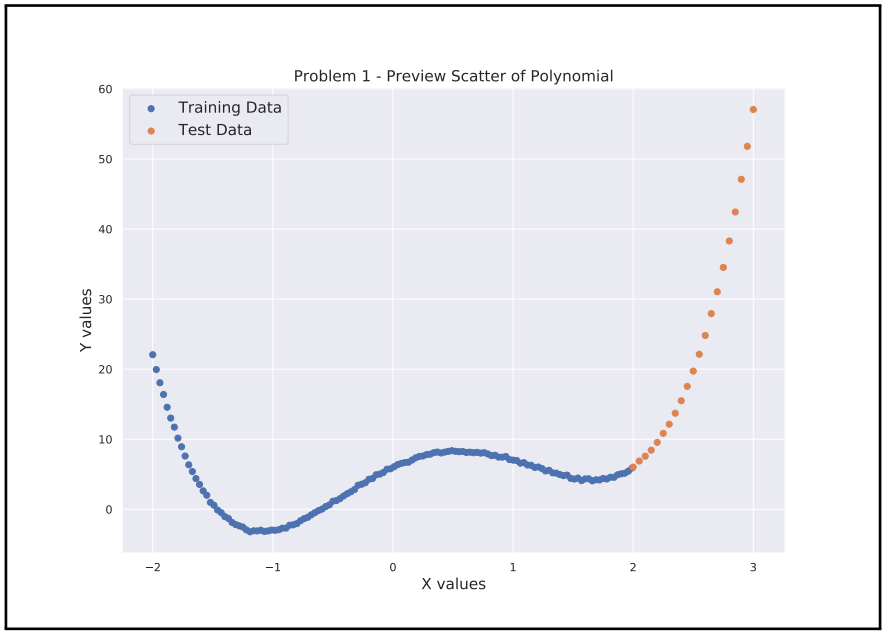
- Document: Final Evaluation
- Course: Artificial Intelligence, Feb - Jun 2020
- Student: Marcos Emmanuel González Laffitte
- Student's Matriculate Number: 311260331
- Professor: Esteban Hernández Vargas, PhD
- Masters in Mathematics
- IMATE-UNAM - Juriquilla, Qro, Mex

Instructions The project test consists of 5 problems/questions, for which a brief written report with pictures and brief explanation of the results should be provided. The student must discuss the report on 12 June 2020 on a ZOOM channel. The student will explain his/her results and will answer a question about the course slides. **Remarks** Provide a brief reasoning in the report why you choose certain algorithm. Please provide in your report full name and matriculate number. The oral examination will be a discussion of the project, tools and a general question. The codes you use for the project should be sent to esteban@im.unam.mx before 1:00 pm 12 June 2020.

Problem 1 Using the data set called problem1.csv (x_training, y_training), do the following :

a) Find the polynomial that fits the best the training data.

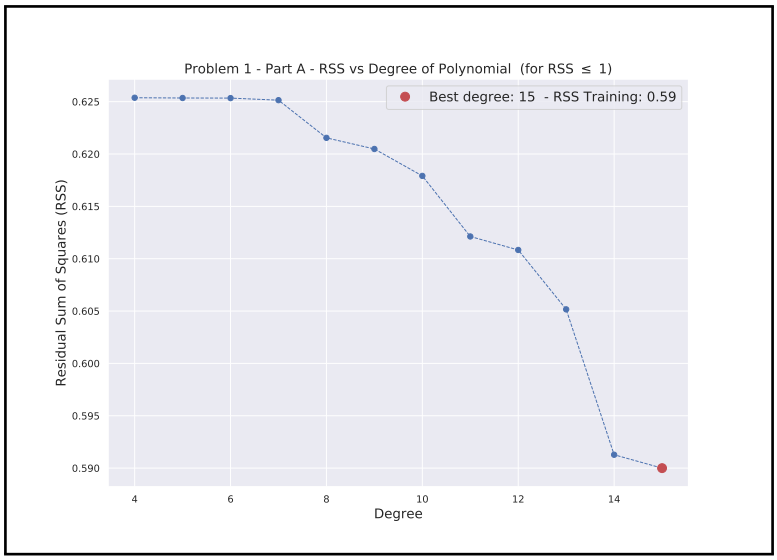
We first loaded the data from the problem1.csv file and made a preview of it. In the following figure we provide an scatter plot that shows the polynomial given by both the training set and test set found in such file.



