Neural Network Modelling of Constrained Spatial Interaction Flows

Design, Estimation and Performance Issues

Manfred M. Fischer

Martin Reismann

Katerina Hlavackova-Schindler

Department of Economic Geography & Geoinformatics,

Vienna University of Economics and Business Administration

Rossauer Lände 23/1, A-1090 Vienna, Austria

phone: +43-1-31336/4836

fax: +43-1-31336/703

e-mail: Manfred.Fischer@wu-wien.ac.at

Paper presented at 41st Congress

of the European Regional Science Association,

Zagreb, Croatia, August 29 – September 1, 2001

Abstract

A novel modular product unit neural network architecture is presented to model singly constrained spatial interaction flows. Modularity is seen as decomposition on the computational level. The network is composed of two processing layers. The first layer is implemented as a layer of functionally independent modules with identical topologies. Each module is a feedforward network with two inputs, H hidden product units and terminates with a single summation unit. The collective outputs of these modules constitute the input to the second processing layer consisting of output units that perform the flow prediction. The efficacy of the model approach is demonstrated for the origin constrained case of spatial interaction using Austrian interregional telecommunication traffic data. The model requires a global search procedure for parameter estimation, such as the Alopex procedure. A benchmark comparison against the standard origin constrained gravity model and the two-stage neural network approach, suggested by Openshaw (1998), illustrates the superiority of the proposed model in terms of generalisation performance measured by ARV and SRMSE.

Keywords: Origin constrained or destination constrained spatial interaction, neural spatial interaction model, product unit network, Alopex procedure, benchmark performance tests

1 Introduction

The subject of spatial interaction is fundamental to economic geography and regional science. Spatial interaction models are used to facilitate the explanation and prediction of human and economic interaction over geographic space. That there have been relatively few papers in this area in recent years is merely a function of the hiatus that followed a very active period of theory development. The 1960s and 1970s saw a huge outpouring of both theoretical and empirical work. These were the heady days of Stewart and Warntz, Stouffer, Isard, Wilson and Alonso. The empiricism that eminated from their theoretical and methodological contributions filled regional science and geographical journals. The lull came not so much because interest decreased, but because very little in the way of novel theoretical insights. One exception was the excitement over the work of Fotheringham on competing destinations in the early 1980s when several new models were developed and new perspectives added (Fischer and Getis 1999).

In more recent years, the major influence stems both from the emerging data-rich environment and from technological innovations. The powerful and fast computing environment now upon us has brought many scholars to spatial interaction theory once again, either by utilising evolutionary computation to breed novel forms of spatial interaction models (see Openshaw 1988; Turton, Openshaw and Diplock 1997) or network-based approaches to spatial interaction (see, for example, Openshaw 1993, 1998, Fischer and Gopal 1994, Black 1995, Fischer, Hlavackova-Schindler and Reismann 1999, Bergkvist 2000, Reggiani and Tritapepe 2000, Mozolin, Thill and Usery 2000) leading to neural spatial interaction models. Neural spatial interaction models are termed *neural* in the sense that they have been inspired by neuroscience. But they are more closely related to conventional spatial interaction of the gravity type than they are to neurobiological models.

Interest in the recent past has largely focused on some crucuial issues in unconstrained neural spatial interaction modelling (see, for example, Fischer, Hlavackova-Schindler, and Reismann 1999, Fischer 2000). These models represent a rich and flexible family 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. The paper presents a novel neural network approach for the case of origin constrained or destination constrained spatial interaction flows. The approach is based on a modular connectionist architecture that may be viewed as a linked collection of functionally independent neural modules with identical 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 some sort of the Bradley-Terry-Luce model as non-linear output transfer function multiplied with a bias term that implements the accounting constraint.

The efficacy of the model approach is demonstrated for the origin-constrained case by using interregional telecommunication traffic data for Austria, noisy real world data of limited record length. The Alopex procedure, a global search procedure, provides an appropriate optimisation scheme to produce Least Square (LS)-estimates of the model parameters. The prediction quality is measured in terms of two performance statistics, average relative variances and the standardised root mean square error. A benchmark comparison shows that the proposed model outperforms origin-constrained gravity model predictions and predictions obtained by applying the two-stage neural network approach suggested by Openshaw (1998).

The reminder of this paper is structured as follows. The next section provides some background information relevant for spatial interaction modelling first, describes then the basic features of unconstrained neural spatial interaction models and finally discusses briefly how a priori information on accounting constraints can be treated from a neural network perspective. Section 3 presents the network architecture and the mathematics of the modular product unit neural network model. Moreover, it points to some crucial issues that have to be addressed when applying the model in a real world context. Section 4 is devoted to the issue of training the network model. The discussion starts by viewing the parameter estimation problem of the model as least squares (LS) learning and continues with a description of the Alopex procedure, a global search procedure, that provides an appropriate optimising scheme for LS-learning. It is emphasised that the main goal of network training is to minimise the learning error while ensuring good network model generalisation. The most common approach in practice is to check the network performance periodically during training to assure that further training improves generalisation as well as reduces learning error. Section 5 presents the results of a benchmark comparison of the model against the standard origin constrained gravity model and the two-stage neural network approach that treats the prediction of flows and the imposing of accounting constraints as two independent issues. The testbed for the evaluation uses interregional telecommunication traffic data from Austria. Section 6 summarises the results achieved, and outlines some directions for future research.

2 Background

2.1 Definitions and the Generic Interaction Model of the Gravity Type

Suppose we have a spatial system consisting of I origins and J destinations and let t_{ij} denote the volume of interaction from spatial unit (region) i to j $\left(i = 1, ..., I; j = 1, ..., J\right)$. This information may be displayed in the form of an interaction matrix of the following kind

$$T_{I\times J} = \begin{bmatrix} t_{11} & \cdots & t_{1j} & \cdots & t_{1J} \\ \vdots & & \vdots & & \vdots \\ t_{i1} & \cdots & t_{ij} & \cdots & t_{iJ} \\ \vdots & & \vdots & & \vdots \\ t_{I1} & \cdots & t_{ij} & \cdots & t_{IJ} \end{bmatrix} \tag{1}$$

In some cases the sets of origins and destinations are the same and, thus, $T_{I\times J}$ is a squared matrix. The interpretation of the main diagonal of $T_{I\times I}$ depends on the specific application. For instance, it might represent internal telecommunication flows within region i $\left(i = 1,...I\right)$. Often such values are not recorded. In other applications, for example shopping trips from residential areas to individual shopping malls, the number of origins and destinations may differ and $T_{I\times J}$ will not be square.

For all applications, the i-th row of the matrix $T_{I\times J}$ describes the outflows from region i to each of the J destinations, while inflows from each of the I origins into destination j are described by the j-th column. From $T_{I\times J}$ we can calculate the volume of interaction originating from region i or terminating in region j, that is

$$t_{i\bullet} = \sum_{j=1}^{J} t_{ij} \qquad i = 1,...,I \tag{2}$$

and

$$t_{\bullet j} = \sum_{i=1}^{I} t_{ij} \qquad j = 1,...,J \tag{3}$$

respectively. In turn these marginal sums can be utilised to calculate the overall level of interaction that is defined as

$$t_{\bullet\bullet} = \sum_{i=1}^{I} \sum_{j=1}^{J} t_{ij} \tag{4}$$

The Generic Interaction Model of the Gravity Type

The distribution of interactions within such a system can be described by the generic interaction model of the gravity type that asserts a multiplicative relationship between the interaction frequency and the effects of origin, destination and separation attributes, respectively. In general form it may be written as (see Wilson 1967, Alonso 1978)[1]

$$\tau_{ij} = b_{(ij)}\, r_i\, s_j\, f_{ij} \qquad i = 1,...,I; \quad j = 1,...,J \tag{5}$$

where τ_{ij} is the estimated flow from *i* to *j*. r_i is an origin factor characterising *i* [=measure of origin propulsiveness], s_j a destination factor characterising *j* [=measure of destination attractiveness], and f_{ij} a separation factor that measures separation from *i* to *j*. The separation factor f_{ij} is generally – but not necessarily – assumed to be a function of some univariate measure d_{ij} of separation from *i* to *j*. The exact functional form of each of these three variables is subject to varying degrees of conjecture (see Fotheringham and O'Kelly 1989). $b_{(ij)}$ is a balancing factor with varying subscripts depending on which constraints $\{\tau_{ij}\}$ has to obey concerning $t_{i\bullet}$, $t_{\bullet j}$ or $t_{\bullet\bullet}$. In the origin constrained case, for example, this conservation principle is enforced from the viewpoint of origins only: $b_{(i)} = t_{i\bullet} / \left(r_i \sum_{j \neq i} s_j \, f_{ij} \right)$ guarantees that $t_{i\bullet} = \sum_j \tau_{ij}$.

Alternative forms of the general gravity model (5) can be specified by imposing different constraints on $\{\tau_{ij}\}$ (see, for example Senior 1979, Fotheringham and O'Kelly 1989, Sen and Smith 1995). In the *globally constrained case* the only condition specified is that the total estimated interaction $\tau_{\bullet\bullet}$ equals the total observed interaction $t_{\bullet\bullet}$, in *the origin constrained case* the estimated outflows $\tau_{i\bullet}$ from each *i* have to match the observed outflows $t_{i\bullet}$ (origin constraint), in the *destination constrained case* the estimated inflows $\tau_{\bullet j}$ to each region *j* must equal the observed inflows $t_{\bullet j}$ (destination constraint), and in the *doubly constrained case* the estimated inflows $\tau_{\bullet j}$ and outflows $\tau_{i\bullet}$ have to match their observed counterparts. Note that in the constrained case $b_{(ij)} = 1$.

It is worth noting that in the origin-constrained [also called production constrained] case the origin factor is linearly dependent with the origin specific balancing factor $b_{(i)}$, and in the destination-constrained [also termed attraction-constrained] case the destination factor with the destination-specific balancing factor $b_{(j)}$, while in the doubly-constrained case, the constant of proportionality $b_{(ij)}$ depends on both origin and destination. The origin constraint and the destination constraint are isomorph.

