Predicting Goldbach partitions

Ehud.Plaksin, Idan. Holander , Gilad. Fuchs

November 2018

Abstract

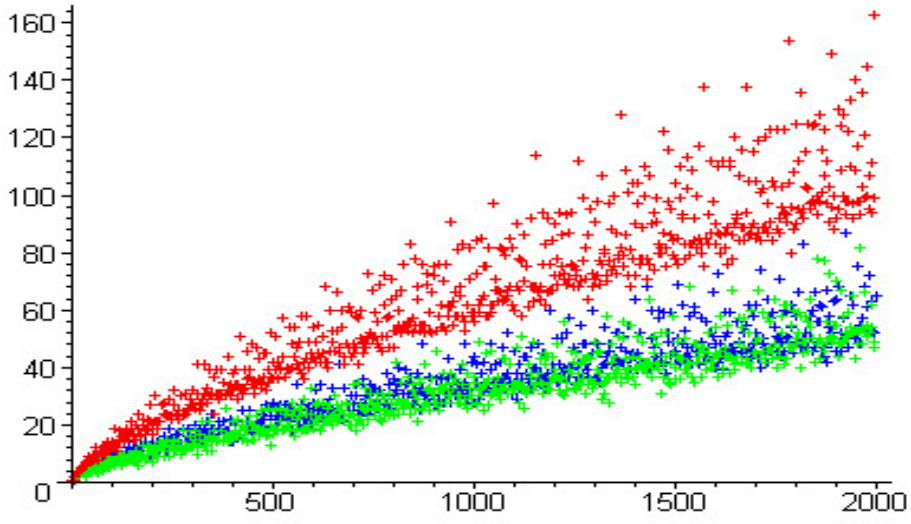
The "Goldbach Conjecture" is one of the oldest and best-known open problem in Number theory. It states that every even integer greater than 2 can be expressed as a sum of two primes. The conjecture was shown to be true for even number up to 4*1018. We built various NN models , that an gave an approximation of the Goldbach function. Where the Goldbach function is defined as: (g(E)) is defined for all even integers $E > 2$ to be the number of different ways, In which E can be expressed as the sum of two primes. For Example $(g(6) = 1)$ since 6 can be expressed as the sum of 2 primes in one way $6 = 3 + 3$.

1 Introduction

After few letters exchanged by the german mathematician Christian Goldbach and Leonhard Euler, the Strong Goldbach Conjecture was sketched: "every even integer greater than 4 can be expressed as the sum of 2 primes". The weaker conjecture states that every odd number greater than 7 can be expressed as the sum of 3 primes, The weak conjecture is a corollary of the strong conjecture as if n-3 is a sum of 2 primes, the n is a sum of 3 primes. The Weak Goldbach Conjecture was proven at 2013 but the strong one remains unproven. Over the years some heuristics were suggested to approximate the Goldbach function.

$$\sum_{m=3}^{n/2} \frac{1}{ln(m)} \frac{1}{ln(n-m)} \approx \frac{n}{2ln(n)^2}$$

since the primes number theorem asserts that an integer m selected at random has roughly a 1ln(m) chance of being prime. then we some the products of all possible pairs. Since the quantity goes to infinity as n increases we expect that every large even integer has even many partitions. This approximation is not very accurate, There are other ones , some of them like Hardys and Little wood even require Prime Factorization which is not very efficient. In our work we tested 3 NN regression models for approximation, a simple LR model, a MLP model (which produced the best result), and an LSTM model.



2 Data set

The data set was gathered using prime Data set from Primes.utm.edu And a python script using it to find $g(E)$ for $E < lim$ and E even. Where lim for now is 32,000. (can be increased later.)

3 Features

We extracted the following features from the data set: Binary representation of the number , Bin(e). For our size of data set 16 bits were enough. Residues in mod 3, 5, 7, 11 which we encoded using "one hot encoding" method.

4 Project Description

We achieved the best results using an MLP,
on a data set of size 32676 (x = 4 to x = 65356) , where we used 70% of the data for training And 30% of the data for testing. we achieved: MSE: 698.6267 RMSE: 26.4315
Our model has 3 hidden layers (16, 8, 16) with RELU activation function.
The weights and biases were randomly initialized (drawn randomly from normal distribution).
We used AdamOptimizer to minimize our MSE.
We used a learning rate of = 0.01 We trained the model for 400 epcos. Adding l2 regularization did not have very big effect on performance.
We used batch gradient descent with batch size of 128.
Example run:
Epoch: 0001 cost= 7292.484375000
Epoch: 0031 cost= 824.398071289, Epoch: 0091 cost= 1036.494018555
Epoch: 0151 cost= 737.299926758, Epoch: 0211 cost= 857.945495605
Epoch: 0271 cost= 770.408081055, Epoch: 0331 cost= 737.079528809
Epoch: 0391 cost= 732.185852051 (using "train set").
MSE: 698.6267 RMSE: 26.4315 (using "test set").
As can be seen we did not suffer from overfitting

