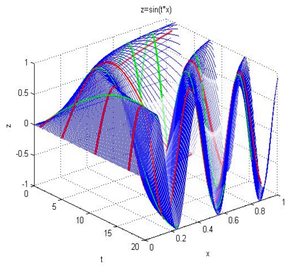
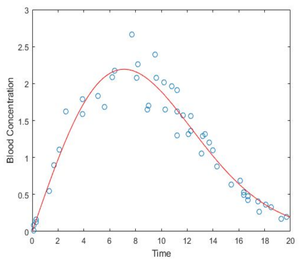
Introduction and Background

Curve Fitting is the construction of a function to estimate the relationship between variables in a dataset. As a form of statistical interpretation, it holds a wide plethora of applications in any field of study in which we observe and analyze numerical data.



A graph of a model fitting to data given by sinusoidal curves to produce the 3D image of an electromagnetic wave. [Courtesy of Mathematics Stack Exchange]



A graph of raw data (blue) fit by a curve fitting model (red). [Courtesy of MathWorks, Inc.]

Polynomial Regression

A common method of curve fitting is **Polynomial Regression**, which is where we determine the coefficients of a polynomial of a pre-specified degree to minimize a cost function, with respect to a particular dataset.

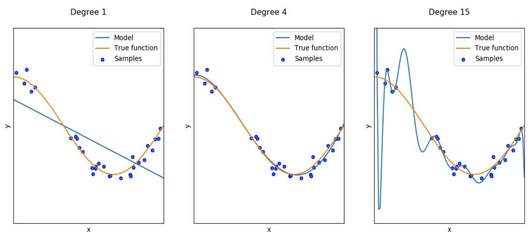
Polynomial regression is universally used due to the many nice properties of polynomials including but not limited to:

- Trivial integration and differentiation
- Existence of iterative root approximation techniques (Newton-Raphson),
- Vast amount of mathematical literature on polynomials.

The most obvious process by which we may reduce the cost of a polynomial is by constructing it such that the curve that characterizes it passes through each datapoint in its training dataset. These polynomials have a special name: **Lagrange's Interpolating Polynomials**, and can be defined as

$$P(x) = \sum_{i=1}^{n} y_i \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_i}{x_i - x_j}$$

Although they may have perfect accuracy on training data, these polynomials are unfeasible to use as they overfit to an exorbitant extent.



Polynomial models of degrees 1, 4, & 15 providing an underfit (left) good fit (center), and overfit (right).

[Courtesy of SciKit-learn]

Polynomial models as techniques for regression are in general highly reliant on the selection of a balanced polynomial degree. This means that they are difficult to use, are poor predictors of multivariate correlations and thus have low scalability.

Engineering Goal

Engineering Goal: The engineering goal in this project was to develop a versatile algorithm for curve fitting with exceeds traditional methods (polynomial regression) in accuracy and scalability with an almost negligible loss in speed.

Criteria:

- Very High Scalability
- Flexibility between Accuracy and Speed
- Insensitivity to Noise: Avoidance of Overfitting

The Proposed Method

Artificial Neural Networks (ANNs) are machine learning frameworks that possess some desirable properties which resolve the design criteria:

- Hierarchical structures, yielding straightforward scalability
- ANNs train in an iterative process, allowing a speed-accuracy tradeoff
- There exist regularization methods for ANNs to reduce overfitting

All graphics not cited on the board were produced by the student researcher

Methodology

Function Selection → Dataset Generation → Parameter Selection → ANN Model Fitting → Polynomial Model Fitting → Data Collection → Analysis & Evaluation

Phase 1: Function Selection

One of 10 distinct classes of functions is selected to be the basis of comparison between the polynomial and neural network models. To increase the scientific rigor of the testing process:

- The 10 functions were deliberately chosen to model common correlations in data acquired from the real world.
- A relative 15% error based on the gaussian (normal) distribution is added to simulate the inevitable observational error.

Phase 2: Dataset Generation

A dataset of 20 training data points is generated in a range of *[0, 6]* (based on the function in phase 1) to be fed into the ANN and polynomial models as training data.