There are different approaches to estimating the generic spatial interaction model (5): the maximum entropy approach developed by Wilson (1967) and the log-linear approach which is a special case of Poisson regression (see, for example, Aufhauser and Fischer 1985). These approaches yield identical estimates of the interaction flows in the case where the interacting units are measured on the same level of aggregation, and identical sets of independent variables are used to calibrate the model.

2.2 The Classical Neural Network Approach to Spatial Interaction Modelling

The neural network approach to model spatial interactions departs from Equation (5) by viewing spatial interaction models as a particular type of input-output model. Given an input vector ***x***, the network model produces an output vector $\tilde{\boldsymbol{y}}$, say $\tilde{\boldsymbol{y}} = \boldsymbol{g}(\boldsymbol{x})$. The function $\boldsymbol{g}$ is not explicitly known, but given by a finite set of samples, say

$M = \{(x^u, y^u) \text{ with } u = 1, ..., U\}$, so that $g(x^u) = y^u$. The set *M* is the set of input and output vectors. The task is to find a continuous function that approximates *M*. In real world application, *U* is generally a small number and the samples contain noise.

The Generic Neural Spatial Interaction Model

In the unconstrained case the challenge is to approximate the real-valued interaction function $g(r_i, s_j, f_{ij}): \Re^3 \rightarrow \Re$, where the 3-dimensional euclidian real space is the input space and the 1-dimensional euclidian real space the output space. In practice only bounded subsets of the spaces are considered. To approximate *g*, we consider the class Ω of feedforward neural network models with three input units, one hidden layer that contains *H* hidden units and a single output unit. The three input units represent measures of origin propulsiveness, destination attractiveness and spatial separation. The output unit, denoted by $\tilde{\boldsymbol{y}}$, represents the estimated flow from *i* to *j*. Formally the neural network model for the unconstrained case of spatial interaction may be written in its general form as:

$$\left\{\Omega; \Re^3 \rightarrow \Re \,\middle|\, \tilde{\boldsymbol{y}} = \Omega(\boldsymbol{x}, \boldsymbol{w}) = \psi\left(\sum_{h=0}^{H} \gamma_h \, \phi_h\left(\sum_{n=0}^{3} \beta_{hn} \, x_n\right)\right), \boldsymbol{x} \in \Re^3; \gamma_h, \beta_{hn} \in \Re\right\} \tag{6}$$

Vector $\boldsymbol{x} = (x_0, x_1, x_2, x_3)$ is the input vector augmented with a bias signal x_0 that can be thought of as being generated by a dummy unit whose output is clamped at 1. Models belonging to $\Omega(\boldsymbol{x}, \boldsymbol{w})$ may have any number of hidden units $(H = 1, 2, ...)$ with connection strengths from hidden to the output unit represented by γ_h. The β_{hn} represent input-to-hidden connection weights. The symbol $\boldsymbol{w} = (w_k \mid k = 1, ..., K = (5H + 1))$ is a convenient short hand notation of the $(5H + 1)$-dimensional vector of all the β_{hn} and γ_h network weights and biases. ϕ_h and ψ are arbitrarily differentiable, generally non-linear transfer functions of the hidden units and the output unit, respectively.

The Classical Unconstrained Neural Spatial Interaction Model

Hornik, Stinchcombe and White (1989) proved that single hidden layer feedforward networks with ψ being the identity function and $\phi_h = \phi \quad (h = 1, ..., H)$ an arbitrary sigmoid transfer function[2] can approximate any measurable function to any degree of accuracy (in appropriate metrics), given sufficiently many hidden units. Thus, the neural spatial interaction models suggested by Fischer, Hlavackova-Schindler and Reismann (1999):

$$\Omega_L(\boldsymbol{x}, \boldsymbol{w}) = \sum_{h=0}^{H} \gamma_h \left(1 + \exp\left(-\sum_{n=0}^{3} \beta_{hn} \, x_n \right)^{-1} \right) \tag{7}$$

represent a rich and flexible family of spatial interaction function approximators. Although it has become common place to view network models such as (7) as kinds of black boxes, this leads to inappropriate applications which may fail not because such network models do not work well but because the issues are not well understood. Failures in applications can often be attributed to inadequate learning (training), inadequate numbers of hidden units, or the presence of a stochastic rather than a deterministic relation between input and target.

Least Squares Learning

If we view (7) as generating a family of approximators (as $\boldsymbol{w}$ ranges over $\boldsymbol{W}$, say) to some specific empirical spatial interaction phenomenon relating inputs $\boldsymbol{x}$ to some response, $\boldsymbol{y}$, then we need a way to pick the best approximation from this family. This is the function of learning in the context of neural network modelling. The goodness of the approximation can be evaluated using an error [penalty] function that measures how well the model output $\tilde{\boldsymbol{y}}$ matches the target output $\boldsymbol{y}$ corresponding to given input $\boldsymbol{x}$. The penalty should be zero when target and model output match, and positive otherwise. A leading case is the least square error function. With this error (penalty) function, learning must arrive at $\boldsymbol{w}^*$ which solves

$$\min_{\boldsymbol{w} \in \boldsymbol{W}} \left[\mathrm{E}\left(\tfrac{1}{2} \left(\boldsymbol{y} - \Omega_L(\boldsymbol{x}, \boldsymbol{w}) \right)^2 \right) \right] = \mathrm{E}\left\{ \tfrac{1}{2} \left(\boldsymbol{y} - \mathrm{E}(\boldsymbol{y} \mid \boldsymbol{x})^2 \right) \right\} + \mathrm{E}\left\{ \tfrac{1}{2} \left[\mathrm{E}(\boldsymbol{y} \mid \boldsymbol{x}) - \Omega_L(\boldsymbol{x}, \boldsymbol{w}) \right]^2 \right\} \tag{8}$$

where E denotes the expectation operator. Finding $\boldsymbol{w}^*$ is precisely the problem of determining the parameters of an optimal least-squares approximator to $\mathrm{E}(\boldsymbol{y} \mid \boldsymbol{x})$, the conditional expectation of $\boldsymbol{y}$ given $\boldsymbol{x}$. The expectation defining the optimand is unknown, so that this problem has to be solved statistically.

Backpropagation of Gradient Descent Errors

Backpropagation of gradient descent errors is such a method that allows parameters to be learned from experience in a process which resembles trial and error (see Rumelhart, Hinton and Williams 1986). Experience is based on empirical observations on the phenomenon of spatial interaction of interest. Thus, we assume to have a training set available consisting of observations x^u $(u = 1, ..., U)$ on the input variables together with observations y^u $(u = 1, ..., U)$ on corresponding target variables, the network model is to learn to associate with x^u. According to the backpropagation of the gradient descent errors procedure one starts with a set of random weights, say $\boldsymbol{w}_0$, and then updates them by the following formula for the n-th step:

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \eta \nabla \Omega_L\left(x^{u+1}, \boldsymbol{w}(n)\right)\left(y^{u+1} - \Omega_L\left(x^{u+1}, \boldsymbol{w}(n)\right)\right) \tag{9}$$

where η is a learning rate and $\nabla \Omega_L$ denotes the gradient of Ω_L with respect to $\boldsymbol{w}$. The weights are adjusted in response to errors in hitting the target where errors are measured in terms of the least square error function. This error is propagated back. Although many modifications of and alternatives to this parameter estimation approach have been suggested in the neural network literature over the past years, experience shows that surprisingly good network model performance can often be achieved with the epoch-based stochastic version of this learning approach (see Fischer and Gopal 1994).

2.3 Departure from the Classical Neural Network Approach

Although classical neural spatial interaction models of type (7) represent a rich and flexible family of spatial interaction function approximators for real world applications, they may be of little practical value in situations where the set of row totals or the set of column totals of the spatial interaction matrix $T_{I \times J}$ is known a priori. For such situations, the families of production constrained and attraction-constrained models of the gravity type had been developed.

The question arises how to build neural network models for the case of singly constrained spatial interactions. Following Openshaw (1998) constrained spatial interaction modelling may be viewed as consisting of two parts:

- The prediction of flows, and
- the imposing of accounting constraints.

These two parts can be treated separately or simultaneously. The question that follows is whether to embed the constraint-handling mechanism within the neural network approach [*one-stage modelling approach*] or whether to estimate the unconstrained neural spatial interaction model first and then to apply the accounting constraints subsequently [*two-stage modelling approach*]. The one-stage modelling approach is harder, requiring major changes to the model structure, while the two-stage approach is much simpler [for an application see Mozolin, Thill and Usery (2000)].

3 The One-Stage Modelling Approach: The Modular Product Unit Network Model

3.1 Why Product rather than Summation Unit Networks?

Classical neural spatial interaction models, such as $\Omega(\boldsymbol{x},\boldsymbol{w})$ and $\Omega_L(\boldsymbol{x},\boldsymbol{w})$, 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 non-linear transfer function, such as the logistic function, is used to squash the sum. Neural network approximation theory has shown the attractivity of such summation networks for unconstrained spatial interaction contexts. But these networks require a larger number of hidden summation units when approximating complex functions g, such as those for mapping constrained interaction phenomena.

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). Although the error surface of product unit networks contains more local minima than when using standard transfer functions, the surface is locally smooth. But the price to be paid is a combinatorial explosion of higher order terms as the number of inputs to the network increases.

The product units introduced by Durbin and Rumelhart (1989) attempt to make use of the above fact. Product unit networks have the advantage that – given an appropriate training algorithm – the units can learn the higher order terms that are required to approximate a specific constrained spatial interaction function. This motivates to utilise the product unit rather than the standard summation unit neural framework for modelling singly constrained interactions over space.

3.2 The Network Architecture

Product units compute the product of inputs, each raised to a variable power. They can be used in a network in many ways, but the overhead required to raise an arbitrary base to an arbitrary power makes it more likely that they will supplement rather than replace summation units (Durbin and Rumelhart 1989).[3] Thus, we use the term *product unit networks* [or product networks] to refer to networks containing both product and summation units.

Figure 1 illustrates the modular network architecture of the product unit neural network that we propose to model the singly constrained case of spatial interactions. 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}, *H* hidden product units $\left[(j-1)H+1, ..., (j-1)H+h, ..., jH\right]$, denoted by the symbol Π, and terminates with a single summation unit, denoted by the symbol Σ. The collective outputs of these modules constitute the input to the second processing layer consisting of *J* output units that perform the flow prediction by applying some sort of the Bradley-Terry-Luce model and enforcing satisfactorily the conservation rule of interaction from the viewpoint of origins [destinations][4].

