Machine learning application in optimization of flexible circuit configuration

Ren Gibbons(gibbons6), Prajwal K.A(prajwalk)

CS 229 Project, Stanford University
December 11, 2018

Stanford
CEE Department

Abstract

- Flexible circuits are of great interest to hardware manufacturers. However, achieving desired stretchability is difficult since transistor elements are relatively stiff and are easily damaged.
- One solution is to place semi-rigid transistors on a flexible polymer substrate.
- Next, we can optimize the transistor placement on the substrate using particle swarm optimization to minimize some objective (area, factor of safety, etc.).
- This has been successfully performed for 3-transistor circuits. However, the algorithm is computationally expensive.
- This motivates machine learning on a dataset of optimizations run offline to quickly predict an optimized geometry.
- We present the results of two supervised learning approaches – multivariate linear regression (MLR) and multivariate adaptive regression splines (MARS).
- Next, we present a starting framework for using reinforcement learning to optimize the circuit geometry which can ensure that constraints are satisfied.

Stanford University

Motivation

Importance of flexible circuits

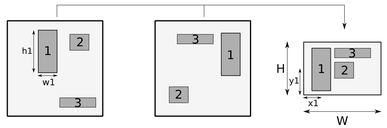
Flexible electronics is an exciting research area because potential applications are widespread. Cellphones, medical devices, and exercise monitors will benefit from flexible components.

Optimization problem

Unfortunately, stiff and fragile transistors elements complicate the production of reliable flexible circuits. One solution is to place the elements on a compliant polymer substrate and optimize the geometry. We use a particle swarm optimization tool to minimize circuit area:

$$\min W \times H$$
s.t. *geometry is valid*
strain energy not exceeded

The algorithm uses 60 particles, where each particle is a possible geometric configuration, and a finite element simulation is run to see if the geometry is safe when strained by 50%.



Features

- **Inputs**: 3 transistors, each with specified width (w), height (h), Young's modulus (E), max strain allowed (ϵ)
  $x^{(i)} = [w_1, w_2, w_3, h_1, h_2, h_3, E_1, E_2, E_3, \epsilon_1, \epsilon_2, \epsilon_3]$
- **Outputs**: Circuit dimensions and each transistor location
  $y^{(i)} = [W, H, x_1, y_1, x_2, y_2, x_3, y_3]$

Simulations

We use Latin hypercube sampling to randomly generate our input features, ensuring an appropriate span of the input space, where $w, h \in [0.2\text{mm}, 2\text{mm}]$, $E \in [5\text{MPa}, 50\text{MPa}]$, and $\epsilon \in [0.005, 0.01]$. We generated 1,000 samples, which took two weeks of wall-clock time on 72 cores in parallel.

Data augmentation

Since our data is expensive to obtain, we use an augmentation technique to increase the size of our dataset. We permute the indices of each of the three transistors, giving an augmented dataset of size

$$n = 3! \times 1{,}000 = 6{,}000\ .$$

Supervised learning approach

Why machine learning for this problem?

Each PSO simulation takes roughly 24 hours on a single core. This expense is unreasonable for a large number of circuits to be designed, so machine learning can help speed up the process. For both MLR and MARS, we separate the data into 80% training set, 10% validation set, 10% test set.

Multivariate linear regression (MLR)

MLR is a learning technique that generates a model to linearly predict an array of outputs of form

$$y_{ij} = \beta_0 + \beta_{1j} x_{i1} + \cdots + \beta_{nj} x_{in}$$

for the ith observation and jth feature. After training, we compute the MSE and absolute difference both the regular and augmented dataset with the results below. The difference in errors between the training and test data are small, so MLR has low bias. However, the augmented data set appears to somewhat help prediction performance.

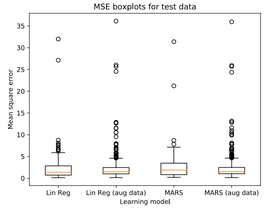
MLR	MSE	MSE (aug)	Abs. diff.	Abs. diff (aug)
Training data	2.102	2.338	1.095	1.187
Test data	2.717	2.236	1.194	1.178

Multivariate adaptive spline regression (MARS)

MARS performs regression on a set of features and searches for nonlinear interactions in the training set by performing two stages. The forward stage looks for points to minimize MSE, and the pruning stage finds a subset of terms by minimizing a cross-validation score. The model is then comprised of a constant, linear functions, and hinge functions. MARS gives the results in the table below. The MARS model does not appear to perform significantly differently from MLR.

