Neuro-Symbolic AI
Logic Tensor Networks
Personality Clustering

1st Semester of 2023-2024

Maria Cioclov maria-simona.cioclov@s.unibuc.ro

Alexandru Moise alexandru-ioan.moise@s.unibuc.ro

Annemarie Messner annemarie-beatrix.messner@s.unibuc.ro

Abstract

This paper analyses the use of **Logic Tensor Networks** in order to identify and create clusters that represent different types of personalities. Logic Tensor Networks is a framework that supports querying, learning and reasoning. Just by defining some logical constraints, we can let the neural network learn without the need to pass in the labels. There are many interesting ways of defining those types of constraints, which we will analyse in a bit.

1 Introduction

1.1 Using Neuro-Symbolic AI for a personality type clusterization

For years, people have devoted a lot of time and effort in order to create a proper thinking machine and a lot of researchers still continue to do so. Research in this particular field has enabled the creation of neural network. These neural networks trained by deep learning have paved the way for multiple breakthroughs, but contemporary deep learning models are limited in their ability to interpret and require huge amounts of data. These limitations have enabled the apparition of the Neuro-Symbolic Artificial Intelligence that uses symbolic reasoning along with the deep learning neural network. To tackle this subject, we tested the efficiency of Logic Tensor Networks on a dataset containing multiple people's answers for 60 personality questions, along with the results. For simplicity, we will train our LTN on 2000 entries of 3 different personalities: ISTJ (Logistician), ESFP (Entertainer), ENTP (Debater). This is in order to make sure the distances between the personalities can be correctly displayed in a 2-dimensional space.

1.2 Our contributions

- **Contributions - Maria:**
  - Researched Neuro-Symbolic AI
  - Studied Logical Tensor Networks mostly by exploring the LTN GitHub repositories
  - Explored different parameter combinations to determine a more effective configuration for our model
  - Searched for an appropriate dataset to use in our work
  - Initial approach to plotting; tests for new data; KMeans clustering
- **Contributions - Moise:**
  - Researched and implemented different clusterization types
  - Suggested methods that might be more suited in the current environment when clusterizing this type of data
  - Tested the network with multiple parameters and different datasets in order to find optimal models
  - Saved the models for future use
  - Wrote a big part of the code explanation
- **Contributions - Anne:**
  - Researched the use and research value of Neuro-Symbolic AI
  - Studied how Logical Tensor Networks work
  - Worked on understanding different types of LTN classification algorithms
  - Implemented data filtering and batches organization
  - Researched and tried to implement ways of calculating the accuracy

1.3 Approach summary

Each answer in the dataset has 5 possible values:

- -3: Fully disagree
- -2: Partially disagree
- -1: Slightly disagree
- 0: Neutral
- 1: Slightly agree
- 2: Partially agree
- 3: Fully agree

We start from the premise that people with similar personalities will answer each question in a similar fashion. The more different the answers, the higher the probability that they have different personalities. We will introduce the concept of "penalty": the more different the answer, the higher the penalty should be. A good way of defining the penalty on a question is by calculating the squared difference between 2 answers:

Example:

- Question: "You regularly make new friends"
- Jack's answer: 2 (partially agree)
- Tom's answer: 0 (neutral)
- Emily's answer: -1 (slightly disagree)

We can conclude that Tom and Emily are quite similar (penalty of 1), that Tom and Jack are not completely different (penalty of $2^2 = 4$), but Emily and Jack are quite far away (penalty of $3^2 = 9$).

One of the benefits of this penalty system is that it allows us to use the euclidean distance between the arrays of answers as a starting point when clustering: the smaller the distance, the higher the chance to be in the same cluster.

1.4 Motivation

We chose to approach the Neuro-Symbolic AI project because the idea of combining deep learning with logical reasoning intrigued us. We considered this project as an opportunity to understand and learn about the integration of symbolic reasoning and neural networks in developing future AI systems more capable, flexible and easy for humans to understand.