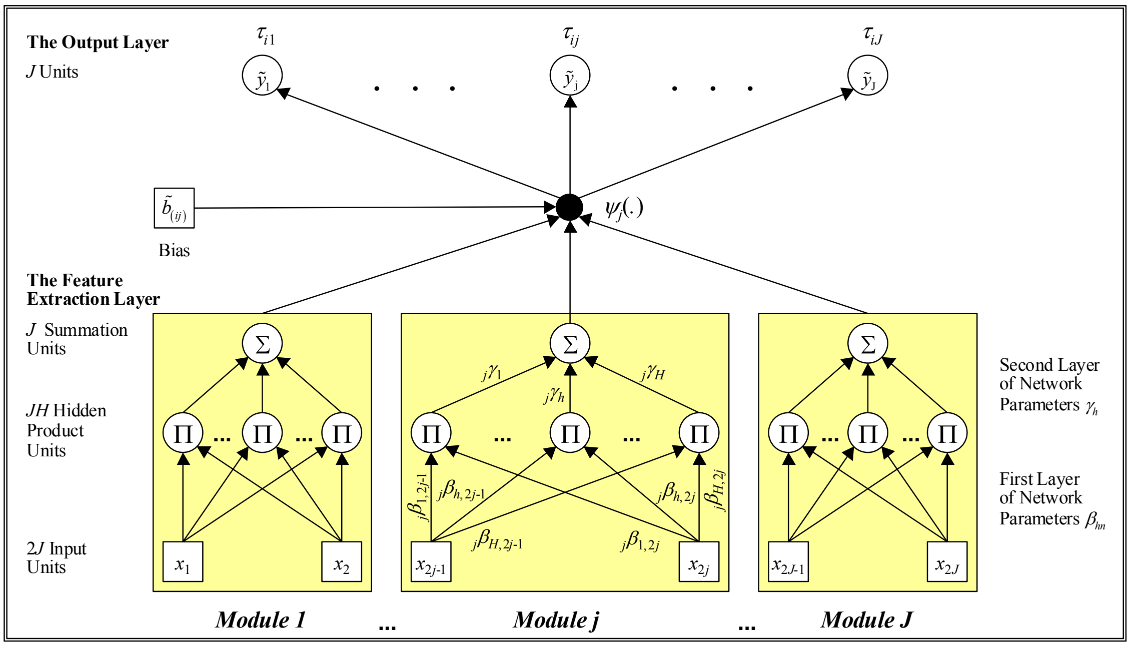


Figure 1: Architecture of the Product Unit Neural Spatial Interaction Model: The Singly Constrained Case

The first layer of network parameters includes 2*JH* connection weights, so that

$$_1\boldsymbol{w} = \left({}_j\beta_{(j-1)H+1,2j-1}, \ldots, {}_j\beta_{(j-1)H+h,2j-1}, \ldots, {}_j\beta_{jH,2j-1}, {}_j\beta_{(j-1)H+1,2j}, \ldots, {}_j\beta_{(j-1)H+h,2j}, \ldots, {}_j\beta_{jH,2j} \right) \tag{10}$$

while the second layer contains *JH* weights:

$$_2\boldsymbol{w} = \left({}_j\gamma_{(j-1)H+1}, \ldots, {}_j\gamma_{(j-1)H+h}, \ldots, {}_j\gamma_{jH} \right) \tag{11}$$

We have incorporated the basic trick of weight sharing into our network design to reduce model complexity. Weight sharing involves forcing the set of connection weights to be identical across the *J* modules. Thus, $\boldsymbol{w} = \left({}_1\boldsymbol{w}, {}_2\boldsymbol{w} \right)$ is a (3*H*)-dimensional rather then a (3*JH*)-dimensional vector. Consequently, notation may be simplified as follows for all $j = 1, ..., J$:

$$(j-1)H + h =: h \tag{12}$$

$$jH =: H \tag{13}$$

$$_{j}\beta_{(j-1)H+h,2j-1} =: \beta_{h,2j-1} \tag{14}$$

$$_{j}\beta_{(j-1)H+h,2j} =: \beta_{h,2j} \tag{15}$$

$$_{j}\gamma_{(j-1)H+h} =: \gamma_{h} \tag{16}$$

3.3 A Mathematical Description

The network architecture described above implements the general class of neural models of singly constrained [SL] spatial interaction

$$\Omega_{SL}(\boldsymbol{x}, \boldsymbol{w})_j = \psi_j \left(\sum_{h=1}^{H} \gamma_h \, \phi_h \left(\prod_{n=2j-1}^{2j} x_n^{\beta_{hn}} \right) \right) \quad j = 1, ..., J \tag{17}$$

with $\phi_h : \Re \rightarrow \Re, \psi_j : \Re \rightarrow \Re$ and a $(2J)$-dimensional vector

$$\boldsymbol{x} = \left(x_1, x_2, ..., x_{2j-1}, x_{2j}, ..., x_{2J-1}, x_{2J} \right) \tag{18}$$

where x_{2j-1} represents a variable s_j pertaining to destination j $(j = 1, ..., J)$ and x_{2j} a variable f_{ij} pertaining to the separation from region *i* to region j $(i = 1, ..., I; j = 1, ..., J)$ of the spatial interaction system under scrutiny. β_{hn} $(h = 1, ..., H; n = 2j-1, 2j)$ are the input-to-hidden connection weights and γ_h $(h = 1, ..., H)$ 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. ψ_j $(j = 1, ..., J)$ represents a non-linear summation unit and ϕ_h $(h = 1, ..., H)$ a linear hidden product unit transfer function.

We restrict our attention to some specific members of the above class. *First*, we assume the hidden transfer function $\phi_h(\cdot)$ to be the identity function. Thus, the output of the h-th hidden product unit of the j-th network module, denoted by ${}_j z_h$, reads as

$$ {}_j z_h = \phi_h\left({}_j net_h \right) = \prod_{n=2j-1}^{2j} x_n^{\beta_{hn}} \qquad j = 1, ..., J;\, h = 1, ..., H \tag{19}$$

Second, we assume that each output unit, $\Omega_{SL}\left(\boldsymbol{x}, \boldsymbol{w}\right)_j =: \tilde{y}_j \quad \left(j = 1, ..., J\right)$, uses a non-linear normalised transfer function $\psi_j(\cdot)$:

$$\Omega_{SL}\left(\boldsymbol{x}, \boldsymbol{w}\right)_j = \psi_j\left(net\ j\right) = \tilde{b}_{(ij)} \frac{net\ j}{\sum_{j'=1}^{J} net\ j'} \qquad j = 1, ..., J \tag{20}$$

that resembles the Bradley-Terry-Luce model augmented by a bias unit $\tilde{b}_{(ij)}$. *net j* is the value of the weighted sum for the j-th output node given by

$$net\ j = \sum_{h=1}^{H} \gamma_h \prod_{n=2j-1}^{2j} x_n^{\beta_{hn}} \qquad j = 1, ..., J \tag{21}$$

The choice of the output transfer function (20) is motivated by the goal to ensure that the network outputs satisfy the conservation principle (Ledent 1985) that is enforced from the viewpoint of origins if $\tilde{b}_{(ij)} = \tilde{b}_{(i)}$ [origin constrained case] or from the viewpoint of destinations if $\tilde{b}_{(ij)} = \tilde{b}_{(j)}$ [destination constrained case]. $\Omega_{SL}\,(\boldsymbol{x}, \boldsymbol{w})_j$ may be interpreted as probability of spatial interactions, conditioned on the output $\sum_{h=1}^{H} {}_j z_h$ of the j-th network module.

With the two above specifications, our modular product unit neural network to model origin constrained spatial interactions reads in a compact way as

$$ {}^1\Omega_{SL}\left(\boldsymbol{x}, \boldsymbol{w}\right)_j = \tilde{b}_{(i)} \frac{\sum_{h=1}^{H} \gamma_h \prod_{n=2j-1}^{2j} x_n^{\beta_{hn}}}{\sum_{j'=1}^{J} \sum_{h'=1}^{H} \gamma_{h'} \prod_{n=2j'-1}^{2j'} x_n^{\beta_{h'n}}} \qquad j = 1, ..., J \tag{22}$$

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 $1/t_{i\bullet}$ The relation of (22) to the generic spatial interaction model (5) becomes evident when setting $x_{2j-1} = s_j$ and $x_{2j} = f_{ij}$:

$$^{1}\tau_{ij} := {}^{1}\Omega_{SL}\left(\boldsymbol{x},\boldsymbol{w}\right)_j \;=\; \tilde{b}_{(i)} \frac{\sum_{h=1}^{H} \gamma_h \, s_j^{\beta_{h1}} \, f_{ij}^{\beta_{h2}}}{\sum_{j'=1}^{J}\sum_{h'=1}^{H} \gamma_{h'} \, s_{j'}^{\beta_{h'1}} \, f_{ij'}^{\beta_{h'2}}} \qquad j = 1, ..., J \tag{23}$$

Analogously one arrives at the modular product unit neural network for the destination constrained case:

$$^{2}\Omega_{SL}\left(\boldsymbol{x},\boldsymbol{w}\right)_i = \tilde{b}_{(j)} \frac{\sum_{h=1}^{H} \gamma_h \prod_{n=2i-1}^{2i} x_n^{\beta_{hn}}}{\sum_{i'=1}^{I}\sum_{h'=1}^{H} \gamma_{h'} \prod_{n=2i'-1}^{2i'} x_n^{\beta_{h'n}}} \qquad i = 1, ..., I \tag{24}$$

where $\tilde{b}_{(j)}$ is the bias signal that can be thought as being generated by a 'dummy unit' whose output is clamped at the scalar $1/t_{\bullet j}$. Set $x_{2i-1} = r_i$ and $x_{2j} = f_{ij}$ then (24) becomes

$$^{2}\tau_{ij} := {}^{2}\Omega_{SL}\left(\boldsymbol{x},\boldsymbol{w}\right)_i = \tilde{b}_{(j)} \frac{\sum_{h=1}^{H} \gamma_h \, r_i^{\beta_{h1}} \, f_{ij}^{\beta_{h2}}}{\sum_{i'=1}^{I}\sum_{h'=1}^{H} \gamma_{h'} \, r_{i'}^{\beta_{h'1}} \, f_{i'j}^{\beta_{h'2}}} \qquad i = 1, ..., I \tag{25}$$

3.4 Two Issues of Crucial Importance for Real-World Applications

Two major issues have to be addressed when applying the spatial interaction model Ω_{SL} in a real world context: *first*, the issue of finding a suitable number H of hidden product units [the so-called representation problem], and *second*, the issue of network training or learning [the so-called learning problem]. The first issue is a challenging task because the number of hidden product units affects the generalisation performance of the model. Small networks learn slowly or may not learn at all to an acceptable error level. Larger networks usually learn better, but such sizes may lead to generalisation degradation which is known as overtraining or overfitting. The correct number of hidden units is problem dependent and a function of the complexity of the input-output mapping to be realised by the model and the required accuracy. The common goal is to simplify the model in terms of H without sacrificing its generalisation performance.

