Paris School of Economics

Michał Miktus, Mateusz Szmidt

Machine learning approach to trade flows estimation

Final project for Trade Policy
in
Analysis and Policy in Economics

March 2019

Contents

List of Figures

List of Tables

Introduction

Since the pioneer work of Tinbergen (1962), the gravity equations has been widely implemented in the estimation of bilateral trade flows. The fundamental insight that the volume of trade between two countries is proportional to the product of an index of their economic sizes diminished by the measures of “trade resistance” between them has shaped the empirical specifications mainly due to the surprisingly good fit to the majority of data sets of both regional, as well as international trade flows. Over time the Tinbergen (1962) approach has been modified and enhanced, not to mention the supplementary theoretical underpinnings such as additional measures of trade resistance in spite of the classical ones (geographic distance, a dummy for common borders or dummies for Commonwealth memberships) or better estimation methods, allowing for the inclusion of zero-trade flows in the framework.

The following paper aims to implement the modern machine learning algorithms in the framework of gravity modeling in order to predict the bilateral trade flows. Machine learning can be viewed as an application of artificial intelligence (AI) which provides systems the ability to automatically learn and improve from experience without being explicitly programmed. In other words, machine learning focuses on the development of computer programs that can access data and use it learn for themselves, without human intervention or assistance, and adjust actions accordingly. The latest advancements in machine learning allowed to effortlessly identify patterns in data and use them to automatically make predictions or decisions. To the authors’ best knowledge, the following paper is the first try in implementing the above-mentioned framework to the trade policy analysis.

In addition, due to the familiarity of both authors to the Polish trade environment, the Poland trade relations has been chosen as a workhorse illustration. Obtained results prove that a neural network approach can be viewed as a grievous challenger to the classical estimation methods, such as Poisson Pseudo-Maximum Likelihood models or ordinary fixed panel data estimators.

The paper is organized as follows: the first chapter consists of the brief literature review, including the common gravity models and the estimation techniques, followed by the data characterization. Next sections provide a detailed description of the neural network approach enhanced by the hyper-parameters tuning and outline the main results. The paper is completed with the concluding remarks with potential extensions, references and appendices with codes in R and Python.

Chapter 1

Literature review

The traditional gravity model was developed in the 1960s to explain factory-to-consumer trade (Tinbergen (1962)). The above-mentioned concept was at the heart of the first clear microfoundations of the gravity equation – the seminal Anderson (1979), proposing a theoretical explanation of the gravity equation based on constant elasticity of substitution preferences of nations producing a single differentiated product. In parallel, the monopolistic competition versions were introduced (Krugman (1980), Bergstrand (1985)), followed by the work of Anderson and Van Wincoop (2003), expanding appropriate econometric techniques and introducing the microeconomic framework to the previously promoted monopolistic competition. Subsequent theoretical refinements have further focused on showing that the gravity equation can be derived from trade models with heterogeneous firms (Helpman et al. (2008)).

Simultaneously, the estimation techniques were progressing, starting from the basic least square estimator and its correspondent panel data version, meaning the fixed effect estimator. The endogeneity issues guided to the establishment of instrumental variables and two step least squares methodologies in the gravity models framework. Therefore, the Poisson Pseudo-Maximum Likelihood (henceforth PPML) model, introduced by Santos Silva and Tenreyro (2006), as well as zero-inflated models were proposed in order to solve the mentioned problems. Over the years, they became the flagship framework for the bilateral trade flows estimation with some dominance of PPML, mainly due to its statistical properties such as robustness to different forms of heteroskedasticity.

However, the aforementioned advantage was often criticized over the years, not to mention Martin and Pham (2008) who admitted that PPML estimator is in fact less biased than formerly used methods, but not necessarily fully unbiased. This view was further supported by Martínez-Zarzoso (2013) who compared it within a family of GLS models, arguing that the appropriate estimation method should be chosen with a greater caution. Consequently, authors attempt to propose a machine learning neural network algorithm as a potential competitor to the Poisson Pseudo-Maximum Likelihood estimator in the context of bilateral trade flows.

Chapter 2

Data exploration

For the first part of the data, namely the set of explanatory variables, the CEPII statistics were used, resulting in annual data of 60 variables at the cross country level. Then, using 3 digit ISO codes the dataset was joined with the trade flows information. Nevertheless, in contrary to the first, fully available online dataset, in order to obtain data on flows from Comtrade database, a data scrapper needed to be created. The authors expanded and modified the scrapping function delivered by Comtrade which in the end allowed to bypass all the limitations build into basic API and optimize the time of data scrapping. The exact code can be found in Appendix A.

The final variables used in the calculations, along with their descriptions, are presented in the table Variables and their description, while the basic summary statistics are illustrated in the table Summary statistics.

Table 2.2: Variables and their description

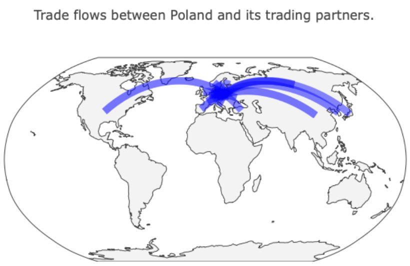
Variable	Description
yr	Year
rt3ISO	Standard ISO code for reporting country (three letters)
pt3ISO	Standard ISO code for partner country (three letters)
contig	Dummy for contiguity
heg_d	Dummy if parter country is current or former hegemon of origin
col_fr	Dummy for reporting and partner countries colonial relationship post 1945
colony	Dummy for reporting and partner countries ever in colonial relationship
sibling	Dummy for reporting and partner countries ever in sibling relationship i.e. two colonies of the same empire
comleg_pretrans	Dummy if reporting and partner countries share common legal origins before transition
comleg_posttrans	Dummy if reporting and partner countries share common legal origins after transition
transition_legalchange	Dummy if common legal origin changed since transition
legold_d	Legal system of partner country before transition. This variable takes the values: fr for French, ge for German, sc for Scandinavian, so for Socialist and uk for British legal origin
legnew_d	Legal system of partner country after transition. This variable takes the values: fr for French, ge for German, sc for Scandinavian, so for Socialist and uk for British legal origin
gatt_d	Dummy if partner country is GATT/WTO member
fta_wto	Dummy for Regional Trade Agreement
eu_to_acp	Dummy for ACP country exporting to EC/EU member
gsp_o_d	Dummy if origin is donator in Generalized System of Preferences (GSP)
flaggsp_o_d	Report changes in Roses data on <gsp_o_d>. No gsp recorded in Rose; Data directly from Rose; Changes in data from Rose; Assumption that gsp continues after 1999
eu_o	Dummy if reporting country a member of the European Union
eu_d	Dummy if partner country a member of the European Union
Trade_value_total	Total value of trade between reporting and partner countries
distw	Weighted bilateral distance between reporting and partner countries in kilometer (population weighted)
pop_o	Population of reporting country total in million
pop_d	Population of partner country total in million
gdp_o	GDP of reporting country (current US$)
gdp_d	GDP of partner country (current US$)
gdpcap_o	GDP per capita of reporting country (current US$)
gdpcap_d	GDP per capita of partner country (current US$)
area_d	Area of partner country in sq. kilometers
tdiff	Time difference between reporting and partner countries in number of hours. For countries which stretch over more than one time zone the respective time zone is generated via the mean of all its time zones (for instance: Russia, Canada, USA)
comrelig	Religious proximity (Disdier and Mayer (2007)) is an index calculated by adding the products of the shares of Catholics, Protestants and Muslims in the exporting and importing countries. It is bounded between 0 and 1 and is maximum if the country pair has a religion which (1) comprises a vast majority of the population and (2) is the same in both countries.