Phase 3: Parameter Selection

Model parameters are chosen for both models. For polynomial regression this means selecting the degree of the polynomial. For ANNs this means selecting one of

- 3 different activation functions - Hyperbolic Tangent, Rectified Linear Unit (ReLU), and Logistic).
- 12 different neuron architectures, *[1]-[2k]-[1]* for *k* in *[1, 10]*.
- 2 different cost minimization methods (Stochastic Gradient Descent and Limited-Memory BFGS)

Phase 4: ANN Model Fitting

The ANN model is fit to the data in the generated training dataset from phase 2 with parameters from phase 3. The prediction of the ANN model is then plotted over the aforementioned interval *[0, 6]*.

To improve experimental accuracy, the ANN model was fit 50 times with the same set of parameters to the same class of function. The final data from that set of conditions is considered to be the average over those 50 trials.

The right column shows the mechanics of a neural network and its simplest training algorithm, Stochastic Gradient Descent (SGD).

Phase 5: Polynomial Model Fitting

The polynomial regression model is fit to the dataset, with the order of the polynomial having prespecified at the parameter selection phase. The predictions of the polynomial model are then plotted in the same graph as the ANN model over the predetermined interval *[0, 6]*.

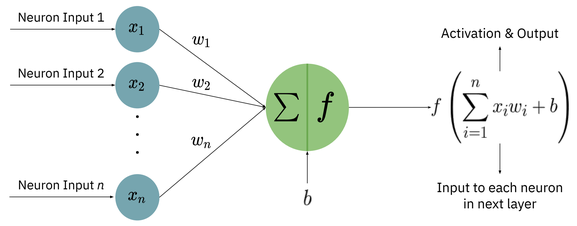
Phase 6: Data Collection

Data is collected on two levels, speed and accuracy.

- Speed is measured in amount of time elapsed during training and plotting. This is so that we can factor in both training time and computational complexity of the models.
- Accuracy is measured by the Mean-Squared-Error (MSE).

The training dataset as well as the pure function are also plotted and compared qualitatively to the ANN and polynomial model predictions.

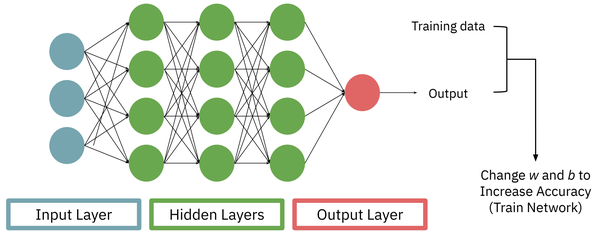
The Artificial Neuron



A diagram illustrating the mechanics of an artificial neuron. The inputs to the neuron are the outputs of the prior layer.

The diagram below exemplifies the hierarchical, and thus scalable structure of neural networks. Each neuron works by the principles depicted by the diagram above.

A Network of Artificial Neurons



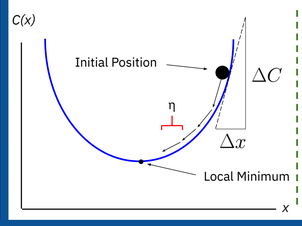
A diagram illustrating an ANN of architecture [3-4-4-4-1].

Sufficient Accuracy On Training Data? → Stop Training → Use ANN on Test Data

Training With Stochastic Gradient Descent (SGD)

The **Mean-Squared-Error** cost function: $C(w,b) \propto \sum_x ||y(x) - a(x)||^2$

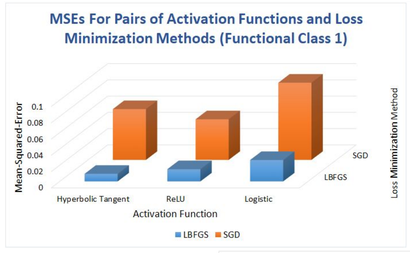
- *C(w, b)* is the cost of the network.
- *a(x)* and *y(x)* are the actual and desired outputs, respectively, for an input *x*.



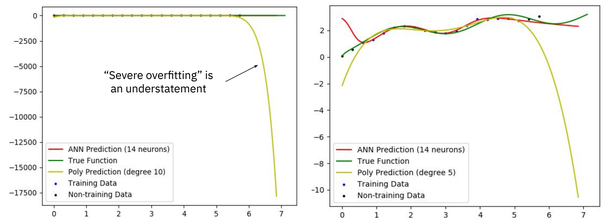
$$\Delta C \approx \frac{\partial C}{\partial x}\Delta x \rightarrow \Delta x := -\eta\left(\frac{\partial C}{\partial x}\right) \rightarrow \Delta C \approx -\eta\left(\frac{\partial C}{\partial x}\right)^2$$