Various techniques have been developed in the neural network literature to control the effective complexity of neural network models, in most cases as part of the network training process itself. Since the maximum complexity of our model can be controlled by limiting the number of hidden units, one obvious approach to the bias-variance trade-off is to train several model candidates with different numbers of hidden product units, and to select that model which gives the best generalisation performance. An obvious drawback of such an approach is its trial and error nature.

Another and more principled approach to the problem, that has been utilised by Fischer and Gopal (1994), is the procedure of *stopped* or *cross-validated training*. Here, an overparametrised model (larger H) is trained until the error on further independent data, called validation set, deteriorates, then training is stopped. This contrasts to the above approach since the choice of H does not require convergence of the training process. The training process is used to perform a directed search of the parameter space for a model that does not overfit the data and, thus, demonstrates generalisation performance. But this approach has its shortcomings too. First it might be hard in practice to identify when to stop training. Second, the results may depend on the specific training set-validation set pair chosen. Third, the model which has the best performance on the validation set might not be the one with the best performance on the test set.

The second issue involves network training or learning [i.e. parameter estimation]. This issue will be addressed in the next section in some more detail.

4 Training the Modular Product Unit Network Model

4.1 The Optimisation Problem

Having identified the model structure for singly constrained spatial interaction prediction in the previous section, we can now follow section 2.2 to view network training in an optimisation context and proceed by considering the parameter estimation problem as least squares learning. The goal of least squares learning is to find $\boldsymbol{w}^*$ so that the least square error function, say $\boldsymbol{Q}$, is minimized[5]:

$$\min_{\boldsymbol{w}} \boldsymbol{Q}\left(x^{u1}, y^{u1}, \boldsymbol{w}\right) = \min_{\boldsymbol{w}} \sum_{\left(x^{u1}, y^{u1}\right) \in M_1} \left(y^{u1} - {}^{1}\Omega_{SL}\left(x^{u1}, \boldsymbol{w}\right)\right)^2 \tag{26}$$

where M_1 denotes the training set available consisting of observations, say $x^{u1}\left(u = 1, ..., U_1\right)$ on the input variables together with observations, say $y^{u1}\left(u = 1, ..., U_1\right)$, on corresponding target variables, the network model is to learn to associate with x^{u1}.

$\boldsymbol{Q}\left(x^{u1}, y^{u1}, \boldsymbol{w}\right)$ is non-negative, continuously differentiable on the (3H)-dimensional parameter space which is a finite dimensional closed bounded domain and, thus, compact. The compactness of the parameter space is of great theoretical convenience. It can be shown that $\boldsymbol{Q}\left(x^{u1}, y^{u1}, \boldsymbol{w}\right)$ assumes its value $\boldsymbol{w}^*$ as the weight minimum under certain conditions. But characteristically there exist many minima in real world applications all of which satisfy

$$\nabla \boldsymbol{Q}\left(x^{u1}, y^{u1}, \boldsymbol{w}^*\right) = 0 \tag{27}$$

where $\nabla \boldsymbol{Q}$ denotes the gradient of $\boldsymbol{Q}$. The minimum for which the value of $\boldsymbol{Q}$ is smallest is termed the global minimum while other minima are called local minima. Unfortunately there is no guarantee about which kind of minimum is encountered.

The fraction of the parameter space that contains a solution depends on the capacity of the network model [in an information theoretic sense] and the complexity of the problem at hand. Given the mountainous error surface that is characteristic for product unit networks, a local search algorithm such as backpropagation of gradient descent errors is ineffective and usually converges to local minima. In contrast, global search algorithms such as, for example, the Alopex procedure have heuristic strategies to help escape from local minima.

The success of global search procedures in finding a global minimum of a given function such as $\boldsymbol{Q}\left(x^{u1}, y^{u1}, \boldsymbol{w}\right)$ over $\boldsymbol{w} \in \boldsymbol{W}$ hinges on the balance between an exploration process, a guidance process and a convergence inducing process (Hassoun 1995). *The exploration process* gives the search a mechanism for sampling a sufficiently diverse set of parameters $\boldsymbol{w}$ in $\boldsymbol{W}$. The Alopex[6] procedure performs an exploration process that is stochastic in nature. The *guidance process* is an implicit process that evaluates the relative quality of search points [i.e., two consecutive search points] and uses correlation guidance to move towards regions of higher-quality solutions in the parameter space. Finally the *convergence-inducing process* ensures the convergence of the search to find a fixed solution $\boldsymbol{w}^*$. The convergence-inducing process is realised effectively by a parameter T, called temperature, that is gradually decreased over time. The dynamic interaction among these three processes is responsible for giving the Alopex search process its global optimising character.

4.2 The Alopex Procedure

Consider a training data set $\left(x^{u1}, y^{u1}\right)$ with $u_1 = 1, ..., U_1$. We assume that the data was generated by some true underlying function $g\left(\boldsymbol{x}\right)$. Our objective is to learn the parameter $\boldsymbol{w} = \left(w_k \mid k = 1, ..., 3H\right)$ of the approximating function $\Omega_{SL}\left(\boldsymbol{x}, \boldsymbol{w}\right)$ whose form is dependent upon the choice of H.

Alopex is a correlation-based method for solving the parameter estimation problem. The error function $\boldsymbol{Q}$ is minimised by means of weight changes that are calculated for the n-th step ($n > 2$) of the iteration process in batch mode as follows[7]:

$$w_k\left(n\right) = w_k\left(n-1\right) + \delta \operatorname{sgn}\left(\varepsilon - p_k\left(n\right)\right) \tag{28}$$

where δ is the step size that has to be chosen a priori and ε an uniformly distributed random value with $\varepsilon \in [0,1]$. The probability of change of the parameter is calculated as

$$p_k(n) = \left(1 + \exp\left(C_k(n)/T(n)\right)\right)^{-1} \tag{29}$$

with $C_k(n)$ given by the correlation

$$\begin{aligned} C_k(n) &= \left[w_k(n-1) - w_k(n-2)\right]\left[\boldsymbol{Q}\left(\boldsymbol{x},\boldsymbol{y},w_k(n-1)\right) - \boldsymbol{Q}\left(\boldsymbol{x},\boldsymbol{y},w_k(n-2)\right)\right] \\ &= \left[\Delta w_k(n)\right]\left[\Delta\boldsymbol{Q}\left(\boldsymbol{x},\boldsymbol{y},w_k(n)\right)\right] \end{aligned} \tag{30}$$

The weight will be incremented in a given fixed magnitude δ, when $\Delta w_k > 0$, and the opposite when it is less than zero. The sign of C_k indicates whether $\boldsymbol{Q}$ varies in the same way as w_k. If $C_k > 0$, both $\boldsymbol{Q}$ and w_k will be raised or lowered. If $C_k < 0$, one will be lowered and the other one raised.

If *T* is too small, the algorithm gets trapped into local minima of $\boldsymbol{Q}$. Thus, the value of *T* for each iteration, *T*(*n*), is chosen using the following heuristic 'annealing schedule':

$$T(n) = \begin{cases} \dfrac{\delta}{3HN}\displaystyle\sum_k \sum_{n'=n-N}^{n-1} \left|C_k(n')\right| & \text{if } n \text{ is a multiple of } N \\ T(n-1) & \text{otherwise} \end{cases} \tag{31}$$

where 3*H* denotes the number of weights. The annealing schedule controls the randomness of the algorithm. 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 $p_k \cong 0.5$. This means that there is the same probability to increment or decrement the weights and that the direction of the steps is now random. In other words, high values of *T* imply a random walk, while low values cause a better correlation guidance (see Bia 2000). The effectiveness of Alopex in locating global minima and its speed of convergence critically depends on the balance of the size of the feedback term $\Delta w_k\ \Delta\boldsymbol{Q}$ and the temperature *T*. If *T* is very large compared to $\Delta w_k\ \Delta\boldsymbol{Q}$

the process does not converge. If T is too small, a premature convergence to a local minimum might occur.

Initial Values

The algorithm has three parameters: the initial temperature T, the number of iterations, N, over which the correlations are averaged for annealing, and the step size δ. The temperature T and the N-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 N iterations and reset it to an appropriate value according to Equation (31). N may be chosen between 10 and 100. In contrast to T and N, δ is a critical parameter that has to be selected with care. There is no way to a priori identify δ.

The Termination Criterion

An important issue associated with network training is the *termination criterion*. The main goal of training is to minimise the learning error while ensuring good model generalisation. It has been observed that forceful training may not produce network models with adequate generalisation 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 generalisation as well as reduces learning error. 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 or its improvement is extremely slow. Then the training process should be stopped. In our implementation of the Alopex procedure network training is stopped when $\kappa = 40,000$ consecutive iterations are unsuccessful.

κ has been chosen so large at the expense of the greater training time, to ensure more reliable estimates. Of course, setting the number of unsuccessful iterations to 40,000 (or more) does not guarantee that there would be any successful steps ahead if training continued. At some stage a training algorithm may recover from some local attractor and accomplish further error minimisation, but we require it should occur within a certain number of iterations. Obviously, when training is stopped, the final set of network weights does not correspond to the best result found. It is, thus, necessary to store the parameter values in a separate array every time a successful training step is made. At the end of the training process the best set of parameter values is then recalled.

