STYLE TRANSFER WITH CNN

Casado Herraez, Daniel
shreyjoshi2004@gmail.com

Gatri, Wathek
shreyjoshi2004@gmail.com

Gimenez Bolinches, Andreu
shreyjoshi2004@gmail.com

March 25th, 2019

ABSTRACT

Abstract

***K*eywords** Artificial Neural Network · Curve Fitting · Dataset · Polynomial · Regression

Contents

1 Introduction

1.1 Style transfer

What is style transfer

2 Theoretical Background

2.1 Artificial Neural Network Architecture

What is VGG19

2.2 Training An Artificial Neural Network

How do we train new styles?

2.2.1 Cost function

Which cost function do we choose and why?

2.3 Purpose of the work (Engineering Goal?)

2.4 Virtual Materials

There were no physical materials in this project, but a multitude of virtual materials were used, which include

- Python 3.7.0 programming software

3 Method

3.1 Implementation

There were three programs developed during the duration of this project, which were for the purposes of

- Data Collection for the ANN model
- Data Collection for the Polynomial model
- Comparison Program which Graphs and a trained ANN and polynomial along with Training and Test data

1. Construct a three layer ANN of architecture $[1 - 2k - 1]$ with randomly initialized parameters/Select a polynomial order k.
2. Select a function, $f(x)$ from the above list from which to generate training data
3. Construct a new function $g(x) := f(x) + m$ where m denotes a random 10% error based on the normal distribution
4. Generate a training dataset of 15 sample points from $g(x)$
5. Train the ANN model/polynomial model with the training dataset
6. Once the model has been trained, obtain data on
   - Time Elapsed Training
   - Mean-Squared-Error
   - Average Percent Accuracy
   - Number of Epochs Elapsed (ANN only)

   where accuracies are measure with respect to the function $f(x)$.

7. Save the aforementioned data to an external text file
8. Perform 30 repetitions of steps 2-6 for each function
9. Perform one repetition of steps 1-7 for each $k \in \{1, 2, 3, ..., 10\}$
10. Summarize resulting data (take averages)

4 Data

The cumulative data acquired can be summarized with the following graphs:

5 Results

5.1 General Project Conclusions

In this project the plausibility of using Artificial Neural Networks (ANNs) as a regression model was tested. It was found that three-layer ANNs usually took longer than polynomial regression and had roughly similar accuracy, however, their rate of overfitting was incredibly lower and their usability was remarkably higher than polynomials. It was also observed that they were much less reliant on initial conditions, unlike polynomials which were very sensitive to their degree, and for this reason, ANNs are so much more resilient than their ANN counterparts. It was observed that in a three layer network, the optimum number of neurons for speed was around 12, and although there was an improvement of accuracy as the number of neurons increased, there was no big improvement after 14 neurons It is theorized that using deeper neural networks will greatly improve performance due to more layers of complexity, allowing the network to model more complicated datasets. This might introduce new problems, however, such as overfitting, but these can be solved with regularization, altering the activation function, and other techniques. It is intended to continue development of the ANN model and perform testing on multi-dimensional datasets to investigate its scalability. The full potential of ANNs as a curve fitting model was not revealed, but rather previewed in this project. It is conjectured that it will however be revealed upon further development and testing upon multivariate datasets with deeper networks, as this is where we can no longer apply standard methods with confidence.

5.2 Implications Of Data

A Two Sample T-test was conducted on the lists of the different Mean-Squared-Errors for the ANN and polynomial models for each individual class of functions. The average p value over these functional classes was computed to be 0.0370, meaning the data was statistically significant.

The data itself strongly implies a number of results:

The pair of activation function and loss minimization method that yielded the highest accuracy (lowest MSE) was hyperbolic tangent + LBFGS.

Testing different network architectures with hyperbolic tangent activation function + LBFGS yielded the result that 17 neurons tended to be optimal for curve fitting 2D datasets around the complexity of those used in this project.

From qualitative analysis of the graphs produced comparing the predictions of the models, it was found that ANNs were remarkably better for extrapolation than polynomials.

It was also seen from qualitative analysis that ANNs succumbed less to overfitting and hence were more usable, even when the number of neurons in the hidden layer was ridiculously large.

For these reasons, the researcher concludes that Artificial Neural Networks are superior to polynomials for curve fitting.

Unlike the polynomial graphs, there exists variation in the ANN graphs as parameters are initialized randomly (local minima)

5.3 Real-World Applications

5.4 Error Analysis

The following are some choices that may have affected the results:

- There were five functions used for testing the models. Using more functions to test the two models would have caused more accurate results since they would model a much wider variety of graph shapes.
- The choice of functions. Sufficiently high degree polynomials fit every function with a high degree of accuracy (> 90%) except for the asymptotic Function 2 due to the fact that polynomials have poor asymptotic properties. The representation of asymptotic function used while testing may not accurately reflect the distribution of data in the real world which models an asymptotic pattern, and for that reason the data could have been skewed. This problem can be solved in the future by increasing the diversity of the functions used.
- The constant choice of η, the learning rate. For different architectures, the optimum learning rates can be different and therefore the data might have been skewed towards the architectures for which $\eta = 6.5$ is the optimum learning rate. This issue can be solved in the future by incorporating a random hyperparameter search algorithm for optimizing the learning rate based on the architecture.