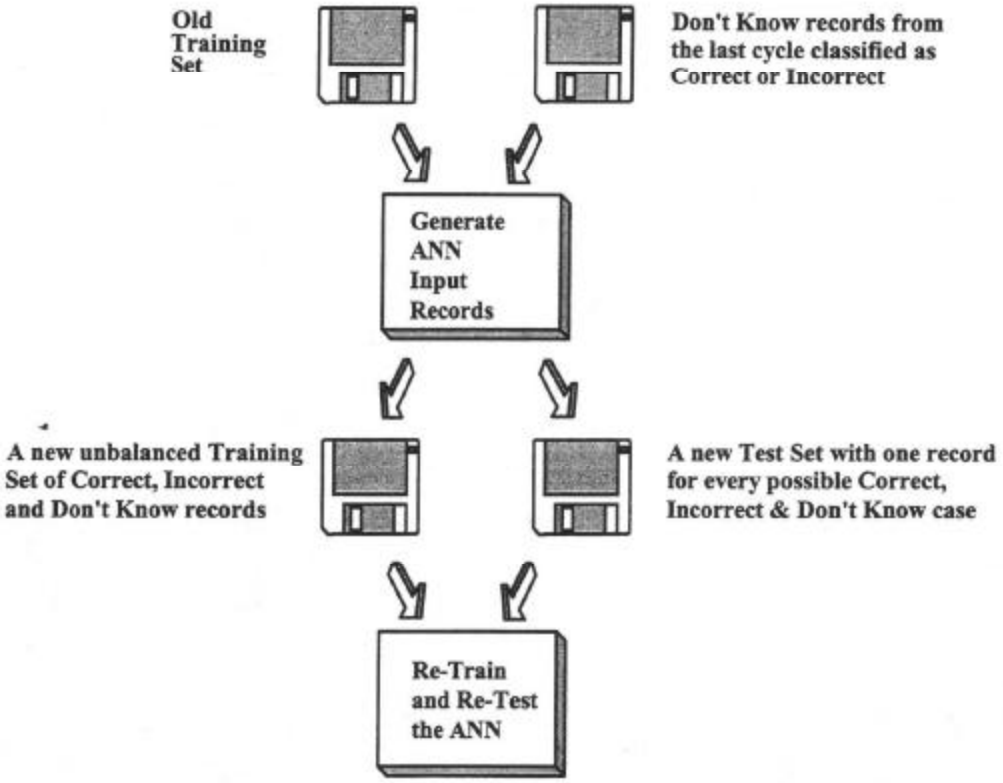
binary variables and calls the edit ANN that is being requested. The ANN edits the values and passes its answer, CR, IC, or DK, back to the control subroutine. The control subroutine converts the answer into the form that the capture system can use and passes the answer back to the capture system. This control layer of insulation is used to ensure that any system using the ANN does not have to know about the peculiarities of the ANN inputs or outputs. It also allows for the regular re-training of the ANN with no impact on the capture system. A generalized control subroutine can be written that can action any of the edit ANNs in the system and interface with the capture system to pass the answers back.

The Production Cycle

As the ANN is run each month it classifies records into one of the three types. The CR records pass through untouched, the IC records go on for correction, either by an imputation ANN or by a conventional imputation routine, and the DK records are output to a database table of DK's for that specific edit. The subject matter specialist analyzes the table and assigns each record to either CR or IC and the database edit table is updated. Once the DK records for the current month have been classified, the ANN is re-trained to produce a new program for the next month. Using a database table for each edit's DK records provides an analytical history of what decisions have been made about that edit.

Assuming that there are no radical shifts in response patterns, the set of DK records output each month should be small and manageable. If the original specification of the CR and IC records is well done, there should only be few DK records in any one month and these should represent new knowledge about the subject matter that should be individually examined.