5 Benchmark Comparisons

The attraction of the approach suggested to model the case of singly constrained spatial interaction depends not only on the awareness of what it can offer, but also on empirical illustration of what can be gained in comparison to alternative model approaches. The standard origin-constrained gravity model and the two-stage neural network approach, suggested by Openshaw (1998) and implemented by Mozolin, Thill and Usery (2000), are used as benchmark models. All three models were estimated by means of the Alopex procedure to eliminate the effect of different estimation procedures on the result. In order to do justice to each model, the δ-parameter was systematically sought for each model.

5.1 Performance Measures

The ultimate goal of any function approximator is its usefulness to generate accurate out-of-sample prediction. One way to directly assess the generalisation ability is to measure how well the approximator predicts the flows for new input data which was not used to fit the model. For this purpose some performance measure is required.

One needs to be very careful when selecting a measure to compare different models. A comparison may become meaningless if the performance measure has been utilised to estimate the parameters in the case of one model, but not in the others. There is some literature that ignores this. In this study model performance is measured on the testing [prediction, out-of-sample] data set, say $M_3 = \{(x^{u3}, y^{u3})$ with $u3 = 1,..,U_3\}$, by means of two overall performance measures. The *first performance measure* are the average relative variances, $\mathrm{ARV}(M_3)$, a normalised mean squared error metric that is widely utilised in the neural network community:

$$\mathrm{ARV}(M_3) = \frac{\sum_{(x^{u3}, y^{u3}) \in M_3} \left(y^{u3} - {}^1\Omega_{SL}(x^{u3}, \boldsymbol{w}) \right)^2}{\sum_{(x^{u3}, y^{u3}) \in M_3} \left(y^{u3} - \bar{y}^{u3} \right)^2} = \frac{1}{\sigma^2 \, U_3} \sum_{(x^{u3}, y^{u3}) \in M_3} \left(y^{u3} - {}^1\Omega_{SL}(x^{u3}, \boldsymbol{w}) \right)^2 \quad (32)$$

where (x^{u3}, y^{u3}) denotes the *u*3-th pattern of the testing set M_3, $\bar{y}^{u3}$ the average over the $U_3 = 248$ desired values. The averaging, that is the division by U_3 [the number of patterns in M_3], makes ARV independent of the size of M_3. The division by the estimated σ^2 of the data removes the dependence on the dynamic range of the data. This implies that if the estimated mean of the observed data would be taken as predictor, ARV would equal to one (Weigend, Rumelhart and Hubermann 1991). The statistic has a lower limit of zero indicating perfectly accurate predictions and an upper limit that is in practice one.[8]

The *second performance measure* is the standardised root mean square error (SRMSE) that is widely utilised by spatial analysts (see Fotheringham and Knudsen 1987):

$$\mathrm{SRMSE}(M_3) = \frac{1}{\bar{y}^{u3}} \left[\sum_{(x^{u3}, y^{u3}) \in M_3} \left(y^{u3} - {}^1\Omega_{SL}(x^{u3}, \boldsymbol{w}) \right)^2 / U_3 \right]^{\frac{1}{2}} \quad (33)$$

This statistic – closely related to the ARV-measure – has a lower limit of zero indicating perfectly accurate predictions and an upper limit that is variable[9] and depends on the distribution of the y^{u3}.

5.2 The Data

The testbed for the benchmark comparisons uses interregional telecommunication traffic data for Austria. From three Austrian data sources – a (32, 32)-interregional telecommunication flow matrix $\left(t_{ij}\right)$, a (32, 32)-distance matrix $\left(d_{ij}\right)$, and gross regional products for the 32 telecommunication regions – a set of 992 3-tupel $\left(s_j, d_{ij}, t_{ij}\right)$ with $i, j = 1,...,32$ $\left(i \neq j\right)$ was constructed. The first two components represent the input variables x_{2j-1} and x_{2j} of the *j*-th module of the network model ${}^1\Omega_{SL}\left(\boldsymbol{x}, \boldsymbol{w}\right)$, and the last component the target output. The bias term $\tilde{b}_{(i)}$ is clamped to the scalar $1/t_{i\bullet}$. s_j represents the potential draw of telecommunication in *j* and is measured in terms of the gross regional product, d_{ij} in terms of distances from *i* to *j*, while t_{ij} and $t_{i\bullet}$ represent telecommunication traffic flows. The input data were preprocessed to data scaled into [0.1, 0.9][10].

The telecommunication data used 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[11] [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 (see Figure 2). Due to measurement problems, intraregional traffic (i.e. *i* = *j*) is left out of consideration.

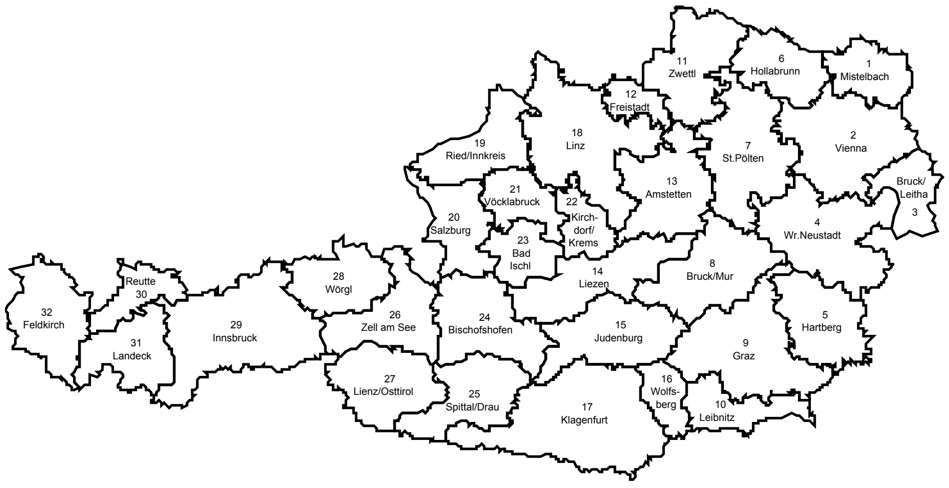


Figure 2: The Regional System for Modelling Interregional Telecommunciation Traffic in Austria

One of the simplest methods for estimating the prediction error is data splitting. This method simulates model validation with new data by partitioning the total data set of

992 samples into three subsets[12]: the training [in-sample] set $M_1 = \left\{ \left(x^{u1}, y^{u1} \right) \text{with } u1 = 1, ..., U_1 = 496 \text{ patterns} \right\}$, the internal validation set $M_2 = \left\{ \left(x^{u2}, y^{u2} \right) \text{with } u2 = 1, ..., U_2 = 248 \text{ patterns} \right\}$ and the testing [prediction, out-of-sample] set $M_3 = \left\{ \left(x^{u3}, y^{u3} \right) \text{with } u3 = 1, ..., U_3 = 248 \text{ patterns} \right\}$. M_1 is used only for parameter estimation, while M_2 for validation. The generalisation performance of the model is assessed on the testing set M_3.

Though the simplicity of this method is appealing, an obvious concern is the necessary reduction in the amount of training data. In deciding how to partition the data, a compromise has been made between creating a test set large enough to fully test the fitted model while still retaining a sufficient amount of training and internal validation data. If the test set is too small then the variance of the prediction error estimate will be high due to the small sample size. Though random splits are commonly used and appear to work reasonably well in the case of unconstrained spatial interaction, a more systematic splitting method had to be used in the case of constrained spatial interaction.

Table 1: Descriptive Statistics: The Training, Validation and Testing Sets

Variables	**Mean**	**Standard Deviation**	**Minimum**	**Maximum**
Whole Set M				
s_j	26,364,563	50,350,660	2,310,400	285,193,984
d_{ij}	229.4	124.6	30.0	630.0
t_{ij}	8.6	22.6	0.0	257.9
$t_{i\bullet}$	266.0	350.1	41.9	1830.1
Training Set M_1				
s_j	26,142,923	49,711,907	2,310,400	285,193,984
d_{ij}	234.1	129.6	35.0	630.0
t_{ij}	9.6	26.2	0.0	257.9
$t_{i\bullet}$	297.0	429.1	41.9	1830.1
Validation Set M_2				
S_j	26,517,946	50,891,071	2,310,400	285,193,984
d_{ij}	219.3	121.4	30.0	590.0
t_{ij}	7.1	16.6	0.0	166.8
$t_{i\bullet}$	220.9	221.4	45.6	759.8
Testing Set M_3				
S_j	26,654,459	51,069,577	2,310,400	285,193,984
D_{ij}	230.3	116.7	37.0	627.0
t_{ij}	8.0	19.7	0.0	195.2
$t_{i\bullet}$	249.0	262.5	55.3	895.7

Note: M consists of 992 patterns, M_1 of 496 patterns, M_2 of 248 patterns and M_3 of 248 patterns.

Some descriptive statistics characterising M_1, M_2 and M_3 are summarised in Table 1. As can be seen from this table there are no large differences between the training, validation and test sets. There are, nevertheless, differences, especially in t_{ij}, which will present some challenge to the estimation procedure used.

5.3 Model Estimation and the Overfitting Problem

Deciding on an appropriate number, *H*, of product units and on the value for the Alopex-parameters δ (the step size) is somewhat discretionary, involving the familiar trade-off between speed and accuracy. The approach adopted for this evaluation was stopped (cross-validation) training. The Alopex-parameters *T* and *N* were set to 1,000 and 10, respectively.

It is worth emphasising that the training process is sensitive to its starting point. Despite recent progress in finding the most appropriate parameter initialisation that would help Alopex to find near optimal solutions, the most widely adopted approach still uses random weight initialisation in order to reduce fluctuation in evaluation. Each experiment employed to determine *H* and δ was repeated 60 times, the model being initialised with a different set of random weights before each trial. 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 training process was stopped when $\kappa = 40,000$ consecutive iterations were unsuccessful.

Extensive computational experiments with different combinations of *H*- and δ-values have been performed on a DEC Alpha 375 Mhz. Table 2 summarises the results of the most important ones. Training Performance is measured in terms of ARV(M_1) and validation performance in terms of ARV(M_2). The performance values represent the mean of the 60 simulations, standard deviations are given in brackets. Since all simulations have similar computational complexity, iterations to converge to the minimal ARV(M_2)-value may be used as a measure of learning time. It is easy to see that the combination of $H = 16$ and $\delta = 0.0025$ provides an appropriate choice for our particular application.

