28 March, 2015
Yesterday our project team became a finalist in the 2015 International Science and Engineering Fair (ISEF). We We will be in Pittsburgh, PA from May 11-15th to compete in the competition. This blog will incorporate commits, project progress, goals, and any potential setbacks in our project process, and will be updated as frequently as we hold team meetings. During today's meeting, we reorganized our functional neural network repository.We also made a rough plan for the next month and a half and set goals for our project deadline.

Goals we determined during today's meeting:

- Adjust the mathematics behind the functional neural network to reduce the algorithm's computational complexity.
- Create a Functional Neural Network library in C++
- Make several applications of the Functional Neural Network using datasets with discrete (input data is an n-d vector) and continuous datapoints (input data is a function).
- Create a distributable program that allows users to train and use functional neural networks from any dataset and customize properties like interpolation algorithm and size of the network. This idea is based off normal neural network software.

6 April, 2015

We met up at the University of Utah math building and began work on reducing the computational cost of the Functional Neural Network feed-forward algorithm. This is the algorithm that determines the output of the network.

We set up the C++ project in visual studio and began a github design log. We began the construction of our math helper class that currently contains an integration method and will incorporate future math concepts like linear regression.

7 April, 2015
We successfully finished rearranging the computational steps to significantly reduce the computational tax of the feed-forward algorithm. This is documented in our paper as the fast feed-forward algorithm.

8 April, 2015

We met up at the University of Utah math building and began work on reducing the computational cost of the Functional Neural Network backpropagation algorithm. This algorithm determines the output of the network. We came up with first steps like removing a few computations from outside of nested sums, yet we are still not satisfied. One idea are we are looking at is using caching to catch values and use them later.

10 April, 2015
Full theory behind the fast back-propagation algoirthm

We successfully reduced the back-propagation algorithm's complexity and represented it mathematically. We are in the process of genreating pseudocode that will help us write the actual code of the algorithm.

14 April, 2015

We added the first parts of the c++ implementation, including a simple feedforward algorithm and the weight surface for the functional neural network.

20 April, 2015

Created two interpolation algorithms (Linear interpolation and Polynomial Interpolation) and finished up feed-forward algorithms. These will be useful to manipulate discrete datasets into continuous approximations. We are currently in the process of implementing spline interpolation which will have the accuracy of polynomial interpolation while minimizing computation. We debugged the feed-forward algorithm and tested it on a small dataset.

25 April, 2015

Finished up spline interpolation and created data logging for testing. Spline interpolation was tested, debugged, and plotted a testing interpolated line in Mathematica, which closely matched the source dataset. The data logging was added to keep a record of all the neural network properties during the training session, such as weight-surface values and the weight-gradients. This will allow us to observe the changes these values experience during testing and debugging.

30 April, 2015

Creation of the rough back-propagation algorithm. We implemented a rough version of the backpropagation algorithm. This included using the data caching theory and the pseudocode we devised earlier.

3 May, 2015

Creation of the datapoint, dataset, trainer and experimenter classes. The purpose of these new classes are to give an abstract value of the datasets that will be used in the program (the datapoint contains a testing and training function as data members) as well as methods with which to train and test the classes.

8 May, 2015

Finished up the error back-propagation method. We were able to test with Laplace transform data sets and get back positive results. Before the fair we plan to continue tweaking our algorithm to optimize the training and error checking.