1.5 Similar research

Similar research into the use of Neuro Symbolic AI has been conducted by MIT-IBM Watson AI Lab along with researchers from MIT CSAIL, Harvard University and Google DeepMind (Yi* et al., 2020). Together they have developed a new, large-scale video reasoning dataset called, CLEVRER — CoLlision Events for Video REpresentation and Reasoning. CLEVRER was used to benchmark the performances of neural networks and Neuro-Symbolic reasoning while only using a fraction of the data usually required for deep learning systems. This has helped AI not only to understand casual relationships but also to apply common sense in order to solve problems.

1.6 Previous work

Previous work done on this subject consists of the examples and tutorials from the official Logic Tensor Network Github page (Serafini et al., 2022) and the LTN Torch documentation (Tommaso Carraro, 2022) Some of the examples were a bit outdated, but they let us see different approaches to clustering and managing of data.

1.7 What we learned so far and what we want to continue studying

- **Maria:** I learned how to combine neural networks and symbolic reasoning to create a Neuro-Symbolic model for clustering and how these two approaches can complement each other in solving complex problems. In the future, I would like to deepen my understanding of symbolic reasoning and logic to be able to come up with more complex axioms that can be used in solving more complex problems.
- **Moise:** I learned about a new way of implementing neural networks using a method that is easy to read and to understand. Because I am still unsure how the different results might be explained, I want to research how the network learns more in-depth, and I want to do a little bit of research on the reasoning part as well using Logic Tensor Networks.
- **Anne:** I learned about the short-comings of symbolic AI and deep learning in simulating the human thought process and decision making. I am glad I had the opportunity to study the way Neuro-Symbolic AI aims to overcome those shortcomings and implement a system that can make smart urgent decisions. In the future I would like to explore more future applications of the Neuro-Symbolic AI in extracting and processing information from images/video as I think this would play a major role in developing technologies such as self-driving cars.

2 Approach

- Link to our repository: https://github.com/MoiseAlexandru/LTN-personality-clusterization
- Software tools we used: Jupyter Notebook, TensorFlow, NumPy, Matplotlib, ltn, scikit-learn, Keras.

Neuro-symbolic Clustering: We build the classifier for our network. We are defining the sizes of the hidden layers and the activation function (we chose elu in this case). We used the softmax function so that the output nodes have a total probability of 1.

Defining logic functions: LTN introduces the term grounding, defining abstract human concepts using

functions and tensors. Since we are working with probabilities, there are many ways to define the logical operators. This happens because each logical operator acts like a function that takes two parameters, and there is no set way of computing that result. However, this is not a bad thing: we can play around, define them more loosely or more strictly. We will use the recommended ones (from the official documentation). We will also define an aggregator which we will use in order to compute the satisfiability value of our constraints.

Defining axioms: The beauty of Logic Tensor Networks is that we can define rules and constraints in a human readable way. Some rules of clustering would be:

- every element should be assigned to a cluster
- every cluster should contain at least one element

In addition to these 2 obvious rules, we need to define how to handle the relationships between the elements and clusters. We have more ways to do so, and we will discuss each of them now.

First approach: We use the euclidean distance to measure the similarity between two sets of answers. Next, we define the following rules (axioms), with x and y being two variables from the current batch:

- for each variable x, there exists a cluster that contains x
- for each cluster, there exists a variable x that is a part of the cluster
- for all variables x and y, the amount of similarity between x and y determines (implies) if the x and y are in the same cluster (x in cluster <=> y in cluster)

In the end, we need to return how much these conditions hold.

Second approach: We will still use the euclidean distance. However, this time we will use two more threshold distances that will help the network decide if two elements need to be in the same cluster or different clusters.

We want to define two limits for the distance between two elements:

- the first limit indicates that any two elements that have the distance under the threshold need to be in the same clusters
- the second limit indicates that any two elements that have the distance above the threshold need to be in different clusters.