Figure 4.-- Periodic ANN Regeneration



Implementation

The edit ANN, as described, has been implemented for a data capture system on a PC using Microsoft Access® for the GUI and the DBMS. The Access capture screen called the control subroutine library, written in C, which in turn called the ANN subroutine. The ANN subroutine was generated using a commercial ANN product, NeuralWorks Professional II/Plus® from NeuralWare Inc. The analytical tools to assign the DK cases were built in Access.

Conclusions

ANN edit systems can be constructed that provide a viable alternative to traditional approaches to data editing. These ANN edit systems shift the emphasis to data analysis rather that specification for the generation of the system and therefore the system development methodology better emulates the modus operandi of the subject matter expert/analyst.

The ability of the edit to evolve over successive survey occasions without re-programming and re-testing should result in substantial development and maintenance savings for the survey. The fact that the ANN can interface with a capture system, an edit system, or both, should increase the flexibility of where the edits are applied.

A classifier ANN system using this method of training, with all of the cases provided, should also be effective for the construction of stratification and auto-coding systems.

References

Nordbotten, S. (1993). *Editing Statistical Records by Neural Networks*, Department of Information Science, University of Bergen, Norway.

Roddick, L. H. (1994). *Data Editing Using Neural Networks,* EXPERSYS '94, 655 - 659. ■

Chapter 10

Editing and Imputation by Means of Neural Networks

Svein Nordbotten, University of Bergen

Abstract

Editing and imputation of statistical data are possible because we take advantage of some prior knowledge about the type of statistical objects we investigate. The processes of editing and imputation are considered expensive parts of survey and census costs. This presentation discusses the use of neural network methodology to improve the efficiency of these processes. Two applications are discussed as demonstrations of the approach.

Editing and Imputation by Means of Neural Networks

Svein Nordbotten, University of Bergen

Models of Statistical Objects

Knowledge of Statistical Objects

In statistical agencies, a huge amount of knowledge exists about the different classes of the objects of which a modern society is considered composed. When preparing surveys for updating this knowledge, the already existing knowledge can and is extensively used. When screening data to identify suspect records, the data are checked to see if it is corresponding to the prior knowledge structure. When suspect records are detected, the structure is again used for imputing or modifying the data.

The situation may be compared to the recipient who is reading a received, handwritten letter. When he arrives to a string of symbols he does not recognize, he uses his knowledge of vocabulary, grammar and the context to decide if the word may be a valid, but for him unknown word, or a misspelled word. In case he decides that the word is a valid, but for him an unknown word, he will probably add it to his vocabulary, while in the case of a misspelled word, he may decide to correct the spelling.

In previous times, the statistical agencies hired specialists on demographic aspects to edit the data collected about individuals and their families. After the appearance of programmed computing equipment, much effort has been spent trying to get the computer systems to simulate the specialists or even to behave as specialists ideally should.

Before automatic editing of statistical records, the most typical and stable patterns among the features of objects are identified, embedded in programs as editing criteria. The individual records could be classified in groups as "acceptable," "suspicious," "defect," etc., and adequate procedures for handling the different groups be devised.

In other word, we can think of the development of the editing process as building editing models reflecting the typical patterns of properties for the object belonging to the class investigated. These models may be further extended to include the imputation, i.e., predict the most probable properties for object that are only partly known. The problem is of course how to extract from specialists the knowledge needed for specifying operative editing and imputation models (Fellegi and Holt, 1976).

Estimating Model Structure by Neural Networks

The models known as neural networks can be considered from a number of different angels. The name, "neural network," indicates an origin in neuro-physiology. Some researchers point out that "parallel processing" is an important feature of the models (Rumelhart and McClelland, 1986). Statisticians can claim that some of important algorithms used, as f.ex. the back propagation learning algorithm, were originally developed as extensions to already existing statistical methods (Werbos, 1974; Cheng and Titterington, 1994). Finally, mathematicians emphasize that the methods used can be considered as universal approximators.

