A Heuristic For Efficient Reduction In Hidden Layer Combinations For Feedforward Neural Networks

Computing Conference 2020

Wei Hao Khoong

NUS National University of Singapore

Table of contents

Background

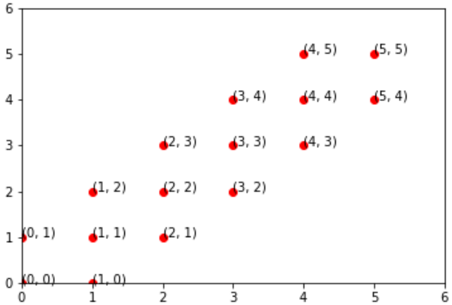
- Analyst, SingHealth Office for Insights & Analytics
- Graduate student, Department of Statistics & Applied Probability, National University of Singapore
- Active Kaggler

Motivation

- A full grid search takes a long time
- Tendency to overfit even with cross-validation
- This approach instantiates another form of grid search with local search

Overview of Approach

- Consider the 2D case:



Preliminaries - The Models

From Scikit-Learn's `sklearn.neural_network` package:

1. `MLPRegressor`
   - A multi-layer perceptron regressor
2. `MLPClassifier`
   - A multi-layer perceptron classifier

Parameters used:

- `activation='relu', solver='lbfgs', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, max_iter=500, random_state=69`
- The rest are defaults

Data

1. Boston house-prices
   - available from Scikit-Learn's `sklearn.datasets` package
2. MNIST handwritten digits
   - downloadable from http://yann.lecun.com/exdb/mnist/

Test Cases

1. Method 1 - Benchmark
   - Initializaion L_0 is the set of all possible hidden-layer combinations
   - `GridSearchCV` takes in this L_0 and returns the 'best' result
2. Method 2 - Heuristic Algorithm
   - Initializaion L_0 is the set of hidden-layer combinations with equal number of neurons in each hidden-layer
   - The algorithm described in the following slides takes in this L_0 and returns the'best' result

The Algorithm (2-Layer Example)

- Set max. number of hidden-layer neurons $= 10$
- Set number of hidden-layers $= 2$
- Initialize set of all hidden-layer combinations with same number of neurons in each layer:

$$L_0 = \{(1,1),(2,2),\ldots,(10,10)\},$$

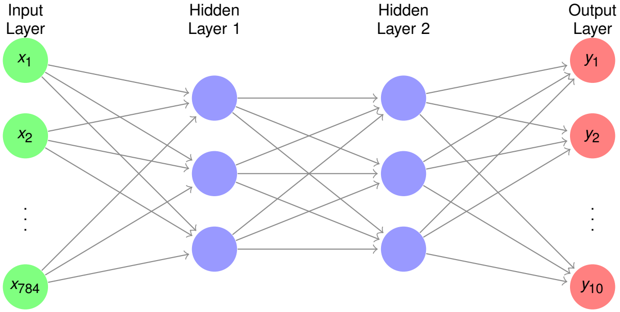
where $n_{min}^{(input)} = 1, n_{max}^{(input)} = 10$

- Perform grid-search on L_0 with stratified K-fold cross-validation (`GridSearchCV`) to obtain $H^{(0)} = \{(j,j)\}$, where

$$\text{RMSE}_{(j,j)} = \min\{\text{RMSE}_{(i,i)}\}, i \in \{1,\ldots,10\}$$

The Algorithm (2-Layer Example)

- Output of `GridSearchCV`: $H^{(0)} = \{(3,3)\}$



The Algorithm (2-Layer Example)

Now let's proceed the main block of the algorithm

The Algorithm (2-Layer Example)

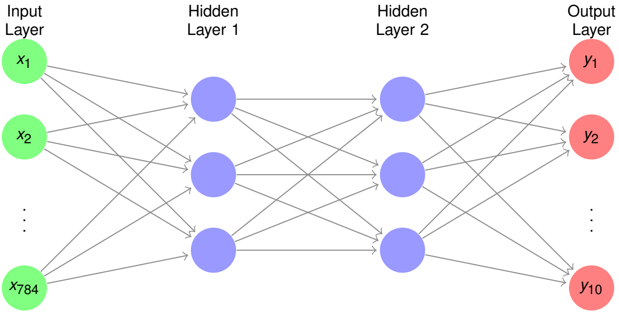
- Define the threshold for each iteration by

$$\Delta = \left| \frac{RMSE_{curr} - RMSE_{prev}}{RMSE_{prev}} \right|,$$

  where $RMSE_{curr}$ and $RMSE_{prev}$ are the RMSE from the current and previous iterations respectively.
- Set tolerance $\alpha = 0.10$ for this example
- If $\Delta > \alpha$, continue to next iteration

