Fischer, Manfred M.; Reismann, Martin

Conference Paper

A methodology for neural spatial interaction modelling

42nd Congress of the European Regional Science Association: "From Industry to Advanced Services - Perspectives of European Metropolitan Regions", August 27th - 31st, 2002, Dortmund, Germany

Provided in Cooperation with:

European Regional Science Association (ERSA)

Suggested Citation: Fischer, Manfred M.; Reismann, Martin (2002) : A methodology for neural spatial interaction modelling, 42nd Congress of the European Regional Science Association: "From Industry to Advanced Services - Perspectives of European Metropolitan Regions", August 27th - 31st, 2002, Dortmund, Germany, European Regional Science Association (ERSA), Louvain-la-Neuve

This Version is available at:
https://hdl.handle.net/10419/115569

Forthcoming in *Geographical Analysis* 34(3), 2002

Manfred M. Fischer
and Martin Reismann

A Methodology for Neural Spatial Interaction Modeling

This paper attempts to develop a mathematically rigid and unified framework for neural spatial interaction modeling. Families of classical neural network models, but also less classical ones such as product unit neural network ones are considered for the cases of unconstrained and singly constrained spatial interaction flows. Current practice appears to suffer from least squares and normality assumptions that ignore the true integer nature of the flows and approximate a discrete-valued process by an almost certainly misrepresentative continuous distribution. To overcome this deficiency we suggest a more suitable estimation approach, maximum likelihood estimation under more realistic distributional assumptions of Poisson processes, and utilize a global search procedure, called Alopex, to solve the maximum likelihood estimation problem. To identify the transition from underfitting to overfitting we split the data into training, internal validation and test sets. The bootstrapping pairs approach with replacement is adopted to combine the purity of data splitting with the power of a resampling procedure to overcome the generally neglected issue of fixed data splitting and the problem of scarce data. In addition, the approach has power to provide a better statistical picture of the prediction variability, Finally, a benchmark comparison against the classical gravity models illustrates the superiority of both, the unconstrained and the origin constrained neural network model versions in terms of generalization performance measured by Kullback and Leibler's information criterion.

The authors gratefully acknowledge the grant no. P15575 provided by the Austrian Fonds zur Förderung der wissenschaftlichen Forschung (FWF).

Manfred M. Fischer is Chaired Professor of the Department of Economic Geography & Geoinformatics at the Vienna University of Economics and Business Administration. E-mail: manfred.fischer@wu-wien.ac.at

Martin Reismann is research assistant at the same department. E-mail: martin.reismann@wu-wien.ac.at

There are several phases that an emerging field goes through before it reaches maturity, and GeoComputation is no exception. There is usually a trigger for birth of the field. In our case, new techniques such as neural networks and evolutionary computation, significant progress in computing technology, and the emerging data rich environment inspired many scholars to revisit old and tackle new spatial problems. The result has been a wealth of new approaches, with significant improvements in many cases (see Longley et al. 1998, Fischer and Leung 2001).

After the initial excitement settles in, the crest breaking question is whether the new community of researchers can produce sufficient results to sustain the field, and whether practitioners will find these results to be of quality, novelty, and relevance to make a real impact. Successful applications of geocomputational models and techniques to a variety of problems such as data mining, pattern recognition, optimization, traffic forecasting and spatial interaction modeling rang the bell signifying the entry of GeoComputation as an established field.

This paper is a response to the perceived omission in the comprehensive understanding of one of the most important subfields in GeoComputation. While various papers on neural network modeling of unconstrained spatial interaction flows have appeared in the past decade, there has yet to be an advanced discussion of the general concepts involved in the application of such models. This paper attempts to fill the gap. Among the elements which should be of interest to those interested in applications are estimation and performance issues.

The paper proceeds as follows. The first section points to some shortcomings evident in current practice and motivates to depart in two directions: *First*, to employ maximum likelihood under more realistic distributional assumptions rather than least squares and normality assumptions, and *second* to utilize bootstrapping to overcome the problems of fixed data splitting and the scarcity of data that affect performance and reliability of the model results. Section 2 describes classical unconstrained neural spatial interaction models and less classical ones. Classical models are those that are constructed using a single hidden layer of summation units. In these network models each input to the hidden node is multiplied by a weight and then summed. Less classical

models utilize a product unit rather than the standard summation neural network framework for modeling interactions over space.

Unconstrained – summation unit and product unit – neural spatial interaction models represent rich and flexible families of spatial interaction function approximators. But they may be of little practical value if a priori information is available on accounting constraints on the predicted flows. Section 3 moves to the case of constrained spatial interaction. To satisfactorily tackle this issue within a neural network environment it is necessary to embed the constraint-handling mechanism within the model structure. This is a far from easy task. We briefly describe the only existing generic model approach for the single constrained case (see Fischer, Reismann and Hlavackova-Schindler 2001), and present summation and product unit model versions. We reserve the doubly constrained case to subsequent work.

We view parameter estimation (network learning) in an optimization context and develop a rationale for an appropriate objective (loss) function for the estimation approach in Section 4. Global search procedures such as simulated annealing or Alopex may be employed to solve the maximum likelihood estimation problem. We follow Fischer, Hlavackova-Schindler and Reismann (2001) to utilize the Alopex procedure that differs from the method of simulated annealing in three important aspects. First, correlations between changes in individual parameters and changes in the loss function are used rather than changes in the loss function only. Second, all parameter changes are accepted at every iteration, and, third, during an iteration step all parameters are updated simultaneously.

The standard approach to evaluate the generalization performance of neural network models is to split the data set into three subsets: the training set, the internal validation set and the testing set. It has become common practice to fix these sets. A bootstrapping approach is suggested to overcome the generally neglected problem of sensitivity to the specific splitting of the data, and to get a better statistical picture of prediction variability of the models. Section 5 illustrates the application of the various families of neural spatial interaction function approximators discussed in the previous sections, and presents the results of a comparison of the performance of the summation and the product unit neural network model versions [unconstrained and origin constrained

cases] against the corresponding standard gravity models. The testbed for the evaluation uses interregional telecommunication traffic data from Austria. Section 6 outlines some directions for future research.

1. DEPARTURES FROM CURRENT PRACTICE

We will begin our analysis with the simplest case, namely that of unconstrained spatial interaction. For concreteness and simplicity, we consider neural spatial interaction models based on the theory of single hidden layer feedforward models. Current research in this field appears to suffer from least squares and Gaussian assumptions that ignore the true integer nature of the flows and approximate a discrete-valued process by an almost certainly misrepresentative distribution. As a result, least squares estimates and their standard errors can be seriously distorted. To overcome this shortcoming we will develop a more appropriate estimation approach under more realistic distributional assumptions.

Thus, throughout the paper we assume observations generated as the realization of a sequence $\{Z_k = (X_k, Y_k), k = 1,..., K\}$ of $(N+1)\times 1$ vectors $(N \in ¥)$ defined on a Poisson probability space. The random variables Y_k represent targets. Their relationship to the variables X_k is of primary interest. When $E(Y_k) < \infty$, the conditional expectation of Y_k given X_k exists, denoted as $g = E(Y_k | X_k)$. Defining $\varepsilon_k \equiv Y_k - g(X_k)$, we can also write $Y_k = g(X_k) + \varepsilon_k$. The unknown spatial interaction function g embodies the systematic part of the stochastic relation between Y_k and X_k. The error ε_k is noise, with the property that $E(\varepsilon_k | X_k) = 0$ by construction. Our problem is to learn (estimate, approximate) the mapping g from a realization of the sequence $\{Z_k\}$.

In practice, we observe a realization of only a finite part of the sequence $\{Z_k\}$, a training set or sample of size K (i.e. a realization of $\{Z_k | k = 1,..., K\}$). Because g is an element of a space of spatial interaction functions, say G, we have essentially no hope of learning g in any complete sense from a sample of fixed finite size. Nevertheless, it is possible to approximate g to some degree of accuracy using a sample of size K, and to construct increasingly accurate approximations with increasing K. We will refer to such a procedure interchangeable as learning, estimation or approximation.

There are many standard procedures of function approximation to this function g. Perhaps the simplest is linear regression. Since feedforward neural networks are characteristically nonlinear it is useful to view them as performing a kind of nonlinear regression. Several of the issues that come up in regression analysis are also relevant to the kind of nonlinear regression performed by neural networks.

One important example comes up in the cases of *underfitting* and *overfitting*. If the neural network model is able to approximate only a narrow range of functions, then it may be incapable of approximating the true spatial interaction function no matter how much training data is available. Thus, the model will be biased, and it is said to be *underfitted*. The solution to this problem seems to be to increase the complexity of the neural network, and, thus, the range of spatial interaction functions, that can be approximated, until the bias becomes negligible. But, if the complexity [too many degrees of freedom] rises too far, then *overfitting* may arise and the fitted model will again lead to poor estimates of the spatial interaction function. If overfitting occurs then the fitted model can change significantly as single training samples are perturbed and, thus shows high variance. The ultimate measure of success is not how closely the model approximates the training data, but how well it accounts for not yet seen cases. Optimizing the generalization performance requires that the neural network complexity is adjusted to minimize both the bias and the variance as much as possible.