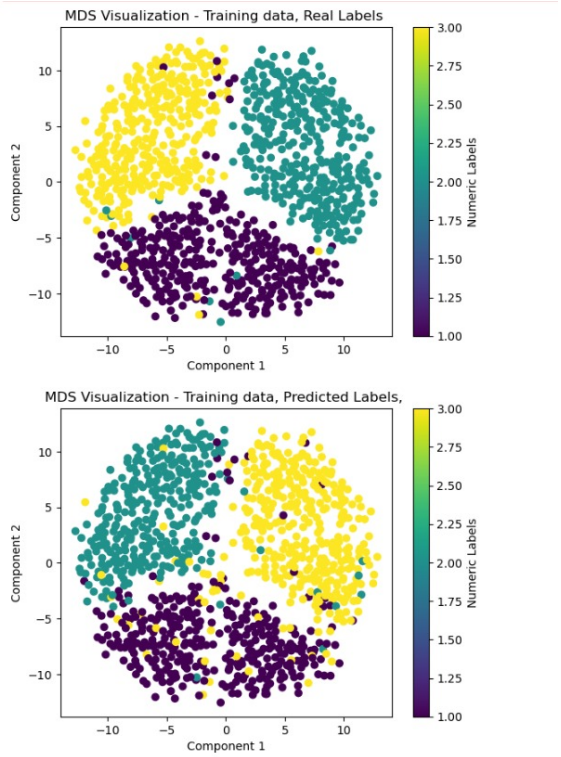

Figure 1: Results Approach 1 on training data

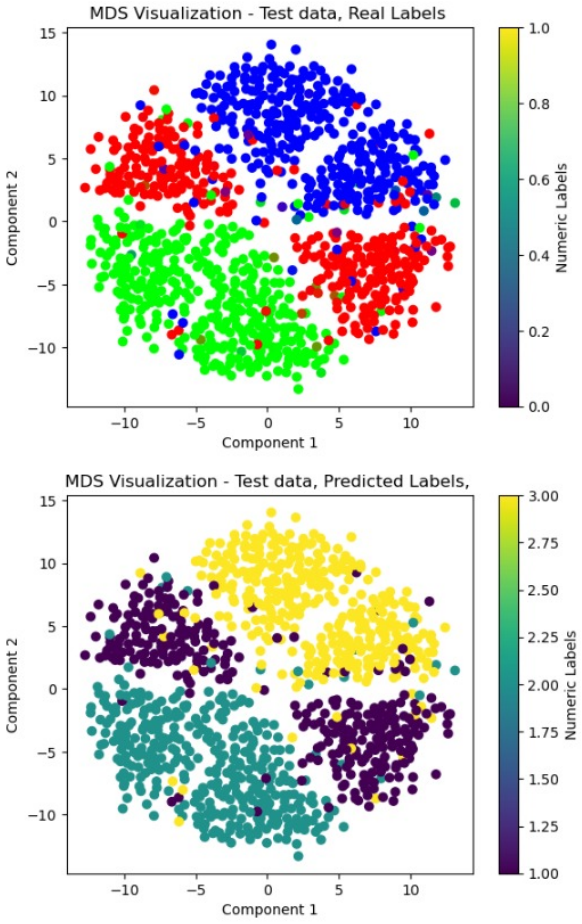

Figure 2: Results Approach 1 on new data

We have taken the data in such a way that the person-

alities are sufficiently separated. Thus, we can observe that there are gaps between clusters. If we manage to set a correct value for the gap, we should have a good clustering.

And, although the `distant_threshold` doesn't help us much, since we can sometimes have small gaps between elements from different clusters (10), and large distances between elements within the same cluster (18), it helps to set a higher value, close to the extreme (e.g., 20), to aid in orienting the clusters. It will practically define the extreme points, and from there, the `close_threshold` should fill in, gradually, towards the center.

Next, we define the following rules (axioms), with x and y being two variables from the current batch:

- for each variable x, there exists a cluster that contains x
- for each cluster, there exists a variable x that is a part of the cluster
- for each x, y and cluster, if the points are close to each other, they should belong to the same cluster
- for each x, y and cluster, if the points are far from each other, they should belong to different clusters

In the end, we need to return how much these conditions hold.

Third approach: This approach relies on us giving the network some examples of initial nodes that belong to each individual cluster. We can give him an example of an answer that is labeled as 'ISTJ', another that is labeled as 'ESPF', and another one that is labeled as 'ENTP'. This approach will ensure that the predicted clusters will have the same label as the original ones. However, if one of the examples is a special case / a more in-between personality, this might fail, since the distance to another personality might be smaller than ones of its own. Just for illustrating this approach quickly, we won't handpick those examples, but we will take the first matching entries we find in the dataset.

