Discovering Optimal Feature Values with Genetic Algorithms: Concrete Composition

William O'Brien
Columbia University
School of Engineering and Applied Science
Applied Mathematics

***Abstract*—In this paper, a technique for finding an optimal composition of concrete is built through the use of XGBoost and a genetic algorithm. A model for predicting concrete performance was generated on top of 1030 coupons tested for compressive strength courtesy of Yeh (1998) [1]. A genetic algorithm library [2] was implemented and tested on functions with known solutions, and the algorithm was set to search the predictive model for the mixture proportions which would maximize the compressive strength of a concrete mix. The XGBoost model's predicted optimal output was compared against a support vector regression (SVR) and a multilayer perceptron (MLP) model and revealed consistency in certain predicted elements of the mixture. Further research would explore the use of a deep learning model to obtain better extrapolations than a tree-based model.**

I. Introduction

In the construction industry, high-performance concrete (HPC) is an area of research which looks at incorporating unique combinations of ingredients to create strong compositions of concrete. Conventional concrete consists of water, cement, and fine and coarse aggregates. HPC incorporates fly ash, blast furnace slag, and a superplasticizer to produce new mixes [1]. The performance of an HPC mix relies on several variables. This makes finding strong compositions an expensive and time-consuming process, relying on trial and error to discover better recipes. Through predictive modeling, the exploration process can be optimized. Using concrete coupons obtained through experimentation, compressive strength measurements can be gained for different mixtures in the data. Predictions on compressive strength can then be used as a heuristic for a search algorithm to obtain high-performing mixes. When relationships between variables in a system are unknown, a statistical approximation of the system is required [3]. In high-dimensional, non-convex optimization problems, a function can be mapped to data and that function can be searched for global maxima or minima. In problems where computational complexity is less a constraint and a global optima is desired, a genetic algorithm is a suitable approach.

Once a predictive model is trained, a genetic algorithm can make a heuristic-based search over the input space to find the model's optimal outputs. A genetic algorithm (GA) is a stochastic, heuristic-based search algorithm. Genetic algorithms are often used in discrete search spaces, such as with the traveling salesman problem, [4]. The GA library [2] used in this paper, however, was built to search over continuous spaces rather than discrete ones. Among other applications, this approach is used in additive manufacturing to find optimal print parameters [5], in travel-time prediction [6], and in control systems [7].

The prediction of values given high-dimensional inputs is the first part of the problem. Machine learning excels at modeling complex, underlying interactions between variables. In this case, given enough HPC mixture data and the concrete compressive strength (MPa, megapascals), the interactions between components can be explored with a regressive ML model. Once the feature space is accurately modeled, the genetic algorithm can search the resultant high-dimensional, non-linear regressive model for the concrete recipe that maximizes concrete compressive strength.

II. State of the Art

Non-convex problems arise in many mathematical functions. Methods of numerical approximation are often required to find optimal values in high dimensional spaces. Gradient-based methods, although fast, tend to fall into local optima. For certain problems, such as in deep learning, gradient methods are appropriate. An algorithm called stochastic gradient descent (SGD) is used to find the sets of weights that minimize predicted loss relative to training data. SGD is a fast but imprecise optimization method which tends to land in local minima. In deep learning, the method works well enough that a slower, global solution is not required. However, for the cases where a local max/min is insufficient, several methods exist to find global optimums. Different types of optimization problems require unique solutions, so an algorithm should be chosen based on its suitability to the problem. Gradient-based solutions such as SGD are fast because of its directional use of the function's derivative, but it does not guarantee global convergence and relies on the function's differentiability.

Simulated annealing (SA) is a probabilistic, heuristic-based search algorithm often used in discrete search spaces, and works by probabilistically moving to a neighboring state (ie a parameter configuration) according to a decided heuristic. SA has been used to solve problems in computational biology, such as with phylogeny reconstruction [8].

The Nelder-Mead method is another heuristic based technique that relies on the use of n points arranged as a simplex, moving the points according to the objective function until the point changes plateau. The advantage of Nelder-Mead lies in its ability to converge to non-stationary points, although modern improvements are known that improve on the method [9].

Differential evolution (DE) works similarly to the genetic algorithm in that it finds optimal solutions by iteratively improving possible solutions according to a quality metric. DE is gradient-free but utilizes a target and unit vector to take advantage of directional [10]. This allows for fast convergence at the cost of poor exploration when compared to genetic algorithms. For this problem, a genetic algorithm is able to converge relatively quickly to near-global solutions and does well at narrowing in on the global solution with a greater number of iterations.