Preview of polynomial in problem1.csv file.

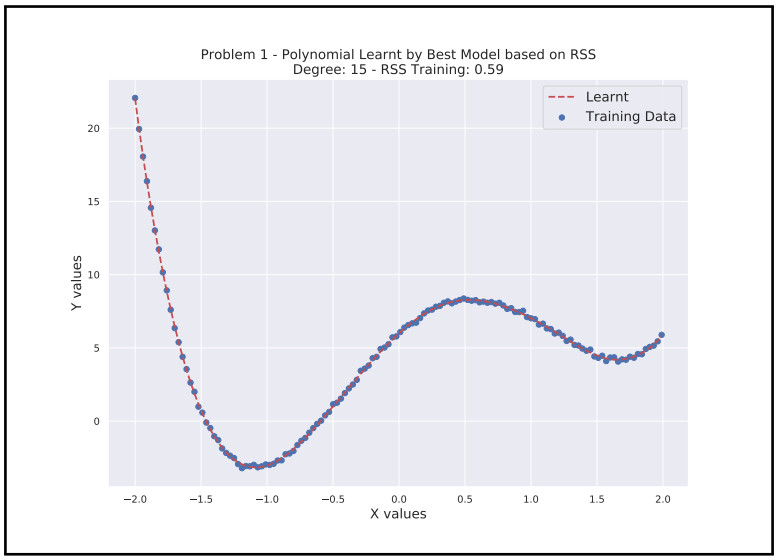
To find the best-fit polynomial for the training data as established by the exercise, we made use of numpy's polynomial model, varying the degree of the polynomial from 1 to 15 (though this can be predifined by the user in the supplementary script).

The next picture shows the relation between the RSS error measure and the degree of the respective polynomial for which each error was determined. For this first part of the problem we found that the best-fit polyomial was that of degree 15, though this changed when taking into acount the AIC measure for each trained model. Only the degrees associated with an RSS ≤ 1 are shown, given that the remaining degrees produced an RSS whose scale did not allow for the better RSS values to be appreciated.



Based on the RSS alone, the polynomial that best fits the training data is that with degree 15.

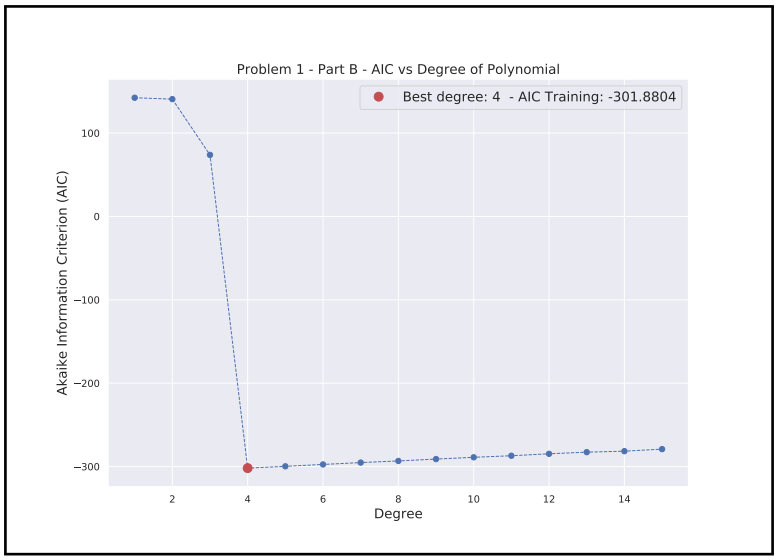
After this we plotted the function obtained when training numpy's polynomial with degree 15. The resulting function is shown bellow together with the training data points.



The learnt polynomial in red dotted lines and the training data set in blue circles.