Next, we can declare the necessary axioms:

- the example entry 'istj' should be in the cluster ISTJ
- the example entry 'esfp' should be in the cluster ESFP
- the example entry 'entp' should be in the cluster ENTP

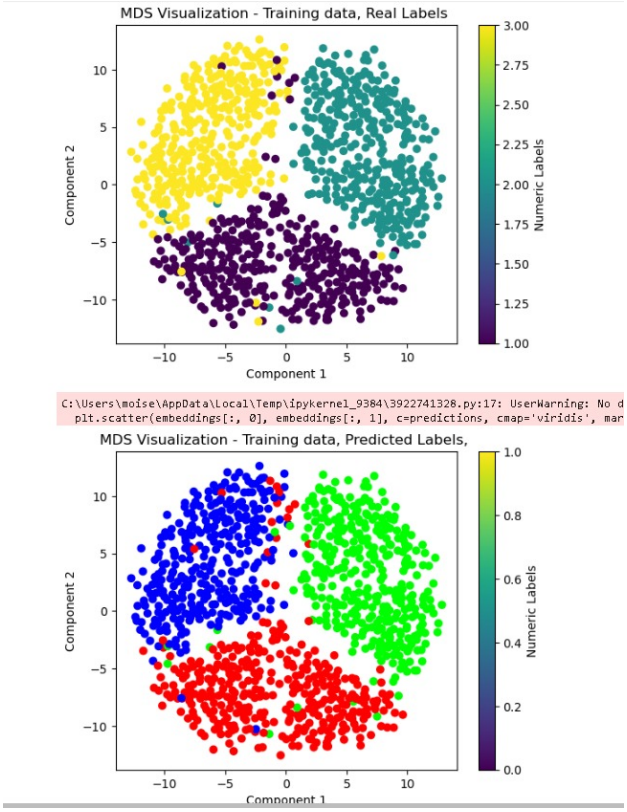

Figure 3: Results Approach 2 on training data

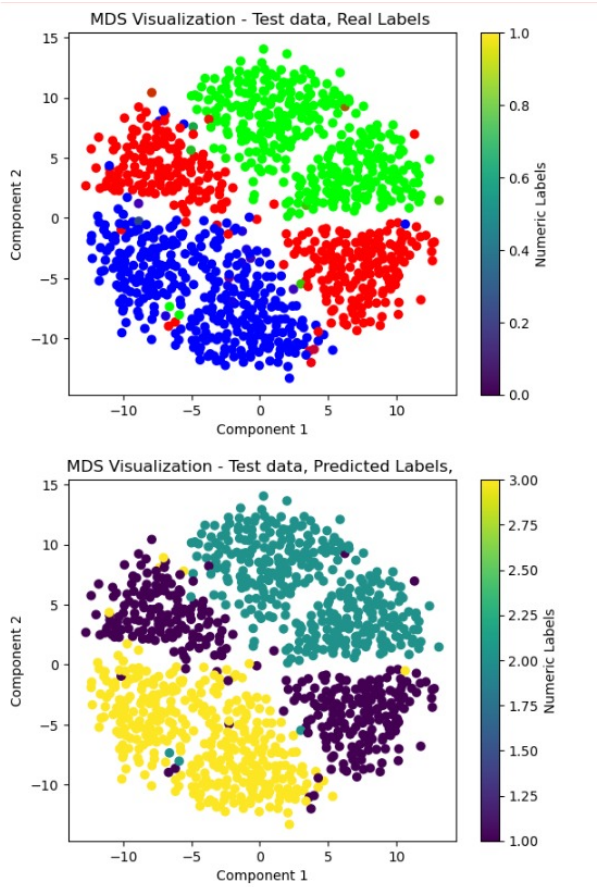

Figure 4: Results Approach 2 on new data

As in the 1st approach, each element should be in a cluster, and the similarity between personalities should determine the similarity between their assigned clusters.