For the modeling of the search space, several regressive modeling techniques exist to map non-convex, non-linear, high dimensional space. Currently, XGBoost is the state of the art for regressive inference on tabular data [11] and is employed for the main analysis. Deep learning methods also perform well mapping functions to non-convex, non-linear spaces and is also analyzed briefly. Finally, support vector regression (SVR) is a linear regressor that takes non-linear data and maps it to linear space. This method is also used for comparison, although it is not the focus of analysis.

III. Methodology

The data used to build the models comes courtesy of Yeh (1998) [1] who published experimental data of 1030 HPC data points and the respective compressive strength measurements. The data captures 7 main components that generate an HPC mix, including Cement, Blast Furnace Slag, Fly Ash, Water, Superplasticizer, Coarse Aggregate, and Fine Aggregate all measured in units kg in a m^3 mixture. The age at which the HPC's compressive strength was measured after its composition is also measured in days. The age of the mixture also serves as a contributing factor to performance. The characteristics of the data are explained in Table I.

XGBoost tends to perform well on tabular data problems, and it is highly performant on many ML tasks. It gained notoriety by being the winning solution for almost every Kaggle competition for several years after it was released in 2019. The method is good at interpolating predictions in the range of seen data and is well suited to fit the data.

XGBoost is an ensemble, supervised-learning model that uses gradients over an objective function to model data and handle the weaknesses of previously ensembled decision trees. Decision trees evaluate a series of if-else questions about input data to produce a prediction, and the algorithm ensembles many by taking a weighted average over its predictions. The unique element of XGBoost comes from the way it computes the gradients of the trees it builds in parallel. This allows for the generation of many gradient-informed decision-trees, resulting in fast, high-performance [11].

After hyperparameter tuning using a 5-fold cross validation, a model was trained on the whole dataset with those parameters to capture more information from the data. The model was then run through the genetic algorithm. From the XGBoost model, the feature importance can be extracted to see which components contribute the most to the compressive

TABLE I
COMPONENT DATA

	Cement (kg/m^3)	Furnace Slag (kg/m^3)	Fly Ash (kg/m^3)	Water (kg/m^3)	Superplasticizer (kg/m^3)	Coarse Aggregate (kg/m^3)	Fine Aggregate (kg/m^3)	Age (days)	Compressive Strength (MPa)
count	1030	1030	1030	1030	1030	1030	1030	1030	1030
mean	281.17	73.9	54.19	181.57	6.2	972.92	773.58	45.66	35.82
std	104.51	86.28	64.0	21.36	5.97	77.75	80.18	63.17	16.71
min	102.0	0.0	0.0	121.75	0.0	801.0	594.0	1.0	2.33
25%	192.38	0.0	0.0	164.9	0.0	932.0	730.95	7.0	23.71
50%	272.9	22.0	0.0	185.0	6.35	968.0	779.51	28.0	34.44
75%	350.0	142.95	118.27	192.0	10.16	1029.4	824.0	56.0	46.14
max	540.0	359.4	200.1	247.0	32.2	1145.0	992.6	365.0	82.6

strength. In Fig. 1, age had a surprisingly large effect on the strength of the mix, followed by cement and water. These are the two core components of concrete, so it is not surprising that they would have a significant impact on HPC strength.

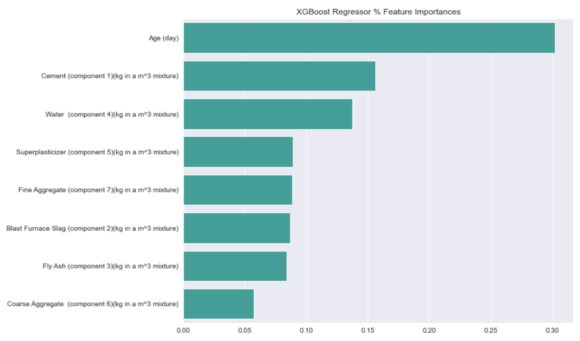

Fig. 1. Feature Importance

An off-the-shelf implementation of a multi-layered perceptron (MLP) was implemented with three hidden layers consisting of 64, 128, and 64 units, in order. An Adam solver was used with relu activation functions over 2000 iterations. A support vector regression (SVR) was also trained with an 8-degree polynomial kernel. Each of the models, the XGBoost, MLP, and SVR, were run through the genetic algorithm to search for an optimal feature-set.