Table 2: The Modular Product Unit Neural Network: Choice of H and δ [$T = 1{,}000$; $N = 10$]

Parameter	Iterations	ARV(M_1)	ARV(M_2)
$H = 4$			
$\delta = 0.0025$	40.782 ±19,886	0.2217 (0.0134)	0.1490 (0.0135)
$H = 8$			
$\delta = 0.0025$	43,905 ±18,854	0.2215 (0.0124)	0.1490 (0.0099)
$H = 12$			
$\delta = 0.0025$	38,905 ±17,896	0.2239 (0.0118)	0.1475 (0.0074)
$H = 16$			
$\delta = 0.0005$	52,702 ±33,311	0.2483 (0.0296)	0.1663 (0.0295)
$\delta = 0.0010$	59,321 ±49,647	0.2368 (0.0244)	0.1585 (0.0263)
$\delta = 0.0025$	45,754 ±21,284	0.2212 (0.0087)	0.1473 (0.0054)
$\delta = 0.0050$	22,948 ±15,360	0.2216 (0.0107)	0.1512 (0.0090)
$\delta = 0.0075$	17,427 ±12,918	0.2206 (0.0115)	0.1547 (0.0094)
$\delta = 0.0100$	13,545 ±11,753	0.2241 (0.0151)	0.1593 (0.0131)
$H = 24$			
$\delta = 0.0025$	40,580 ±20,047	0.2230 (0.0097)	0.1481 (0.0053)

ARV-performance values represent the mean (standard deviations in brackets) of 60 simulations differing in the initial parameter values randomly chosen from [-0.3; 0.3].

Iterations:	Number of iterations required to reach the parameter vector that provides the best ARV(M_2) performance.
ARV(M_1):	In-sample performance measured in terms of relative average variances.
ARV(M_2):	Out-of-sample performance measured in terms of relative average variances.

M consists of 992 patterns, M_1 of 496 patterns, M_2 of 248 patterns and M_3 of 248 patterns.

Figure 3 shows the learning curves of a typical run of the model (H = 16; measured by Alopex with T = 1,000; N = 10; δ =0.0025) of the model in terms of ARV(M_1), ARV(M_2) and ARV(M_3) respectively. The term learning curve is used to characterise the performance as a function of iterations of the Alopex procedure. Figure 3(a) plots the ARV-performance on the training set, Figure 3(b) the ARV-performance on the validation set and Figure 3(c) the ARV-performance on the testing set.

Typically, at the beginning of the training process, the validation error oscillates rapidly. Later, around 5,000 iterations the training process stabilises and the changes in the validation error become smaller. Instead of a clear increasing trend in the validation error that characterises overfitting, it starts around 12,500 to wander around some constant value. These undulations are caused by an increase of T in order to escape from shallow, local minima of the error surface (see Figure 3(d)). Later, the training process stabilises and the changes in validation error become smaller. According to our termination criterion, training is stopped after 18,841 iterations. At this stopping point, *P*, the model is used for testing (prediction).

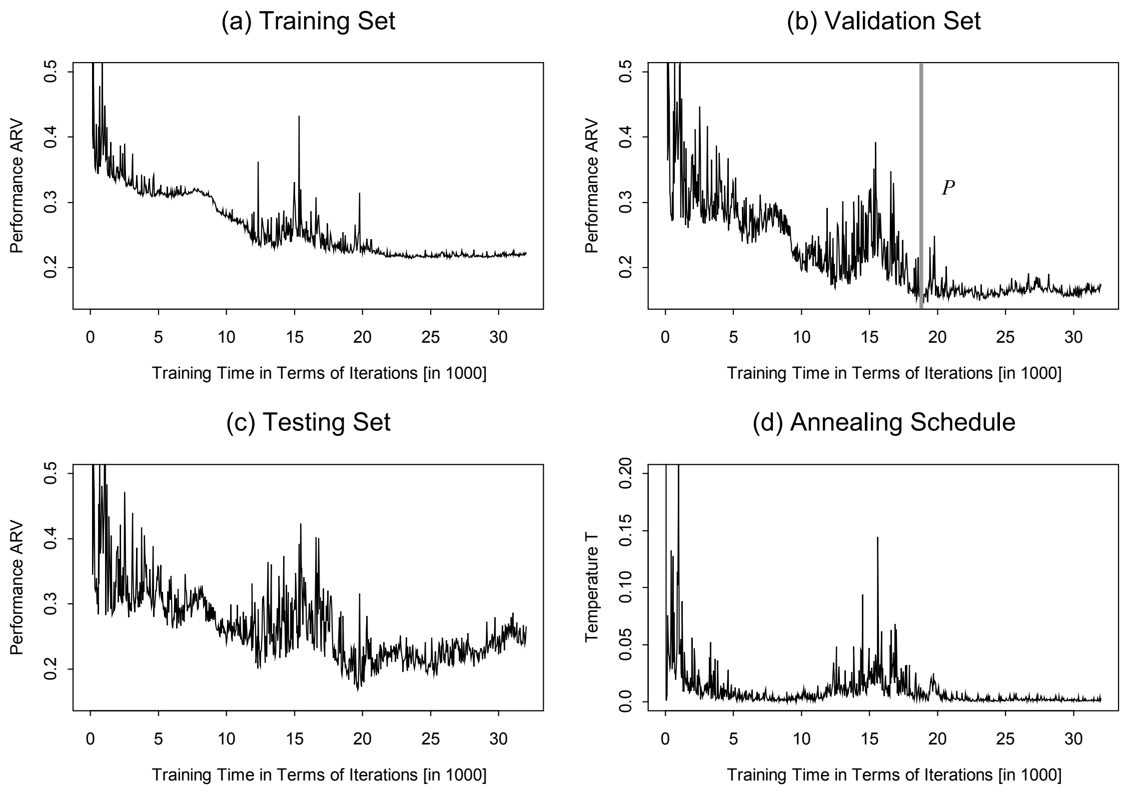


Figure 3: Training, Validation and Testing Set Curves as a Function of Training Time (the vertical line *P* indicates the stopping point): The Modular Product Unit Neural Network for Origin Constrained Spatial Interactions

5.4 The Benchmark Models

The first benchmark model is the *standard origin constrained gravity model*, a special case of the generic interaction model of the gravity type (see Equation (5))[13]:

$$\tau_{ij}^{grav} = b_{(i)}\, r_i\, s_j^{\alpha}\, d_{ij}^{-\beta} \tag{34}$$

with

$$b_{(i)} = \frac{t_{i\bullet}}{r_i \sum_{j \neq i} s_j^{\alpha} \, d_{ij}^{-\beta}} \tag{35}$$

$b_{(i)}$ is the origin specific balancing factor, α reflects the relationship of s_j with τ_{ij}^{grav} and β is the distance sensitivity parameter, $\beta > 0$. s_j is measured in terms of the gross regional product in *j*, d_{ij} in terms of distances from *i* to *j*, while $t_{i\bullet}$ in terms of erlang. We utilised the Alopex procedure for ML-estimation[14] with *T* = 1,000; *N* = 10, δ=0.0075 and the termination criterion $\kappa = 40,000$ iterations.

The *two-stage neural network modelling approach* serves as second benchmark model. In the *first stage* the classical unconstrained neural spatial interaction model, Ω_L (see Equation (7)) is used. The input data were preprocessed to logarithmically transformed data scaled to [0.1, 0.9]. The number of hidden summation units is 16. Least squares learning and the Alopex procedure $(T = 1,000; N = 10; \delta = 0.001, \kappa = 40,000)$ were used to determine the 81 model parameters. The parameters were randomly initialised in the range of [-0.3, 0.3]. In the *second stage* the following constraint mechanism is used to obtain origin constrained flows

$$\Omega_L^{constr}(\boldsymbol{x}, \boldsymbol{w})_{ij} = \frac{\Omega_L(\boldsymbol{x}, \boldsymbol{w})_{ij}}{\sum_j \Omega_L(\boldsymbol{x}, \boldsymbol{w})_{ij}} \, t_{i\bullet} \tag{36}$$

5.5 Performance Tests and Results

Table 3 summarises the simulation results for the modular product unit network model $^1\Omega_{SL}$ in comparison with the two-stage neural network approach Ω_L^{constr} and the origin constrained gravity model τ_{ij}^{grav}. Out-of-sample performance is measured in terms of ARV(M_3) and SRMSE(M_3). In addition, training performance values are displayed for matters of completeness. The figures represent averages taken over 60 simulations differing in the initial parameter values randomly chosen from [-0.3, 0.3]. Note that this random initialisation puts $^1\Omega_{SL}$ in contrast to Ω_L^{constr} at a slight disadvantage as its optimal range is [-2, +2].

If out-of-sample [generalisation] performance is more important than fast learning, then the modular product unit neural network exhibits clear superiority. As can be seen by comparing the ARV-values and the SRMSE-values the modular product unit neural network model ranks best, followed by the two-stage neural network approach and the gravity model. The average generalisation performance, measured in terms of ARV(M_3), is 0.2022 and, measured in terms of SRMSE(M_3), 1.1017, compared to 0.2251 and 1.1614 in the case of the two-stage approach, and 0.2262 and 0.1658 in the case of the gravity model[15]. This difference in performance between the modular product unit neural network model and the benchmark models is statistically significant[16]. If, however, the goal is to minimise execution time and a sacrifice in generalisation accuracy is acceptable, then the standard origin constrained gravity model is the method of choice. The gravity model outperforms the neural network models in terms of execution time, the modular product unit network model by a factor of 10 and the 2-stage neural network model by a factor of 10^2. 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 ($\kappa = 40{,}000$) to stop the training process.