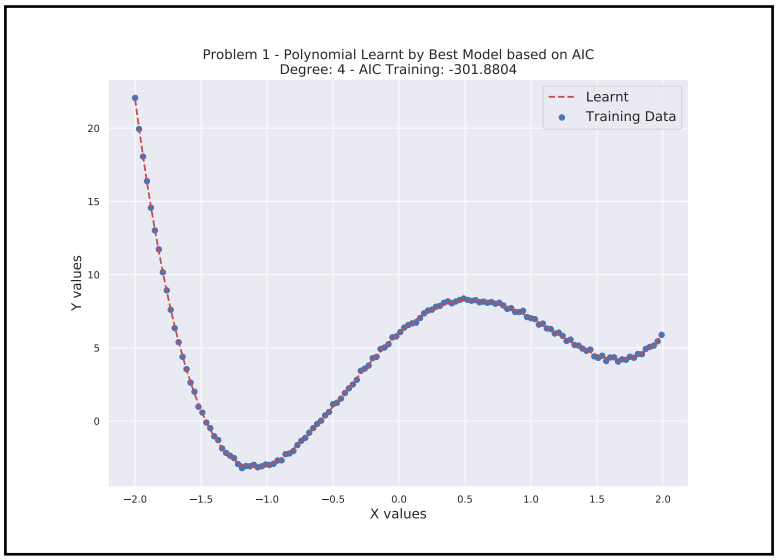
b) Using the AIC criteria, find the best polynomial that can fit the data.

When using the AIC criteria to determine the best-fit polynomial over the training set, we could notice the effect that a greater number of parameters has in said measure. This can be observed in the next figure, where the AIC attained a minimum value, therefore showing a better suited model, for the polynomial with degree 4 instead of that with degree 15.



Though the AIC is better for high degrees, it still grows according to the number of parameters.

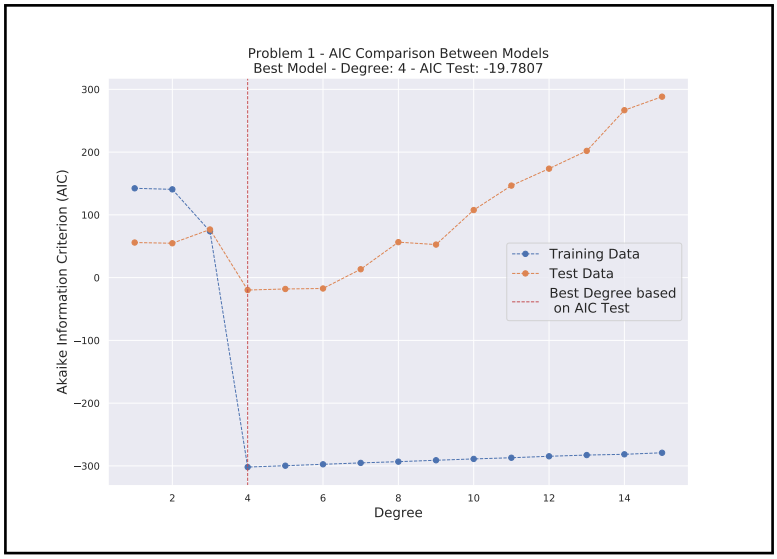
As in the last part, we also obtained the polynomial with best degree and compared it to the training data set in the file.



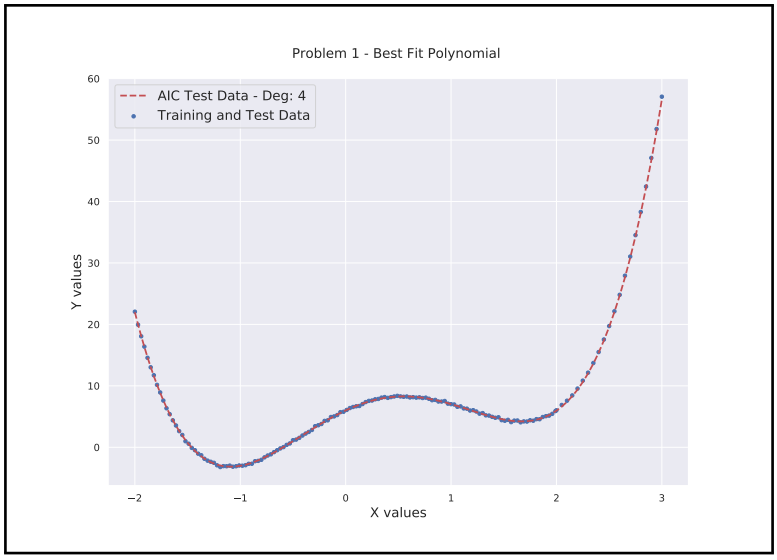
The polynomial with degree 4 plotted against the training data.

c) Cross validate the polynomial with the data set called problem1.csv (x_test, y_test).

