R programs for splitting abridged fertility data into a fine grid of ages using the neural network method

Vasily P. Gorlishchev  l  gpa15@yandex.ru
Pavel Grigoriev  l  grigoriev@demogr.mpg.de
Anatoli I. Michalski  l  mpocctok@yandex.ru

For additional material see www.demogr.mpg.de/tr/
R programs for splitting abridged fertility data into a fine grid of ages using the neural network method

Vasily P. Gorlishchev (gpa15@yandex.ru)
Pavel Grigoriev (Grigoriev@demogr.mpg.de)
Anatoli I. Michalski (mpoctok@yandex.ru)
Abstract. The need to split aggregated fertility data into a fine grid of ages is a challenge that is often encountered by demographers. Several methods for addressing this problem have been developed. In this technical report, we present an application of a new approach to splitting abridged fertility data, the neural network (NN) model. Although neural networks have been widely used in various fields, they have seldom been applied in demography. The algorithm presented here is very flexible and simple to use, but it requires substantial computational resources. The NN method allows us to split abridged fertility rates of any kind using a pre-learned model generated with high-quality data drawn from the Human Fertility Database. The results of testing show that in most cases, the NN model returns estimates that correspond very closely to the original values. However, the model also tends to return erroneous estimates for fertility patterns that are ‘unfamiliar’ to the pre-learned model.
1. Background

The need to split aggregated fertility data into a fine grid of ages sometimes arises. To handle this task, several disaggregation methods have been developed and tested (McNeil et al. 1975; Smith, Hyndman, Wood, 2004; Liu, et al. 2011; Schmertmann 2012; Jasilioniene et al. 2012, Grigoriev and Jdanov, 2015). Here, we present an application of the neural network (NN) method, which represents a new approach to splitting abridged fertility data. While neural networks have been widely used in various disciplines, they have seldom been applied in demography. To our knowledge, this is the first attempt to apply the neural network method to the problem of demographic data disaggregation. We intend to provide a fine split of aggregated fertility rates \( f_x \) into fertility rates by single years of age \( f_x \). We have established three criteria that must be met when applying the NN method in this context:

1) *Fit* – the predicted values should be as close as possible to the observed values;
2) *Shape* - the estimated fertility curves should be plausible and smooth; and
3) *Non-negativity* – the method should not generate negative values.

The NN method can be used to split any kind of input data with a structure that corresponds to a pre-learned model. If needed, a pre-learned model with the desired format can be easily generated by the user. Here, we describe a neural net that splits nine five-year age groups: 10-14, … , 50-54 into single ages from 12 to 54. The NN estimation algorithm contains a smoothing spline and negative value extinction procedures.

2. Description of the algorithm

Below, we provide a very brief and general description of the NN algorithm. A more detailed description of the process can be found elsewhere (Riedmiller and Braun, 1994). Our base model is a pre-learned neural network with a resilient back-propagation algorithm. The output of the network consists of smoothed estimates adjusted for negative values.

2.1. The neural network and the resilient back-propagation algorithm

The basic component of a neural net is a formal neuron, shown in Figure 1.

![Figure 1. Formal neuron](Source: Adopted from Zaencev (1999))
A formal neuron contains a weighted sum and a non-linear element (activation function). The operation of a formal neuron can be described by the following formulas:

\[ NET = \sum_i w_i x_i \]  
\[ OUT = F(NET - \theta), \]

where

- \( x_i \) – the input signals, whereby the vector of all of the input values stands for \( x \);
- \( w_i \) – the weight coefficients;
- \( NET \) – the weighted sum of all input values, whereby the \( NET \) value is transferred to the non-linear elements;
- \( \Theta \) – the threshold level of the neuron;
- \( F \) – the non-linear function, which stands for the activation function – the sigmoidal (logistic) function that accepts and recalculate \( (NET - \theta) \)

\[ OUT = \frac{1}{1 + e^{-NET}} \] - Output with a sigmoidal activation function, whereby \( \theta \) equals zero for this function.

The combination of the formal neurons builds a neural net, shown in Figure 2.