Since the training data will be fitted more closely as the model complexity increases, the ability of the trained model to predict this data cannot be utilized to identify the transition from underfitting to overfitting. In order to choose a suitably complex model, some means of directly estimating the generalization performance are needed. For neural spatial interaction models data splitting is commonly used. Though this procedure is simple to use in practice, effective use of data splitting may require a significant reduction in the amount of data which is available to train the model. If the available data is limited and sparsely distributed – and this tends to be the rule rather than the exception in spatial interaction contexts, then any reduction in amount of training data may obscure or remove features of the true spatial interaction function from the training set.

In this contribution, we address this issue by adopting the bootstrapping pairs approach (see Efron 1982) with replacement. This approach will combine the purity of data splitting with the power of a resampling procedure and, moreover, allows to get a better statistical picture of the prediction variability. An additional benefit of the bootstrap is that it provides approximations to the sampling distribution of the test statistic of interest that are considerably more accurate than the analytically obtained large sample approximations. Formal investigation of this additional benefit is beyond the scope of this contribution. We have a full agenda just to analyze the performance of summation and product unit neural network models for the cases of unconstrained and constrained spatial interaction. But we anticipate that our bootstrapping procedure may well afford such superior finite sample approximations.

2. FAMILIES OF UNCONSTRAINED NEURAL SPATIAL INTERACTION MODELS

In many spatial interaction contexts, little is known about the form of the spatial interaction function which is to be approximated. In such cases it is generally not possible to use a parametric modeling approach where a mathematical model is specified with unknown coefficients which have to be estimated to fit the model. The ability of neural spatial interaction models to model a wide range of spatial interaction functions relieves the model user of the need to specify exactly a model that includes all the necessary terms to model the true spatial interaction function.

The Case of Unconstrained Spatial Interaction

There is a growing literature in geography and regional science that deals with alternative model specifications and estimators for solving unconstrained spatial interaction problems. Examples include, among others, Fischer and Gopal (1994); Black (1995); Nijkamp, Reggiani and Tritapepe (1996); Bergkvist and Westin (1997); Bergkvist (2000); Reggiani and Tritapepe (2000); Thill and Mozolin (2000); Mozolin, Thill and Usery (2000). All these models are members of the following *general class of unconstrained neural spatial interaction models* given by

$$W^H\left(\boldsymbol{x}, \boldsymbol{w}^H\right) = \psi\left(w_{00} + \sum_{h=1}^{H} w_{0h}\, \phi\left(\sum_{n=1}^{N} w_{1hn}\, x_n\right)\right) \tag{1}$$

where the *N*-dimensional euclidean space (generally, $N = 3$) is the input space and the 1-dimensional euclidean space the output space. Vector $\boldsymbol{x} = (x_1, ..., x_N)$ is the input vector that represents measures characterizing the origin and the destination of spatial interaction as well as their separation. $\boldsymbol{w}^H \equiv (\boldsymbol{w}_0, \boldsymbol{w}_1)$ is the $(HN + H + 1) \times 1$ vector of the network weights (parameters). There are H hidden units. The vector $\boldsymbol{w}_0$ contains the hidden to output unit weights, $\boldsymbol{w}_0 \equiv (w_{00}, w_{01}, ..., w_{0H})$, and the vector $\boldsymbol{w}_1$ contains the input to hidden unit weights, $\boldsymbol{w}_1 \equiv (w_{10}, ..., w_{1H})$ with $\boldsymbol{w}_{1h} \equiv (w_{1h1}, ..., w_{1hN})$. We allow a bias at the hidden layer by including w_{00}. A bias at the input array may be taken into consideration by setting $x_1 \equiv 1$. ϕ is a hidden layer transfer function, ψ an output unit transfer function, both continuously differentiable of order 2 on $\Re$. Note that the model output function and the weight vector are explicitly indexed by the number, *H,* of hidden units in order to indicate the dependence. But to simplify notation we drop the superindex hereafter.

FIGURE 1 TO BE PLACED ABOUT HERE

FIG. 1 depicts the corresponding network architecture. Hidden units, denoted by the symbol Σ, indicate that each input is multiplied by a weight and then summed. Thus, models of type (1) may be termed unconstrained *summation unit* spatial interaction models. The family of approximations (1) embodies several concepts already familiar from the pattern recognition literature. It is the combination of these that is novel. Specifically, $\sum_{n=1}^{N} w_{1hn}\, x_n$ is a familiar linear discriminant function (see Young and Calvert 1974) which – when transformed by ϕ – acts as a nonlinear feature detector. The 'hidden' features are then subjected to a linear discriminant function and filtered through ψ. The approximation benefits from the use of nonlinear feature detectors, while retaining many of the advantages of linearity in a particularly elegant manner.

A leading case occurs when both transfer functions are specified as logistic functions[1]. This leads to the model

$$W_L(\boldsymbol{x},\boldsymbol{w}) = \left\{1+\exp\left[-\left(w_{00}+\sum_{h=1}^{H} w_{0h}\left(1+\exp\left(-\sum_{n=1}^{N} w_{1hn}\, x_n\right)\right)^{-1}\right)\right]\right\}^{-1} \qquad (2)$$

that has been often used in practice (see, for example, Mozolin, Thill and Usery 2000; Fischer, Hlavackova-Schindler and Reismann 1999; Fischer and Leung 1998; Gopal and Fischer 1996; Black 1995; Fischer and Gopal 1994; Gopal and Fischer 1993; Openshaw 1993).

Product Unit Model Versions

Neural spatial interaction models of type (1) are constructed using a single hidden layer of summation units. In these networks each input to the hidden node is multiplied by a weight and then summed. A nonlinear transfer function, such as the logistic function, is employed at the hidden layer. Neural network approximation theory has shown the attractivity of such summation networks.

In the neural network community it is well known that supplementing the inputs to a neural network model with higher-order combinations of the inputs increases the capacity of the network in an information capacity sense (see Cover 1965) and its ability to learn (see Giles and Maxwell 1987). This may motivate to utilize a product unit rather than the standard summation unit neural network framework for modeling interactions over space. The *general class of unconstrained product unit spatial interaction models* is given as

$$W_p(\boldsymbol{x},\boldsymbol{w}) = \left\{w_{00}+\sum_{h=1}^{H} w_{0h}\left(\prod_{n=1}^{N} x_n^{w_{1hn}}\right)\right\} \qquad (3)$$

which contain both product and summation units. The product units compute the product of inputs, each raised to a variable power. FIG. 2 illustrates the corresponding network architecture.

FIGURE 2 TO BE PLACED ABOUT HERE

Specifying $\jmath(\cdot)$ to be the identity function and $\psi(\cdot)$ to be the logistic function we obtain the following special case of (3)

$$ {}^{p}W_L(\boldsymbol{x},\boldsymbol{w}) = \left\{1+\exp\left[-\left(w_{00}+\sum_{h=1}^{H} w_{0h}\left(\prod_{n=1}^{N} x_n^{w_{1hn}}\right)\right)\right]\right\}^{-1} \tag{4} $$

3. NEURAL NETWORKS OF CONSTRAINED SPATIAL INTERACTION FLOWS

Classical neural network models of the form (1) and less classical models of the type (3) represent rich and flexible families of neural spatial interaction approximators. But they may be of little practical value if a priori information is available on accounting constraints of the predicted flows. For this purpose Fischer, Reismann and Hlavackova-Schindler (2001) have recently developed a novel class of neural spatial interaction models that are able to deal efficiently with the singly constrained case of spatial interaction.

The models are based on a modular connectionist architecture that may be viewed as a linked collection of functionally independent modules with identical feedforward topologies [two inputs, *H* hidden product units and a single summation unit], operating under supervised learning algorithms. The prediction is achieved by combining the outcome of the individual modules using a nonlinear output transfer function multiplied with a bias term that implements the accounting constraint.

Without loss of generality we consider the origin constrained model version only. FIG. 3 illustrates the modular network architecture of the models. Modularity is seen here as decomposition on the computational level. The network is composed of two processing layers and two layers of network parameters. The first processing layer is involved with the extraction of features from the input data. This layer is implemented as a layer of *J* functionally independent modules with identical topologies. Each module is a feedforward network with two inputs x_{2j-1} and x_{2j} [representing measures of destination attractiveness and separation between origin and destination,

respectively], *H* hidden product units, denoted by the symbol Π, and terminates with a single summation unit, denoted by the symbol Σ. The collective output of these modules constitutes the input to the second processing layer consisting of *J* output units that perform the flow prediction.

FIGURE 3 TO BE PLACED ABOUT HERE

This network architecture implements the *general class of product unit neural models of origin constrained* [*OC*] *spatial interaction*

$$ {}^{p}W^{OC}\left(\boldsymbol{x},\boldsymbol{w}\right)_j = \boldsymbol{y}_j\left(\sum_{h=1}^{H}\boldsymbol{g}_h\,\boldsymbol{j}_h\left(\prod_{n=2\,j-1}^{2\,j}x_n^{\boldsymbol{b}_{hn}}\right)\right)\quad j=1,\ldots,J \tag{5}$$

with $\boldsymbol{j}_h:\Re\to\Re$, $\boldsymbol{y}_j:\Re\to\Re$ and $\boldsymbol{x}\in\Re^{2J}$, that is $\boldsymbol{x}=\left(x_1,x_2,\ldots,x_{2\,j-1},x_{2\,j},\ldots,x_{2\,J-1},x_{2J}\right)$ where $x_{2\,j-1}$ represents a variable pertaining to destination j $\left(j=1,\ldots,J\right)$ and $x_{2\,j}$ a variable f_{ij} pertaining to the separation from region i to region j $\left(i=1,\ldots,I;\ j=1,\ldots,J\right)$ of the spatial interaction system under scrutiny. $\boldsymbol{b}_{hn}$ $\left(h=1,\ldots,H;\ n=2\,j-1,2\,j\right)$ are the input-to-hidden connection weights, and $\boldsymbol{g}_h$ $\left(h=1,\ldots,H\right)$ the hidden-to-output weights in the *j*-th module of the network model. The symbol $\boldsymbol{w}$ is a convenient shorthand notation of the (3*H*)-dimensional vector of all the model parameters. $\boldsymbol{y}_j$ $\left(j=1,\ldots,J\right)$ represents a nonlinear summation unit transfer function and $\boldsymbol{j}_h$ $\left(h=1,\ldots,H\right)$ a linear hidden product unit transfer function.