Table 2.1: Summary statistics

Variable	count	unique	top	freq	mean	std	min	25%	50%	75%	max	cova
gatt_o	4060.00	NaN	NaN	NaN	1.00	0.00	1.00	1.00	1.00	1.00	1.00	0.00
area_o	4060.00	NaN	NaN	NaN	312685.00	0.00	312685.00	312685.00	312685.00	312685.00	312685.00	0.00
yr	4060.00	NaN	NaN	NaN	2004.49	6.31	1994.00	1999.00	2005.00	2010.00	2015.00	0.00
pop_o	4060.00	NaN	NaN	NaN	38.33	0.22	38.00	38.15	38.23	38.54	38.66	0.01
gdp_o	4060.00	NaN	NaN	NaN	3.19e+11	1.55e+11	1.09e+11"	1.72e+11"	3.04e+11	4.77e+11	5.45e+11	0.48
gdpcap_o	4060.00	NaN	NaN	NaN	8339.74	4052.41	2819.70	4483.24	7976.12	12554.55	14341.86	0.49
gatt_d	4060.00	NaN	NaN	NaN	0.75	0.43	0.00	1.00	1.00	1.00	1.00	0.57
distw	4060.00	NaN	NaN	NaN	6140.89	3899.61	387.07	2603.11	5845.77	8583.18	17653.91	0.64
eu_o	4060.00	NaN	NaN	NaN	0.55	0.50	0.00	0.00	1.00	1.00	1.00	0.91
tdiff	4060.00	NaN	NaN	NaN	3.38	3.12	0.00	1.00	2.00	6.00	12.00	0.92
comrelig	4060.00	NaN	NaN	NaN	0.25	0.28	0.00	0.01	0.11	0.45	0.79	1.15
gdpcap_d	4060.00	NaN	NaN	NaN	10410.33	16066.97	64.81	864.06	3223.29	13299.54	116612.88	1.54
gsp_o_d	4060.00	NaN	NaN	NaN	0.24	0.43	0.00	0.00	0.00	0.00	1.00	1.80
fta_wto	4060.00	NaN	NaN	NaN	0.23	0.42	0.00	0.00	0.00	0.00	1.00	1.84
eu_to_acp	4060.00	NaN	NaN	NaN	0.21	0.40	0.00	0.00	0.00	0.00	1.00	1.96
comleg_pretrans	4060.00	NaN	NaN	NaN	0.17	0.38	0.00	0.00	0.00	0.00	1.00	2.20
transition_legalchange	4060.00	NaN	NaN	NaN	0.13	0.34	0.00	0.00	0.00	0.00	1.00	2.55
area_d	4060.00	NaN	NaN	NaN	719321.80	1956801.49	25.00	25713.00	119902.00	547244.00	17075400.00	2.72
eu_d	4060.00	NaN	NaN	NaN	0.11	0.32	0.00	0.00	0.00	0.00	1.00	2.81
comleg_posttrans	4060.00	NaN	NaN	NaN	0.10	0.30	0.00	0.00	0.00	0.00	1.00	2.96
pop_d	4060.00	NaN	NaN	NaN	34.23	129.66	0.01	1.69	6.66	21.70	1371.22	3.79
gdp_d	4060.00	NaN	NaN	NaN	2.64e+11	1.13e+12	1.09e+07	3.16e+09	1.46e+10	1.04e+11	1.80e+13	4.30
Trade_value_total	4060.00	NaN	NaN	NaN	1.12e+09	5.30e+09	0.00	1.52e+06	2.34e+07	2.35e+08	1.03e+11	4.74
contig	4060.00	NaN	NaN	NaN	0.04	0.19	0.00	0.00	0.00	0.00	1.00	5.04
sibling	4060.00	NaN	NaN	NaN	0.03	0.16	0.00	0.00	0.00	0.00	1.00	6.02
colony	4060.00	NaN	NaN	NaN	0.01	0.07	0.00	0.00	0.00	0.00	1.00	13.55
col_fr	4060.00	NaN	NaN	NaN	0.01	0.07	0.00	0.00	0.00	0.00	1.00	13.55
heg_d	4060.00	NaN	NaN	NaN	0.01	0.07	0.00	0.00	0.00	0.00	1.00	13.55
rt3ISO	4060.00	1	POL	4060	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
pt3ISO	4060.00	190	ARE	22	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
iso2_o	4060.00	1	PL	4060	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
iso2_d	4060.00	190	BZ	22	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
comlang_off	4060.00	NaN	NaN	NaN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	NaN
comlang_ethno	4060.00	NaN	NaN	NaN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	NaN
comcol	4060.00	NaN	NaN	NaN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	NaN
col45	4060.00	NaN	NaN	NaN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	NaN
heg_o	4060.00	NaN	NaN	NaN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	NaN
col_to	4060.00	NaN	NaN	NaN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	NaN
curcol	4060.00	NaN	NaN	NaN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	NaN
cursib	4060.00	NaN	NaN	NaN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	NaN
comcur	4060.00	NaN	NaN	NaN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	NaN
legold_o	4060.00	1	so	4060	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
legold_d	4060.00	5	fr	1759	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
legnew_o	4060.00	1	ge	4060	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
legnew_d	4060.00	5	fr	2153	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
acp_to_eu	4060.00	NaN	NaN	NaN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	NaN
gsp_d_d	4060.00	NaN	NaN	NaN	0.00	0.00	0.00	0.00	0.00	0.00	0.00	NaN
flaggsp_o_d	4060.00	3	no gsp recorded in Rose	3099	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
flaggsp_d_d	4060.00	1	no gsp recorded in Rose	4060	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Where the columns denote respectively the variables described, the number of observations, the amount of unique values, the most frequent value (for categorical variables), the frequency of the most frequent value, the mean, the standard deviation, the minimum value, the first quantile, the median, the third quantile, the maximum value and finally the coefficient of variation.

Furthermore, due to the fact that the authors concentrated their attention on the trade flows of Poland with its partner, the following graphs demonstrate that Poland mainly trades with Unites States, China and Europe.



(a) Trade flows between Poland and its world trading partners

(b) Trade flows between Poland and its European trading partners

Figure 2.1: Trade flows between Poland and its trading partners

Moreover, the histogram of standarized trade flows of Poland from 1994 to 2015 certifies that there is a relatively large group of countries with which Poland is not involved in trading relations.