In context of editing and imputation, it may be convenient to regard neural networks as sets of non-linear, multi-variate regression models used for prediction of values of independent variables for individual objects subject to existence of values for some known dependent variables of the same objects. The neural network models can be designed for prediction of a single dependent variable as well as for simultaneous prediction of a number of variables. In contrast to the multi-variate, statistical models, the neural network models are non-parametric and non-linear. This can be a serious drawback because it does not permit any direct inference about the quality of predictions based on statistical theory. The type of neural network most likely to be useful for editing, have, on the other hand, the great advantage that the networks can "learn" from a set of records representing the prior knowledge. The term "learn" means in this context that algorithms exist for estimating the huge number of parameters or internal connection weights of the networks.

Applications

Neural networks have many potential applications in statistical production besides editing and imputation. In this presentation, we limit the discussion to two applications demonstrating:

- simultaneous editing and imputation, and
- large scale imputation as a separate process.

Application of neural networks for editing and imputation in surveys and censuses, is probably just in the initial stage. One of the very few who has done research and promoted the use of neural networks in editing is Hugh Roddick (Roddick, 1993). It has also recently been reported that the National Statistical Office in UK is now evaluating neural networks for their 2001 Population Census (Clark and Street, 1996).

CASE 1: Simultaneous Editing and Imputation

Editing

The first application we shall discuss demonstrates simultaneous editing and imputation of data collected in an imaginary survey. Further details are discussed in Nordbotten (1996a). The scenario considered is a sample survey comprising 12.000 individuals, for whom values of the 9 categorical, socio-economic attributes -- sex, age group, marital status, geographic region, number of children, education, industry, employment status, and income group -- were collected. It was assumed that each record during collection and pre-processing, was subjected to a risk for partial non-response and erroneous data fields.

The questions studied in connection with this scenario were:

- Is it possible to design a neural network model to simulate the editing and imputation of a human?
- Can such a neural network be trained from a subsample of 2,000 objects for which raw as well as edited records are available?
- How well does a trained neural network edit and impute when applied to the remaining 10,000 raw records?

A Synthetic Population Model

Real data are not easily available for a researcher outside statistical agencies. In some situation, synthetic data can be used as a second best solution. Synthetic data have the great advantage that it renders a much closer experimental control.

For this study, synthetic data were used. A stochastic model which reflected the statistical properties of a typical individual was established, and used to generate the needed syntethic set of 12,000 data records for individuals.

The a random sample of 2,000 individual was drawn from the population of 12,000, and the true records were established in corresponding files for the 2,000 and the 10.000 individuals. Table 1 shows the 9 distributions by category for the true values for the 10,000 individuals.

Table 1.--Population Distributions Based on True Data

	Cat. 1	Cat. 2	Cat. 3	Cat.4	Cat. 5	Cat. 6	Total
Sex	5,031	4,969					10,000
Age	3,002	3,961	2,003	1,034			10,000
Marriage	6,954	3,046					10,000
Region	4,048	2,511	2,455	986			10,000
Children	7,161	1,344	945	360	190		10,000
Education	2,998	5,958	1,044				10,000
Industry	4,247	474	283	1,171	1,814	2,011	10,000
Employment	5,329	4,671					10,000
Income	5,087	3,297	1,522	94			10,000

An Error Model of Data Collection

Collection of the data about a population introduces errors of different types, i.e., interview errors, observation errors, non-response errors, transcription errors, etc. To simulate the effects of the data collection process, a stochastic error model was developed. The model introduced random errors in the records for each individual. The probabilities for errors were specified to be similar to observed error frequencies in real surveys.

When the true data records were processed by the error model, each record was transformed to an observed or "raw" record which might contain errors of two types. The first type was partial non-response errors resulting in blank fields in the records, the second error type was called interchange errors. An interchange error was created when a true category was substituted by another category of the same attribute, f.ex. when the response to the question about sex was changed from "male" to "female."