Specifying $\boldsymbol{j}_h(\cdot)$ to be the identity function and $\boldsymbol{j}_j(\cdot)$ a nonlinear normalized function we obtain the following important special case of (5)

$$ {}^{p}W_1^{OC}\left(\boldsymbol{x},\boldsymbol{w}\right)_j = \beta_{(i)}\frac{\sum_{h=1}^{H}\boldsymbol{g}_h\prod_{n=2\,j-1}^{2\,j}x_n^{\boldsymbol{b}_{hn}}}{\sum_{j'=1}^{J}\sum_{h'=1}^{H}\boldsymbol{g}_{h'}\prod_{n=2\,j'-1}^{2\,j'}x_n^{\boldsymbol{b}_{h'n}}}\quad j=1,\ldots,J \tag{6}$$

where $\tilde{b}_{(i)}$ is the bias signal that can be thought as being generated by a 'dummy unit' whose output is clamped at the scalar t_{i_g} A more detailed description of the model may be found in Fischer, Hlavackova-Schindler and Reismann (2001).

Summation Unit Model Versions

The summation unit version of the *general class of product unit neural network models* of *origin constrained spatial interaction* may be easily derived from Equation (5):

$$ {}^{S}\boldsymbol{W}^{OC}\left(\boldsymbol{x},\boldsymbol{w}\right)_j = \boldsymbol{y}_j\left(\sum_{h=1}^{H} g_h\, \boldsymbol{j}_h\left(\sum_{n=2\,j-1}^{2j} \boldsymbol{b}_{hn}\, x_n\right)\right) \quad j=1,\ldots,J \tag{7} $$

where $\boldsymbol{j}_h : \Re \to \Re$, $\boldsymbol{y}_j : \Re \to \Re$ and $\boldsymbol{x} \in \Re^{2J}$ as above. Specifying $\boldsymbol{j}_h(\cdot)$ as logistic function for h = 1,...,N, and $\boldsymbol{y}_j(\cdot)$ as nonlinear normalized output transfer function we obtain the following *origin constrained member* of class (7)

$$ {}^{S}\boldsymbol{W}_1^{OC}\left(\boldsymbol{x},\boldsymbol{w}\right)_j = \tilde{b}_{(i)}\, \frac{\sum_{h=1}^{H} g_h \left(1+\exp\left(-\sum_{n=2\,j-1}^{2j} \boldsymbol{b}_{hn}\, x_n\right)\right)^{-1}}{\sum_{j'=1}^{J}\sum_{h'=1}^{H} g_{h'} \left(1+\exp\left(-\sum_{n=2\,j'-1}^{2j'} \boldsymbol{b}_{h'n}\, x_n\right)\right)^{-1}} \quad j=1,\ldots,J \tag{8} $$

4. A RATIONALE FOR THE ESTIMATION APPROACH

If we view a neural spatial interaction model, unconstrained or constrained, as generating a family of approximations (as $\boldsymbol{w}$ ranges over $\boldsymbol{W}$, say) to a spatial interaction function g , then we need a way to pick a best approximation from this family. This is the function of network learning (training, parameter estimation) which might be viewed as an optimization problem.

We develop a rationale for an appropriate objective (loss, cost) function for this task. Following Rumelhart et al. (1995) we propose that the goal is to find that model which

is the most likely explanation of the observed data set, say *M*. We can express this as attempting to maximize the term

$$P\left(\boldsymbol{W}(\boldsymbol{w})\middle|M\right)=\frac{P\left(M\middle|\boldsymbol{W}(\boldsymbol{w})\right)\ P\left(\Omega(\boldsymbol{w})\right)}{P\left(M\right)} \tag{9}$$

where $\boldsymbol{W}$ represents the neural spatial interaction model (with all the weights $\boldsymbol{w}^H$) in question, unconstrained or constrained. $P\left(M\middle|\boldsymbol{W}(\boldsymbol{w})\right)$ is the probability that the model would have produced the observed data M. Since sums are easier to work with than products, we will maximize the log of this probability, and since this log is a monotonic transformation, maximizing the log is equivalent to maximizing the probability itself. In this case we have

$$\ln P\left(\boldsymbol{W}(\boldsymbol{w})\middle|M\right)=\ln P\left(M\middle|\boldsymbol{W}(\boldsymbol{w})\right)+\ln\ P\left(\boldsymbol{W}(\boldsymbol{w})\right)-\ln P\left(M\right) \tag{10}$$

The probability of the data, $P\left(M\right)$, is not dependent on the model. Thus, it is sufficient to maximize $\ln P\left(M\middle|\boldsymbol{W}(\boldsymbol{w})\right)+\ln P\left(\boldsymbol{W}(\boldsymbol{w})\right)$. The first of these terms represents the probability of the data given the model, and hence measures how well the network accounts for the data. The second term is a representation of the model itself; that is, it is a prior probability, that can be utilized to get information and constraints into the learning procedure.

We focus solely on the first term, the performance, and begin by noting that the data can be broken down into a set of observations, $M=\left\{z_k=\left(x_k,\ y_k\right)\middle|k=1,...,K\right\}$, each z_k, we will assume, chosen independently of the others. Hence we can write the probability of the data given the model as

$$\ln P\left(M\middle|\boldsymbol{W}(\boldsymbol{w})\right)=\ln\prod_k P\left(z_k\middle|\boldsymbol{W}(\boldsymbol{w})\right)=\sum_k \ln P\left(z_k\middle|\boldsymbol{W}(\boldsymbol{w})\right) \tag{11}$$

Note that this assumption permits to express the probability of the data given the model as the sum of terms, each term representing the probability of a single observation

given the model. We can still take another step and break the data into two parts: the observed input data x_k and the observed target y_k. Therefore we can write

$$\ln P\left(M \middle| W(\boldsymbol{w})\right) = \sum_k \ln P\left(y_k \middle| x_k \text{ and } (\boldsymbol{w}) W_k\right) + \sum_k \ln P\left(x_k\right) \tag{12}$$

Since we assume that x_k does not depend on the model, the second term of the equation will not affect the determination of the optimal model. Thus, we need only to maximize the term $\sum_k \ln P\left(y_k \middle| x_k \text{ and } W(\boldsymbol{w})\right)$.

Up to now we have – in effect – made only the assumption of the independence of the observed data. In order to proceed, we need to make some more specific assumptions, especially about the relationship between the observed input data x_k and the observed target data y_k, a probabilistic assumption. We assume that the relationship between x_k and y_k is not deterministic, but that for any given x_k there is a distribution of possible values of y_k. But the model is deterministic, so rather than attempting to predict the actual outcome we only attempt to predict the expected valued of y_k given x_k. Therefore, the model output is to be interpreted as the mean bilateral interaction frequencies (that is, those from the region of origin to the region of destination). This is, of course, the standard assumption.

To proceed further, we have to specify the form of the distribution of which the model output is the mean. Of particular interest to us is the assumption that the observed data are the realization of a sequence of independent Poisson random variables. Under this assumption we can write the probability of the data given the model as

$$P\left(y_k \middle| x_k \text{ and } W(\boldsymbol{w})_k\right) = \frac{\prod_k W(\boldsymbol{w})_k^{y_k} \exp\left(-W(\boldsymbol{w})_k\right)}{y_k!} \tag{13}$$

and, hence, define a maximum likelihood estimator as a parameter vector $\hat{\boldsymbol{w}}$ which maximizes the log-likelihood L

$$\max_{\boldsymbol{w}\in\boldsymbol{W}} \; L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{w}) = \max_{\boldsymbol{w}\in\boldsymbol{W}} \sum_k \left(y_k \ln \boldsymbol{W}(\boldsymbol{w})_k - \boldsymbol{W}(\boldsymbol{w})_k \right) \tag{14}$$

Instead of maximizing the log-likelihood it is more convenient to view learning as solving the minimization problem

$$\min_{\boldsymbol{w}\in\boldsymbol{W}} \; \boldsymbol{l}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{w}) = \min_{\boldsymbol{w}\in\boldsymbol{W}} \left[-L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{w}) \right] \tag{15}$$

where the loss (cost) function $\boldsymbol{l}$ is the negative log-likelihood L. $\boldsymbol{l}$ is non-negative, continuously differentiable on the Q-dimensional parameter space ($Q = HN + H + 1$ in the unconstrained case and $Q = 3H$ in the constrained one) which is a finite dimensional closed bounded domain and, thus, compact. It can be shown that $\boldsymbol{l}$ assumes its value as the weight minimum under certain conditions.