$$\therefore \Delta C < 0 \quad \forall \quad (\eta > 0)$$

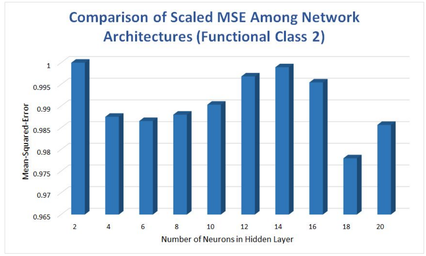
Data and Results



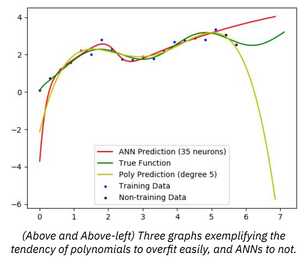
- All 6 pairs of activations functions and loss minimization methods were tested.
- It was found that the Limited-Memory BFGS + tanh activation function was optimal.



- A peculiar pattern emerged upon graphing MSE vs. #neurons in hidden layer
- MSE was often low at ~7 neurons, high at ~13 neurons, and then low at ~17 neurons.
- The only exception to this trend was functional class 9.



- ANNs and polynomials had comparative accuracy regarding interpolation only when a good polynomial degree was chosen.
- Selecting a polynomial degree off by a couple numbers usually resulted in horrid overfitting.
- ANNs, however, displayed no such ridiculous sensitivity to their initial parameters.



(Above and Above-left) Three graphs exemplifying the tendency of polynomials to overfit easily, and ANNs to not.

Data Evaluation and Result Analysis

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.

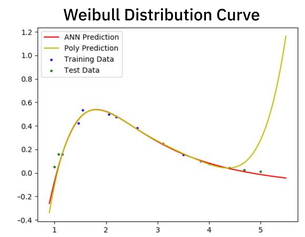
Real-World Application

The ANN model was tested on real-world datasets including, but not limited to:

- Crank angle vs. displacement of a load of a deep drawing press.
- Compressive strength of concrete given age and manufacturing details
- Data resembling Weibull Distributions (common probability distributions)

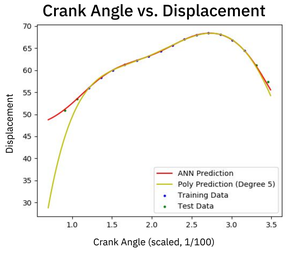
The tests demonstrated that ANNs are applicable to real world situations.

Weibull Distribution Curve

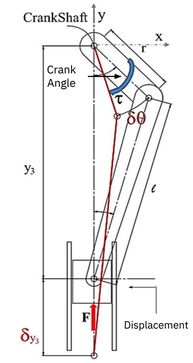


(Left) ANN and polynomial (degree 5) predictions after training upon data resembling a Weibull Distribution.

Crank Angle vs. Displacement



(Right) ANN and polynomial predictions after training upon data of crank angle vs. displacement of a deep drawing press.



(Above) A simplified diagram of the mechanics of a deep drawing press. [Courtesy of Lale Canon Dulger of the Izmur University of Economics]

Conclusion

In this project, the researcher designed, constructed, and evaluated an algorithm to function as an alternative to polynomial regression for fast curve fitting. Neural networks were used as the basis for the algorithm as they possess a set of inherent mechanics that resolved the design criteria, which was scalability, flexibility, and insensitivity to noise.

The researcher rigorously tested the network with a plethora of different activation functions, architectures and loss minimization methods upon 10 types of datasets generated by different classes of functions *f: R → R* and 4 real-world datasets. In addition, a program was made to curve fitting with ANNs and compare ANN and polynomial regression any dataset.

A two sample t-test was run on the MSEs and average percent accuracies for the ANN and polynomial models, which yielded a *p* value of 0.0370, (rounded to 3 significant figures) meaning the implications of data are statistically significant.

The data itself strongly implied that ANNs have:

- A considerably lower rate of overfitting than polynomials
- Modestly higher accuracy than polynomials
- Lower sensitivity to initial parameters, and therefore higher usability

At the cost of a negligible loss in speed, suggesting ANNs as a preferable framework to polynomials for curve fitting.

Future Research

Future research is intended to follow the following rough chronological sequence:

- Training ANNs to predict the optimal polynomial degree to fit a dataset (Low model complexity).
- Further testing and development of ease of use for multivariate data.
- RNNs or CNNs for analysis of 2D and 3D temporal or spatial data.