Table 3: Benchmark comparisons of the Modular Product Unit Neural Network ${}^1\Omega_{SL}$ with the Two-Stage Neural Network Approach Ω_L^{constr} and the Gravity Model τ_{ij}^{grav} for Modelling Origin Constrained Spatial Interactions

	Modular Product Unit Neural Network	*Two-Stage Neural Network Approach*	*Origin Constrained Gravity Model*
In-Sample (Training) Performance			
ARV	0.2212 (0.0087)	0.2682 (0.0222)	0.2121 (0.0017)
SRMSE	1.2858 (0.0254)	1.4150 (0.0578)	1.2594 (0.0049)
Out-of-Sample (Testing) Performance			
ARV	0.2022 (0.0150)	0.2251 (0.0255)	0.2262 (0.0027)
SRMSE	1.1017 (0.0407)	1.1614 (0.0670)	1.1658 (0.0069)

Note: Figures represent averages taken over 60 simulations differing in the initial parameter values randomly chosen from [-0.3, 0.3] (standard deviations in brackets); the testing set consists of 248 patterns and the training set of 496 patterns.

Residual Analysis

One means of further investigating the predictive power of the *modular product unit neural network model* ${}^{1}\Omega_{SL}(\boldsymbol{x},\boldsymbol{w})$ in comparison to the two benchmark models $\Omega_{L}^{constr.}(\boldsymbol{x},\boldsymbol{w})$ and τ_{ij}^{grav} is the use of residual analysis. Figure 4 displays these in terms of

- the absolute residuals of the individual flows $\left(t_{ij} - {}^{1}\Omega_{SL}(\boldsymbol{x},\boldsymbol{w})_{j}\right)$ compared to $\left(t_{ij} - \Omega_{L}^{constr}(\boldsymbol{x},\boldsymbol{w})_{ij}\right)$ and $\left(t_{ij} - \tau_{ij}^{grav}\right)$, and
- the relative residuals of the individual flows $\left(t_{ij} - {}^{1}\Omega_{SL}(\boldsymbol{x},\boldsymbol{w})_{j}\right)/t_{ij}$ compared to $\left(t_{ij} - \Omega_{L}^{constr}(\boldsymbol{x},\boldsymbol{w})_{ij}\right)/t_{ij}$ and $\left(t_{ij} - \tau_{ij}^{grav}\right)/t_{ij}$,

where both absolute and relative residuals are ordered by the size of the observed flows, t_{ij}. The main conclusion from this analysis can be summarised as follows:

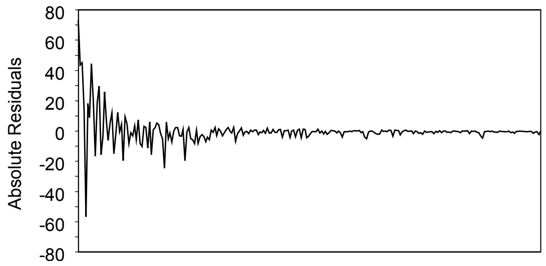
First, all three models show a tendency to underpredict larger flows. The neural network models underpredict 17 out of 25 flows in the largest decile, compared to 16 gravity model underpredictions. The benchmark models tend to give results with a little larger absolute deviation.

Second, all three models show a tendency to overpredict smaller flows. This is evidenced in the smallest decile by 23 overpredictions in the smallest decile, obtained by the modular product unit network, compared to 24 overpredictions of the benchmark models.

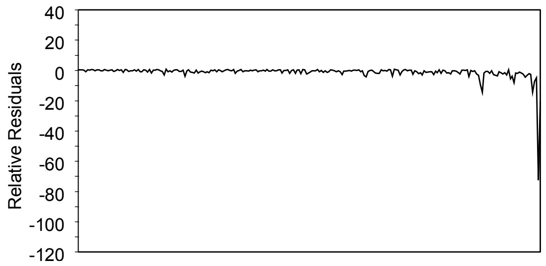
Third, the modular product unit neural network model and the benchmark models show a relatively similar pattern of residuals. Despite this similarity the modular model tends to produce slightly more accurate predictions in the case of larger flows, but slightly less accurate ones in the case of smaller flows.

In summary, the analysis unequivocally shows that the modular product unit network outperforms the benchmark models in terms of both the ARV(M_3) and SRMSE(M_3) prediction performance, as well as the prediction accuracy, but the latter to a lesser degree than previously expected. One reason for this might be that the method of stopped training did not indicate unequivocally the stopping point, and thus, the determination of H and δ . This is an issue for further research.

(i) Residuals of the Modular Product Unit Neural Network Model ${}^{1}\Omega_{SL}\left(\boldsymbol{x},\boldsymbol{w}\right)$

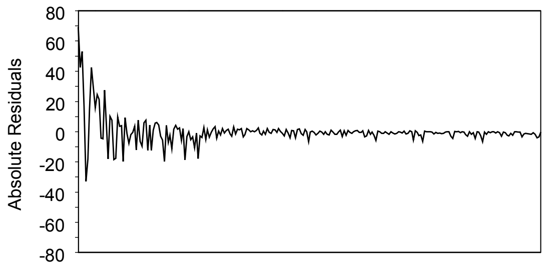


(i1) Absolute Residuals $\left(t_{ij} - {}^{1}\Omega_{SL}\left(\boldsymbol{x},\boldsymbol{w}\right)\right)$; ordered by the size of t_{ij}

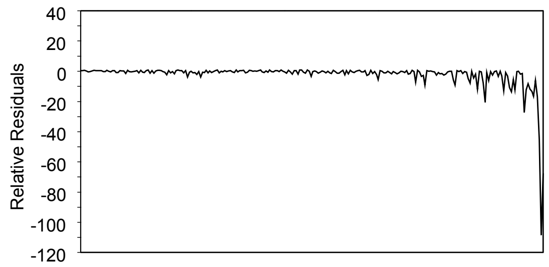


(i2) Relative Residuals $\left(t_{ij} - {}^{1}\Omega_{SL}\left(\boldsymbol{x},\boldsymbol{w}\right)\right)/\ t_{ij}$; ordered by the size of t_{ij}

(ii) Residuals of the Two-Stage Neural Network Model Approach $\Omega_{L}^{constr}\left(\boldsymbol{x},\boldsymbol{w}\right)$

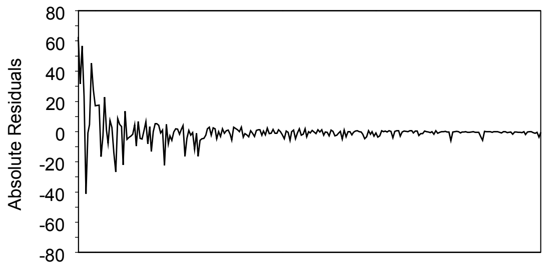


(ii1) Absolute Residuals $\left(t_{ij} - \Omega_{L}^{constr}\left(\boldsymbol{x},\boldsymbol{w}\right)\right)$; ordered by the size of t_{ij}

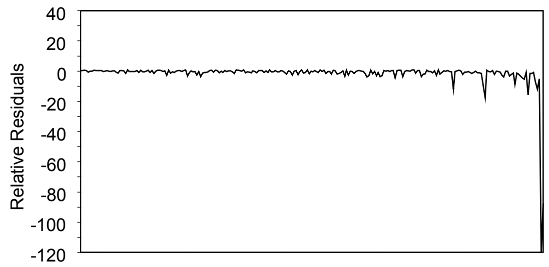


(ii2) Relative Residuals $\left(t_{ij} - \Omega_{L}^{constr}\left(\boldsymbol{x},\boldsymbol{w}\right)\right)/\ t_{ij}$; ordered by the size of t_{ij}

(iii) Residuals of the Origin Constrained Gravity Model τ_{ij}^{grav}



(iii1) Absolute Residuals $\left(t_{ij} - \tau_{ij}^{grav}\left(\boldsymbol{x},\boldsymbol{w}\right)\right)$; ordered by the size of t_{ij}



(iii2) Relative Residuals $\left(t_{ij} - \tau_{ij}^{grav}\left(\boldsymbol{x},\boldsymbol{w}\right)\right)/\ t_{ij}$; ordered by the size of t_{ij}

Figure 4: Residuals of the Modular Product Unit Neural Network ${}^{1}\Omega_{SL}$, the Two-Stage Neural Network Approach Ω_{L}^{constr} and the Origin Constrained Gravity Model Predictions τ_{ij}^{grav}

6 Conclusions and Outlook

In this paper, a neural network methodology for modelling singly constrained spatial interactions has been presented. The proposed function approximator is based on a modular network design with functionally independent product unit network modules where modularity refers to a decomposition on the computational level. Each module is a feedforward network with two inputs and a hidden layer of 16 product units and terminates with a single summation unit. The collective outputs of these modules constitute the input to the layer of output units that perform the flow prediction by applying some sort of the Bradley-Terry-Luce model. The paper also demonstrates a simple way to implement the conservation rule from the viewpoint of origins [destinations] that avoids the need to modify the parameter estimation procedure to integrate the constraints on the predicted flows.

The Alopex procedure provides an optimisation scheme that allows to produce LS-estimates of the model parameters. The dynamic interaction among a stochastic exploration process, the correlation based guidance to move towards regions of higher-quality solutions in the parameter space and the convergence-inducing process is responsible for the attractivity of the global search process of the procedure.

The attraction of this novel model approach depends not only on the awareness of what it can offer, but also on empirical illustrations of what can be gained in terms of out-of-sample (testing) approximation accuracy. Benchmark comparisons against the standard origin constrained gravity model and the two-stage neural network approach, suggested by Openshaw (1998) and implemented by Mozolin, Thill and Usery (2000), illustrate the superiority of the product unit neural network model, measured in terms of both the ARV- and the SRMSE-performance over 60 simulations.

The importance of avoiding overfitting cannot be overemphasised if a good predictive model is desired, and consequently, we believe that testing further techniques to control the model complexity without comprising network generalisation or learning accuracy is merited. Our research may, furthermore, be extended in two other directions in future, *first*, by modifying the approach to model the issue of origin and destination [that is, doubly] constrained interactions over geographic space and *second*, by analysing the prediction quality of the function approximator in other spatial interaction contexts.

Acknowledgement: The authors gratefully acknowledge the grant no. P12681-INF provided by the Austrian Fonds zur Förderung der wissenschaftlichen Forschung (FWF).

Endnotes

1 Alternative models based on additive adjustment formulations were introduced by Tobler (1983).

2 Sigmoid transfer functions 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 non-linear in their response as they approach their points of maximum curvature on either side of the midpoint.