5. TRAINING THE NEURAL NETWORK MODELS

Since the loss function $\boldsymbol{l}$ is a complex nonlinear function of $\boldsymbol{w}$ for the neural spatial interaction models, $\hat{\boldsymbol{w}}$ cannot be found analytically and computationally intensive iterative optimization techniques such as global search procedures must be utilized to find (15). Simulated annealing, genetic algorithms and the Alopex[2] procedure are attractive candidates for this task. We utilize the latter as described in Fischer, Hlavackova-Schindler and Reismann (2001).

The loss function $\boldsymbol{l}(\boldsymbol{w})$ is minimized by means of weight changes that are computed for the s-th step ($s > 2$) of the iteration process as follows[3],

$$w_k(s) = w_k(s-1) + \boldsymbol{d}_k(s) \tag{16}$$

where $\boldsymbol{d}_k(s)$ is a small positive or negative step of size $\boldsymbol{d}$ with the following properties:

$$\boldsymbol{d}_k(s) = \begin{cases} -\boldsymbol{d} \text{ withprobability } p_k(s) \\ +\boldsymbol{d} \text{ withprobability } \left[1 - p_k(s)\right] \end{cases} \tag{17}$$

The probability $p_k(s)$ for a negative step is given by the Boltzmann distribution

$$p_k(s) = \left[1 + \exp\left(-C_k(s)/T(s)\right)\right]^{-1} \tag{18}$$

where

$$C_k(s) = \Delta w_k(s) \quad \Delta \boldsymbol{l}(s) \tag{19}$$

with

$$\Delta w_k(s) = w_k(s-1) - w_k(s-2) \tag{20}$$

and

$$\Delta \boldsymbol{l}(s) = \boldsymbol{l}(s-1) - \boldsymbol{l}(s-2) \tag{21}$$

The parameter T in Equation (18), termed temperature in analogy to simulated annealing, is updated using the following annealing schedule:

$$T(s) = \begin{cases} \dfrac{\boldsymbol{d}}{QS} \displaystyle\sum_k \sum_{s'=s-S}^{s-1} \left|C_k(s')\right| & \text{if } s \text{ isamultipleof } S \\ T(s-1) & \text{otherwise} \end{cases} \tag{22}$$

where ($Q = HN + H + 1$ in the case of the unconstrained models, and $Q = 3H$ in the case of the constrained models) denotes the number of weights. When T is small, the probability of changing the parameters is around zero if C_k is negative and around one if C_k is positive. If T is large, then $p_k \cong 0.5$ (see Bia 2000).

The effectiveness of Alopex in locating global minima and its speed of convergence critically depend on the balance of the size of the feedback term $\Delta w_k \, \Delta \boldsymbol{l}$ and the temperature *T*. If *T* is very large compared to $\Delta w_k \, \Delta \boldsymbol{l}$ the process does not converge. If *T* is too small, a premature convergence to a local minimum might occur. The procedure is governed by three parameters: the initial temperature *T*, the number of iterations, *S*, over which the correlations are averaged for annealing, and the step size $\boldsymbol{d}$. The temperature *T* and the *S*-iterations cycles seem to be of secondary importance for the final performance of the algorithm. The initial temperature *T* may be set to a large value of about 1,000. This allows the algorithm to get an estimate of the average correlation in the first *S* iterations and reset it to an appropriate value according to Equation (22). *S* may be chosen between 10 and 100. In contrast to *T* and *S*, $\boldsymbol{d}$ is a critical parameter that has to be selected heuristically with care. There is no way to a priori identify $\boldsymbol{d}$ in the case of multimodal parameter spaces.

The Termination Criterion

It has been observed that forceful training may not produce network models with adequate generalization ability, although the learning error achieved is small. The most common remedy for this problem is to monitor model performance during training to assure that further training improves generalization as well. For this purpose an additional set of validation data, independent from the training data is used. In a typical training phase, it is normal for the validation error to decrease. This trend may not be permanent, however. At some point the validation error usually reverses. Then the training process should be stopped. In our implementation of the Alopex procedure network training is stopped when $\boldsymbol{k} = 40{,}000$ consecutive iterations are unsuccessful. $\boldsymbol{k}$ has been chosen so large at the expense of the greater training time, to ensure more reliable estimates.

6. EXPERIMENTAL ENVIRONMENT, PERFORMANCE TESTS AND BENCHMARK COMPARISONS

To illustrate the application of modeling and estimation tools discussed in the previous sections we utilize interregional telecommunication traffic data from Austria and standard gravity models as benchmarks.

The Benchmark Models

The *standard unconstrained gravity model*

$$\boldsymbol{t}_{ij}^{grav} = k \; r_i^a \; s_j^b \; d_{ij}^{-g} \quad i=1,...,I; \; j=1,...,J; \; j \neq i \tag{23}$$

with

$$k = \frac{t_{gg}}{\sum_{i \neq} \sum_{j} \sum_{j} r_i^a \; s_j^b \; d_{ij}^{-g}} \tag{24}$$

$$t_{gg} := \sum_{i \neq} \sum_{j} \sum_{j} t_{ij} \tag{25}$$

serves as a benchmark model for the unconstrained neural spatial interaction models[4], that is, the classical models of type (2) and the less classical ones of type (4). $\boldsymbol{t}_{ij}^{grav}$ denotes the estimated flow from i to j, k is a factor independent of all origins and destinations, $\boldsymbol{a}$ reflects the relationship of r_i with $\boldsymbol{t}_{ij}^{grav}$ and b the relationship of s_j with $\boldsymbol{t}_{ij}^{grav}$. $\boldsymbol{g}$ is the distance sensitivity parameter, $\boldsymbol{g}$ >0. r_i and s_j are measured in terms of the gross regional product, d_{ij} in terms of distances from i to j, whereas t_{ij} in terms of erlang (see Fischer and Gopal 1994 for more details).

The *standard origin constrained gravity model*

$$^{orig}\boldsymbol{t}_{ij}^{grav} = b_{(i)} \; s_j^a \; d_{ij}^{-g} \quad i=1,...I; \; j=1,...,J; \; j \neq i \tag{26}$$

with

$$b_{(i)} = \frac{t_{i\,g}}{\sum_{j \neq i} s_j^a \; d_{ij}^{-g}} \tag{27}$$

where

$$t_{i_g} := \sum_{j \neq i} t_{ij} \tag{28}$$

is used as benchmark model for the constrained neural spatial interaction models (6) and (8). $b_{(i)}$ is the origin specific balancing factor. α, γ, s_j, d_{ij} and t_{ij} are defined as above.

Performance Measure

One needs to be very careful when selecting a measure to compare different models. It makes not much sense to utilize least squares related performance measures, such as the average relative variances or the standardized root mean square, in the context of our ML estimation approach. Model performance is measured in this study by means of Kullback and Leibler's (1951) information criterion (*KLIC*) which is a natural performance criterion for the goodness-of-fit of ML estimated models:

$$KLIC(M) = \sum_{u=1}^{U} \frac{y_u}{\sum_{u'=1}^{U} y_{u'}} \ln \left[\frac{y_u \left(\sum_{u'=1}^{U} y_{u'} \right)^{-1}}{W(x_u, w) \left[\sum_{u'=1}^{U} W(x_{u'}, w) \right]^{-1}} \right] \tag{29}$$

where (x_u, y_u) denotes the u-th pattern of the data set M, and W is the neural spatial interaction model under consideration. The performance measure has a minimum at zero and a maximum at positive infinity when $y_u > 0$ and $W(x_u w) = 0$ for any (x_u, y_u) pair.

The Data, Data Splitting and Bootstrapping

To model interregional telecommunication flows for Austria we utilize three Austrian data sources – a (32, 32)-interregional telecommunication flow matrix (t_{ij}), a (32, 32)-distance matrix (d_{ij}), and gross regional products for the 32 telecommunication regions – to produce a set of 992 4-tupel $(r_i, s_j, d_{ij}; t_{ij})$ with $i, j = 1,...,32 \quad (i \neq j)$. The first three components represent the input vector of the unconstrained models, the second and the third component represent the input variables

x_{2j-1} and x_{2j} of the *j*-th module of the origin constrained network models, and the last component the target output. The bias term $\hat{B}_{(i)}$ is clamped to the scalar t_{ig}. s_j represents the potential draw of telecommunication in j and is measured in terms of the gross regional product, d_{ij} denotes distances from i to j, while t_{ij} represents telecommunication traffic flows. The input data[5] were rescaled to lie in [0.1, 0.9].

The telecommunication data stem from network measurements of carried traffic in Austria in 1991, in terms of erlang, an internationally widely used measure of telecommunication contact intensity, which is defined as the number of phone calls (including facsimile transfers) multiplied by the average length of the call (transfer) divided by the duration of measurement (for more details, see Fischer and Gopal 1994). The data refer to the telecommunication traffic between the 32 telecommunication districts representing the second level of the hierarchical structure of the Austrian telecommunication network. Due to measurement problems, intraregional traffic (i.e. $i = j$) is left out of consideration.