As for the true records, the raw records were also kept apart for the 2,000 sample and the remaining 10,000 individuals. Table 2 shows the 9 attribute distributions by category based on the raw records for the 10,000 individuals. About 3,400 attributes out of 90,000 have non-response while another 900 attributes have other categorization errors. Since the two error types were mutually excluding, there were approximately 4,300 erroneous records in the file of the 10,000 raw records.

Table 2.--Population Distributions Based on Raw Data

	Cat. 1	Cat. 2	Cat. 3	Cat.4	Cat. 5	Cat. 6	Total
Sex	4,819	4,656					9,475
Age	2,775	3,629	2,014	1,147			9,565
Marriage	6,418	2,964					9,382
Region	3,908	2,430	2,375	1,041			9,754
Children	6,942	1,290	923	371	273		9,799
Education	2,878	5,551	1,180				9,609
Industry	4,084	451	266	1,135	1,777	1,979	9,692
Employment	5,257	4,643					9,900
Income	4,585	3,136	1,499	192			9,412

Neural Network Training

A neural network was used in this study to investigate if errors in raw records for the 10,000 individuals could be identified and corrected. The neural network can here be considered as a set of non-linear, multi-variate regression equations in which the fields of each raw record correspond to a set of independent variables and the outputs of the regressions are a predicted set of true values for the same object.

Because of the non-linearity, the statistical estimation of the regression coefficients in neural networks is substituted by an iterative "learning" process which determines the values of a large set of weights in the network.

It was assumed that the 2,000 raw records were expertly edited and imputed with the result that the 2,000 true records also became available. In our case the paired set of 2,000 true and raw records was used as training set for a neural network. By representing each of the 9 attributes by 6 categories, each record could be expressed as a binary vector with 54 elements. Not all 6 categories were used for each attribute. F.ex. for sex, only two categories were used while the remaining 6 always were set equal to 0. The neural network used, was a so-called feedforward net with 54 input neurons, one layer of 300

hidden neurons, and 54 output neurons. The network used sigmoid transfer functions in hidden and output neurons.

Training of such a network requires the determination of the value of about 32,000 connection weights, and a large number of iterative cycle through a training set of records (Rumelhart and McClelland, 1986).

The twin sets of 2,000 true and raw records were used to train the network through in total 300 iterative cycles.

Model Verification

After training, evaluation of the network ability to simultaneously edit and impute could be carrieds out by means of the file of 10,000 raw records which had not been used for training. By using these records as input, the trained network produced edited records, or prediction for the true records. The closer the 10,000 edited records were to the corresponding 10,000 true records, the more successful would the neural network edit and imputation be evaluated.

Table 3.--Population Distributions Based on Edited and Imputed Data

	Cat. 1	Cat. 2	Cat. 3	Cat.4	Cat. 5	Cat. 6	Total
Sex	5,127	4,872					9,999
Age	3,002	3,878	2,024	989			9,893
Marriage	7,090	2,910					10,000
Region	4,046	2,488	2,443	946			9,923
Children	7,130	1,364	918	336	146		9,894
Education	3,088	5,870	974				9,932
Industry	4,243	460	265	1,173	1,784	1,985	9,910
Employment	5,371	4,621					9,992
Income	5,029	3,352	1,427	67			9,875

Table 3 shows the attribute distributions based on the 10,000 edited records. Comparison with Table 1 shows that 580 out of the 3,400 non-responses are still unresolved while there are still 460 other errors in the edited records compared with 900 in the raw records.

We have demonstrated by means of this example how we simultaneously can edit and impute a set of raw records if we have access to, or can produce, a training set of paired raw and true records. As pointed out in the beginning of this paper, neural networks can also successfully to the editing step only (Roddick, 1993).

Using synthetic data have the advantage that the discussion about what are the true values, and how to obtain true data for comparison, can be disregarded. We can therefore make statements about this particular approach to simultaneous editing and imputation. A more interesting comparison with alternative editing and imputation procedures, would require that these procedures were applied to the same 10,000 raw records and their results compared with the neural network results.