Figure 2. Formal structure of the neural network

Source: Adopted from Zaencev (1999)
Formally, the neural network is just a sequential evaluation of linear and non-linear function combinations:

\[ f(x) = F\left( \sum_{i_k} w_{i_k} x \right) \sum_{i_2} w_{i_2} F\left( \sum_{i_1} w_{i_1} - \theta_{j_1} \right) - \theta_{j_2} \ldots - \theta_{j_k} \]  

The sequential evaluation provides a close approximation of the multidimensional function.

The resilient back-propagation algorithm (2) is a method used to tune the weights \( w_i \) in a way that allows the function \( f(x) \) to approximate data. After all of the train data have been entered into the neural net and all of the derivative errors of the formal neurons have been counted, the updated values \( \Delta^{(t)}_{ij} \) for the neural net coefficients are calculated by the following system of equations:

\[
\Delta^{(t)}_{ij} = \begin{cases} 
\eta^* \Delta^{(t-1)}_{ij} & \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\
\eta^- \Delta^{(t-1)}_{ij} & \text{if } \frac{\partial E^{(t-1)}}{\partial w_{ij}} \cdot \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\
\Delta^{(t-1)}_{ij} & \text{else}
\end{cases}
\]  

where

\( \eta \) is a step parameter. The updated values of the weights are based on the information on a local error derivative

\[
\Delta w^{(t)}_{ij} = \begin{cases} 
-\Delta^{(t)}_{ij} & \text{if } \frac{\partial E^{(t)}}{\partial w_{ij}} > 0 \\
+\Delta^{(t)}_{ij} & \text{if } \frac{\partial E^{(t)}}{\partial w_{ij}} < 0 \\
0 & \text{else}
\end{cases}
\]  

The weights are updated by the following rule:
\[ w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)} \quad (6) \]

The cycle continues until convergence is reached.

### 2.2. Spline and elimination of negative values procedures

The smoothing spline estimates value \( \hat{f} \) for the neural net estimates \( f \), and is by definition a minimizer for the function:

\[
\sum_{i=1}^{n} (Y_i - \hat{f}(x_i))^2 + \lambda \int_{x_i}^{x_n} \hat{f}''(x)^2 dx \rightarrow \min_f \quad (7)
\]

where \( \lambda \geq 0 \) is a smoothing parameter that trades the quality of the approximation and the smoothness of function (3).

The vector of the smoothed spline parameters is calculated by the following formula:

\[
\hat{m} = (I + \lambda A)^{-1} Y \quad (8)
\]

with the matrix

\[
A = \int f_i'(x) f_j''(x) dx \quad (9)
\]

The negative values are eliminated by the following simple substitution:

\[
\hat{f}(x_i) = \begin{cases} \hat{f}(x_i), & \text{if } \hat{f}(x_i) \geq 0 \\ 0, & \text{if } \hat{f}(x_i) < 0 \end{cases} \quad (10)
\]

### 2.3. Disaggregation of the neural network structure

We constructed a neural net with nine input neurons (age groups: 10-14, 15-19, …, 50-54), two hidden layers with 27 and 36 neurons, and 43 output neurons (single age groups: 12, 13, 14, …, 54). The structure of this neural net appears in Figure 3.
3. Requirements

The scripts were tested using R version 3.1. The library ‘neuralnet’ is required for the calculation of the neural network, and the library ‘bigsplines’ is required for the smoothing procedure. Both packages can be installed from the standard R repository (CRAN).

4. Usage

The .zip file included in the technical report contains R scripts, and the data files used in the example files HFD_BirthsExp1x1.csv and HFC_ASFR_TOT 9 age groups.txt were drawn from the HFD and the HFC, respectively. All of the other data files were derived on the basis of HFD_BirthsExp1x1.csv, and can be reproduced using the provided R scripts ‘1 Generating abridged data.R’ and ‘2 Preparing data for learning.R’. The file NN Functions.R contains four functions that needed to be loaded into the R environment using the source command. A description of these functions is provided below.