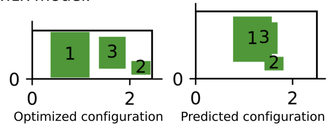
MARS	MSE	MSE (aug)	Abs. diff.	Abs. diff (aug)
Training data	1.745	2.341	0.997	1.189
Test data	2.815	2.234	1.242	1.181

The box plot below shows the distribution of errors. Both MLR and MARS suffer from high outliers.



Visualization of predicted design

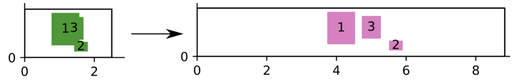
The figure below is an example of a predicted configure from the MLR model.



Reinforcement learning approach

Problem: violated constraints

The figure of the predicted configuration shows overlapping transistor elements. This is one limitation of a supervised learning approach: no simple way to enforce constraints. One workaround is to linearly scale the configuration until no overlap is observed as shown below. Performing this procedure on the MLR test set results in a mean size increase of 802% compared to the optimized shape. This is not a satisfactory result, which motivates a reinforcement learning approach.



Reinforcement learning

Our optimization problem ideally falls into the category of continuous space and continuous action problem. To compound this, the continuous space for the transistor coordinates also keeps changing with area. As this is a complex problem, a simpler model having the area fixed is solved for maximizing the Factor of Safety (FS) of transistors.

$$FS = \frac{Max\ strain\ energy\ allowable}{Average\ strain\ energy}$$

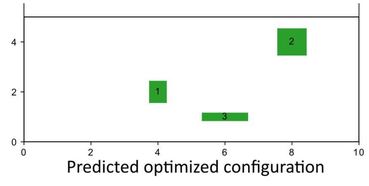
In the RL algorithms, high positive rewards are given to global maximum state and high negative rewards for violated constraints. The state with the highest reward is output as the optimized solution.

Value iterations for discretized space

The simplest way to apply Reinforcement Learning to a complex continuous problem is by discretizing it. Both the space and actions are discretized. Random initializations contribute to uniform search over the domain. The optimal value function (V^*) is learnt using Bellman's equation.

$$V^*(s) = R(s) + \max_a \gamma \sum_{s'} P_{sa}(s') V^*(s')$$

A sample simulation by discretizing the space into 4097 states and actions into 125 states produced an optimized configuration as shown below.



Fitted value iteration for continuous space

This is currently being worked on. The space is considered as continuous but the actions are discretized. In this method, we approximate the optimal value function as a function of the states, initially using Linear Regression.

Conclusion

Discussions

- Both MLR and MARS consider the output features to be independent of each other and just depend on the input features. This assumption doesn't hold in this problem and hence, they do not perform well.
- The amount of data we had for training the model is very less. This compelled us against using Deep Learning techniques like Recurrent Neural Networks (RNN) which considers the interactions between the outputs.
- The data we get by optimization is bereft of failed/unsafe configurations. Thus, supervised learning algorithms cannot learn anything about the safety of a resulting configuration.
- Linear scaling of the configuration to remove overlap is not a good solution for the problem of violated constraints as it results in a very high increase in area.
- Observing these drawbacks of supervised learning, it can be considered to not be a good method for this application. This urged us to look at Reinforcement Learning.
- Though value iterations by discretizing considers all the constraints, it may not be a good algorithm for optimization problems that are not convex as it takes a large amount of iterations and time.

Future Work

- The next immediate work would be to get results from the fitted value iteration method for continuous space.
- Next, we wish to extend this to the original problem of minimizing the area with continuous action space. This basically would be equivalent to an optimization algorithm.
- With more data, we would wish to observe how well Recurrent Neural Networks can handle the constraints.
- The main application of this project is in optimizing the design of complex circuits with a large number of transistors. So, we wish to extend these methods to larger circuits either by discretizing them into smaller circuits with 3 transistors and stitching them back, or by applying the RL for continuous space and actions directly to the method if it is scalable.

References

- Friedman, J. Multivariate adaptive regression splines. The annals of statistics, 19(1), 1991.
- Olsson, A, et al.. On Latin hypercube sampling for structural reliability analysis. Structural Safety 25 (2003) 47–68. (2003)

Acknowledgements

- Reza Rastak, CEE PhD candidate, author of PSO algorithm used in this project, offered advice and access to his code.
- CS 229, PS 4, 2018, provided a template for the RL implementation.