CASE 2: Large Scale Imputation

A Population Census Case

In the previous section, it was pointed out that neural networks could be used for editing as well as for simultaneous editing and imputation. In the second case to be presented, we shall describe how neural networks can be used for imputation only. A more complete discussion is available in Nordbotten (1996b).

The 1990 Population Census in Norway was based on data from administrative registers and supplemented by a special population sample survey. In Norway, there are about 435 municipalities, and those are further subdivided into census tracts, some with only 100-200 inhabitants. The census tracts have been convenient geographical subdivisions for a number of users of statistics. In municipalities with less than 6,000 inhabitants, the population survey was extended to all, while the survey size varied with the population for those with more than 6,000 inhabitants. A few municipalities also paid themselves for a complete survey.

The 1990 Population Census was evaluated by an independent group in 1993. One point emphasized by the group and others was that, for smaller areas, many estimates could not be released for publication because of the high uncertainty associated with estimates.

The purpose of this second case to be discussed below was to investigate if improved estimates of proportions for small subpopulations could be produced by means of neural networks.

Design of the Population Census Experiment

This experiment was made possible by a cooperation contract with the Central Bureau of Statistics of Norway. The population records from a municipality with 17,326 inhabitants which paid for a complete survey, were selected for the experiment. The census record in the Population Census of 1990 was composed by one set of fields with information transferred from administrative registers and a second set of fields from information collected by the special population census survey. For the investigation carried out, 97 administrative field and 49 fields with data from the survey for each inhabitant were extracted from the population record and adjusted to the form required by the experiment. The survey fields selected, represented 10 attribute groups with from 2 to 9 categories. The 49 fields were representing observations as binary variables, while the register fields represented data as both binary, categorical and continuous variables. Other experiments have also included continuous dependent variables.

A random sample of 2,007 inhabitants was drawn to represent what the situation could have been if this selected municipality had not requested and paid itself for a complete survey. In the situation simulated, both survey and register data, therefore, existed for each of the inhabitants of this municipality. From this sample, 1,845 records were randomly extracted and named Sample 1; the remaining 162 records of the original sample were named Sample 2.

Ten neural imputation networks were defined, one for each attribute group of survey variables. The structure of all models corresponds to a feedforward neural network with 97 input neurons, one hidden layer of 25 neurons and from 2 to 9 output neurons depending on thge number of categories in the

particular group represented by the model. Each neuron in the hidden and the output layers, produced its output by converting its input by means of a sigmoid transfer function. A neural network of this type is specified by the weights used in connecting all input and hidden neurons, and all hidden and output neurons. The number of weights in the networks varied from 2,502 for the nets designed to predict 2 survey variables to 2,684 for the net which predicted 9 survey variables simultaneously.

Training the Imputation Models

The 10 imputation networks were estimated or "trained" to produce estimates of the survey variables for individual records based on the register variables of the corresponding records. Sample 1, which contained 1,845, was used as a training set. The training was done by means of a standard backpropagation algorithm (Rumelhart and McClelland, 1986). Since training is an iterative adjustment process, some of the models required several thousand iterations and some hours of computer time.

Imputing Survey Variable Values

The 10 trained imputation models were then used to impute individual survey variable values for each individual in municipality. In total, these add up to almost 850,000 imputed values which required only minutes to compute. The models did not produce binary values, but values in the interval 0 to 1. The real values were converted to binary values according to usual rules the values less than 0.5 became 0, and the remaining 1.

At this stage of the experiment, we had both observed values and imputed values for the complete population. Comparison showed that the imputed values for individuals in the training sample were much closer to the observations than for individuals in Sample 2 and Sample 3. This is a phenomenon called overfitting, and a well known problem in applications of neural network. The cause is that the network during training adjusts too well to the training set.