Training: We need to define the learning rate, and start the training process. We played around a bit with the learning rate, and anything more than 0.01 is too high, and under 0.001 is a bit low. For our case, 0.001 looks like to be the sweet spot. For each step, we need to set the `p_exists` (parameter for how loose the axioms are), iterate through all the batches, then adjust the weights based on the loss function: $1 - \text{axioms}(\text{p\_exists}, \text{batch})$. The training processing took between 5 and 10 minutes, depending on the method used and the number of epochs.

Getting the results: From our test data, we test each input individually, retrieving the output data from the network. The value of each output node indicates the likelihood that the input is assigned to the corresponding cluster. Obviously, we will consider the node that has the highest probability as the result.

Checking the accuracy: We spent a bit of time deciding on how to define the accuracy, since the SAT value only indicates that clusters have been made, not if they are correctly corresponding to the actual labels. We decided that for each pair of people that are in the same predicted clusters, we will check if they have the same labels. Counting the number of the positive and the number of negative responses, we can get a percentage that would be our success rate, or the Accuracy.

Visualizing data: Accuracy does not tell everything. It also does not help us improve the network in any way. Therefore, we need a way to visualize the data, the clusters, and compare them to the original labels. We'll use a method that transforms arrays in a way that they can be plotted in a 2-dimensional environment.

Comparing the results with those provided by the traditional clustering model:

- Matching clusters in proportion of 0.976447892690373 - for the test data, using KMeans model
- Matching clusters in proportion of 0.9671478646713283 - for the test data, using the Neuro-Symbolic model

3 Limitations

1. Number of clusters. We started with the assumption that there are gaps between clusters. This means that the distance between two points that belong to different clusters is higher or equal to K. This happens to be true in the current scenario because the types of personalities we chose happen to be very different. But if we consider the case involving all 16 types of personalities, not only can't we visualize them in 2D, but the gaps could disappear all together. In this case, the second approach can't be used as the close threshold could also disappear. The first and third approach could still stand. The third approach could however be limited by the need of handpicking representative elements for each cluster.

2. Computing power. Being a Neuro Symbolic AI, time is needed to process the given data and this time keeps increasing as more axioms are added. As an example: in order to process data of size 1000 using the second approach, we require 5 minutes per 400 epochs. This is way we resumed to training and testing data only for the first 2000 to 60000 entries from the dataset.

3. Accuracy and Randomness. We have been require to run the same code many times, obtaining completely different results. The only difference that could cause this would be the randomness of the nodes during the initialization phase. Therefore we had to implement a way of saving the model used when reaching a satisfying result.

4. Understanding the Network. Even though Logic Tensor Networks have been designed to build a network that is easy to comprehend, debugging it requires a lot of effort. This is mainly due to the number of different results we reached by running the same program multiple times. We observed that LTN's can get stuck in a state by reaching a conclusion early on. Adjusting the learning speed could help solve this problem, but the initial randomization seems to be playing an essential role in the training process.

4 Conclusions and Future Work

We figured out pretty late that the randomization of the input data directly influenced the training. Saving a model with a satisfying performance earlier would have saved us a lot of time and helped us study its use on new data more.

In the future we could look further into how the input data affects the training and try comparing different sets of data in order to see how the predictions change.

Even though the topic of experimenting with this type of AI seemed new and complicated at first, we were intrigued about its potential use. We

dived into the theory behind it and the algorithm's implementation and, even though we found it a bit confusing at first, we enjoyed trying to train a model and switching different features in order to achieve a better accuracy.

We learned that there are more ways of implementing a reasoning machine than we knew so far and we are open to further studying different models of artificial intelligence and their particular uses, strengths and shortcomings.

For future projects at this course regarding the Logic Tensor Networks we suggest trying to implement other typed of classification algorithms such as binary classification or multi-class classification and see in which category does the Neuro-Symbolic AI perform better than other neural networks.

We wished we had more materials on this subject to work with. For this project we could only find the official documentation on Github which was little outdated and some interview recordings that were hard to follow.

References

Luciano Serafini, Ben Wagner, sbadredd, and mspranger. 2022. Logic tensor networks (ltn).

Tommaso Carraro. 2022. LTNtorch: PyTorch implementation of Logic Tensor Networks.

Kexin Yi*, Chuang Gan*, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. 2020. Clevrer: Collision events for video representation and reasoning. In *International Conference on Learning Representations*.