A genetic algorithm consists of a population of feature-sets, called individuals, initialized to random values in the range of provided bounds. An appropriate set of bounds for each variable is in the range of the minimum and maximum of each feature column in the data. That way, we interpolate the data with what is known instead of extrapolating to what is unknown. The genetic algorithm then uses the input model to predict the performance values of each individual. The individuals are then ranked, and a selection algorithm samples from the population. Selected individuals are then merged by an average of each of their parameter values, and a mutation is made to the resulting values. The mutation is simply a small addition of noise. In this implementation of the GA, a modification is made to the GA so that a dynamic exploration rate can be set by the user. With this setting, the GA will perturb a high performing individual with a small amount of noise, and a worse performing individual will receive a greater perturbation. This allows the GA to search a potential global solution more carefully while letting the worse-performers search the wider space for better potential solutions. Once a new set of individuals is made to fill a population, the evolution is complete and the algorithm repeats for a designated number of evolutions. An elitism parameter is also implemented, so the user can set a number of top-performing individuals to be saved for future generations.

Each GA was set to search the data within the minimum and maximum values of the data for each column. See Table I. Each GA employed rank selection. For the XGBoost model, the GA was initialized with 100 individuals ran for 2000 generations. The elitism parameter was set to keep the top 5 performers each generation, and an exploration rate of .35 was used. For the MLP and SVR models, populations of size 50 were used with an elitism parameter of 3, an exploration rate of .35, and 500

TABLE II
OPTIMAL MIXTURE PREDICTIONS

	XGBoost	**MLP**	**SVR**
Cement (kg/m^3)	540.0	540.0	540.0
Blast Furnace Slag (kg/m^3)	285.6	284.2	359.4
Fly Ash (kg/m^3)	123.4	107.5	200.1
Water (kg/m^3)	151.1	121.8	121.75
Superplasticizer (kg/m^3)	11.3	32.1	32.2
Coarse Aggregate (kg/m^3)	1029.8	1145.0	1145.0
Fine Aggregate (kg/m^3)	749.5	992.6	992.6
Age (day)	150	205	365

generations were run. Greater population sizes and generations increase run-times but will have a more likely tendency to converge to global optima.

IV. Results

Of the three trained models, the XGBoost far outperformed the MLP and SVR models. With an average 5-fold cross validation testing score, the XGBoost had $r^2_{XGBoost} = .94$, the MLP scored $r^2_{MLP} = .80$, and the SVR scored $r^2_{SVR} = .71$. The GA was able to converge to an approximate solution in under 500 generations with a predicted max strength of 108 MPa. The MLP and SVR had max predictions of 149 MPa and 893 MPa in around 200 generations each, respectively. The evolution of the GAs on each model are depicted in Figs. 2, 3, 4.

The outputs reflect a few interesting traits of the models. The XGBoost in particular is incapable of extrapolating values not seen in the training data, so it is not projecting a potential maximum from the data the same way the MLP model does. The SVR blew up in its prediction, which shows the SVR likely trailed up towards a boundary in the data that does not reflect a realistic physical reality. In Table II, there are clear overlaps between the XGBoost and the MLP, and the SVR and the MLP. This is likely due to the fact that each method is capturing an underlying pattern in the data for different components.

The XGBoost, because of its strength in interpolation between seen data, is the most conservative estimate for a predicted output. Comparing its result to the maximum in the data, the XGBoost model expects a higher performing mixture than what exists in the data. The MLP, however, would be the optimistic estimation of the output as it provides and estimate that is a plausible physical reality.

