developing trade consultants
policy • research • capacity building

Lecture 3: Introduction to Text Based Prediction and Classification

Ben Shepherd, Principal.

Ben@Developing-Trade.Com

Key Takeaways

1. Artificial neural networks (ANNs) are powerful prediction tools. They have a fundamentally different philosophy from regression-related methods: the emphasis is on learning through repeated prediction and adjustment based on observed error.

2. While technically complex, the intuition behind ANNs is easy to grasp. With the basics in place, it is possible to start using them quite quickly.

3. Workflow is familiar from the general ML workflow.

4. ANNs have been found to work very well with text as input, i.e. transforming text into numbers and using it to predict or classify.

5. R's Keras package offers a very powerful but easy to use approach to implementing ANNs.

6. Economics research in this area is in its infancy, so there is huge scope to make a contribution!

Outline

1. Motivation: A Text-Based Classification Problem

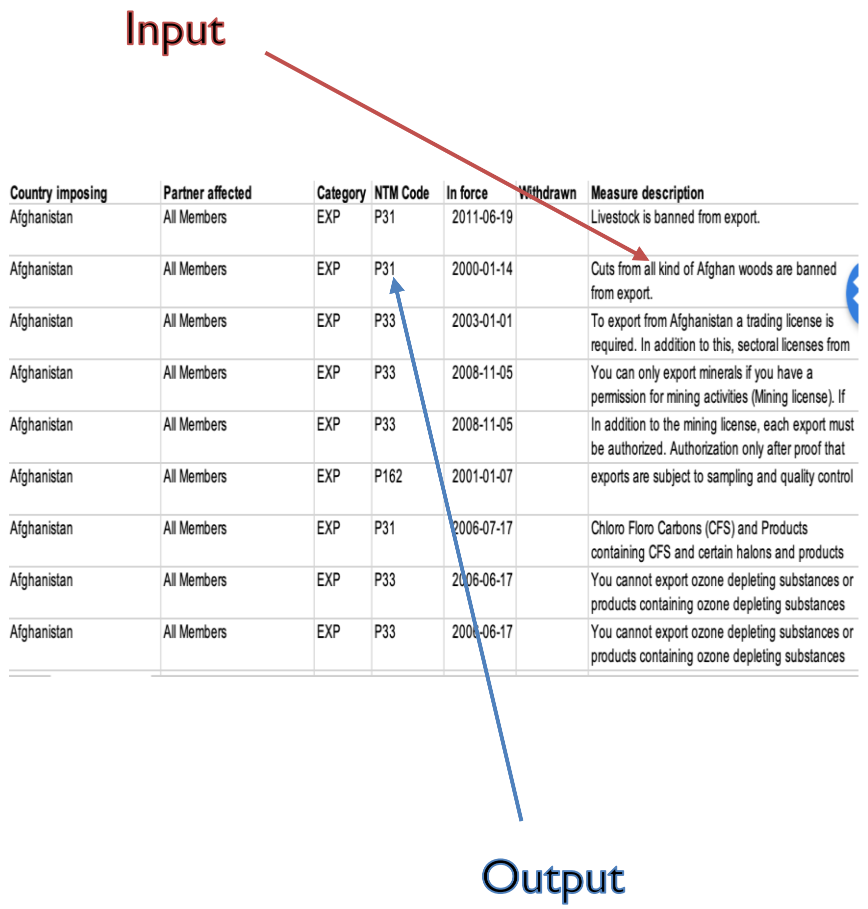
- So far, we've looked at two ingredients:
  - Basic ML tools.
  - Basic text-as-data tools.

- This lecture connects the two by asking the following question: can we use text to predict or classify, in the same way we use standard quantitative data?

- To answer it, we need to introduce a new tool: the artificial neural network (ANN) that is widely used in text-based prediction and classification problems.

- ANNs have a recognizable workflow from basic ML, but work in a fundamentally different way, so we start by setting up a standard quantitative problem to fix ideas, then we move to text.