5 Previous Attempts

5.1 LSTM

Previous work was done with using LSTM to predict stock market prices, as a stock price is a function of time, that grows in the macro level and can have agressive changes of magnitude. Which is similar to the Goldbach function. We took a working code from "Thusan Ganegedara" tutorial for predicting stock market prices. We predicted G(e) by observing previous G(e-1), G(e-2)..
We used an unrolled version of LSTM we use 50 continuous time steps for a single optimization step.
A batch size of 50, dropout of 0.2, we used 3 layers of LSTM with 20,20,15 nodes respectively. (originally was 200,200,150 but took to long to run)
We used 90% of the data for training and 10% of the data for testing.
Without normalizing the data the model preformed very poorly.
Therefore we tried a new approach, We created a new model, an LSTM layer and then connected our trustworthy MLP.
We fed the LSTM with the features we used previously.
we used $batch_size$ of 100, we looked 10 patritios backward, and predicted 15 into the future, we ran for 50 epocs.
our results:
step 0, training loss 27223.5
step 10, training loss 170.759, step 30, training loss 77.0031
step 40, training loss 56.2985, step 49, training loss 46.6744
MSE: 103442.9600 RMSE: 321.6255
as can be seen we have an overfitting issue, the thing is we took the last 20% as our test data.
And since our data is sequantial and G(e) grows as e grows, our model isnt familiar with the higher partitions.
Since the data is sequantial shuffeling like we did before is not possible, normalization is also not a good solution.
We want the partitions and not their magnitude. we tried regularization, and diffrent hyperparameter tuning with not sucsess.
The model also sometimes falls into "bad minimum" and predicts the same for all input. (like he found an average value with good MSE).

5.2 Linear Regression

Hyper Parameters used:
we used a learning rate of = 0.01
we trained the model for 1000 epchos.
when adding regularization we used = 0.01 adding l2 regularization did not have very big effect on performance.
We used batch gradient descent with batch size of 2048.
On same dataset as our MLP we achieved
MSE: 5405.7327 RMSE: 73.5237

6 Related Work

Goldbach Function approximation was done by Avigail Stekel [1], Merav Chkroun and Amos Azaria.
Articles for prediction stock prices by Thusan Ganegedara[3] and Lilian Weng[2]

7 Resources

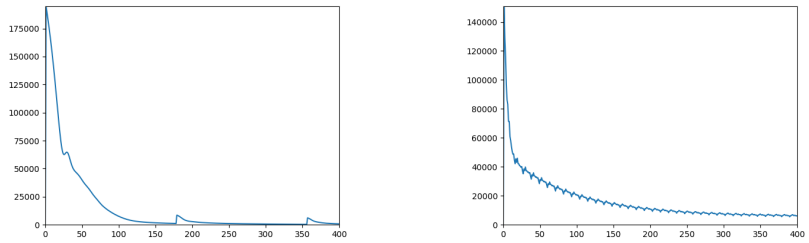


Figure 1: Loss: MLP on the left LR on the right

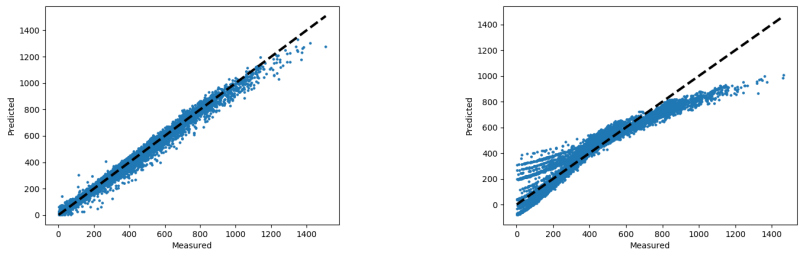


Figure 2: Predicted as function of measured: on left MLP on the right LR

comet.png

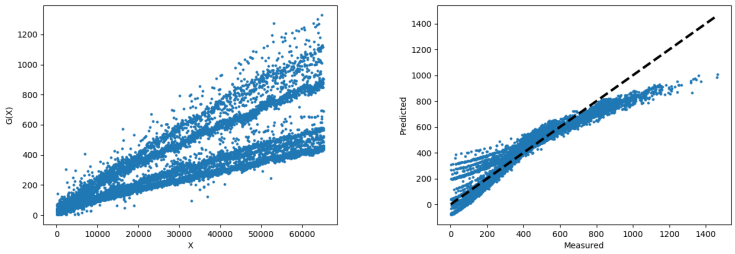


Figure 3: G(e): on left MLP on the right LR

comet.png

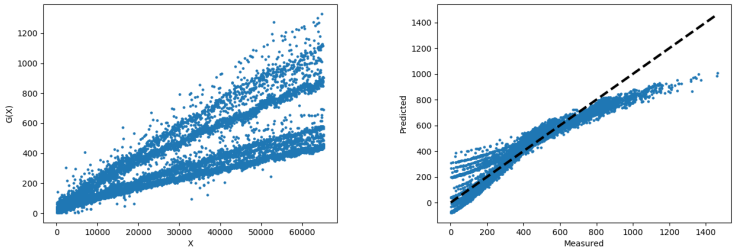


Figure 4: G(e): on left MLP on the right LR

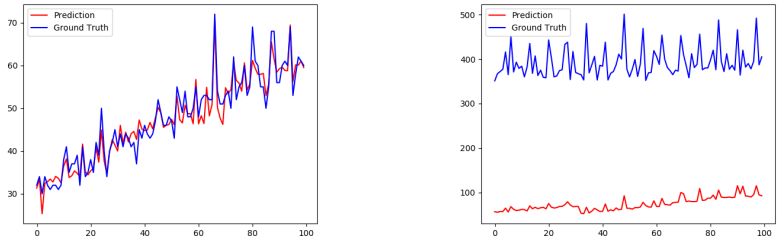


Figure 5: on the left LSTM on train data, on the right on test data, as can be seen LSTM under predicts test data

References

[1] Avigail Stekel. Goldbach's function approximation using deep learning.

[2] Lilian Weng. predict stock prices using lstm.

[3] Tushar Ganegedara. Lstm python stock market.