The standard approach to evaluate the out-of-sample [prediction] performance of a neural spatial interaction model (see Fischer and Gopal 1994) is to split the total data set M of 992 samples into three subsets: the *training* [in-sample] *set* $M_1 = \{(x_{u1}, y_{u1}) \text{ with } u1 = 1,..., U_1 = 496 \text{patterns}\}$, the *internal validation set* $M_2 = \{(x_{u2}, y_{u2}) \text{ with } u2 = 1,..., U_2 = 248 \text{patterns}\}$ and the *testing* [prediction, out-of-sample] *set* $M_3 = \{(x_{u3}, y_{u3}) \text{ with } u3 = 1,..., U_3 = 248 \text{patterns}\}$. M_1 is used only for parameter estimation, while M_2 for validation. The generalization performance of the model is assessed on the testing set M_3. It has become common practice to fix these sets. But recent experience has found this approach to be very sensitive to the specific splitting of the data. To overcome this problem as well as the problem of scarce data we make use of the bootstrapping pairs approach (Efron 1982) with replacement. This approach combines the purity of splitting the data into three disjoint data sets with the power of a resampling procedure and allows us also to get a better statistical picture of the prediction variability.

The idea behind this approach is to generate B pseudo-replicates of the training, validation and test sets, then to re-estimate the model parameters $\boldsymbol{w}$ on each training bootstrap sample, stopping training on the basis of the validation and testing out-of-

sample performance of the test bootstrap samples. In this bootstrap world, the errors of prediction and the errors in the parameter estimates are directly observable. Statistics on parameter reliability can easily be computed.

Implementing the approach involves the following steps (see Fischer and Reismann 2000):

Step 1: Conduct three totally independent re-sampling operations in which B independent *training bootstrap samples*, B independent *validation bootstrap samples* and B independent *testing bootstrap samples* are generated, by randomly sampling U_1, U_2 and U_3 times, respectively, with replacement from the observed input-output pairs M.

Step 2: For each training bootstrap sample the minimization problem (15) is solved by applying the Alopex procedure. During the training process the *KLIC* performance of the model is monitored on the corresponding bootstrap validation set. The training process is stopped as specified in Section 5.

Step 3: Calculate the *KLIC*-statistic of generalization performance for each test bootstrap sample. The distribution of the pseudo-errors can be computed, and used to approximate the distribution of the real errors. This approximation is the bootstrap.

Step 4: The variability of the B bootstrap *KLIC*-statistics gives an estimate of the expected accuracy of the model performance. Thus, the standard errors of the generalization performance statistic is given by the sample standard deviation of the B bootstrap replications.

Performance Tests and Results

We consider first

- the summation unit neural network W_L [see Equation (2)] and
- the product unit neural network ${}^{p}W_L$ [see Equation (4)],

to model the unconstrained case of spatial interaction, and then

- the modular product unit neural network version ${}^{p}W_{1}^{OC}$ [see Equation (6)] and
- the modular summation unit neural network version ${}^{S}W_{1}^{OC}$ [see Equation (8)]

of singly constrained neural spatial interaction models to model the origin constrained case. Conventional gravity model specifications [see Equations (23)-(25) for the unconstrained case and Equations (26)-(28) for the origin constrained case] serve as benchmark models.

All the models were calibrated by means of the ML-estimation approach utilizing the Alopex procedure to eliminate the effect of different estimation procedures on the result. In order to do justice to each model specification, the critical Alopex parameter δ [step size] was systematically sought for each model. The Alopex parameters T and S were set to 1,000 and 10, respectively. We made use of the bootstrapping pairs approach [B = 60] to overcome the problem of sensitivity to the specific splitting of the data into in-sample, internal validation and generalization data sets, and the scarcity of data, but also to get a better statistical picture of prediction variability.

It should be emphasized that the main goal of training is to minimize the loss function λ. But it has been observed that forceful training may not produce network models with adequate generalization ability. We adopted the most common remedy for this problem and checked the model performance in terms of $KLIC(M_2)$ periodically during training to assure that further training improves generalization, the so-called cross-validation technique.

Alopex is an iterative procedure. In practice, this means that the final results of training may vary as the initial weight settings are changed. Typically, the likelihood functions of feedforward neural network models have many local minima. This implies that the training process is sensitive to its starting point. Despite recent progress in finding the most appropriate parameter initialization that would help Alopex – but also other iterative procedures – to find near optimal solutions, the most widely adopted approach still uses random weight initialization. In our experiments random numbers were generated from [-0.3, 0.3] using the rand_uni function from Press et al. (1992).

The order of the input data presentation was kept constant for each run to eliminate its effect on the result.

The Case of Unconstrained Spatial Interactions: Extensive computational experiments with different combinations of H- and δ-values have been performed on DEC Alpha 375 Mhz, with $H \in \{2,4,6,8,10,12,14\}$ and $\delta \in \{0.0005,0.0010,0.0025,0.0050,0.0100,0.0250,0.0500,0.1000\}$. Selected results of these experiments $[H = 2,4,6,8,10,12,14$ and $\delta = 0.0005,0.0010,0.0050,0.0100]$ are reported in TABLE 1. Training performance is measured in terms of $KLIC(M1)$, validation performance in terms of $KLIC(M2)$ and testing performance in terms of $KLIC(M3)$. The performance values represent the mean of $B = 60$ bootstrap replications, standard deviations are given in brackets.

PLACE TABLE 1 ABOUT HERE

Some considerations are worth making. *First*, the best result (averaged over the 60 independent simulation runs) in terms of average out-of-sample *KLIC*-performance was obtained with $H = 12$ and $\delta = 0.0010$ in the case of the summation unit neural network model, and with $H = 14$ and $\delta = 0.0005$ in the case of the product unit neural network model. *Second*, there is convincing evidence that the summation unit model outperforms the product unit model version at any given level of model complexity. This is primarily due to the fact that the input data of W_L were preprocessed to logarithmically transformed data scaled to [0.1, 0.9]. *Third*, it can be seen that model approximation improves as the complexity of ${}^{p}W_L$ grows with increasing H (except H = 12). This appears to be less evident in the case of the summation unit model version. *Fourth*, the experiments also suggest that $\delta = 0.0010$ tends to yield the best or at least rather good generalization performances in both cases of neural network models. The poorest generalization performance of the summation unit network is obtained for $\delta = 0.0005$ (except: H = 8) while $\delta = 0.0100$ leads to the poorest results in the case of the product unit network model (except H = 2 and 12). *Fifth*, as already mentioned above, forceful training may not produce the network model with the best generalization ability. This is evidenced for H =2, 10, 12, 14 in the case of W_L, and H = 2, 10, 12 in the case of ${}^{p}W_L$. *Finally*, note that the standard deviation illustrates the impact of both, random variations in training, validation and test sets and random

variations in parameter initializations. Most of the variability in prediction performance is clearly coming from sample variation and not from variation in parameter initializations as illustrated in Fischer and Reismann (2000). This implies that model evaluations based on one specific static split of the data only, the current practice in neural spatial interaction modeling (see, for example, Bergkvist 2000; Reggiani and Tritapepe 2000; Mozolin, Thill and Usery 2000), have to be considered with great care.

PLACE TABLE 2 ABOUT HERE

TABLE 2 summarizes the simulation results for the unconstrained neural network models in comparison with the gravity model. Out-of-sample performance is measured in terms of $KLIC(M_3)$. For matters of completeness, also training performance values are displayed. The figures represent averages taken over 60 independent simulations differing in the bootstrap samples and in the initial parameter values randomly chosen from [-0.3, 0.3].

If out-of-sample [generalization] performance is more important than fast learning, then the neural network models exhibit clear and statistically significant superiority. As can be seen by comparing the *KLIC*-values the summation unit neural network model ranks best, followed by the product unit model and the gravity model. The average generalization performance, measured in terms of $KLIC(M_3)$, is 0.2348 (H = 12), compared to 0.2514 in the case of ${}^{p}\boldsymbol{W}_{L}$ (H = 14), and 0.3036 in the case of $\boldsymbol{t}^{grav}$. These differences in performance are statistically significant[6]. If, however, the goal is to minimize execution time and a sacrifice in generalization accuracy is acceptable, then the gravity model is the model of choice. The gravity model outperforms the neural network models in terms of execution time, the summation unit network model by a factor of 50 and the product unit network model by a factor of 30. But note that this is mainly caused by two factors: *first*, that our implementations were done on a serial platform even though the neural network models are parallelizeable, and, second, that we implemented a rather time consuming termination criterion ($\boldsymbol{k} = 40{,}000$) to stop the training process.

The Origin Constrained Case of Spatial Interactions: TABLE 3 presents some selected results of experiments with different combinations of $H \in \{2,4,6,8,10,12,14\}$

and $\delta \in \{0.0005, 0.0010, 0.0500, 0.1000\}$. Again some considerations are worth making. *First*, a comparison with TABLE 1 illustrates that the consideration of a priori information in form of origin constraints clearly improves the generalization performance more or less dramatically. *Second*, the best result (averaged over the 60 independent simulation runs) in terms of average $KLIC(M3)$ was achieved with $H = 8$ and $\delta = 0.0500$ in both cases, the origin constrained summation unit neural network model ${}^{S}W_1^{OC}$, and the origin constrained product unit neural network model, ${}^{p}W_1^{OC}$. *Third*, the summation unit model version slightly outperforms the product unit version. Again this is primarily due to the logarithmic transformation of the input data in the case of ${}^{Ó}\grave{U}_1^{OC}$. *Fourth*, model approximation improves as the complexity of the model grows with increasing H [up to $H = 8$, except $H = 6$ in the case of ${}^{S}W_1^{OC}$]. *Fifth*, there is clear evidence that $\delta = 0.0500$ tends to lead to the best results (except $H = 6$ in the case of ${}^{S}W_1^{OC}$), while $\delta = 0.0005$ tends to yield the poorest results, with only two exceptions ($H = 2$ and 4) in the case of ${}^{S}W_1^{OC}$. *Sixth*, there is strong evidence that the origin constrained neural network models are much less robust with respect to the choice of the Alopex parameter δ in comparison to their unconstrained counterparts, while the variability in prediction performance over changes in training, internal validation and test samples, and parameter initialization is lower. Finally it is interesting to note that forceful training encourages ${}^{p}W_1^{OC}$ to produce the best generalization ability in all cases considered.

