Activities Report

Introduction and Context

The goal of the project is to implement a deep neural network which is able to understand – given a specific set of forecasts – which forecast may be better.

Details about the dataset and the idea come from the ACE IARPA project available at this link: IARPA Good Judgement Project.

The dataset used by the neural network comes from the Good Judgement Project which is freely available at the following link: Good Judgement Project Dataset.

The project initial focus was implementing a deep neural network which was capable of predicting the outcome of a specific geopolitical event, but this was not possible in the domain of deep learning due to the lack of significant amount of questions (i.e., events) we had from the ACE IARPA repository.

The source code of the application can be found at following private Github Repository.

In the first month of the activity I had to study deep learning, so I covered the topics offered by the Coursera Specialization named “Deep Learning” taught by Professor Andrew Ng from the University of Stanford. The list of topics can be consulted at the following link.

In order to be able to compare two predictions, a siamese neural network Is used followed by a sequence of fully connected layers and an output layer composed by a tanh activation function.

The input to the neural network is a couple of forecasts related to the same question and the output is which of the forecasts was more accurate.

Topic Modeling

In order to give a meaningful representation to the questions the Latent Dirichlet Allocation (LDA) has been used, we can launch the LDA algorithm once in the ‘forecasting’ directory of the Github repository by executing ‘python gjnn/lda.py’. One of the most important parameters for the tuning is the number of topics, which has been set to 6 after different tests. LDA can be thought as an unsupervised process, so topics do not get names, but a set of words with their relative importance as a weight. Another parameter of tuning for the LDA process is the number of passes which can be found in the lda.py source code set to 20. In order to properly process the text, we merged the question main text and long description and removed all the stopwords, and meaningless (to our purpose) words such as for example “i.e.,” or “e.g.,”. Furthermore words have been processed with a technique called “stemization” in order to extract the stem for each word. Topic modeling contributed to the project by creating new features in the dataset, indeed for each question we had how much it was related to each topic.

An example output of the topic modeling phase was:

```
(0, '0.024*"outcom" + 0.019*"resolut" + 0.015*"open" + 0.013*"follow" + 0.013*"report" + 0.012*"met" + 0.012*"statu" +
0.012*"assum" + 0.0
12*"criteria" + 0.011*"quo" + 0.011*"reuter" + 0.011*"bbc" + 0.011*"sourc" + 0.011*"determin" + 0.011*"gjp" +
0.011*"credibl" + 0.011*"med
ia" + 0.011*"state" + 0.010*"ap" + 0.010*"countri"')
(1, '0.025*"question" + 0.024*"sourc" + 0.019*"report" + 0.015*"outcom" + 0.013*"may" + 0.013*"militari" +
0.012*"administr" + 0.012*"reut
er" + 0.012*"bbc" + 0.011*"elect" + 0.011*"refer" + 0.010*"news" + 0.009*"59" + 0.009*"outsid" + 0.009*"forc" +
0.009*"economist" + 0.009*
"onlin" + 0.008*"resolv" + 0.007*"use" + 0.007*"case"')
(2, '0.023*"question" + 0.017*"report" + 0.016*"sourc" + 0.015*"open" + 0.015*"outcom" + 0.014*"may" + 0.013*"credibl"
+ 0.010*"data" + 0.
010*"59" + 0.010*"administr" + 0.009*"2014" + 0.009*"case" + 0.008*"determin" + 0.008*"day" + 0.008*"refer" +
0.008*"use" + 0.008*"et" + 0
.007*"price" + 0.007*"mean" + 0.007*"23"')
(3, '0.032*"sourc" + 0.032*"report" + 0.027*"question" + 0.019*"may" + 0.018*"administr" + 0.018*"outcom" +
0.017*"news" + 0.016*"bbc" + 0
.016*"reuter" + 0.016*"economist" + 0.015*"59" + 0.015*"onlin" + 0.014*"offic" + 0.012*"resolv" + 0.011*"case" +
0.010*"expert" + 0.010*"l
eader" + 0.010*"matter" + 0.010*"subject" + 0.010*"one"')
(4, '0.032*"announc" + 0.028*"offici" + 0.020*"sourc" + 0.019*"report" + 0.017*"question" + 0.014*"must" +
0.013*"outcom" + 0.013*"govern"
+ 0.012*"may" + 0.012*"press" + 0.012*"59" + 0.011*"news" + 0.011*"reuter" + 0.011*"bbc" + 0.011*"administr" +
0.011*"economist" + 0.010*
"onlin" + 0.010*"make" + 0.007*"resolut" + 0.007*"resolv"')
(5, '0.032*"outcom" + 0.021*"resolut" + 0.020*"open" + 0.019*"determin" + 0.018*"offici" + 0.017*"criteria" +
0.017*"met" + 0.017*"gjp" +
0.016*"statu" + 0.016*"credibl" + 0.016*"media" + 0.016*"quo" + 0.016*"assum" + 0.015*"ap" + 0.015*"follow" +
0.015*"sourc" + 0.014*"announc" + 0.014*"report" + 0.013*"reuter" + 0.013*"bbc"')
```