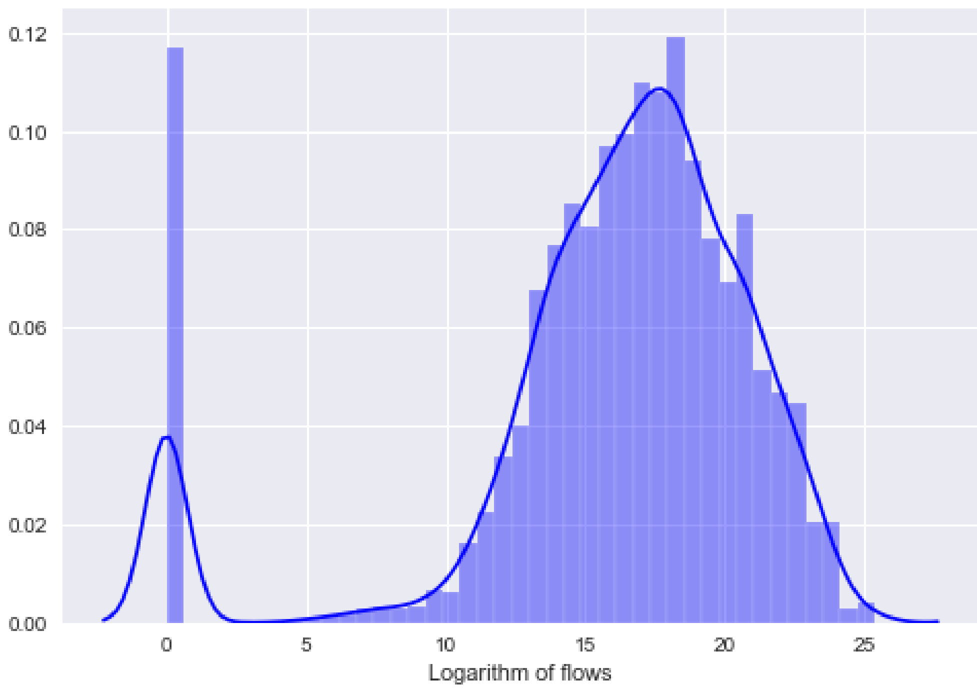
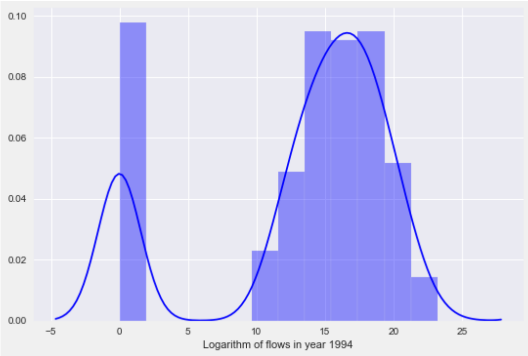
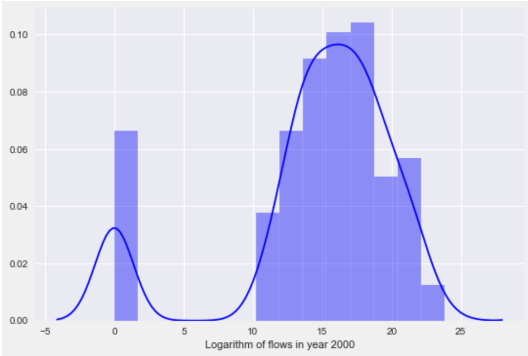


Figure 2.2: Histogram of flows over the history (overall)

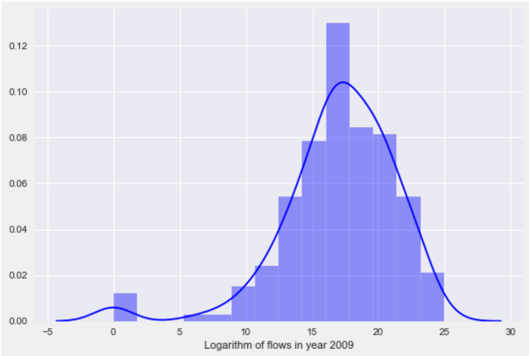
The aforementioned fact it complemented with the histograms in specific years: 1994, 2000, 2009 and 2015. However, it can be noticed that the amount of trading partners was gradually increasing over time.



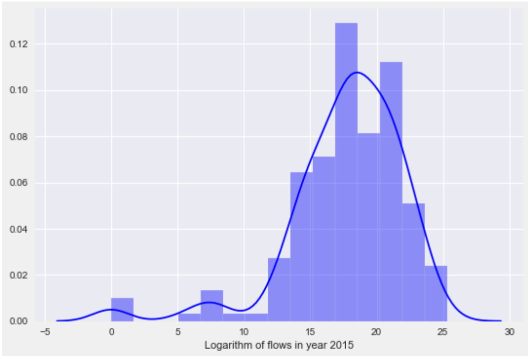
(a) Histogram of flows in 1994



(b) Histogram of flows in 2000



(c) Histogram of flows in 2009



(d) Histogram of flows in 2015

Figure 2.3: Histogram of flows in specific years

Finally, as the main interest of the following paper is the gravity model, the relations between total trade value, distance and partner country's GDP (standarized) are demonstrated in the pairplots graphs.

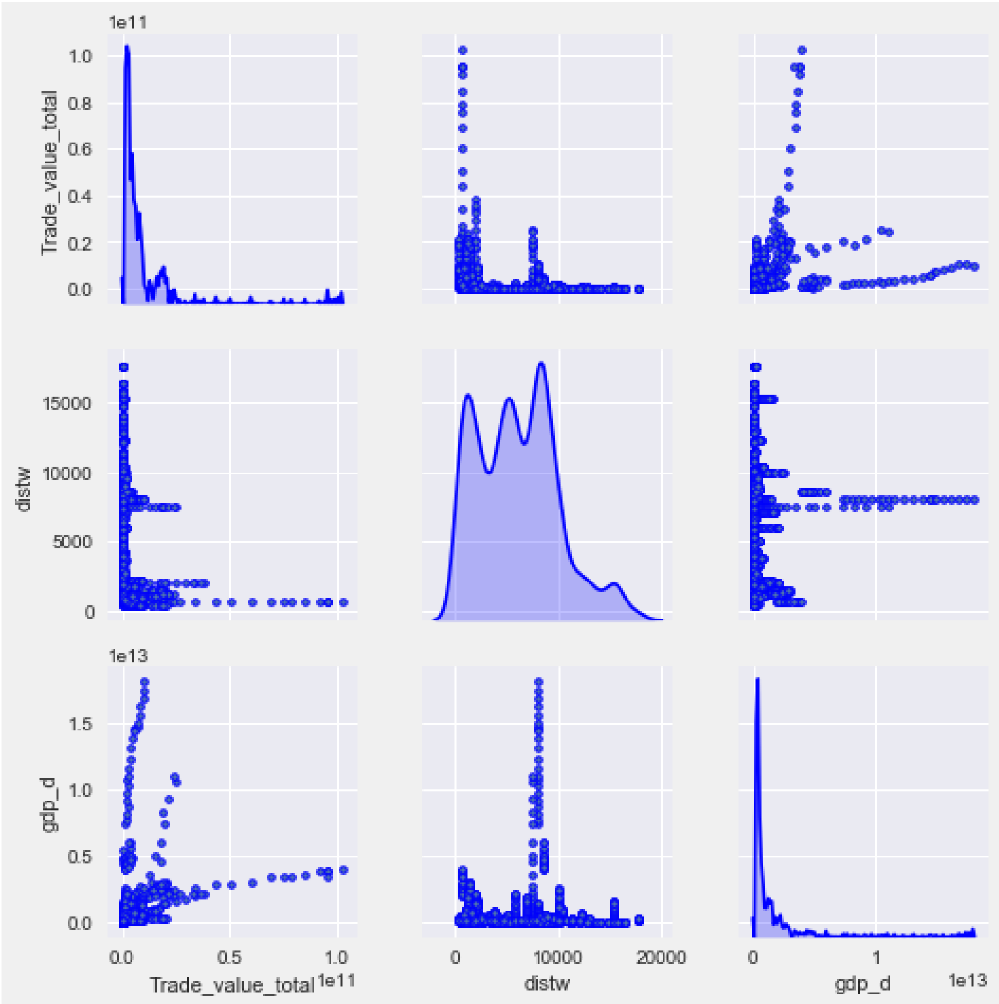


Figure 2.4: Pairplots between total trade value, distance and partner country's GDP (standarized)

Accordingly to the intuition, the value of trade is lower for the more distanced countries, while on contrary it seems to be positively correlated with the trading country's gross domestic product, which can be regarded as the proxy for the nation's size.