Finally we obtained the AIC for each polynomial but this time associated to the test set. When doing so, we determined that the polynomial that best fits both the training set and test data is the one with degree 4, as seen below.



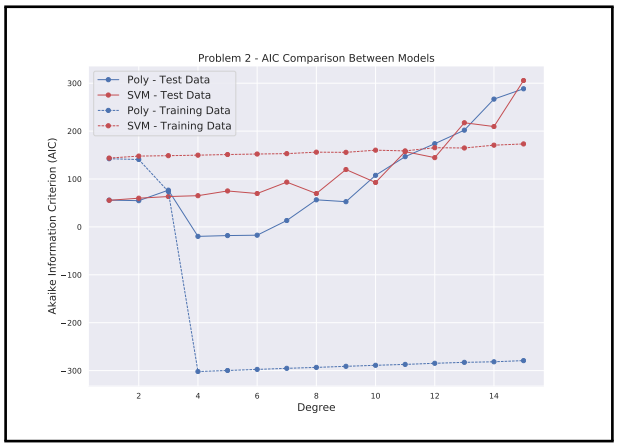
In both cases the polynomial with degree 4 is that which fits the data the best according to the AIC criteria, and it is also the one with the smallest degree between those that produce and AIC with the same order of magnitude (and sign).



The resulting polynomial compared to the complete data in the file. Here we show the one with degree 4, but omit to show the one with degree 15 because it will make impossible to appreciate the one with degree 4, due to the fact that it produces a function that does not match the scale of the data in the file, specially for the test data set.

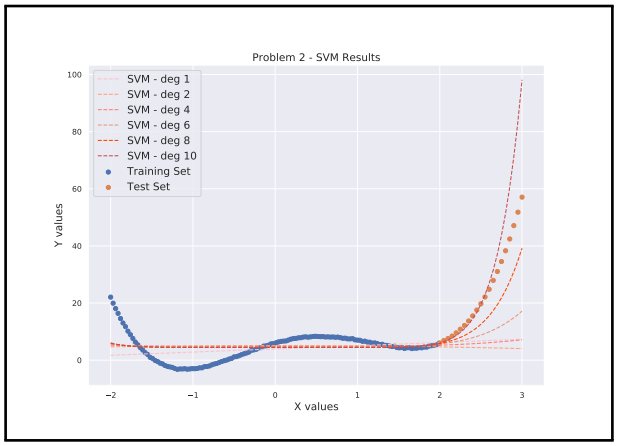
Problem 2 For the problem 1, it could be possible to develop a SVM to predict the test data. Using a polynomial kernel, Would it be possible to give a good prediction of the test data of problem 1? If not, give an explanation about it.

We implemented the SVR (Support Vector Machine for Regression) with polynomial kernel of the scikit-learn library of python, and compared it to the polynomials in the past problem. This object is also instantiated by defining a degree for its kernel, so we could match the AIC given by both models with the corresponding degree.



Polynomial with degree 4 still produces a better outcome than the SVM with polynomial kernel.