Cost Function

The cost (or loss) function can be represented by a function belonging to R^3 . Its domain is given by (sigma, b-a) where sigma is the output of the tanh activation function contained in the output layer and b-a is the euclidean distance between the two forecasters predictions. The codomain is instead represented by the loss function output result.
Some of the desired features we needed can be summarized as follows:

- on the x axis we have sigma so the neural network output, this is bound in the interval [-1,1] since it is the output of a tanh activation function;

- on the y axis we have the distance (b - a), and here we are not using the absolute value so we could also have negative values. Also this value is bound in a limited interval, since this is just a difference between two euclidean distances;

- on the z axis we have the loss function ouput which should be characterized by the following properties:

- if sigma is = 1 and b-a is positively high (so the network thinks the prediction of
  A is better than B and the distance b-a is positively high) this means that actually a is a better prediction then the network loss should be "LOW"

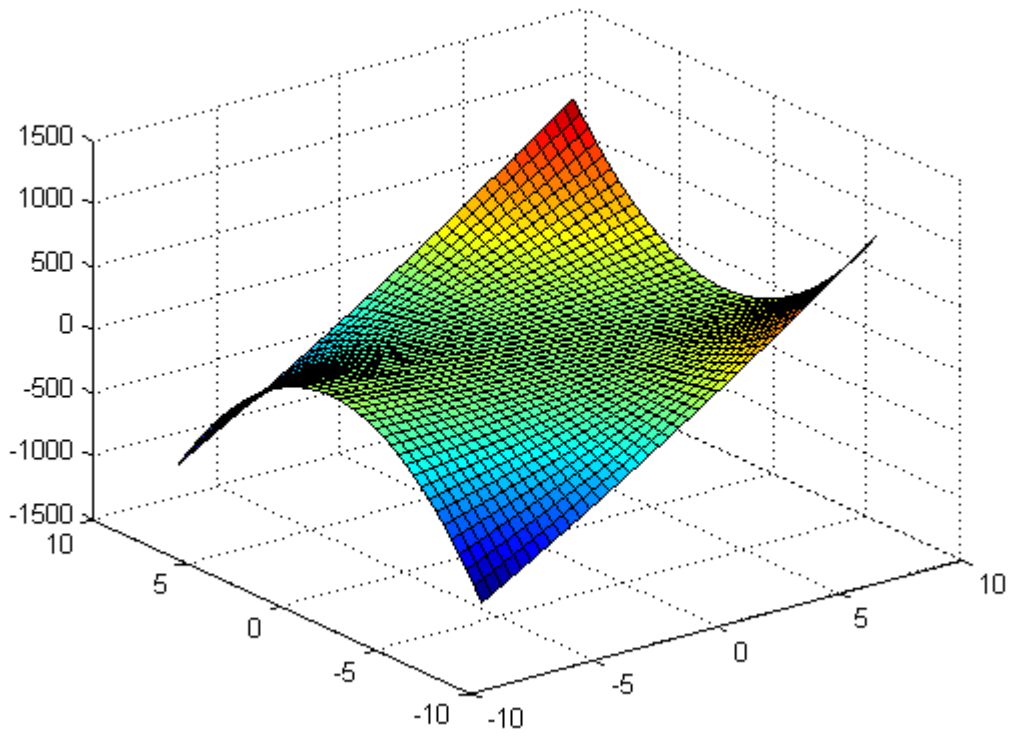
- if sigma is = -1 and b-a is negatively high (so the network thinks the preidction of B is better than A and the distance b-a is negatively high) this means that actually b is a better prediction then the network loss should be "LOW"
- if sigma is = 1 and b-a is negatively high (so the network thinks the prediction of