Chapter 3

Neural network approach

The neural networks approach is a statistical framework allowing to find complex patterns of relations in the data. The intuition behind the above-mentioned concept is often compared to the way of how human nerve system functions. In a nutshell, it can be characterized as follows - in the first phase the external signal is received by receptors and transferred to the set of neurons. Then, during further stages, it is iteratively processed and passed to next set of neurons until the signal is finally decoded. The structure of the neural network model similarly compounds of 3 elements: the input layer of independent variables, set of "hidden layers" and finally the output layer with calculated results of a model. Given the structure, in each phase besides the last one, the values of nodes from former layer are affinely transformed and then nonlinear function in performed in order to obtain the values for each node of a new layer. The calculations are repeated until the last phase when the final value is accessed through a nonlinear function of affine product of nodes from previous layers. The aforementioned process, starting from an input data and aiming to compute the output, is called the *forward propagation* and can be seen as a function of coefficients coined within every single affine transformation taking place between all neighbouring layers.

As a result, the estimated trade flows from the neural network approach rely on finding the appropriate values of parameters under arbitrary selected structure of a model. Thus in the first stage, the values of the coefficients are randomly assigned and then the forward propagation is performed. Next, based on model's output and true values of the observable dependent variable, the arbitrary chosen loss function is calculated. It has to be underlined that due to the fact that the generated output is a result of forward propagation, the loss function can be also defined as a function of the same parameters. It allows to compute a derivative with respect to them and in the end, to recalibrate their values – such a process is called *backward propagation* and it is iteratively repeated together with forward propagation to minimize the loss function, optimizing the values of parameters.

Although the intuition and general process behind the estimation of neural network model were presented above, a plethora of aspects referred to depends on arbitrary chosen structure or so called *architecture of a model*. Therefore, some choices implemented in the final, best suited to the data architecture of the model need to be elaborated.

Firstly, a number of hidden layers intuitively allows to approximate any continuous function more carefully, nevertheless adding any next layer is computationally costly. The charge born is strictly related to another element of a model's structure, namely the number of neurons in each layer. It has to be emphasised that the above-mentioned amount can be different depending on a layer but again bigger number directly translates into higher cost. Consequently, to take advantage of computer architecture and to optimize processing time, a power of 2 neurons in each layer were implemented, as suggested in the literature.

It has been already mentioned that each node is in fact defined as a function of the values of neurons from the former layer. It is thus beneficial to remark that it can be enforced that a node from hidden layer is a function of only a subset of nodes from a former one. Depending on the problem such an idea might be intuitive, not to mention the picture recognition, but it does not seem to be relevant in trade flows case. What is more, during the learning process such an exclusion of particular nodes may appear anyway, when the weights in affine transformations are relatively close to zero. Thus, the network with nodes being functions of all previous ones will be considered.

Moreover, the nonlinear transformation of a product of former nodes has to be defined. In the neural network framework, it is often called *an activation function*, aiming to activate the particular neuron on a hidden layer and assign to it some positive value when the particular pattern within a former nodes is observed. In a neural network literature, a particular set of functions can be observed, which by construction allows the model to be trained faster due to computational advantage while deriving derivatives and which satisfy the basic intuition behind activation. The most common ones are presented below.

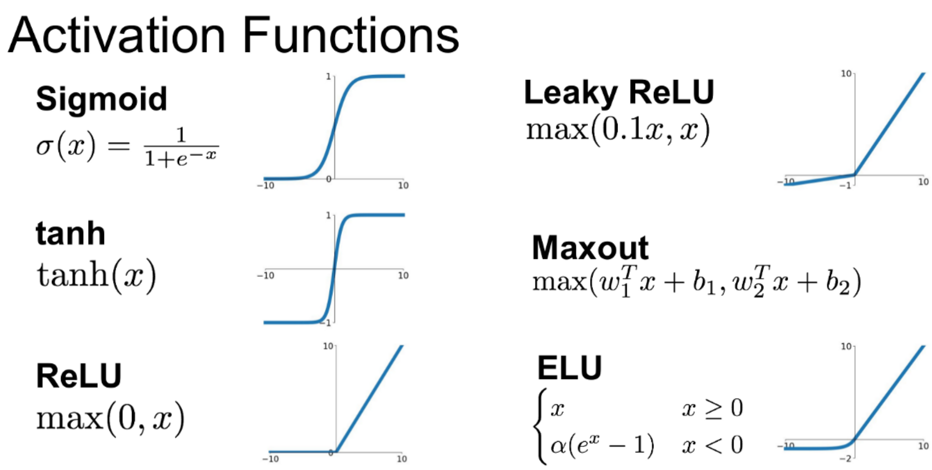


Figure 3.1: Activation functions[1]

[1]Source: https://bit.ly/2uh7NyV

The actually implemented in the end are sigmoid and relu. The first one was especially popular in the past, while the second one gained the popularity recently, outperforming the former with respect to the computational time.

At this stage, the part of hyper-parameters of models' structure directly connected to the forward propagation was covered. As far as the backward propagation choices are concerned, a loss function given the generated output has to be chosen. In the paper, the mean squared error was selected to validate the output. Moreover, in order to prevent the problem of overfitting, the regularization was implemented. The role of the aforementioned concept is simply to penalize the actual loss function of the model so that increase of the coefficients to some extent negatively affects the loss function. The value of a hyper-parameter of a penalty function identifies the size of marginal increase in a loss function alone to be compensated by the penalty.

Another regularization approach which can be implemented simultaneously is dropout. It serves to omit a fraction of randomly chosen nodes (along with their connections) on each layer while performing both forward and backward propagation. Thus, it enforces iterative deactivation of different neurons which diminish the pace of convergence but also stanches from overfitting the model. Nevertheless, it may simultaneously negatively influence the convergence, therefore a maximum number of iterations, in neural network framework called *epochs*, has to be specified. Choice of its value always brings a trade-off between a computational time and optimality of solution. One way (actually implemented in the paper) of meeting halfway is to set a threshold for marginal increase in loss function, ending the learning process sooner if the condition is fulfilled.

In fact, it is a learning process itself which determines the final performance of neural network framework. To fully define it, few more elements of a model's structure have to be recalled. As mentioned before, the estimation is based on calculating numerous derivatives with respect to all the parameters, according to the chain-rule, which in the end determines for a given set of parameters' values a point-gradient. However, as the neural networks tend to be defined over an enormous parameter space, the straightforward calculation of a gradient might be a complex task itself. Therefore, different optimization algorithms were implemented. The first one is Stochastic Gradient Decent (SGD), which calculates new iteration of parameters according to the specified learning rate, which is in turn another hyper-parameter of the model, defining the convergence speed. However, in standard SGD the learning rate is not scalable and it poorly handles updating the parameters of high variance. To deal with it, the second method is proposed, namely the Adam optimizer, which becomes gradually common recently. It allows to adjust the specified learning rate for each parameter and is often more efficient. Lastly, to speed up the whole process the hyper-parameter called batch size can be defined. The model chooses a subsample in a size of batch and performs an iteration using only selected observations. Therefore, it allows to train the model each time on different observation set and reduces the complexity of the whole process.

At this stage all the elements of model architecture are defined, allowing to implement the neural network on the presented grid of hyper-parameters and proceed with an estimation, with exact codes included in Appendix B.

Chapter 4

Results

As far as the main outcomes from the trade flows prediction through a neural network approach are concerned, the best performing ten models are presented in the following Results Table.