1. Motivation: A Text-Based Classification Problem


Input
Output

Country imposing	Partner affected	Category	NTM Code	In force	Withdrawn	Measure description
Afghanistan	All Members	EXP	P31	2011-06-19		Livestock is banned from export.
Afghanistan	All Members	EXP	P31	2000-01-14		Cuts from all kind of Afghan woods are banned from export.
Afghanistan	All Members	EXP	P33	2003-01-01		To export from Afghanistan a trading license is required. In addition to this, sectoral licenses from
Afghanistan	All Members	EXP	P33	2008-11-05		You can only export minerals if you have a permission for mining activities (Mining license). If
Afghanistan	All Members	EXP	P33	2008-11-05		In addition to the mining license, each export must be authorized. Authorization only after proof that
Afghanistan	All Members	EXP	P162	2001-01-07		exports are subject to sampling and quality control
Afghanistan	All Members	EXP	P31	2006-07-17		Chloro Floro Carbons (CFS) and Products containing CFS and certain halons and products
Afghanistan	All Members	EXP	P33	2006-06-17		You cannot export ozone depleting substances or products containing ozone depleting substances
Afghanistan	All Members	EXP	P33	2006-06-17		You cannot export ozone depleting substances or products containing ozone depleting substances

- UNCTAD's TRAINS database catalogues NTMs using the MAST classification.
- One of the options is to download a dataset with (lots of information) + a verbal description of a measure from a human coder, and a MAST code for the type of measure.
- Can we take out one element of human intervention by taking the short descriptions and using them to predict NTM codes with reasonable accuracy?
- First part of a bigger challenge: take the full text of laws or regulations and use them to predict NTM codes, products, etc.

1. Motivation: A Text-Based Classification Problem

- From the work we've already done, you can hopefully see the two steps in this problem:
  - Turn the short description into input data.
  - Use a model to use input data to predict output data.

- Much of the workflow will be familiar:
  - Preprocessing of text.
  - Cross-validation.
  - Etc.

- But we need a more sophisticated approach to ML before we can really crack this kind of problem…

2. Artificial Neural Networks for Prediction and Classification

- Lasso-type approaches to ML are quite recognizable from econometrics, and can be easily related to the standard OLS model.
  - Remember the different emphasis, though: prediction rather than inference.

- Artificial Neural Networks (ANNs) have the same emphasis on prediction, but work in a fundamentally different way.
  - Standard practice is not to even report anything like "coefficients".
  - Focus is almost exclusively on prediction accuracy.

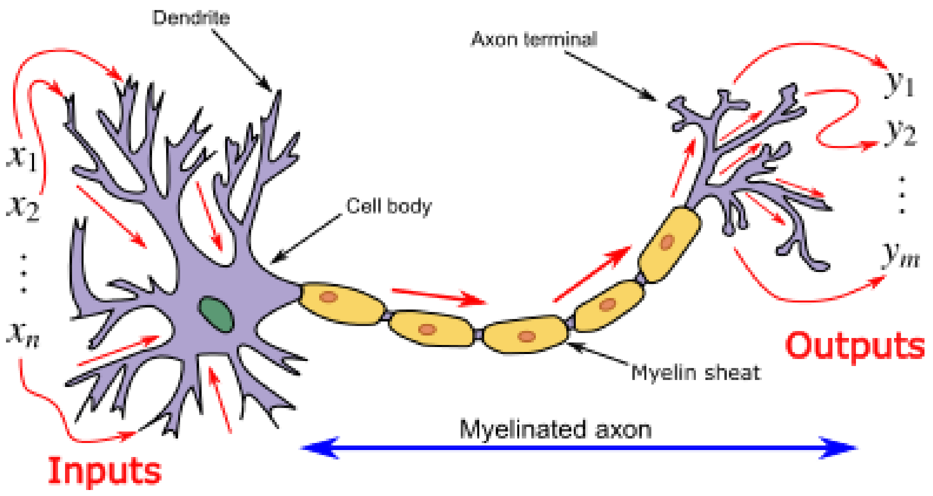
- The math behind ANNs is sophisticated, in terms of understanding the mechanics of the algorithms that power them.