TABLE 4 reports the simulation results for the origin constrained neural network models, ${}^{S}W_1^{OC}$ and ${}^{p}W_1^{OC}$, in comparison with ${}^{orig}t^{grav}$. Training and generalization performance are displayed. The figures represent again averages taken over 60 simulations differing in parameter initialization and bootstrap samples as in the other tables. The modular summation unit neural network performs best, closely followed by the product unit model version. Both outperform the gravity model predictions[7]. The average out-of-sample performance of ${}^{S}W_1^{OC}$ with $H = 8$, measured in terms of $KLIC(M_3)$, is 0.1989, compared to 0.2076 in the case of ${}^{p}W_1^{OC}$ with $H = 8$, and 0.2726 in the case of ${}^{orig}t^{grav}$. The gravity model would be the model of choice if the goal would be to minimize execution time and a sacrifice in generalization would be acceptable.

7. SUMMARY AND DIRECTIONS FOR FUTURE RESEARCH

In this contribution a modest attempt has been made to provide a unified framework for neural spatial interaction modeling including the case of unconstrained and that of origin constrained spatial interaction flows. We suggested and used a more suitable estimation approach than available in literature, namely maximum likelihood estimation under distributional assumptions of Poisson processes. In this way we could avoid the weakness of least squares and normality assumptions that ignore the true integer nature of the flows and approximate a discrete-valued process by an almost certainly misrepresentative continuous distribution. Alopex, a powerful global search procedure, was used to solve the maximum likelihood estimation problem.

Randomness enters in two ways in neural spatial interaction modeling: in the splitting of the data into training, internal validation and test sets on the one side and in choices about parameter initialization on the other. The paper suggests the bootstrapping pairs approach to overcome this problem as well as the problem of scarce data. In addition one receives a better statistical picture of the variability of the out-of-sample performance of the models. The approach is attractive, but computationally intensive. Each bootstrap iteration requires a run of the Alopex procedure on the training bootstrap set. In very large real world problem contexts this computational burden may become prohibitively large.

Although the discussion has been centered on several general families of neural spatial interaction models, only one of the vast number of neural network architectures and only one – even though powerful – estimation approach were considered. Thus, we emphasize that our results are only a first step towards a more comprehensive methodology for neural spatial interaction modeling. There are numerous important areas for further investigation. Especially desirable is the design of a neural network approach suited to deal with the doubly constrained case. Another area for further research is greater automation of the cross-validation training approach to control maximum model complexity by limiting the number of hidden units. Finding good global optimization methods for solving the non-convex training problems is still an important area for further research even though some relevant work can be found in Fischer, Hlavackova-Schindler and Reismann (1999). Finally the model choice problem

deserves further research activities to come up with methods that go beyond the current rules of thumb. We hope that this paper will inspire others to pursue the investigation in neural spatial interaction modeling further as we finally believe that this field is an interesting theoretical area rich with practical applications.

Endnotes

[1] Sigmoid transfer functions such as the logistic function are somewhat better behaved than many other functions with respect to the smoothness of the error surface. They are well behaved outside of their local region in that they saturate and are constant at zero or one outside the training region. Sigmoidal units are roughly linear for small weights [net input near zero] and get increasingly nonlinear in their response as they approach their points of maximum curvature on either side of the midpoint.

[2] Alopex is an acronym for **al**gorithm f**o**r **p**attern **ex**traction.

[3] For the first two iterations, the weights are chosen randomly.

[4] There is virtual unanimity of opinion that site specific variables, such as s_j in this case, are generally best represented as power functions. The specification of f_{ij} is consistent with general consensus that the power function is more appropriate for analyzing longer distance interactions (Fotheringham and O'Kelly 1989).

[5] In the case of the summation unit model versions the input data were preprocessed to logarithmically transformed data scaled into [0.1, 0.9].

[6] assessed by means of the Wilcoxon-Test (comparison of two paired samples). The differences between W_L and $\boldsymbol{t}^{grav}$ are statistically significant at the 1 % level (Z = -3.740, Sig. 0.000) as are the differences between ${}^{p}W_L$ and $\boldsymbol{t}^{grav}$ (Z = -3.269, Sig. 0.001). But the differences between W_L and ${}^{p}W_L$ are not statistically significant at the 1 % level (Z = -1.436, Sig. 0.151).

[7] The differences between ${}^{s}W_1^{OC}$ and ${}^{orig}\boldsymbol{t}^{grav}$ are statistically significant at the 1 % level (Z = -6.684, Sig. 0.000) as are the differences between ${}^{p}W_1^{OC}$ and ${}^{orig}\boldsymbol{t}^{grav}$ (Z = -6.714, Sig. 0.000), while the differences between the neural network models are not statistically significant at the 1 % level (Z = -2.481, Sig. 0.130).

REFERENCES

Bergkvist, E. (2000). "Forecasting Interregional Freight Flows by Gravity Models." *Jahrbuch für Regionalwissenschaft* 20, 133-48.

Bergkvist, E. and L. Westin (1997). Estimation of Gravity Models by OLS Estimation, NLS Estimation, Poisson and Neural Network Specifications. CERUM Regional Dimensions, Working Paper No. 6.

Bia, A. (2000). A Study of Possible Improvements to the Alopex Training Algorithm. In Proceedings of the VIth Brazilian Symposium on Neural Networks, pp. 125-130. IEEE Computer Society Press.

Black, W.R. (1995). "Spatial Interaction Modelling Using Artificial Neural Networks." *Journal of Transport Geography* 3(3), 159-66.

Cover, T.M. (1965). "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition." *IEEE Transactions on Electronic Computers* 14, 326-34.

Efron B (1982). *The Jackknife, the Bootstrap and Other Resampling Plans.* Philadelphia: Society for Industrial and Applied Mathematics.

Fischer, M.M. and S. Gopal (1994). "Artificial Neural Networks: A New Approach to Modelling Interregional Telecommunication Flows." *Journal of Regional Science* 34(4), 503-27.

Fischer, M.M. and Y. Leung (1998). "A Genetic-Algorithms Based Evolutionary Computational Neural Network for Modelling Spatial Interaction Data." *The Annals of Regional Science* 32(3), 437-58.

Fischer, M.M. and Y. Leung (eds.) (2001). *GeoComputational Modelling: Techniques and Applications.* Berlin, Heidelberg and New York: Springer.

Fischer, M.M. and M. Reismann (2000). Evaluating Neural Spatial Interaction Modelling by Bootstrapping. Paper Presented at the 6th World Congress of the Regional Science Association International, Lugano, Switzerland [accepted for publication in *Networks and Spatial Economics*].

Fischer, M.M., K. Hlavackova-Schindler and M. Reismann (1999). "A Global Search Procedure for Parameter Estimation in Neural Spatial Interaction Modelling." *Papers in Regional Science* 78, 119-34.

Fischer, M.M., M. Reismann and K. Hlavackova-Schindler (2001). Neural Network Modelling of Constrained Spatial Interaction Flows, Paper Presented at the 41th Congress of the European Regional Science Association, Zagreb, Croatia [accepted for publication in the *Journal of Regional Science*].

Fotheringham, A.S. and M.E. O'Kelly (1989). *Spatial Interaction Models: Formulations and Applications*. Dordrecht, Boston and London: Kluwer.

Giles, C. and T. Maxwell (1987). "Learning, Invariance, and Generalization in High-Order Neural Networks." *Applied Optics* 26(23), 4972-8.

Gopal, S. and M.M. Fischer (1993). Neural Net Based Interregional Telephone Traffic Models. In *Proceedings of the International Joint Conference on Neural Networks IJCNN 93 Nagoya, Japan, October 25-29*, pp. 2041-4.

Gopal, S. and M.M. Fischer (1996). "Learning in Single Hidden-Layer Feedforward Network Models: Backpropagation in a Spatial Interaction Context." *Geographical Analysis* 28(1), 38-55.

Kullback, S. and R.A. Leibler (1951). "On Information and Sufficiency." *Annals of Mathematical Statistics* 22, 78-86.

Longley, P.A., S.M. Brocks, R. McDonnell and B. MacMillan (eds.) (1998). *Geocomputation: A Primer*. Chichester: John Wiley.

Mozolin, M., J.-C. Thill and E.L. Usery (2000). "Trip Distribution Forecasting with Multilayer Perceptron Neural Networks: A Critical Evaluation." *Transportation Research B* 34, 53-73.