Table 4.1: Results of neural network

N	N_iter	Val_loss	Val_MSE	Loss	MSE	LR	L1	L2
1	250	2,865	0,077	2,889	0,079	0,5	0,1	20,05
2	202	1,528	0,078	1,542	0,079	0,5	10075	10075
3	37	1,500	0,074	1,518	0,079	0,5	10075	10075
4	75	0,099	0,078	0,100	0,080	0,5	0,1	10075
5	201	1,567	0,078	1,598	0,080	0,5	10075	20,05
6	41	0,466	0,079	0,468	0,080	3125	0,1	0,1
7	174	0,107	0,079	0,108	0,080	0,5	0,1	30025
8	250	1,026	0,079	1,032	0,081	0,5	0,1	10075
9	129	0,677	0,080	0,679	0,081	0,5	0,1	0,1
10	65	0,379	0,080	0,381	0,081	1375	0,1	0,1

N	First	Hidden	Batch	Epochs	Dropout	Opt	Losses	Activation
1	4	1	64	250	0,000	Adam	MSE	relu
2	8	1	32	250	0,000	Adam	MSE	relu
3	4	1	32	250	0,000	Adam	MSE	relu
4	16	1	64	250	0,000	Adam	MSE	relu
5	4	1	32	250	0,000	Adam	MSE	relu
6	4	2	64	250	0,000	Adam	MSE	relu
7	4	2	32	250	0,000	Adam	MSE	relu
8	4	1	64	250	0,000	Adam	MSE	relu
9	8	2	32	250	0,000	Adam	MSE	relu
10	4	2	64	250	0,000	Adam	MSE	relu

Where columns denote respectively: *Upper:* position in ranking, number of iterations to converge, loss for validation set, MSE for validation set, loss for test set, MSE for test set, learning rate, L1 penalty, L2 penalty; *Lower:* position in ranking, first layer size, number of hidden layers, batch size, maximum number of epochs, dropout, optimizer, losses, activation function;

Chapter 5

Concluding remarks

Here goes the conclusion.

REMEMBER TO CHANGE CODES

Bibliography

Cepii dataset. "http://www.cepii.fr".

Comtrade dataset. "https://comtrade.un.org".

James E Anderson. A theoretical foundation for the gravity equation. *The American Economic Review*, 69(1):106–116, 1979.

James E Anderson and Eric Van Wincoop. Gravity with gravitas: a solution to the border puzzle. *The American Economic Review*, 93(1):170–192, 2003.

Jeffrey H Bergstrand. The gravity equation in international trade: some microeconomic foundations and empirical evidence. *The review of economics and statistics*, pages 474–481, 1985.

Anne-Célia Disdier and Thierry Mayer. Je t'aime, moi non plus: Bilateral opinions and international trade. *European Journal of Political Economy*, 23(4):1140–1159, 2007.

Elhanan Helpman, Marc Melitz, and Yona Rubinstein. Estimating trade flows: Trading partners and trading volumes. *The Quarterly Journal of Economics*, 123(2):441–487, 2008.

Paul Krugman. Scale economies, product differentiation, and the pattern of trade. *The American Economic Review*, 70(5):950–959, 1980.

Will Martin and Cong S. Pham. Estimating the gravity model when zero trade flows are frequent. *Working Papers*, 3, 2008.

Inmaculada Martínez-Zarzoso. The log of gravity revisited. *Applied Economics*, 45(3): 311–327, 2013.

Joao Santos Silva and Silvana Tenreyro. The log of gravity. *The Review of Economics and Statistics*, 88(4):641–658, 2006.

Jan Tinbergen. An analysis of world trade flows. *Shaping the world economy*, 3:1–117, 1962.

Appendix A

```
# Data Scrapper for comtrade.un.org
# Authors Michal Miktus & Mateusz Szmidt
# February 2019

# Environment setup

closeAllConnections()
library(rjson)
library(data.table)

# Defining all functions necessary to scrap the data


# Support function closing all conections (urls) opened during a scrapping process to avoid errors
#
# It uses a vector of connections defined at the beginning of each process
# and closes the opened ones when process ends

connections_dropper <- function(vector){
  new_connections <- getAllConnections()
  if(length(vector)<length(new_connections)){
    connections_to_kill <- setdiff(new_connections, vector)
    for(i in 1:length(connections_to_kill)){
      con <- getConnection(i)
      close(con)
    }
  }
}

# Support function Splitting numeric or string vector into vector of n-elements batches with "," separator
# It allows to lower the number of queries

vector_processing <- function(vector, n){

  # We consider a case when the set size of a batch is greater than the length of vector

  if(length(vector)> n){