A is better than B but the distance b-a is negatively high) this means that actually b is a better prediction then the network loss should be "HIGH"

- if sigma is = 0 and b-a is positively high (so the network thinks the preidction of B is better than A but the distance b-a is positevely high) this means that actually b is a better prediction then the network loss should be "HIGH"

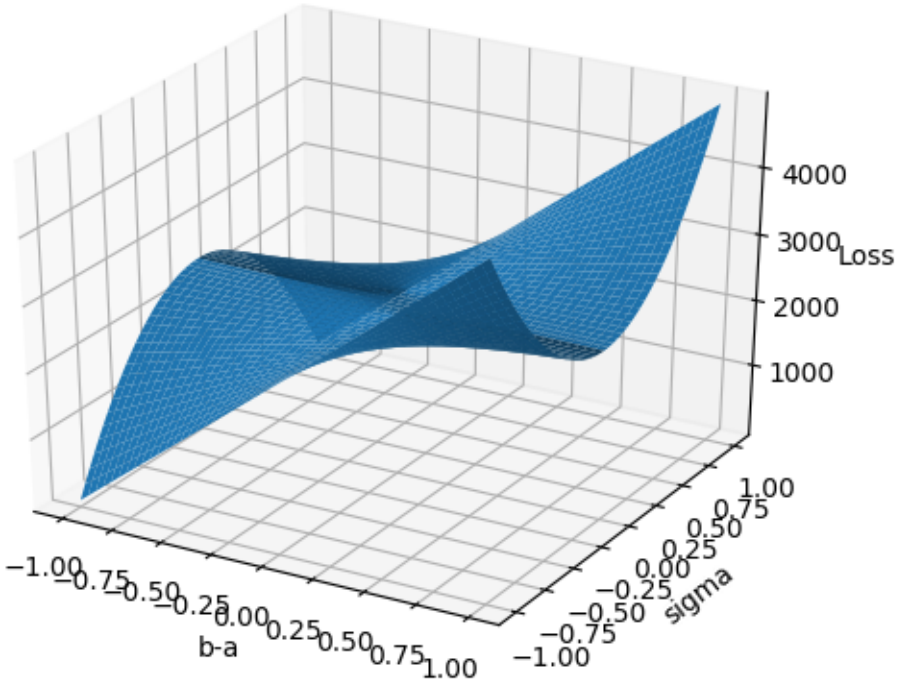
The starting function we found, which (with some adjustments) could satisfy our requirements is:

$$z = x^2 - y^2 + 2xy + 1$$



In order to properly shift the domain, and satisfy our requirements we had to apply some shifting and scaling, so the resulting function is:

$$z = \left(\left(((x-\alpha)\,\beta)^2 - ((y-\alpha)\,\beta)^2 + 2\,(x\beta)\,(y\beta)^2 + 1\right)\varphi\right) + \gamma$$



where alpha, beta, gamma and phi are constants used to shift and scale the function, a possible combination could be:

- alpha = 8
- beta = 500
- gamma = 2500
- phi = 10^5

For further documentation and testing (or manipulation) of the cost function the script called '*3d_plot.py*' can be used, contained in the Github repository mentioned above.