- The intuition is more accessible, so that is our focus here. Our objectives are to identify:
  - The intuition behind how an ANN works.
  - The different choices that go into making a functional ANN to solve a prediction or classification problem.

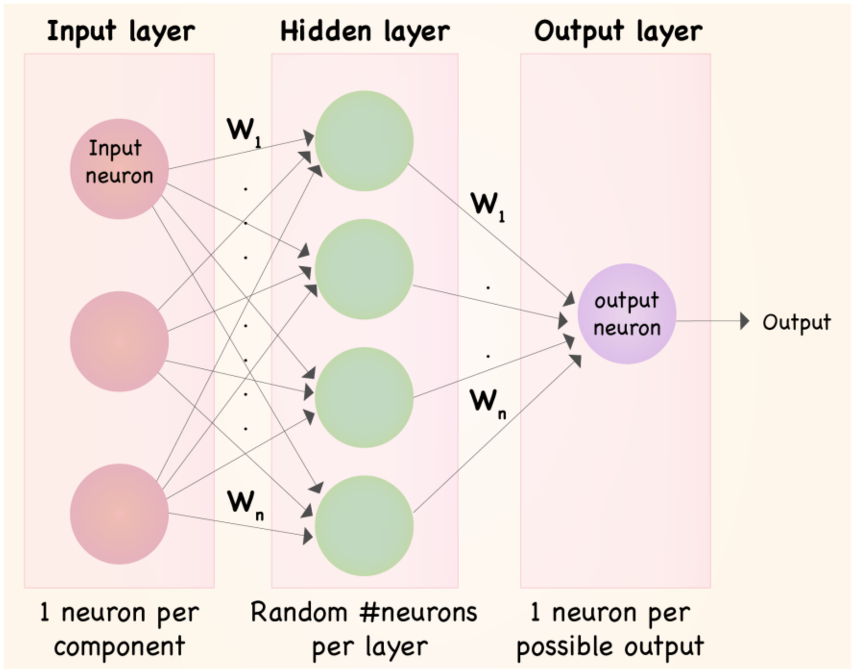
- ANNs are everywhere in our daily lives, usually in a "deep learning" format:
  - Image recognition.
  - Predictive text.
  - Automatic translation.
  - …

2. Artificial Neural Networks for Prediction and Classification

Inspiration

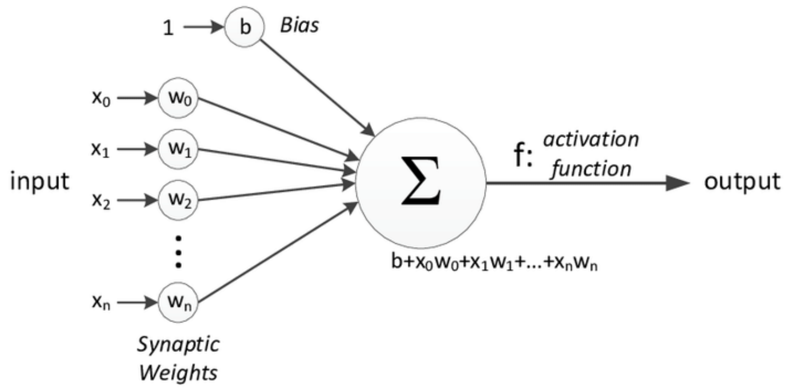


Implementation



Sources: https://www.kdnuggets.com/2019/10/introduction-artificial-neural-networks.html#:~:text=%E2%80%9CArtificial%20Neural%20Networks%20or%20ANN,to%20solve%20a%20specific%20problem.%E2%80%9D
& https://towardsdatascience.com/understanding-neural-networks-what-how-and-why-18ec703ebd31.

2. Artificial Neural Networks for Prediction and Classification



- ANNs do not see the world through the lens of a regression problem.
- Rather, they try to "learn" by using inputs to product outputs.
- By doing it over and over, the model can refine the weight given to each input so that predictions get better and better.
- Each neuron (node, unit) takes a set of inputs and applies weights, plus a set bias factor.
- It then sends this linear combination as an output through an activation function, which can be nonlinear.
- Deep learning means, in essence, stacking a number of layers of neurons one on top of the other.