```

```
    list <- split(vector,cut(seq_along(vector),ceiling(length(vector)/n) ,
       labels = F))
    j = 1
    vector <- c()

    for(i in list){
      subsample <- NULL
      for(ii in i){
        if(is.null(subsample)){
          subsample <- paste(subsample, ii, sep="")
        }
        else subsample <- paste(subsample, ii, sep=",")
      }

      # Self check if the split was performed correctly
      if(length(i) > n){
        print("Something went wrong!")
      }
      vector[j] <- subsample
      j = j + 1
    }
  }
  else{
    vector_ <- vector
    vector <- c()
    subsample <- NULL
    for(i in 1:length(vector_)){
      if(is.null(subsample)){
        subsample <- paste(subsample, vector_[i], sep="")
      }
      else subsample <- paste(subsample, vector_[i], sep=",")
    }
    vector[1] <- subsample
  }
  return(vector)
}



# Basic data scrapper for a single query
# Default values of parameters adjusted to download annuall data on trade
   flows
# Source: https://comtrade.un.org/data/Doc/api/ex/r

get.Comtrade <- function(url="http://comtrade.un.org/api/get?"
                         ,maxrec=50000
                         ,type="C"
                         ,freq="A"
                         ,px="HS"
                         ,ps="now"
                         ,r
                         ,p
                         ,rg="all"
```

```r
                              ,cc="TOTAL"
                              ,fmt="json"
)
{
  string<- paste(url
                    ,"max=",maxrec,"&" #maximum no. of records returned
                    ,"type=",type,"&" #type of trade (c=commodities)
                    ,"freq=",freq,"&" #frequency
                    ,"px=",px,"&" #classification
                    ,"ps=",ps,"&" #time period
                    ,"r=",r,"&" #reporting area
                    ,"p=",p,"&" #partner country
                    ,"rg=",rg,"&" #trade flow
                    ,"cc=",cc,"&" #classification code
                    ,"fmt=",fmt         #Format
                    ,sep = ""
  )

  if(fmt == "csv") {
    raw.data<- read.csv(string,header=TRUE)
    return(list(validation=NULL, data=raw.data))
  } else {
    if(fmt == "json" ) {
      raw.data<- fromJSON(file=string)
      data<- raw.data$dataset
      validation<- unlist(raw.data$validation, recursive=TRUE)
      ndata<- NULL
      if(length(data)> 0) {
        var.names<- names(data[[1]])
        data<- as.data.frame(t( sapply(data,rbind)))
        ndata<- NULL
        for(i in 1:ncol(data)){
          data[sapply(data[,i],is.null),i]<- NA
          ndata<- cbind(ndata, unlist(data[,i]))
        }
        ndata<- as.data.frame(ndata)
        colnames(ndata)<- var.names
      }
      return(list(validation=validation,data =ndata))
    }
  }
}


# Definining an object for an output of basic_scrapper function
output <- setRefClass("scrapper_output", fields = list(data ="ANY", checked = "ANY", hits = "ANY"))


# Function scrapping the data on all possible connections
# between countries defined in the input <vector> and all the partners available
# for the years defined as <year> .
```

```r
#
# To control for the number of queries we use the parameter <hits>.
# It allows to stop the process after 100 hits to not exceed an hourly limit of 100 queries

basic_scrapper <- function(vector, year, hits){

  # Console output and definition of an output object
  print(paste("Trying for vector of", length(vector), "length."))
  current_connections <-getAllConnections()
  data <- NULL
  checked <- NULL

  # Looping over all batches of countries in a tryCatch block to avoid a failure of a process
  for(i in 1:length(vector)){
    tryCatch({
      print(i)
      out <- NULL
      unit <- get.Comtrade(r=vector[i], p="all", ps=toString(year), freq="A")

      if(is.null(unit$data)){
        checked <- rbind(checked, vector[i])
        print(paste("No data available for year", year, "for" ,vector[i]))
      }
      else{
        checked <- rbind(checked, vector[i])
        out <- unit$data
      }
    },
    error = function(e){
      print(paste("Error for", i))
    }
    )

    # Stopping the process for 1 hour after 100 hits
    hits = hits + 1

    if(hits >= 100){
      Sys.sleep(3600)
      hits = 0
    }

    # Output generation
    data <- rbind(data, out)

    # Dropping all connections opened during a process
    connections_dropper(current_connections)
  }

  out <- output(data = data, checked = checked, hits = hits)
  return(out)
```

```
}


# Main scrapping process using basic_scrapper function
# It splits the year range into batches of length 5 to optimize the number of queries.
# It also splits the list of countries into batches with initial length of 5,
# the batches where the error occured are joined and split again into batches of smaller size (up to 1).

main_scrapper <- function(main_vector, from, to){

  # Definition of an output object and years range splitting into batches of 5
  main_data <- NULL
  years <- vector_processing(seq(from, to), 5)
  hits = 0

  # Looping over the years
  for(i in 1:length(years)){
    vector <- main_vector
    cond <- TRUE
    data <- NULL
    try <- NULL
    split <- 5

    # Scrapping the data for all connections between the countries for a given batch of years
    # It is continued until for none of a countries an error is reported
    while(cond){
      print(paste("Scrapping for years:", years[i]))
      print(paste("The number of countries checked in one hit is", split))

      unit <- basic_scrapper(vector, years[i], hits)
      data <- rbind(data, unit$data)
      hits <- unit$hits
      try <- unique(rbind(unlist(try), unique(unlist(unit$checked))))

      # Vector of countires for which error is reported and so the queries will be repeated
      vector <- setdiff(unlist(strsplit(main_vector, "\\,")), unlist(strsplit(try, "\\,")))

      #  Checking if data for all countries is scrapped,
      #  then if not splitting the vector of countries into batches of smaller size.
      if (length(vector) < 1){
        cond = FALSE
      }
      else{
        split <- max(split - 1, 1)
        vector <- vector_processing(vector, split)
```

```
      }
    }

    # Overriding the state of the scrapping after each finished batch of years
    main_data <- rbind(main_data, data)
    write.csv(file = "trade_data.csv", main_data)
  }
  return(main_data)
}

# Scrapping the data


# Scrapping the list of the countries listed in the comtrade database

download_reporters <- TRUE
if (download_reporters){
  string <- "http://comtrade.un.org/data/cache/partnerAreas.json"
  reporters <- fromJSON(file=string)
  reporters <- as.data.frame(t(sapply(reporters$results,rbind)))
}

# Adjusting the list of reporters for which the process works (removing "world" and "all")
vector <-vector_processing(unlist(as.numeric(reporters$V1[3:length(reporters$V1)])), 5)

# Data scrapping for the range of dates available in comtrade database
data <- main_scrapper(vector, 1962, 2018)
fwrite(file = "trade_data.csv", data)
```