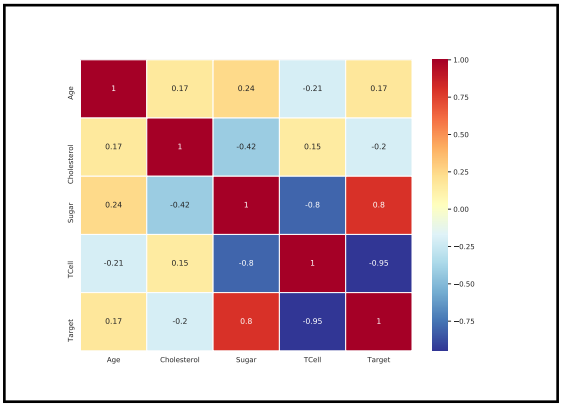
The not-so-good behavior of the SVM for the overall data set might be due to the fact that these models are better suited for high dimensional feature spaces, while in this case we are trying to reproduce a real function. Even so, we found that the SVM with kernel of degree 8 makes a close prediction of the test set. But at the same time, no degree value allowed the SVM to reproduce the training set adequately, plus any improvement that could be found around degree 8 is cancelled out in the AIC by a high number of parameters. Bellow we show how the polynomials grow closer to the test set depending on their degree.



In addition, no improvement was found when training the SVM's with both normalized and not normalized data sets, being the one shown here trained with not normalized data.

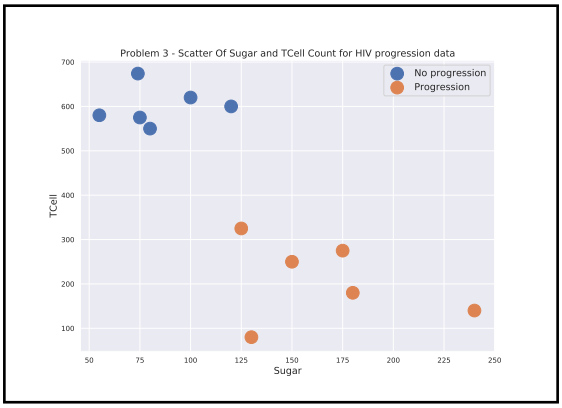
Problem 3 From a clinical trial, we have 12 patients with HIV infection. After treatment, the disease progressed in 6 patients (1) and in 6 patients the infection did not progress (0). Four measurements are taken in the 12 patients (Age, sugar levels, T cell levels and Cholesterol). Which measurement can be used as a marker to describe progression of the disease? Which will be the criteria to predict the progression? The data can be found in, problem3.csv (x_age, x_sugar, x_Tcell, x_cholesterol, outcome). Arrange the data and briefly explain your results. The variable "y" (target) is a vector of 0 and 1 to represent the progression.

We loaded the data into numpy arrays holding integer data types, one per measurement including the target variable. Then we obtained a correlation matrix for the six vectors. In the following picture we display this correlation matrix in form of a heatmap.



We found a high negative correlation between both the target variable and the TCell count, and the sugar measurement with the TCell count. It can also be seen that there is a positive correlation between the target and the sugar measurement.

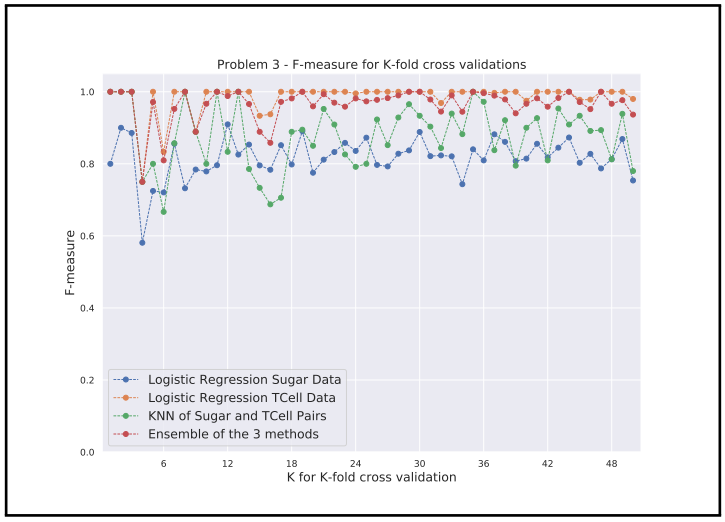
Based on these results, we decided to use both the sugar measurement and the TCell count as the main criteria for the prediction of the HIV progression. In the next image we show the clusttering of the 12 data points, with coordinates in said features. Furthermore, we developed a cross validation study of the prediction obtained with four machine learning methods using this data: a logistic regression over sugar data, a logistic regression over TCell count, a K nearest neighbor for both variables, and finally the ensemble prediction of these three approaches.