2. Artificial Neural Networks for Prediction and Classification

- The ANN is run recursively to refine the weights.
  - This is what "learning" means.
  - A common algorithm is stochastic gradient descent, which is a way of minimizing a defined loss function given the data and the structure of the model.
    - Gradient = error gradient, so move down it to try and find a minimum.

- Here's the basic idea (details are complex):
  - Initialize the ANN with random weights.
  - Take the inputs and apply processing in each layer, combining inputs and weights, regularizing using the activation function, and passing to the next layer, until a prediction of the output can be made.
  - Assess the prediction error.
  - Calculate the gradients of the loss function, and use this information to decide how to adjust each weight so as to move down the loss function.
  - Rinse and repeat.

2. Artificial Neural Networks for Prediction and Classification

- ANNs have many hyper-parameters that define the structure of the model and how it "learns". For instance:
  - How many layers will the model have?
  - How many neurons (nodes) will be in each layer?
  - Will dropout rules be applied (percentage of nodes randomly set to zero)?
  - How many samples (observations) in each batch?
  - How many epochs (full iterations) will the model be run over?
  - What is the metric to assess loss?
  - And many others!

- To set them rigorously, we can use a grid search, but it is very time consuming: many parameters and configurations. You can quickly end up estimating millions of models!

- To cut it down, try using a stochastic approach: define a model field, then sample randomly from it at a pre-defined weight.
  - For a given use of computing power, random search actually provides better results, as it enables a wider model field relative to grid search.

- There are typically many combinations of hyperparameters that produce relatively similar results, so choose the simplest model that delivers good results.

- More complex is not necessarily better: overfitting is a major risk with ANNs.

2. Artificial Neural Networks for Prediction and Classification

- A key advantage of ANNs is their ability to introduce nonlinearities.

- But this depends on the activation function that sends information from one layer to the next.

- Some common options:
  - Linear: as simple as the name suggests. Loses nonlinearity, but produces non-bounded outputs. Typically the output layer of a model producing a quantitative prediction without bounds.
  - Rectified linear (relu): good for general purpose applications; linear above zero; negatives shift to zero. Typically used in hidden layers.
  - Sigmoid: like a logistic function, bounded between zero and one. Typically used as the output layer of a model producing a binary classification.
  - Softmax: generalizes the logistic function to multiple dimensions. Typically used as the output layer of a model producing a multinomial classification.

2. Artificial Neural Networks for Prediction and Classification

- Ideally, we would use k-fold CV to analyze the prediction ability of an ANN.
- But because building and running the model is so time consuming, typically we just use a testing-training split. The testing subsample is typically referred to as "validation".
- A standard approach is to hold back part of the data for out of sample predictions. Then run the ANN again keeping part of the sample for validation.
- Be very careful of overfitting, as indicated by validation loss much higher than training loss. A typical pattern is that training loss tends to fall as model complexity and number of epochs increase, but validation loss will not follow the same pattern.

2. Artificial Neural Networks for Prediction and Classification

- The basic workflow is easily recognizable from what we have seen before:
  - Get the data into the right configuration (typically matrices).
  - Normalize the data (Lasso did this automatically!)
  - Split into testing and training samples.
  - Set up the model.
  - Run the model and examine prediction ability.
  - Repeat the process over a plausible set of hyperparameters.
  - Choose the model with best predictive ability.
  - Produce out of sample predictions, and check accuracy.

3. Text as an Input

- We have seen that with the right pre-processing, text can be a kind of data just like numbers.

- To use text as an input for an ANN, we need to go one step further: we actually turn text into numbers!

- Actually, we already have all the tools we need to do this from the last session.

- First, go through whatever pre-processing we want to apply:
  - Tokenize.
  - Remove stop words and numbers.
  - Lemmatize.