4.1. R functions

After sourcing ‘NN Functions.R’ (command ‘source’), the following R functions are ready for use: NN.learn(), NN.split(), NN.plot(), and NN.errors().
4.1.1. **NN.learn()**

This function builds a neural network on the basis of real observations (HFD data). The model establishes the associations between the disaggregated and the aggregated fertility rates generated from the same data source.

Usage: `NN.learn(input = train.input, output = train.output)`

The arguments are as follows:

- `train.input` – the data frame (or the vector) containing the abridged age-specific fertility rates;
- `train.output` – the data frame (or the vector) containing the disaggregated age-specific fertility rates (Explanatory note: As both `train.input` and `train.output` are generated from the original birth counts and the population exposures by single years of age, they have the same attributes as `CountryYear`. The format of these data is described in section 4.2.); and
- `layers` – the vector of the hidden layers and the units in them, default: `layers=c(10,50,50,60)`.

The accuracy of the neural network prediction is very sensitive to this parameter. How the `layers` are designed depends on both the quantity and the quality of the raw data. A larger amount of data generally implies a higher degree of heterogeneity. If there are 300+ examples (empirical fertility schedules), we recommend that the sizes of the layers be kept as small as possible: `c(10,50,50,60)`. If the data are homogeneous and the number of schedules is small (about 30), the sizes of the layers can be larger: `c(10,50,50,50,50,50,50,50,60)` (Explanatory note: `layers = c(20,20,20)` – three-layer network with 20 units in each layer).

Example: `NN.learn(input = train.input, output = train.output)`

`NN.learn` returns a large R object that has to be passed into the `NN.split` function.

4.1.2. **NN.split ()**

This function splits abridged data (rates) into a fine grid of ages.

Usage: `NN.split (input = x, model = model1)`

The arguments are as follows:

- `x` – the data frame (or the vector) containing the abridged age-specific fertility rates (Explanatory note: The format of the input data is described in section 4.2.); and
- `model1` – the pre-learned model.

Example: `NN.split (test.input[c('AUT1964','CAN1980'), ], model = model1)`

`NN.split` returns the data frame containing the disaggregated age-specific fertility rates in the following format:

- columns – ages by single years of age (12, 13, 14, …..52, 53, 54)
- rows – `CountryYear` (e.g., AUT1964)
4.1.3. **NN.plot**() 

This function visualizes the results of splitting using the NN method. 

Usage: `NN.plot(input, input.ages, output.ages, prediction, output=FALSE, additional=FALSE, file.name='NN.plot output.pdf')`  

The arguments are as follows:

`input` – the data frame with the abridged fertility data;  

`input.ages` – the mid-point values of the abridged age intervals;
output.ages – the desired output ages;

prediction – the predicted values returned by NN.split;

output (optional) – the data frame containing the original data by single years of age
(Explanatory note: Normally, these data are not available, but this option might still be used to
evaluate the performance of the NN model), default: output=FALSE;

additional (optional) – the data frame containing the predicted values obtained from an
alternative splitting method (e.g., HFD method, calibrated spline method, etc.), default:
additional=FALSE; and

file.name (optional) – the name of the output .pdf file, default: file.name='NN.plot output.pdf'.

Example:

```r
> NN.plot(input=test.input, input.ages=seq(12.5,52.5,5),
prediction=prediction1, output=test.output, output.ages=(12:54),
file.name = 'NN split 9 age groups Example 2.pdf')
```

NN.plot returns a .pdf file with plots showing the results of splitting.

4.1.4. NN.errors()

This function calculates the root mean squared error (RSME) between the observed and the
predicted values by the following formula:

\[
RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}},
\]

where \( y_i \) and \( \hat{y}_i \) - observed and predicted values, respectively. \( n \) - number of observations.

NN.errors is used to evaluate the performance of the NN method (for testing purposes).

Usage: `NN.errors<- function(output=test.output, prediction=prediction1)`

The arguments are as follows:

- `prediction` – the data frame (or the vector) containing the predicted values of the ASFRs; and
- `output` – the data frame (or the vector) containing the observed (real) values of the ASFRs.