Scrapper.R

Appendix B

```
# Neural net created for the gravity model prediction for the Trade Policy
    class at PSE
# Author: Michal Miktus at michal.miktus@gmail.com
# Date: 21.02.2019

# Import libraries

import plotly.io as pio
import plotly.graph_objs as go
import plotly.plotly as py
from plotly.offline import init_notebook_mode, iplot
from matplotlib import pyplot as plt
from scipy.stats import mstats
from statsmodels.distributions.empirical_distribution import ECDF
from sklearn.preprocessing import MinMaxScaler, StandardScaler

import os
import numpy as np
import pandas as pd
#import torch
import seaborn as sns
import keras
import tensorflow as tf
import talos as ta
from keras.optimizers import Adam, Nadam, SGD
from keras.activations import relu, elu, sigmoid, tanh
from keras.losses import mse
from talos.model.normalizers import lr_normalizer
from talos.model.layers import hidden_layers
from talos.model.early_stopper import early_stopper
%matplotlib inline

# from plotly import tools


# Set seed

random_state = 123
np.random.seed(random_state)
tf.set_random_seed(random_state)
#torch.manual_seed(random_state)
```

```python

# Supress scientific notation for pandas

pd.options.display.float_format = '{:.5f}'.format

# Templates for graphs

pio.templates.default = 'plotly_dark+presentation'
sns.set(style="ticks", context="talk")
plt.style.use("dark_background")
init_notebook_mode(connected=True)


# Path specifiation

#path = "/Users/miktus/Documents/PSE/Trade policy/Model/"
path = "C:/Repo/Trade/Trade-policy/"

# Import data

data = pd.read_csv(path + "/Data/final_data_trade.csv")

# Data exploration only for Poland

data = data.loc[data['rt3ISO'] == "POL"]

data.columns

# Number of trade partners

data["pt3ISO"].unique().shape

data.info()

# Dropping the duplicates from the dataset

data = data.drop_duplicates(keep='first')

# Handling missing data

data.isnull().sum()

data.dropna(thresh=data.shape[0] * 0.7, how='all', axis=1, inplace=True)

data.dropna(axis=0, inplace=True)
# data.fillna(data.mean(), inplace=True) # Or replace by the column mean

# Desribe data

description = data.describe(include='all')
coef_variation = description.loc["std"] / description.loc["mean"]
description.loc["cova"] = coef_variation
(description.sort_values(by="cova", axis=1)).T
```

```


# Number of unique entries

print(data.nunique())

# Names of binary data (unstandarized)

binary = []
for columns in data:
    if (data.loc[:, columns].min() == 0) & (data.loc[:, columns].max() ==
        1):
        binary.append(columns)

for columns in data.loc[:, binary]:
    print(data.loc[:, binary][columns].unique())

# Remove iso_2o, iso_2d and family

data.drop(columns=['iso2_d', 'iso2_o'], inplace=True)

# Numeric variables

data_numeric = data._get_numeric_data()
data_numeric.drop(columns="yr", inplace=True)
data_numeric.drop(columns=binary, inplace=True)

# Visualisations

# Numerical data distribution
data_numeric.hist(figsize=(10, 10), bins=50, xlabelsize=8, ylabelsize=8)

for i, col in enumerate(data_numeric.columns):
    plt.figure(i)
    sns.distplot(data_numeric[col], color="y")

sns.distplot(data_numeric["tdiff"], color="y")

sns.pairplot(data_numeric);
sns.pairplot(data_numeric, vars=["pop_o", "tdiff"]) # kind="reg"/kind="kde"



# Flows
flows = data[['yr','rt3ISO','pt3ISO','Trade_value_total']]
data_loc = pd.read_csv(path + "/Data/CountryLatLong.csv")
data_loc.drop(columns=['Country'], inplace=True)
data_loc.columns = ["CODE", "rt_Lat", "rt_Long"]

flows = pd.merge(flows, data_loc, left_on="rt3ISO", right_on="CODE").drop('
    CODE', axis=1)
data_loc.columns = ["CODE", "pt_Lat", "pt_Long"]
```

```
flows = pd.merge(flows, data_loc, left_on="pt3ISO", right_on="CODE").drop('CODE', axis=1)

flow_directions = []
for i in range( len( flows ) ):
    flow_directions.append(
        dict(
            type = 'scattergeo',
            locationmode = 'ISO-3',
            lon = [ flows['rt_Long'][i], flows['pt_Long'][i]],
            lat = [ flows['rt_Lat'][i], flows['pt_Lat'][i]],
            text = flows['pt3ISO'][i],
            mode = 'lines',
            line = dict(
                width = flows['Trade_value_total'][i]*10,
                color = 'red',
            ),
            opacity = 200 * np.power(float(flows['yr'][i]) - float(flows['yr'].min()),2)/float(np.power(float(flows['yr'].max()),2)),
        )
    )

layout = dict(
        title = 'Trade flows between Poland and its trading partners.',
        showlegend = False,
        geo = dict(
            scope='world',
            projection=dict( type='robinson' ),
            showland = True,
            landcolor = 'rgb(243, 243, 243)',
            countrycolor = 'rgb(204, 204, 204)',
        )
    )

fig = dict( data=flow_directions, layout=layout )
iplot( fig, filename='Flows map' )

print(data['Trade_value_total'].describe())

flows_winsorized = mstats.winsorize(data['Trade_value_total'], limits=[0.05, 0.05])
layout = go.Layout(
    title="Basic histogram of flows (winsorized)")

data_hist = [go.Histogram(x=flows_winsorized)]
fig = go.Figure(data=data_hist, layout=layout)

iplot(fig, filename='Basic histogram of flows')
sns.distplot(data['Trade_value_total'], axlabel= "Basic histogram of flows", color="y")


# Corr - to correct
```

```
corr = ecdf_normalized_df.corr()

sns.heatmap(corr[(corr >= 0.5) | (corr <= -0.5)],
            cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1,
            annot=True, annot_kws={"size": 8}, square=True)

# Coef of variation

layout = go.Layout(
    title="Coefficient of variation")

data_cova = [go.Histogram(x=description.loc["cova"])]
fig = go.Figure(data=data_cova, layout=layout)
iplot(fig,
      filename="Coefficient of variation")

high_cova = description.loc["cova"].where(lambda x: x > 0.30).dropna().sort_values(ascending=False)
high_cova



# Normalization

minmax_normalized_df = pd.DataFrame(MinMaxScaler().fit_transform(data_numeric),
                                    columns=data_numeric.columns, index=data_numeric.index)

standardized_df = pd.DataFrame(StandardScaler().fit_transform(data_numeric), columns=data_numeric.columns,
                               index=data_numeric.index)

ecdf_normalized_df = data_numeric.apply(
    lambda c: pd.Series(ECDF(c)(c), index=c.index))

# Replace data by its standardized values

data[list(ecdf_normalized_df.columns.values)] = ecdf_normalized_df


# Select only POL as rt3ISO

data_PL = data.query("rt3ISO == 'POL'")

data_PL.drop('rt3ISO', axis=1, inplace=True)

data_PL.info()

# One hot encoding
data_PL = pd.get_dummies(
```

```
    data_PL, columns=["pt3ISO", "legold_o", "legold_d", "legnew_o", "legnew_d", "flaggsp_o_d", "flaggsp_d_d"],
    prefix=["pt3ISO", "legold_o", "legold_d", "legnew_o", "legnew_d", "flaggsp_o_d", "flaggsp_d_d"])

# Splitting the data

# train_size = 0.9
# train_cnt = math.floor(data_PL.shape[0] * train_size)

splitting_yr = 2010

x_train = data_PL.drop('Trade_value_total', axis=1).loc[data_PL['yr'] <= splitting_yr].values
y_train = data_PL.loc[:, 'Trade_value_total'].loc[data_PL['yr'] <= splitting_yr].values
x_test = data_PL.drop('Trade_value_total', axis=1).loc[data_PL['yr'] > splitting_yr].values
y_test = data_PL.loc[:, 'Trade_value_total'].loc[data_PL['yr'] > splitting_yr].values

# Build NN class in PyTorch

# A fully-connected ReLU network with one hidden layer, trained to predict y from x
# by minimizing squared Euclidean distance.


class ThreeLayerNet(torch.nn.Module):
    def __init__(self, D_in, H_in, H_out, D_out):
        """
        In the constructor we instantiate two nn.Linear modules and assign them as
        member variables.
        """
        super(ThreeLayerNet, self).__init__()
        self.linear1 = torch.nn.Linear(D_in, H_in)
        self.linear2 = torch.nn.Linear(H_in, H_out)
        self.linear3 = torch.nn.Linear(H_out, D_out)

    def forward(self, x):
        """
        In the forward function we accept a Tensor of input data and we must return
        a Tensor of output data. We can use Modules defined in the constructor as
        well as arbitrary operators on Tensors.
        """
        h_relu_1 = self.linear1(x).clamp(min=0)
        h_relu_2 = self.linear2(h_relu_1).clamp(min=0)
        y_pred = self.linear3(h_relu_2)
        return y_pred

```

```

#x = torch.tensor(x_train).float()
#y = torch.tensor(y_train).float()

# N is batch size; D_in is input dimension;
# H is hidden dimension; D_out is output dimension.
N, D_in, H_in, H_out, D_out = int(data_PL.shape[0]), int((data_PL.shape[1] - 1)), 50, 50, 1

# Construct our model by instantiating the class defined above
#model = ThreeLayerNet(D_in, H_in, H_out, D_out)

# Construct our loss function and an Optimizer. The call to model.parameters()
# in the SGD constructor will contain the learnable parameters of the three
# nn.Linear modules which are members of the model.
#criterion = torch.nn.MSELoss(reduction='sum')
#optimizer = torch.optim.SGD(model.parameters(), lr=1e-4)
#for t in range(5):
#    # Forward pass: Compute predicted y by passing x to the model
#    y_pred = model(x)
#
#    # Compute and print loss
#    loss = criterion(y_pred, y)
#    print(t, loss.item())
#
#    # Zero gradients, perform a backward pass, and update the weights.
#    optimizer.zero_grad()
#    loss.backward()
#    optimizer.step()

# Build NN class in Keras


def build_model(x_train, y_train, x_val, y_val, params):

    model = keras.Sequential()
    model.add(keras.layers.Dense(10, activation=params['activation'],
                                 input_dim=x_train.shape[1],
                                 use_bias=True,
                                 kernel_initializer='glorot_uniform',
                                 bias_initializer='zeros',
                                 kernel_regularizer=keras.regularizers.l1_l2(l1=params['l1'], l2=params['l2']),
                                 bias_regularizer=None))

    model.add(keras.layers.Dropout(params['dropout']))

    # If we want to also test for number of layers and shapes, that's possible
    hidden_layers(model, params, 1)

    # Then we finish again with completely standard Keras way
```

```
    model.add(keras.layers.Dense(1, activation=params['activation'], use_bias=True,
                                 kernel_initializer='glorot_uniform',
                                 bias_initializer='zeros',
                                 kernel_regularizer=keras.regularizers.l1_l2(l1=params['l1'], l2=params['l2']),
                                 bias_regularizer=None))

    model.compile(optimizer=params['optimizer'](lr=lr_normalizer(params['lr'], params['optimizer'])),
                  loss=params['losses'],
                  metrics=['mse'])

    history = model.fit(x_train, y_train,
                        validation_data=[x_val, y_val],
                        batch_size=params['batch_size'],
                        epochs=params['epochs'],
                        callbacks=[early_stopper(epochs=params['epochs'], mode='strict')],
                        verbose=0)

    # Finally we have to make sure that history object and model are returned
    return history, model

# Then we can go ahead and set the parameters space


params = {'lr': (0.5, 4, 4),
          'l1': (0.1, 40, 4),
          'l2': (0.1, 40, 4),
          'first_neuron': [4, 8, 16],
          'hidden_layers': [0, 1, 2],
          'batch_size': [32, 64],
          'epochs': [250],
          'dropout': (0, 0.5, 5),
          'optimizer': [Adam, SGD],
          'losses': [mse],
          'activation': [relu, sigmoid]}

# Alternatively small parameters space


params_small = {'lr': (0.5, 5, 2),
                'l1': (0.1, 50, 2),
                'l2': (0.1, 50, 2),
                'first_neuron': [4],
                'hidden_layers': [0],
                'batch_size': [32],  # [32, 64, 128, 256],
                'epochs': [100],
                'dropout': (0, 0.5, 2),
                'optimizer': [Adam],
                'losses': [mse],
```

```
                    'activation': [relu]}

# Run the experiment

os.chdir(path + "/Data/")

t = ta.Scan(x=x_train,
            y=y_train,
            model=build_model,
            grid_downsample=1,
            val_split=0.3,
            params=params,
            dataset_name='POL',
            experiment_no='2')
```