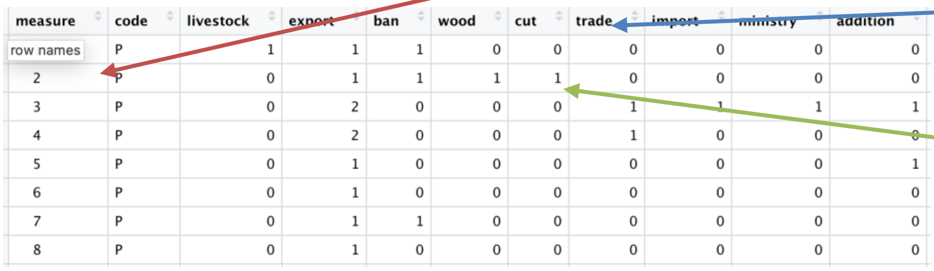
3. Text as an Input

- Next, calculate word counts: the number of times each word appears in a text input.
  - "A service provider may deliver any service without quantitative restriction" would show a count of two for "service" after lemmatization.

- Typically, we will only use a subset of the most frequent words rather than the full vocabulary.

- Transform the word counts to a document term matrix, as we did for LDA.

3. Text as an Input



measure	code	livestock	export	ban	wood	cut	trade	import	ministry	addition
row names	P	1	1	1	0	0	0	0	0	0
2	P	0	1	1	1	1	0	0	0	0
3	P	0	2	0	0	0	1	1	1	1
4	P	0	2	0	0	0	1	0	0	0
5	P	0	1	0	0	0	0	0	0	1
6	P	0	1	0	0	0	0	0	0	0
7	P	0	1	1	0	0	0	0	0	0
8	P	0	1	0	0	0	0	0	0	0

- This is how we ultimately want the data to be organized:
  - One row per text fragment = one row per observed outcome.
  - One column per word.
  - Cells indicate the number of times each word appears in the text fragment contained in each row.

- This should be pretty intuitive, as it is the same way we would set up a regression problem!

3. Text as an Input

- This is the most transparent way of preparing text as an ANN input.
- There are other ways of doing it in R that involve much less programming, but the pre-processing is typically different: stop words may stay in, words may not be lemmatized etc.
- Extension: natural language applications frequently use a different approach ("embedding") and a different type of ANN ("convolutional"), but they are outside scope here.
  - Rationale for retaining all words, as context is important, as is order.
  - But as the demonstration will show, we can actually do very well with just a simple ANN

4. Demonstration in R 1: Revisiting the LPI

- Recall our problem from Lecture 1: the LPI is being discontinued in its current form, and only ever had partial coverage by year and by country.
- Can we use WDI data to predict the LPI accurately?
- We've seen results using a Lasso.
- To get the hang of ANNs, let's try exactly the same exercise with an ANN.
- Note that all the preprocessing will be same, so we can move straight to how modeling is different.

4. Demonstration in R 1: Revisiting the LPI

- ANNs sound intimidating, but as usual "there's a package for that".
- The main package we are interested in is Keras: it makes building ANNs very easy.
  - The backend is provided by TensorFlow.
- In essence, we will be using an R interface to Python tools that also use C++ code…
- Together, the packages give you access to cutting edge tools, coming from the good people at Google.

4. Demonstration in R 1: Revisiting the LPI

```
model <- keras_model_sequential()


model %>%
  layer_dense(units = 50, activation = 'relu',
input_shape = c(55)) %>%
  layer_dense(units = 25, activation = 'relu')
%>%
  layer_dense(units = 12, activation = 'relu')
%>%
  layer_dense(units = 1)
```

- The easiest way to build a Keras ANN is to proceed sequentially, i.e. add layer after layer as a stack.
- The simple ANNs we are interested in use "dense" layers.
- Here we have code to create an ANN with:
  - A first layer that takes 55 inputs and has 50 neurons with a relu activation function.
  - A second layer with 25 neurons and a relu activation function.
  - A third layer with 12 neurons and a relu activation function.
  - An output layer with a single output and a linear activation function.
- This is a pretty typical set up for a "regression"-type ANN problem: we want predictions of some real valued variable, without bounds.