Nijkamp, P., A. Reggiani and T. Tritapepe (1996). Modelling Intra-Urban Transport Flows in Italy. TRACE Discussion Papers TI 96-60/5, Tinbergen Institute, The Netherlands.

Openshaw, S. (1993). "Modelling Spatial Interaction Using a Neural Net." In *Geographic Information Systems, Spatial Modeling, and Policy Evaluation*, edited by M.M. Fischer and P. Nijkamp, pp. 147-64. Berlin, Heidelberg and New York: Springer.

Press, W.H., S.A. Teukolsky, W.T. Vetterling and B.P. Flannery (1992). *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge: University Press.

Reggiani, A. and T. Tritapepe (2000). "Neural Networks and Logit Models Applied to Commuters' Mobility in the Metropolitan Area of Milan." In *Neural Networks in Transport Applications*, edited by V. Himanen, P. Nijkamp and A. Reggiani, pp. 111-29. Aldershot: Ashgate.

Rumelhart, D.E., R. Durbin R., R. Golden and Y. Chauvin (1995). "Backpropagation: The Basic Theory." In *Backpropagation: Theory, Architectures and Applications*, edited by Y. Chauvin and D.E. Rumelhart, pp. 1-34. Hillsdale [NJ]: Lawrence Erlbaum Associates.

Thill, J.-C. and M. Mozolin (2000). "Feedforward Neural Networks for Spatial Interaction: Are they Trustworthy Forecasting Tools?" In *Spatial Economic Science: New Frontiers in Theory and Methodology*, edited by A. Reggiani, pp. 355-81. Berlin, Heidelberg and New York: Springer.

Young, T.Y. and T.W. Calvert (1974). *Classification, Estimation and Pattern Recognition.* Amsterdam: Elsevier.

List of Figures:

List of Tables:

FIG. 1. Architecture of the Unconstrained Summation Unit Neural Spatial Interaction Models as defined by Equation (1) for $N = 3$

FIG. 2. Architecture of the Unconstrained Product Unit Neural Spatial Interaction Models as defined by Equation (3) for $N = 3$

FIG. 3. Origin Constrained Product Unit Neural Spatial Interaction Models as Defined by Equation (5)

TABLE 1

The Unconstrained Case of Spatial Interaction – Summation Unit and Product Unit Neural Networks, W_L and ${}^{p}W_L$ [see Equations (2) and (4)], Estimated by the Alopex Procedure: The Choice of H and δ [T = 1,000; S = 10]

		Summation Unit Network			Product Unit Network		
		$KLIC(M_1)$	$KLIC(M_2)$	$KLIC(M_3)$	$KLIC(M_1)$	$KLIC(M_2)$	$KLIC(M_3)$
H = 2	δ = 0.0005	0.2396	0.2555	0.2744	**0.2931**	**0.2784**	0.3228
		(0.0619)	(0.1934)	(0.1718)	(0.0763)	(0.1398)	(0.1623)
	δ = 0.0010	0.2415	**0.2334**	**0.2535**	0.2955	0.2847	0.3199
		(0.0565)	(0.1426)	(0.1334)	(0.0740)	(0.1335)	(0.1635)
	δ = 0.0050	**0.2327**	0.2418	0.2605	0.3084	0.3007	0.3241
		(0.0545)	(0.1427)	(0.1356)	(0.0749)	(0.1334)	(0.1535)
	δ = 0.0100	0.2436	0.2447	0.2648	0.3132	0.2955	**0.3094**
		(0.0558)	(0.1361)	(0.1428)	(0.0814)	(0.1367)	(0.1572)
H = 4	δ = 0.0005	**0.2252**	0.2433	0.2771	0.2367	0.2316	0.2779
		(0.0598)	(0.1921)	(0.1756)	(0.0684)	(0.1284)	(0.1449)
	δ = 0.0010	0.2268	**0.2238**	0.2622	**0.2278**	**0.2311**	**0.2725**
		(0.0581)	(0.1412)	(0.1368)	(0.0546)	(0.1201)	(0.1451)
	δ = 0.0050	0.2268	0.2259	**0.2539**	0.2629	0.2516	0.2943
		(0.0572)	(0.1363)	(0.1333)	(0.0853)	(0.1459)	(0.1575)
	δ = 0.0100	0.2294	0.2250	0.2606	0.2694	0.2831	0.3140
		(0.0534)	(0.1207)	(0.1284)	(0.0917)	(0.1515)	(0.1685)
H = 6	δ = 0.0005	0.2206	0.2424	0.2568	0.2229	0.2281	0.2727
		(0.0637)	(0.1875)	(0.1544)	(0.0595)	(0.1102)	(0.1455)
	δ = 0.0010	0.2219	0.2188	0.2528	**0.2165**	**0.2264**	**0.2614**
		(0.0602)	(0.1334)	(0.1240)	(0.0490)	(0.1143)	(0.1294)
	δ = 0.0050	**0.2102**	**0.2111**	**0.2447**	0.2399	0.2379	0.2666
		(0.0557)	(0.1176)	(0.1232)	(0.0693)	(0.1177)	(0.1423)
	δ = 0.0100	0.2221	0.2225	0.2470	0.2658	0.2488	0.3021
		(0.0501)	(0.1189)	(0.1280)	(0.0784)	(0.1263)	(0.1554)
H = 8	δ = 0.0005	0.2179	0.2426	0.2608	0.2230	**0.2211**	0.2682
		(0.0673)	(0.1848)	(0.1518)	(0.0584)	(0.1052)	(0.1482)
	δ = 0.0010	**0.2144**	0.2177	**0.2491**	**0.2190**	0.2229	**0.2591**
		(0.0584)	(0.1350)	(0.1121)	(0.0516)	(0.1085)	(0.1304)
	δ = 0.0050	0.2221	0.2256	0.2617	0.2281	0.2331	0.2728
		(0.0552)	(0.1350)	(0.1277)	(0.0615)	(0.1084)	(0.1512)
	δ = 0.0100	0.2159	**0.2171**	0.2534	0.2600	0.2526	0.2879
		(0.0531)	(0.1242)	(0.1285)	(0.0847)	(0.1313)	(0.1679)
H = 10	δ = 0.0005	0.2149	0.2416	0.2623	**0.2150**	**0.2199**	0.2551
		(0.0663)	(0.1895)	(0.1589)	(0.0498)	(0.1079)	(0.1310)
	δ = 0.0010	0.2189	0.2247	**0.2395**	0.2167	0.2248	**0.2528**
		(0.0544)	(0.1363)	(0.1190)	(0.0541)	(0.1129)	(0.1304)
	δ = 0.0050	0.2160	**0.2174**	0.2423	0.2358	0.2341	0.2616
		(0.0573)	(0.1262)	(0.1189)	(0.0671)	(0.1233)	(0.1601)
	δ = 0.0100	**0.2138**	0.2214	0.2415	0.2563	0.2504	0.2812
		(0.0576)	(0.1205)	(0.1327)	(0.0748)	(0.1205)	(0.1836)
H = 12	δ = 0.0005	0.2146	0.2430	0.2588	0.2127	0.2171	**0.2552**
		(0.0640)	(0.1849)	(0.1581)	(0.0462)	(0.1056)	(0.1416)
	δ = 0.0010	0.2163	0.2190	**0.2348**	**0.2110**	**0.2098**	0.2667
		(0.0591)	(0.1346)	(0.1070)	(0.0460)	(0.1084)	(0.1843)
	δ = 0.0050	0.2181	0.2227	0.2535	0.2206	0.2340	0.2995
		(0.0555)	(0.1216)	(0.1175)	(0.0617)	(0.1039)	(0.1564)
	δ = 0.0100	**0.2092**	**0.2158**	0.2513	0.2480	0.2527	0.2595
		(0.0529)	(0.1191)	(0.1252)	(0.0944)	(0.1300)	(0.1979)
H = 14	δ = 0.0005	0.2139	0.2395	0.2537	**0.2067**	**0.2111**	**0.2514**
		(0.0626)	(0.1867)	(0.1565)	(0.0445)	(0.1083)	(0.1390)
	δ = 0.0010	0.2160	0.2203	0.2488	0.2099	0.2182	0.2554
		(0.0564)	(0.1350)	(0.1254)	(0.0457)	(0.1102)	(0.1470)
	δ = 0.0050	0.2144	**0.2153**	**0.2409**	0.2335	0.2360	0.2833
		(0.0561)	(0.1187)	(0.1190)	(0.0691)	(0.1132)	(0.1624)
	δ = 0.0100	**0.2120**	0.2254	0.2483	0.2391	0.2477	0.3008
		(0.0547)	(0.1158)	(0.1233)	(0.0774)	(0.1178)	(0.2717)

Note: KLIC-performance values represent the mean (standard deviation in brackets) of B = 60 bootstrap replications differing in both the initial parameter values randomly chosen from [-0.3; 0.3] and the data split. $KLIC(M_1)$: In-sample performance measured in terms of average *KLIC* (the best value for a given H in bold); $KLIC(M_2)$: Validation performance measured in terms of average *KLIC* (the best values for a given H in bold); $KLIC(M_3)$: Out-of-sample performance measured in terms of average *KLIC* (the best values for a given H in bold); M consists of 992 patterns, M_1 of 496 patterns, M_2 of 248 patterns and M_3 of 248 patterns.