NN.errors returns a list containing `rmse.CCY` and `rmse.age` objects summarizing the RSME statistic by CountryYear or age, respectively.
Example:

```r
> Errors <- NN.errors(output = test.output[c('AUT1965', 'CAN1980'), ], prediction=prediction1[c('AUT1965', 'CAN1980'), ])
```

### 4.2 Format of the input data

The input should be either a data frame or a vector containing the age-specific fertility rates. A fragment of the aggregated input data to be disaggregated appears below:

```r
> head(test.input)

<table>
<thead>
<tr>
<th>AUT1952</th>
<th>ASFR.10</th>
<th>ASFR.15</th>
<th>ASFR.20</th>
<th>ASFR.25</th>
<th>ASFR.30</th>
<th>ASFR.35</th>
<th>ASFR.40</th>
<th>ASFR.45</th>
<th>ASFR.50</th>
<th>ASFR.55</th>
<th>ASFR.60</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00010965274788</td>
<td>0.0344274289</td>
<td>0.1184082739</td>
<td>0.1157629083</td>
<td>0.08457438561</td>
<td>0.0439332252</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.00006910694635</td>
<td>0.0321290082</td>
<td>0.1275242264</td>
<td>0.1219545035</td>
<td>0.0840935185</td>
<td>0.0453386951</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.00002832197614</td>
<td>0.0310630561</td>
<td>0.1347707133</td>
<td>0.1304663075</td>
<td>0.0902490081</td>
<td>0.0306541803</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.00001895571782</td>
<td>0.0236305973</td>
<td>0.1479409296</td>
<td>0.1416949233</td>
<td>0.0969249095</td>
<td>0.0155519725</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.00001476225286</td>
<td>0.0566418584</td>
<td>0.1616259669</td>
<td>0.1544956969</td>
<td>0.0995025964</td>
<td>0.0501716501</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.00001359713301</td>
<td>0.0111936069</td>
<td>0.0359679027</td>
<td>0.1121375917</td>
<td>0.0501716501</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

The data used for learning have the same format:

```r
> head(train.input)

<table>
<thead>
<tr>
<th>AUT1951</th>
<th>ASFR.10</th>
<th>ASFR.15</th>
<th>ASFR.20</th>
<th>ASFR.25</th>
<th>ASFR.30</th>
<th>ASFR.35</th>
<th>ASFR.40</th>
<th>ASFR.45</th>
<th>ASFR.50</th>
<th>ASFR.55</th>
<th>ASFR.60</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0001070880663</td>
<td>0.0340936701</td>
<td>0.1147491184</td>
<td>0.1132871454</td>
<td>0.08244793567</td>
<td>0.0451412790</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0001132746726</td>
<td>0.0336167959</td>
<td>0.1233653329</td>
<td>0.1185901024</td>
<td>0.0832775909</td>
<td>0.0427455135</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01601244271</td>
<td>0.0130270937</td>
<td>0.0005017174</td>
<td>0.0150981966</td>
<td>0.0133313235</td>
<td>0.0007469118</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Note: The row and the column names in `train.input` and `train.output` must be identical. The same requirement applies to `test.input` and `test.output`.

5. Examples

For the sake of simplicity and the convenience of the user, the procedure for splitting aggregated fertility data is divided into three modules:

1) generating the abridged data,
2) preparing the data for learning, and
3) modeling and predicting.

The corresponding R scripts that perform these functions are included in the .zip file. Module (1) allows users to generate abridged fertility data of any kind, which can then be used for learning. By default, the age-specific fertility rates for the nine age groups used in this report are calculated on the basis of the birth counts and the population exposures by single years of age (see file ‘HFD_BirthsExp1x1.csv’). Module 2 contains the utilities for reshaping the input data in the internal format. Finally, using Module 3, the users can build their own neural network model and split the aggregated data into single ages.

As the scripts with examples are accompanied by detailed comments, even inexperienced R users should find the NN method very simple to use.
References