4. Demonstration in R 1: Revisiting the LPI

```
model %>% compile(
  loss = "mse",
  optimizer = "adam”
)

training <- model %>% fit(
  x_train, y_train, epochs =
500, batch_size = 100,
validation_split = 0.25
)
```

- Now we tell the model how we want it to work its magic:
  - The loss function is mean squared error, which is typical for a “regression” problem.
  - The choice of optimizer is less important: adam is a good general purpose algorithm.
- Then we train the model by telling it to run in batches of 100 for 500 epochs, keeping 25% of the training sample for validation.

4. Demonstration in R 1: Revisiting the LPI

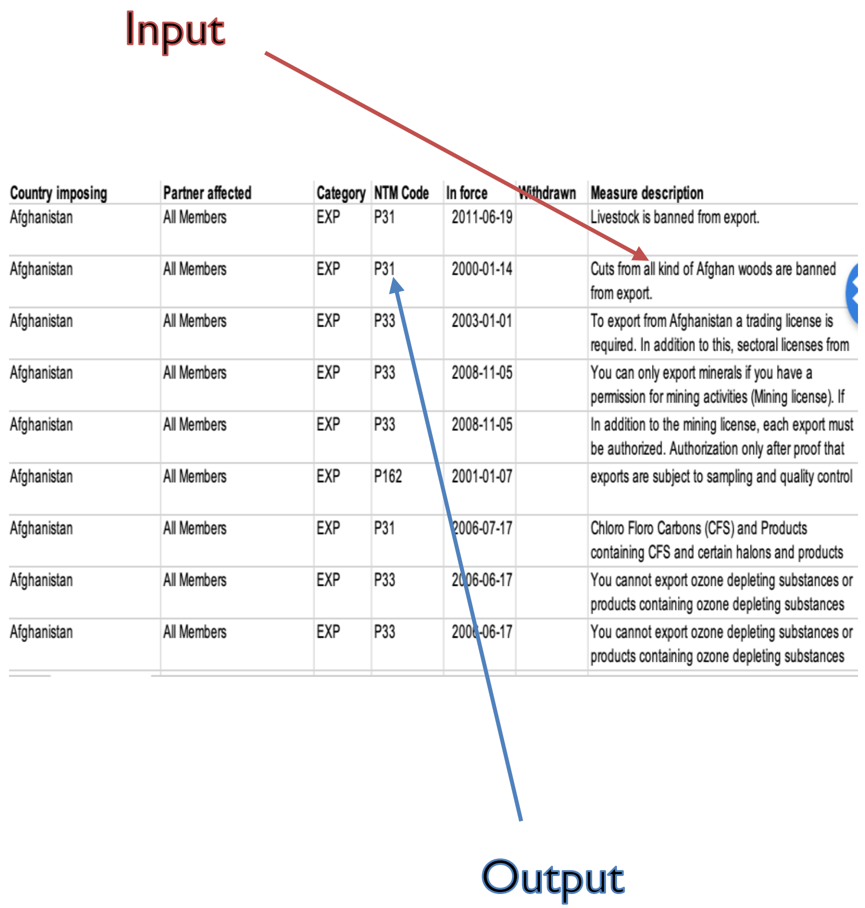
```
grid_search <- list(
  dense_units1 = c(10, 25, 50),
  dense_units2 = c(10, 25, 50),
  dense_units3 = c(10, 25, 50),
  dense_units4 = c(10, 25, 50),
  dense_units5 = c(10, 25, 50),
  dense_units6 = c(10, 25, 50),
  dense_units7 = c(10, 25, 50),
  dense_units8 = c(10, 25, 50),
  dense_units9 = c(10, 25, 50),
  dense_units10 = c(10, 25, 50),
  dropout = c(0, 0.1, 0.2)
)

runs <- tuning_run("core code.R", flags =
grid_search, confirm = FALSE, sample = 0.01)
```

- What about grid searching over the hyperparameter space?
- Tfruns is a package that helps automate it:
  - Write a core R script with your ANN, including a set of flags for parameters that will be changed during the search.
  - In your main file, call Tfruns with a list defining the hyperparameter space and a reference to your core script.
  - Assess results by choosing the smallest validation loss.
  - Use the sample = 0.01 option to use random sampling of (say) 1% of the total number of specifications.
- Here we try ten layers with different combinations of nodes (10, 25, 50), with use of dropout regularization.
- Search is random over 1% of the models defined by the list (=0.01 * 3^11 = 1,771).
- This can be very time consuming!