3 In addition one property is being lost in comparison to summation units, namely that product units are vulnerable to translation and rotation of the input space, in the sense that a learnable problem may no longer be learnable after translation. Rotational and translational vulnerability of single product units is in part compensated for, if a number of them are being used in parallel.

4 In the production constrained case the conservation principle is enforced from the viewpoint of origins of spatial interactions, and in the attraction constrained case from the viewpoint of destinations only (see Ledent 1985).

5 Bishop (1995) has shown that the least square error function can be derived from the principle of maximum likelihood on the assumption of Gaussian distributed target data. Of course, the use of the error function does not require the target data to have a Gaussian distribution.

6 Alopex is an acronym for **al**gorithm f**or** **p**attern **ex**traction.

7 For the first two iterations, the weights are chosen randomly.

8 ARV-values greater than one arise whenever the average error is greater than the mean.

9 SRMSE-values greater the one arise whenever the average error is greater than the mean.

10 except for the standard origin constrained model

11 Flows are discrete counts, but note that flows are measured here in terms of erlang, a metric variable.

12 This static approach for evaluating the performance of a neural model has been used for many years in the connectionist community in general and in neural spatial interaction modelling in particular (see Fischer and Gopal 1994). Recent experience has found this approach to be rather sensitive to the specific splitting of the data. Thus, usual tests of forecast reliability may appear over-optimistic in general. Fischer and Reismann (2000) suggest an approach that combines the purity of splitting the data into three disjoint sets with the power of bootstrapping to get a better statistical picture of forecast variability, including the ability to estimate the effect of the randomness of the splits of the data.

13 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).

14 Assuming that each t_{ij} has a Poisson distribution and the t_{ij}'s are independent leads to the following objective function: $\sum_{i,j}\left(t_{ij} \ln \tau_{ij}^{grav} - \tau_{ij}^{grav}\right)$ that has to be maximized. This distributional assumption often considered to be realistic is open to question in our context in view of the fact that the measurements of flows are not discrete counts.

15 ARV-Out-of-sample variance of ${}^{1}\Omega_{SL}$ varies between 0.1725 and 0.2361, that of Ω_{SL}^{constr} between 0.1852 and 0.2502 and that of τ_{ij}^{grav} between 0.2225 and 0.2327.

16 assessed by means of the Mann-Whitney-U-Test (pairwise comparison of two independent samples). The differences are statistically significant at the 1 % significance level. (U=280, Sig. 0.0 [compared to the standard origin constrained gravity model] and U=144, Sig. 0.009 [compared to the two-stage approach]).

References

Alonso, W. (1978): A theory of movement. In Hansen, N.H. (ed.): *Human settlement systems*, pp. 197-212. Ballinger, Cambridge [MA]

Aufhauser, E. and Fischer, M.M. (1985): Log-linear modelling and spatial analysis, *Environment and Planning* A 17, 931-951

Bergkvist, E. (2000): Forecasting interregional freight flows by gravity models, *Jahrbuch für Regionalwissenschaft* 20, 133-148

Bia, A. (2000): A study of possible improvements to the Alopex training algorithm, Proceedings of the VIth Brazilian Symposium on Neural Networks, pp. 125-130. IEEE Computer Society Press

Bishop, C.M. (1995): *Neural networks for pattern recognition*. Clarendon Press, Oxford

Black, W.R. (1995): Spatial interaction modelling using artificial neural networks, *Journal of Transport Geography* 3(3), 159-166

Cover, T.M. (1965): Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, *IEEE Transactions on Electronic Computers* 14, 326-334

Durbin, R. and Rumelhart, D.E. (1989): Product units: A computationally powerful and biologically plausible extension to backpropagation, *Neural Computation* 1, 133-142

Fischer, M.M. (2000): Methodological challenges in neural spatial interaction modelling: The issue of model selection. In Reggiani, A. (ed.): *Spatial economic science: New frontiers in theory and methodology*, pp. 89-101. Springer, Berlin, Heidelberg and New York

Fischer, M.M. and Getis, A. (1999): Introduction. New advances in spatial interaction theory, *Papers in Regional Science* 78, 117-118

Fischer, M.M. and Gopal, S. (1994): Artificial neural networks: A new approach to modelling interregional telecommunication flows, *Journal of Regional Science* 34(4), 503-527

Fischer, M.M. and Leung, Y. (1998): A genetic-algorithm based evolutionary computational neural network for modelling spatial interaction data, *The Annals of Regional Science* 32(3), 437-458

Fischer, M.M. and Reismann, M. (2000): Evaluating neural spatial interaction modelling by bootstrapping. Paper presented at the 6th World Congress of the Regional Science Association International, Lugano, Switzerland, May 16-20, 2000

Fischer, M.M., Hlavackova-Schindler, K. and Reismann, M. (1999): A global search procedure for parameter estimation in neural spatial interaction modelling, *Papers in Regional Science* 78, 119-134

Flowerdew, R. and Aitkin, M. (1982): A method of fitting the gravity model based on the Poisson distribution, *Journal of Regional Science* 22, 191-202

Fotheringham, A.S. (1983): A new set of spatial interaction models: The theory of competing destinations, *Environment and Planning* A 22, 527-549

Fotheringham, A.S. and O'Kelly, M.E. (1989): *Spatial interaction models: Formulations and applications*. Kluwer, Dordrecht, Boston and London

Giles, C. and Maxwell, T. (1987): Learning, invariance, and generalization in high-order neural networks, *Applied Optics* 26(23), 4972-4978

Harth, E. and Pandya, A.S. (1988): Dynamics of ALOPEX process: Application to optimization problems. In Ricciardi, L.M. (ed.): *Biomathematics and related computational problems*, pp. 459-471. Kluwer, Dordrecht, Boston and London

Hassoun, M.H. (1995): *Fundamentals of neural networks*. MIT Press, Cambridge [MA] and London [England]

Hecht-Nielsen, R. (1990): *Neurocomputing*. Addison-Wesley, Reading [MA]

Hornik, K., Stinchcombe, M. and White, H. (1989): Multilayer feedforward networks are universal approximators, *Neural Networks* 2, 359-366

Ismail, A. and Engelbrecht, A.P. (1999): Training product units in feedforward neural networks using particle swarm optimization, In Bajic, V.B. and Sha, D. (eds.): *Proceedings of the International Conference on Artificial Intelligence*, Durban, pp. 36-40. South Africa

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983): Optimization by simulated annealing, *Science* 220, 671-680

Knudsen, D.C. and Fotheringham, A.S. (1986): Matrix comparison, goodness-of-fit, and spatial interaction modeling, *International Regional Science Review* 10(2), 127-147

Ledent, J. (1985): The doubly constrained model of spatial interaction: A more general formulation, *Environment and Planning A* 17, 253-262

Mozolin, M., Thill, J.-C. and Usery, E.L. (2000): Trip distribution forecasting with multilayer perceptron neural networks: A critical evaluation, *Transportation Research B* 34, 53-73

Openshaw, S. (1993): Modelling spatial interaction using a neural net. In Fischer, M.M. and Nijkamp, P. (eds.): *Geographic information systems, spatial modeling, and policy evaluation*, pp. 147-164. Springer, Berlin, Heidelberg and New York

Openshaw, S. (1998): Neural network, genetic, and fuzzy logic models of spatial interaction, *Environment and Planning A* 30, 1857-1872

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992): *Numerical recipes in C: The art of scientific computing.* Cambridge University Press, Cambridge

Reggiani, A. and Tritapepe, T. (2000): Neural networks and logit models applied to commuters' mobility in the metropolitan area of Milan. In Himanen, V., Nijkamp, P. and Reggiani, A. (eds.): *Neural networks in transport applications*, pp. 111-129. Ashgate, Aldershot

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986): Learning internal representations by error propagation. In Rumelhart, D.E. and McClelland, J.L. (eds.): *Parallel Distributed Processing*, pp. 318-362. MIT Press, Cambridge [MA]

Sen, A. and Smith T.E. (1995): *Gravity models of spatial interaction behavior*. Springer, Berlin, Heidelberg and New York

Senior, M.L. (1979): From gravity modelling to entropy maximizing: A pedagogic guide, *Progress in Human Geography* 3(2), 175-210

Tiefelsdorf, M. and Boots, B. (1995): The specification of constrained interaction models using the SPSS loglinear procedure, *Geographical Systems* 3, 21-38

Thill, J.-C. and Mozolin, M. (2000): Feedforward neural networks for spatial interaction: Are they trustworthy forecasting tools? In Reggiani, A. (ed.): *Spatial economic science: New frontiers in theory and methodology*, pp. 355-381. Springer, Berlin, Heidelberg and New York

Tobler, W. (1983): An alternative formulation for spatial interaction modelling, *Environment and Planning A* 15, 693-703

Turton, I., Openshaw, S. and Diplock, G. (1997): A genetic programming approach to building new spatial models relevant to GIS. In Kemp, Z. (ed.): *Innovations in GIS 4*, pp. 89-104. Taylor & Francis, London

Tzanakou, E., Michalak, R. and Harth, E. (1973): The Alopex process: Visual receptive fields by response feedback, *Biological Cybernetics* 35, 161-174

Unnikrishnan, K.P. and Venugopal, K.P. (1994): Alopex: A correlation-based learning algorithm for feedforward and recurrent neural networks, *Neural Computation* 6(3), 469-490

Verikas, A. and Gelzinis, A. (2000): Training neural networks by stochastic optimisation, *Neurocomputing* 30, 153-172

Weigend, A.S., Rumelhart, D.E. and Huberman, B.A. (1991): Back-propagation, weight-elimination and time series prediction. In Touretzki, D.S., Elman, J.L, Sejnowski, T.J. and Hinton, G.E. (eds.): Connectionist models: Proceedings of the 1990 Summer School, pp. 105-116. Morgan Kaufmann Publishers, San Mateo [CA]

White, H. (1980): Using least squares to approximate unknown regression functions, *International Economic Review* 21, 149-170

White, H. (1989): Some asymptotic results for learning in single hidden layer feedforward networks, *Journal of the American Statistical Association* 84, 1008-1013

White, H. (1990): Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings, *Neural Networks* 3, 535-550

Wilson, A.G. (1967): A statistical theory of spatial distribution models*, Transportation Research* 1, 253-269

Wilson, A.G. (1970): *Entropy in urban and regional modelling*. Pion, London