For our study of K-fold cross validation, we produced K random partitions of the sugar and TCell pairs associated to each patient for each given K value. Here we selected 60% of the pairs as traning set and the other 40% as test set. From each partition we isolated the sugar data and the TCell data, which later let us train the corresponding logistic regression. Then we fitted the KNN model to the mentioned training pairs, producing from these methods three predictions for the same expected output of the test data. After this, we averaged the three predictions (not any score but the actual predictions) into one sole prediction, obtaining therefore a *consensus* or ensembled prediction for this same partition.

To evaluate such predictions we made use of the F-score (also known as F-measure or F1-score), whose implementation is already available in the sklearn library. This is calculated as 2 times the fraction of the product between the precision and the recall over their sum. Here by precision it is understood the proportion of true positives w.r.t the overall positive results, while recall refers to the true positives divided by the sum of true positives and false negatives. Then, this measure alows one to assess the preformance of classifiers, ranging from 0 to 1, and having a value close to 1 if the classifier is performing correctly.

In the next picture we show the average F-score for each method obtained for multiple K-fold validation experiments, where K goes from 1 to 50. For a K between 1 and 20, it can be seen that the predictions may vary in accuracy. But for greater values of K, such variations appear to be reduced, allowing these methods to obtain more stable F-score values.



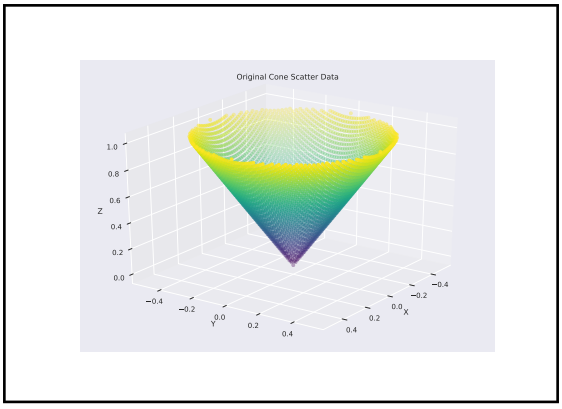
Result of multiple K-fold validation experiments.

Here we can see that the F-score for the logistic regression over the sugar data oscilates around a value of 0.8, while the KNN wraps around a value of 0.9. On the other hand, the logistic regression for the TCell count produces an almost perfect prediction, while the ensemble prediction hangs just below of said optimal result. Finally, even though one might be inclined to select the logistic regression over TCell data as a unique method for the prediction of the HIV progression, it should be taken into acount that it is possible for these ideal values to implicitly present some overfitting, in particular for bigger data sets. Therefore, we can conclude that the ensemble prediction of these methods is the most reliable option for the prediction, given the variables selected here.

Problem 4 Using a multilayer perceptron with Keras produce a cone with the dimension of your election. Remark: In other words, use the equation of a cone, then generate artificial data to generate X, Y and Z. Then you use data of X, Y Z to train a Neural Network, and then form the shape of a cone but produce by the Neural Network. In your report you should mention the equation of the cone you selected, and the figures generate by the equation and by your neural network.

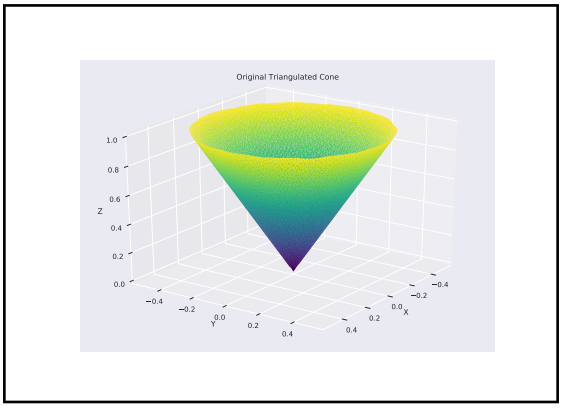
The cartesian equation for a cone centered at the origin is given by $x^2 + y^2 = (R/h)^2 z^2$, where x and y are coordinates of a point at height z, and where the constants R and h depict the radius R reached at height h. Here we determined $R = 0.5$ and $h = 1$ for our cone, and therefore $-0.5 \leq x, y \leq 0.5$ under the constraint $\sqrt{x^2 + y^2} \leq 0.5$, together with $0 \leq z \leq 1$.