Neural_net.py

Appendix C

```r

# Code for the Trade Policy class at PSE
# Author: Michal Miktus at michal.miktus@gmail.com
# Date: 23.02.2019


#path <- '/Users/miktus/Documents/PSE/Trade policy/Model/'
path <- 'C:/Repo/Trade/Trade-policy/'

setwd(path)
set.seed(12345)


# Load packages ---------------------------------------------------------------


list.of.packages <- c("readstata13", "data.table")

new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages, repos = "http://cran.us.r-project.org")

invisible(lapply(list.of.packages, library, character.only = TRUE))

# Useful functions

RMSE = function(m, o){
  sqrt(mean((m - o)^2, na.rm=TRUE))
}

# Perform computations or load the data ------------------------------------

data_cepii <- as.data.table(read.dta13(paste0(path,"Data/gravdata.dta")))
data_trade <- fread(paste0(path,"Data/trade_data.csv"))

# Delete cases for which the trading partner is unknown

data_trade <- data_trade[complete.cases(data_trade[,pt3ISO])]

```

```r
# Convert TradeValues to numeric, with emphasis on scientific notation issues

data_trade[, TradeValue := as.numeric(format(as.numeric(gsub(',', '.', TradeValue)), scientific = FALSE))]
data_trade <- data_trade[, c('yr', 'TradeValue', 'rt3ISO', 'pt3ISO')]
data_trade <- unique(data_trade[, 'Trade_value_total' := sum(TradeValue), by = c("yr", "rt3ISO", "pt3ISO")], by = c("yr", "rt3ISO", "pt3ISO", "Trade_value_total"))
data_trade[, TradeValue := NULL]
data_trade <- data_trade[!data_trade[, pt3ISO == 'WLD']]

# Merge data

# Inner

data_inner <- merge(data_trade, data_cepii, by.y = c('year', 'iso3_o', 'iso3_d'), by.x = c('yr', 'rt3ISO', 'pt3ISO'))

# table(data[,"yr"])

data_cepii["year" > 1993]

#Left

data_left <- merge(data_trade, data_cepii["year" > 1993], by.y = c('year', 'iso3_o', 'iso3_d'), by.x = c('yr', 'rt3ISO', 'pt3ISO'), all.y = T)

data_left[, Trade_value_total := lapply(data_left[,"Trade_value_total"], function(x) {ifelse(is.na(x), 0, x)})]

# Write whole dataset

fwrite(data_left, 'Data/final_data_trade.csv')
```

DataCleaning.R

Appendix D

```r
# Code for the Trade Policy class at PSE
# Author: Michal Miktus at michal.miktus@gmail.com
# Date: 23.02.2019


path <- '/Users/miktus/Documents/PSE/Trade policy/Model/'
# path <- 'C:/Repo/Trade/Trade-policy/'

setwd(path)
set.seed(12345)

# Load packages ---------------------------------------------------------------

list.of.packages <- c("readstata13", "data.table", "gravity", "dplyr", 'stargazer', 'caret')

new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages, repos = "http://cran.us.r-project.org")

invisible(lapply(list.of.packages, library, character.only = TRUE))

# Useful functions

RMSE = function(m, o){
  sqrt(mean((m - o)^2, na.rm=TRUE))
}

# Load the data ---------------------------------------------------------------

data <- fread(paste0(path,"Data/data_PL.csv"))
names(data) <- make.names(names(data), unique=TRUE)

# Year variable

year <- data[, 'yr']

# Near zero variance variables

near <- nearZeroVar(data)
```

```r
data <- data[, -near, with = FALSE]

# Remove highly correlated data

corr = cor(data)
hc = findCorrelation(corr, cutoff=0.8) # put any value as a "cutoff"
hc = sort(hc)
data = data[, -hc, with = FALSE]

# Add year (just for splitting)

data[, yr := year]

# Data split to compare the reults

data_bef2010 <- data[yr <= 2010]
# data_bef2010[, yr := NULL]
data_aft2010 <- data[yr > 2010]
# data_aft2010[, yr := NULL]
data_aft2010[, dist_log := log(distw)]
var <- setdiff(names(data_bef2010), c("Trade_value_total", "distw", "V1", "yr"))

# PPML: Poisson Pseudo Maximum Likelihood

PPML <- ppml(dependent_variable= "Trade_value_total", distance="distw", additional_regressors = var, es = T, robust=TRUE, data = data_bef2010)
summary(PPML)
predictions <- predict(PPML, newdata = data_aft2010)
residuals <- predictions - data_aft2010[,"Trade_value_total"]
MSE <- mean(sum(residuals^2)/length(unlist(residuals)))
max(unlist(residuals))

# Summary to latex

stargazer(PPML)

# FE ------------------------------------------------------------

dependent <- c("Trade_value_total")
continous <- c("distw", "pop_o", "pop_d", "gdp_o", "gdp_d", "area_d", "tdiff", "comrelig")
log_variables <- paste("log(",continous, ")", sep = "")
dummies <- setdiff(setdiff(names(data_bef2010), continous), dependent)

linear_het <- as.formula(paste(paste("log(",dependent, ")", sep = ""),
                               paste(paste(log_variables, collapse = " + "), paste(dummies, collapse = " + "), sep = " + "), sep = " ~ "))

linear_het <- as.formula(paste(dependent,
                               paste(paste(log_variables, collapse = " + "), paste(dummies, collapse = " + "), sep
```

```
                                    = " + "), sep = " ~ "))

FE <- lm(linear_het, data = data_bef2010)
#FE$coefficients <- lapply(coef(FE), function(x) {ifelse(is.na(x), as.
    numeric(0), as.numeric(x))})
summary(FE)
MSE_FE_train <- (mean(FE$residuals^2))
MSE_FE_train

 predictions <- predict(FE, newdata = data_aft2010)
 residuals =  predictions - (data_aft2010[,'Trade_value_total'])
max(residuals)

MSE_FE_test <- (sum(residuals^2)/length(unlist(residuals)))
MSE_FE_test

# Summary to latex

 stargazer(FE)

# FE on test

# FE <- lm(linear_het, data = data_aft2010)
# MSE_FE_aft <- mean(sum(FE$residuals^2)/length(FE$residuals))
# MSE_FE_aft
```

Gravity.R