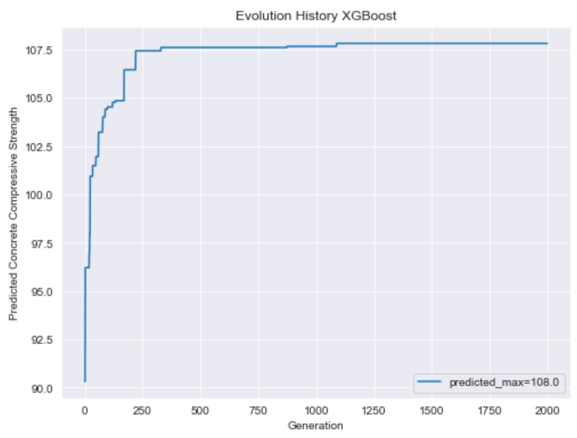

Fig. 2. XGBoost Convergence

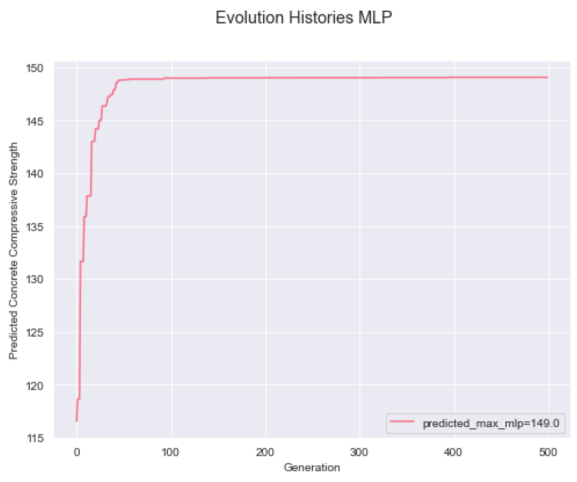

Fig. 3. MLP Convergence

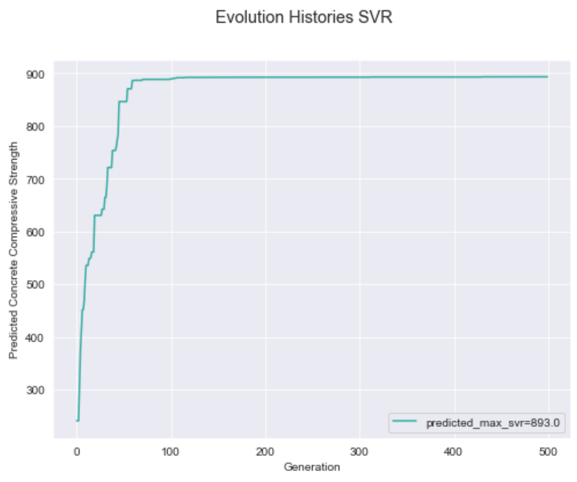

Fig. 4. SVR Convergence

V. Discussion

In reflection on the project, the XGBoost model was not the most informative for the goal of predicting an optimal HPC mix. It would have been more valuable as a validation tool for a better architected neural network. Because the XGBoost interprets well in known data, it would be valuable insight into what is already known rather than extrapolating to the unknown.

The focus of the Yeh (1998) [1] and papers that followed from it dive specifically into the predictive modeling of HPC. Prediction is a helpful tool, but it only marginally helps the issue of trial and error in finding new HPC mixes. Building an optimization tool with the genetic algorithm helps to create a more informed search by returning potential maximums. Ideally, the project could be supplemented with experimental data by testing the predictions with physical data. Experimental data could be fed back to the model to improve the model's knowledge of the data space which would lead to a cheaper, more efficient feedback loop resulting in stronger HPC.

References

[1] I.-C. Yeh, "Modeling of strength of high-performance concrete using artificial neural networks," *Cement and Concrete Research*, vol. 28, pp. 1797–1808, 12 1998.

[2] W. O'Brien, "Genetic algorithm," https://github.com/wgobrien/genetic-algorithm, 2022.

[3] G. C. Ohlmacher and J. C. Davis, "Using multiple logistic regression and gis technology to predict landslide hazard in northeast kansas, usa," *Engineering Geology*, vol. 69, no. 3, pp. 331–343, 2003.

[4] J.-Y. Potvin, "Genetic algorithms for the traveling salesman problem," *Annals of Operations Research*, vol. 63, no. 3, pp. 337–370, 1996.

[5] A. D. Tura and H. B. Mamo, "Characterization and parametric optimization of additive manufacturing process for enhancing mechanical properties," *Heliyon*, vol. 8, no. 7, p. e09832, 2022.

[6] C.-H. Wu, J.-M. Ho, and D. Lee, "Travel-time prediction with support vector regression," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 276–281, 2004.

[7] K. Krishnakumar and D. E. Goldberg, "Control system optimization using genetic algorithms," *Journal of Guidance, Control, and Dynamics*, vol. 15, no. 3, pp. 735–740, 1992. [Online]. Available: https://doi.org/10.2514/3.20898

[8] M. A. Strobl and D. Barker, "On simulated annealing phase transitions in phylogeny reconstruction," *Molecular Phylogenetics and Evolution*, vol. 101, pp. 46–55, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1055790316300823

[9] T. Kolda, R. Lewis, and V. Torczon, "T.g. kolda, r.m. lewis, v. torczon: Optimization by direct search: New perspectives on some classical and modern methods. siam review 45, 385-482," *SIAM Review*, vol. 45, pp. 385–482, 09 2003.

[10] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 01 1997.

[11] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785