Comparison of Estimates

The observed values from Sample 1 and Sample 2 were added and divided with the number of records in Sample 1 and Sample 2 to obtain simple, unbiased estimates for the proportions of the population. To obtain alternative, imputed estimates, the imputed values for Sample 2 and Sample 3, and observed values from Sample 1, were added and divided with the size of the population. The relative precision, defined as the standard error of an estimate divided by the estimate, is usually used as an indicator of accuracy for sample based estimates. Statistics Norway restricted f.ex. general publication of estimates from the Population Census 1990 to those with a relative precision corresponding to 0.3 or less. With the sizes of the population of the selected municipality and Sample 1 + Sample 2, we can by means of standard theory predict that only estimates of proportions 0.008 or less, or 0.992 or larger, will have an unacceptable, relative precision.

The estimates based on a sum of imputed and observed values, have no similar sampling error. If we repeated the drawing of the training sample, the training and the imputation, we would get a basis for estimating an assumed structural sampling errors of the neural network. This has not been done in this experiment. The imputed values may, however, be biased and create an inaccuracy in the aggregated estimates. The exact biases can not be observed in a real life situation, but they can be estimated. The purpose of Sample 2 was to obtain estimates for the biases. We used ratios of estimated proportion biases from Sample 2 divided by the estimatd proportions as relative accuracy indicators for the imputed

proportion estimates. In contrast to the relative precision for unbiased estimates, the values of the relative accuracy indicators for imputed estimates are not expected to vary with the size of a subsample. In other words, while the unbiased estimates frequently are useless for small subsamples, the imputed estimates should be expected to be useful also when computed for small subpopulations if the network used for imputation have been trained on a sufficiently large sample.

Comparing Estimates for the Population of a Municipality

The relative sampling errors for the unbiased estimates and the relative biases for the imputed estimates were computed for the municipality population. From these accuracy predictions, it was evident that the unbiased estimates would generally be better than the imputed.

The particular municipality used, permitted us to test empirically the predicted accuracies. The target proportions could be computed because we had access to the set of observations for the whole population. The unbiased estimates for the municipality computed from the observations from Sample 1+2, and the imputed estimates of the proportions which were arrived to by adding observations for the individuals in Sample 1+2 and the imputed values for individuals in Sample 3, could therefore be compared with the target proportions.

The comparison test confirmed the expectations -- 27 out of the 49 unbiased estimates were closer to the target proportions than the imputed estimates.

Comparing Estimates for the Population of a Small Area

The purpose of the experiment was, however, to test the estimates for the small census tract area. Calculating unbiased estimates from the 18 individuals of this area in Sample 1+2, and the relative sampling error for these estimates, 44 estimates were predicted not to satisfy the 0.3 requirement. The corresponding number for the relative bias for imputed estimates, which exceeded 0.3, were 22.

Also for the census tract, the estimates were computed and compared with the targets. Table 4 shows the results for the first attribute group proportions. While most of the unbiased estimates, as expected, deviate significantly from the target proportions, the imputed estimates were close to the targets. When

Table 4.--With Whom Are You Living? Target and Estimated Proportions, Census Tract of 162 Individuals

Attribute	Target	Unbiased estimate	Imputed estimate
1. Nobody	0.14	0.17	0.11
2. Spouse	0.48	0.33	0.47
3. Cohabitant	0.10	0.28	0.11
4. Children	0.30	0.28	0.32
5. Parents	0.13	0.28	0.14
6. Siblings	0.06	0.17	0.07
7. In-laws	0.04	0.11	0.03
8. Grandparent-child	0.04	0.11	0.03
9. Other	0.04	0.11	0.03

considering the relative deviation, Table 5 shows that 32 of the unbiased estimates deviated from the target with more than 0.3 requirement from the target proportion, while the corresponding figure for the imputed estimates was 16. Compared with the predictions, both estimators yielded better results than expected. However, both the predictions and the observed deviations clearly point to the imputed estimator as the better for areas or subgroups of this size (Nordbotten, 1996c).