TABLE 2

Benchmark Comparisons of the Summation Unit and Product Unit Neural Networks, W_L and ${}^{p}W_L$, with the Gravity Model t^{grav} for Modeling Unconstrained Spatial Interactions

	Summation Unit Neural Network [H = 12; δ= 0.001]	***Product Unit Neural Network*** [H = 14; δ= 0.0005]	***Gravity Model*** [δ= 0.0005]
In-Sample (Training) Performance			
$KLIC(M_1)$	0.2163	0.2067	0.2991
	(0.0591)	(0.0445)	(0.0717)
Out-of-Sample (Testing) Performance			
$KLIC(M_3)$	0.2348	0.2514	0.3036
	(0.1070)	(0.1390)	(0.1952)

Note: *KLIC*-performance values represent the mean (standard deviation in brackets) of B = 60 bootstrap replications differing in the initial parameter values randomly chosen from [+0.3, -0.3] and the data split; the testing set consists of 248 patterns and the training set of 496 patterns.

TABLE 3

The Origin Constrained Case of Spatial Interaction – Summation Unit and Product Unit Neural Networks, ${}^{S}W_1^{OC}$ and ${}^{P}W_1^{OC}$ [see Equations (6) and (8)], Estimated by the Alopex Procedure: The Choice of H and δ [T = 1,000; S = 10]

		Summation Unit Network			Product Unit Network		
		$KLIC(M_1)$	$KLIC(M_2)$	$KLIC(M_3)$	$KLIC(M_1)$	$KLIC(M_2)$	$KLIC(M_3)$
H = 2	δ = 0.0005	0.3521	0.3684	0.3809	0.2693	0.2810	0.2939
		(0.0724)	(0.0998)	(0.1034)	(0.0989)	(0.1126)	(0.1113)
	δ = 0.0010	0.3515	0.3688	0.3815	0.2340	0.2425	0.2538
		(0.0711)	(0.1008)	(0.1038)	(0.0763)	(0.1057)	(0.0937)
	δ = 0.0500	**0.1958**	0.2049	**0.2131**	**0.2037**	**0.2071**	**0.2181**
		(0.0520)	(0.0877)	(0.0735)	(0.0575)	(0.0785)	(0.0747)
	δ = 0.1000	0.1982	**0.1976**	0.2141	0.2188	0.2250	0.2328
		(0.0518)	(0.0777)	(0.0808)	(0.0596)	(0.0890)	(0.0760)
H = 4	δ = 0.0005	0.3530	0.3669	0.3843	0.2422	0.2543	0.2718
		(0.0719)	(0.0947)	(0.1036)	(0.0860)	(0.1058)	(0.1289)
	δ = 0.0010	0.3540	0.3655	0.3855	0.2175	0.2265	0.2404
		(0.0710)	(0.0949)	(0.1042)	(0.0809)	(0.0857)	(0.0905)
	δ = 0.0500	**0.1854**	0.1867	**0.2032**	**0.1953**	**0.1975**	**0.2125**
		(0.0502)	(0.0713)	(0.0747)	(0.0513)	(0.0750)	(0.0749)
	δ = 0.1000	0.1862	**0.1867**	0.2051	0.2105	0.2133	0.2271
		(0.0505)	(0.0702)	(0.0686)	(0.0579)	(0.0835)	(0.0785)
H = 6	δ = 0.0005	0.3523	0.3695	0.3820	0.2351	0.2462	0.2662
		(0.0723)	(0.1002)	(0.1070)	(0.0749)	(0.1013)	(0.1166)
	δ = 0.0010	0.3505	0.3663	0.3776	0.2078	0.2134	0.2277
		(0.0704)	(0.1004)	(0.1006)	(0.0604)	(0.0753)	(0.0801)
	δ = 0.0500	0.1862	0.1883	0.2067	**0.1915**	**0.1941**	**0.2084**
		(0.0481)	(0.0726)	(0.0703)	(0.0474)	(0.0751)	(0.0722)
	δ = 0.1000	**0.1858**	**0.1868**	**0.2050**	0.2019	0.2046	0.2211
		(0.0463)	(0.0743)	(0.0701)	(0.0468)	(0.0764)	(0.0784)
H = 8	δ = 0.0005	0.3525	0.3678	0.3822	0.2257	0.2315	0.2495
		(0.0730)	(0.1004)	(0.1031)	(0.0621)	(0.0871)	(0.0940)
	δ = 0.0010	0.3521	0.3667	0.3817	0.2136	0.2190	0.2369
		(0.0721)	(0.0997)	(0.1034)	(0.0801)	(0.0870)	(0.1034)
	δ = 0.0500	**0.1790**	**0.1825**	**0.1989**	**0.1916**	**0.1905**	**0.2076**
		(0.0456)	(0.0697)	(0.0684)	(0.0475)	(0.0741)	(0.0707)
	δ = 0.1000	0.1822	0.1844	0.2024	0.2032	0.2039	0.2193
		(0.0441)	(0.0677)	(0.0716)	(0.0527)	(0.0809)	(0.0834)
H = 10	δ = 0.0005	0.3532	0.3673	0.3850	0.2267	0.2369	0.2505
		(0.0728)	(0.0989)	(0.1043)	(0.0684)	(0.0971)	(0.0990)
	δ = 0.0010	0.3516	0.3675	0.3818	0.2014	0.2076	0.2219
		(0.0713)	(0.0986)	(0.1037)	(0.0488)	(0.0697)	0.0755
	δ = 0.0500	**0.1795**	**0.1820**	**0.2004**	**0.1918**	**0.1949**	**0.2087**
		(0.0440)	(0.0671)	(0.0656)	(0.0478)	(0.0755)	(0.0727)
	δ = 0.1000	0.1840	0.1856	0.2040	0.2047	0.2055	0.2207
		(0.0494)	(0.0701)	(0.0697)	(0.0524)	(0.0782)	(0.0797)
H = 12	δ = 0.0005	0.3586	0.3720	0.3877	0.2357	0.2443	0.2637
		(0.0821)	(0.1038)	(0.1108)	(0.0895)	(0.1009)	(0.1428)
	δ = 0.0010	0.3527	0.3662	0.3812	0.2094	0.2153	0.2251
		(0.0724)	(0.0980)	(0.1018)	(0.0563)	(0.0760)	(0.0776)
	δ = 0.0500	0.1793	0.1824	**0.2005**	**0.1911**	**0.1920**	**0.2101**
		(0.0443)	(0.0686)	(0.0647)	(0.0529)	(0.0757)	(0.0717)
	δ = 0.1000	**0.1788**	**0.1822**	0.2012	0.2038	0.2081	0.2202
		(0.0444)	(0.0670)	(0.0691)	(0.0532)	(0.0845)	(0.0763)
H = 14	δ = 0.0005	0.3523	0.3683	0.3809	0.2174	0.2284	0.2389
		(0.0714)	(0.1003)	(0.1031)	(0.0647)	(0.0826)	(0.0895)
	δ = 0.0010	0.3514	0.3682	0.3801	0.2036	0.2101	0.2211
		(0.0714)	(0.1003)	(0.1037)	(0.0518)	(0.0754)	(0.0690)
	δ = 0.0500	**0.1798**	**0.1831**	**0.1992**	**0.1912**	**0.1924**	**0.2126**
		(0.0467)	(0.0713)	(0.0663)	(0.0479)	(0.0773)	(0.0725)
	δ = 0.1000	0.1829	0.1865	0.2052	0.2055	0.2063	0.2244
		(0.0464)	(0.0704)	(0.0698)	(0.0497)	(0.0824)	(0.0822)

Note: KLIC-performance values represent the mean (standard deviation in brackets) of B = 60 bootstrap replications differing in both the initial parameter values randomly chosen from [-0.3; 0.3] and the data split. $KLIC(M_1)$: In-sample performance measured in terms of average *KLIC* (the best values for a given H in bold); $KLIC(M_2)$: Validation performance measured in terms of average *KLIC* (the best values for a given H in bold); $KLIC(M_3)$: Out-of-sample performance measured in terms of average *KLIC* (the best values for a given H in bold); M consists of 992 patterns, M_1 of 496 patterns, M_2 of 248 patterns and M_3 of 248 patterns.

TABLE 4

Benchmark Comparisons of the Summation Unit and Product Unit Neural Networks, ${}^{S}\boldsymbol{W}_1^{OC}$ and ${}^{P}\boldsymbol{W}_1^{OC}$, with the Gravity Model ${}^{orig}\boldsymbol{t}^{grav}$ for Modeling Origin Constrained Spatial Interactions

	Summation Unit Neural Network [H = 8; $\boldsymbol{d}$= 0.05]	***Product Unit Neural Network*** [H = 8; $\boldsymbol{d}$= 0.05]	***Gravity Model*** [$\boldsymbol{d}$= 0.1]
In-Sample (Training) Performance			
$KLIC(M_1)$	0.1790	0.1916	0.2532
	(0.0456)	(0.0475)	(0.0601)
Out-of-Sample (Testing) Performance			
$KLIC(M_3)$	0.1989	0.2076	0.2726
	(0.0684)	(0.0707)	(0.0949)

Note: *KLIC*-performance values represent the mean (standard deviation in brackets) of B = 60 bootstrap replications differing in the initial parameter values randomly chosen from [+0.3, -0.3] and the data split; the testing set consists of 248 patterns and the training set of 496 patterns.