4. Demonstration in R 1: Revisiting the LPI

- Now we look at implementation.
- Remember that we already have a model that performs ok (the Lasso).
- ANNs are excellent tools for prediction, but they are not always and everywhere better, particularly in contexts with small amounts of data.
- So it is always good to have a baseline of comparison!

5. Demonstration in R 2: Classifying NTMs


Input
Output

Country imposing	Partner affected	Category	NTM Code	In force	Withdrawn	Measure description
Afghanistan	All Members	EXP	P31	2011-06-19		Livestock is banned from export.
Afghanistan	All Members	EXP	P31	2000-01-14		Cuts from all kind of Afghan woods are banned from export.
Afghanistan	All Members	EXP	P33	2003-01-01		To export from Afghanistan a trading license is required. In addition to this, sectoral licenses from
Afghanistan	All Members	EXP	P33	2008-11-05		You can only export minerals if you have a permission for mining activities (Mining license). If
Afghanistan	All Members	EXP	P33	2008-11-05		In addition to the mining license, each export must be authorized. Authorization only after proof that
Afghanistan	All Members	EXP	P162	2001-01-07		exports are subject to sampling and quality control
Afghanistan	All Members	EXP	P31	2006-07-17		Chloro Floro Carbons (CFS) and Products containing CFS and certain halons and products
Afghanistan	All Members	EXP	P33	2006-06-17		You cannot export ozone depleting substances or products containing ozone depleting substances
Afghanistan	All Members	EXP	P33	2006-06-17		You cannot export ozone depleting substances or products containing ozone depleting substances

- UNCTAD's TRAINS database has an option to download a coding of measures, with a short text description of the measure by a human coder.
- Can we use the descriptive texts to classify measures by code using an ANN?
- To simplify the problem, let's use letter codes only, rather than the full alphanumeric codes.

5. Demonstration in R 2: Classifying NTMs

- Preprocessing uses the tools we're familiar with from Lecture 2.

- But there are a few points of difference:
  - We need to identify just a subset of the total vocabulary, to keep the number of input nodes manageable.
  - We need to put everything in matrix form. DTM is a separate object class, so there is an extra step to transform it to a standard matrix.
  - A final trick is to use to_categorical to change a factor list (the codes) into a matrix with numerical dummies; this is the format Keras likes for multinomial classification.

- Note that the loss function will be different from a regression-type problem (MSE):
  - For binary classification: binary cross entropy.
  - For multinomial classification: categorical cross entropy.

Key Takeaways

1. Artificial neural networks (ANNs) are powerful prediction tools. They have a fundamentally different philosophy from regression-related methods: the emphasis is on learning through repeated prediction and adjustment based on observed error.

2. While technically complex, the intuition behind ANNs is easy to grasp. With the basics in place, it is possible to start using them quite quickly.

3. Workflow is familiar from the general ML workflow.

4. ANNs have been found to work very well with text as input, i.e. transforming text into numbers and using it to predict or classify.

5. R's Keras package offers a very powerful but easy to use approach to implementing ANNs.

6. Economics research in this area is in its infancy, so there is huge scope to make a contribution!

Additional Resources

- A very simple overview of deep learning: https://towardsdatascience.com/a-gentle-introduction-to-deep-learning-part-1-introduction-43eb199b0b9.

- A nice collection of examples in R: https://blog.rstudio.com/2017/09/05/keras-for-r/.

- An example of text as input in R: https://tensorflow.rstudio.com/tutorials/beginners/basic-ml/tutorial_basic_text_classification/.
  - Note that it takes a different approach to preprocessing and ANN type and structure.

- Application of text as input, using FOMC minutes to predict economic variables: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3534914.