We generated scatter x and y data for this surface by means of the numpy library. Next we selected those x, y paris that meet the constraints and produced for them a respective z. This data is show in the follwing image, generated with the 3D tools of the matplotlib library.



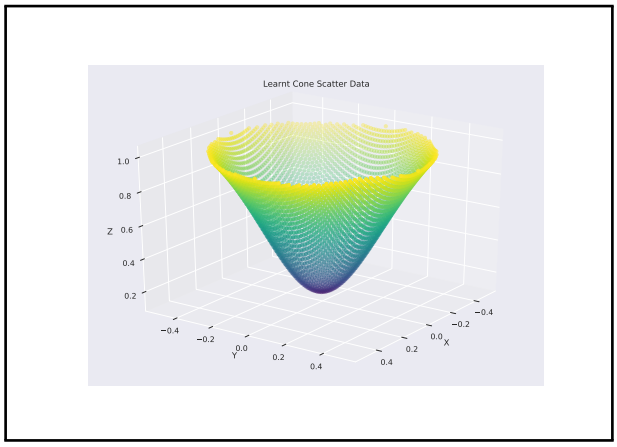
3D Scatter plot of the selected cone.

With this library, one can produce an almost smooth representation of the scatter data points, by using the triangulation commands included in it. In the following we display a triangulation of our cone. In addition, both figures can be rotated and manipulated by the user when running the script.



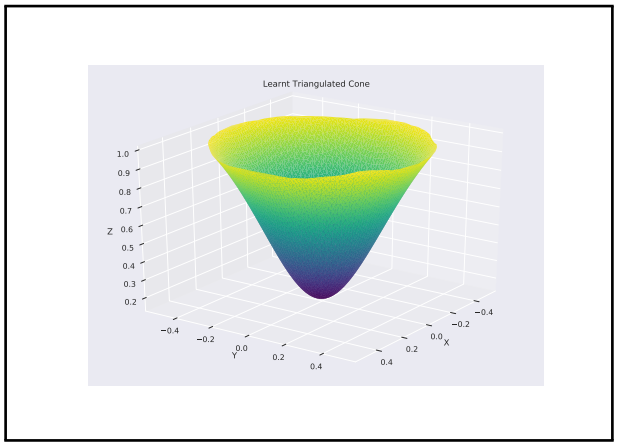
3D Triangulation plot of the selected cone.

After this we built our neural network, using for this purpose the sequential model of the keras library included at the same time under the tensorflow packages. The input for this network are arrays of length 2 (x and y), and the output is the variable z. It has 3 (dense) hidden layers with 50 neurons each, all of them working with the sigmoid activation function, while the output layer consists of just one unit running a linear activation function. To wrap up everything, this network is compiled using the Adam optimizer (a variation of the stochastic gradient descent) and it learns by minimizing the mean square error function. In order to train this model we determined a batch size of 6 over 100 epochs. All of these parameters were selected by a trial and error loop until the network surpassed visible error plateaus or local minima in the learning curve, aspects that can be reproduced while running the script, given the continuous output of information of the network while trained with the parameter verbose set to 1.



3D Scatter plot of the cone learnt by the neural network.

Next the network is trained with a 60% random sample of the data. The learnt cone can be retrived by asking the network for a prediction over the whole original set of x and y pairs. In this page we show both the scatter data points and a triangulation of them, that make up the *cone* learnt by the network. These images can also be manipulated and rotated when running the scripts.



3D Triangulation plot of the leanrt cone.

Python Info

[1] python 3.7
[2] anaconda 2020.02
[3] scipy 1.4.1
[4] seaborn 0.10.0
[5] numpy 1.18.1
[6] pandas 1.0.1
[7] matplotlib 3.1.3
[8] scikit-learn 0.22.1
[9] keras 2.3.1
[10] tensorflow 2.2.0