Table 5.--Predicted and Observed Relative Acuracy of Estimates for Proportions, Census Tract of 162 Individuals

Unbiased estimates:

		Observation		
		<=0.3	>0.3	Total
Prediction	<=0.3	4	1	5
	>=0.3	13	31	44
		17	32	49

Imputed estimates:

		Observation		
		<=0.3	>0.3	Total
Prediction	<=0.3	22	5	27
	>=0.3	11	11	22
		33	16	49

A Third Estimator

The investigation showed clearly that the correct selection between unbiased and imputed estimators varies from variable to variable and by the size of then population. Following a proposal by Spjøtvold and Thomsen (1987), the optimal strategy can be to use a composite estimate in which the two estimators are weighted. Optimal weights can be determined by f.ex. minimizing the squared sampling error of the unbiased estimate and the squared bias of the imputed estimate.

Such a composite estimate was computed for the census tract case discussed. For 14 of the in total 49 proportions estimated, the composite estimator gave better results than both of the simpler estimators. Unfortunately, the composite estimator in some cases also produced results which had a significant lower accuracy than the better of the two estimators of which it was composed.

Final Remarks

The two case studies reported do not permit any final conclusions about the use of the neural networks for building editing and imputation models. There are many questions still which need to be investigated and answered before neural networks eventually can be a standard editing and imputation tools. Studies have taught us that this approach may have a substantial potential. My prediction is that neural networks will be important components in production of many future, statistical products.

The two case studies give an interesting lesson in research design. The first case discussed, supports the idea of simultaneous automatic editing and imputation, as well as the adequacy of experiments based on synthetic data. Simultaneous editing and imputation will save both time and resources. The use of synthetic data gives researchers the possibility to systematically focus attention to one by one of many error factors in the editing process and study their effects. The drawback of the synthetic approach is that we may easily overlook factors and complexities of the real world.

The second case discussed, indicates how we can provide statistics from sample surveys supported by administrative data; also for small groups and areas if we can learn the relationships between survey data and administrative data from a more general sample. It also demonstrates how we can, if we are lucky, use real data in research and still have the advantage of knowing the correct answers to an ideal process for evaluation purposes. We simply assume that some of the existing information is saved for the evaluation of the process to be investigated.

Acknowledgments

This paper is based on research in the SIS Statistical Information Systems project at the Department of Information Science, University of Bergen. Part of the work reported was carried out under a cooperation contract with Statistics Norway.

References

Cheng, B. and Titterington, D.M. (1994). Neural Networks: A Review from a Statistical Perspective, *Statistical Science*, 9, 1, 3-54.

Clark, A. and Street, L. (1996). Planning for the 2001 Census of the United Kingdom, Presentation for the U. S. Bureau of the Census, Annual Research Conference 1996, Washington DC.

Fellegi, I. P. and Holt, D. (1976). A Systematic Approach to Automatic Editing and Imputation, *Journal of the American Statistical Association*, 71, 17-35.

Nordbotten, S. (1996a). Editing Statistical Records by Neural Networks, *Journal of Official Statistics*, 3, Stockholm.

Nordbotten, S. (1996b). Neural Network Imputation Applied on Norwegian 1990 Population Census Data Utilizing Administrative Registers, Department of Information Science, University of Bergen, Bergen.

Nordbotten, S. (1996c). Predicting the Precision of Imputed Proportions, Department of Information Science, University of Bergen, Bergen.

Roddick, L. H. (1993). Data Editing Using Neural Networks, Statistics Canada, Ottawa.

Rumelhart, D. E. and McClelland, J.L. (1986). Parallel Distributed Processing -- Explorations in Microstructure of Cognition, 1, *Foundation*, MIT Press, Cambridge, Mass.

Spjøtvold, E. and Thomsen, I. (1987). Application of Some Empirical Bayes Methods to Small Area Samples, *Proceedings of the 46th Session of the International Statistical Institute.*

Werbos, P. (1974). Beyond Regression: New Tools for Prediction and Analysis in Behaviour Sciences, Ph.D. dissertation, Harvard University. ■