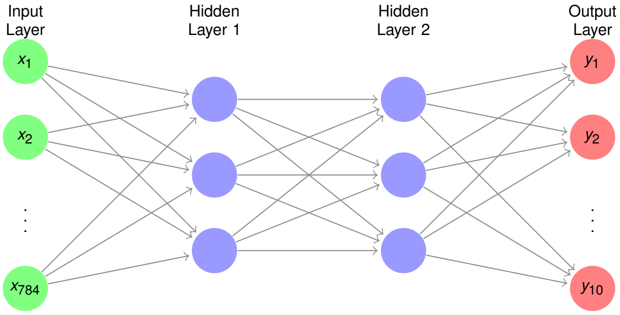
The Algorithm (2-Layer Example)

- Now at iteration: **1**, $\Delta > 0.10$
- $H^{(1)} = H^{(0)} = \{(3,3)\}, H_{prev} = \{\}$
- $n_{min} = 3, n_{max} = 3$



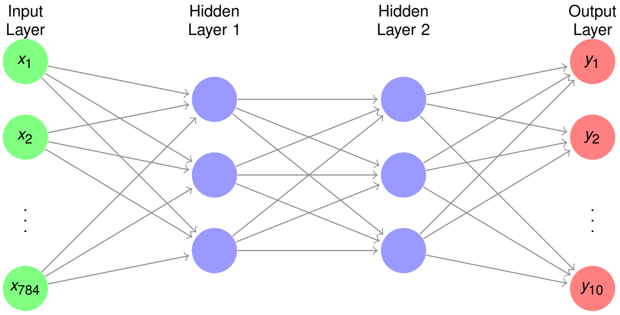
The Algorithm (2-Layer Example)

- Now at iteration: **1**, $\Delta > 0.10$
- $H^{(1)} = H^{(0)} = \{(3,3)\}, H_{prev} = \{\}$
- $n_{min} = 3 - 1, n_{max} = 3$



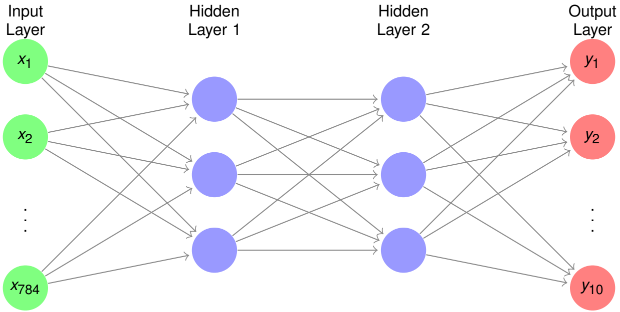
The Algorithm (2-Layer Example)

- Now at iteration: **1**, $\Delta > 0.10$
- $H^{(1)} = H^{(0)} = \{(3,3)\}, H_{prev} = \{\}$
- $n_{min} = 2, n_{max} = 3$



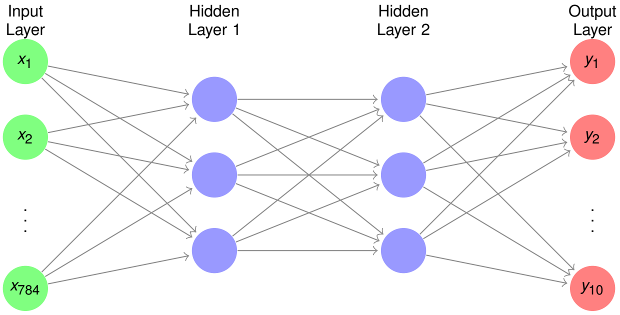
The Algorithm (2-Layer Example)

- Now at iteration: **1**, $\Delta > 0.10$
- $H^{(1)} = H^{(0)} = \{(3,3)\}, H_{prev} = \{\}$
- $n_{min} = 2, n_{max} = 3{\color{red}+1}$



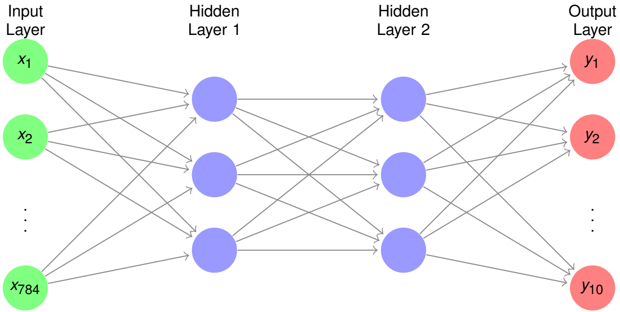
The Algorithm (2-Layer Example)

- Now at iteration: **1**, $\Delta > 0.10$
- $H^{(1)} = H^{(0)} = \{(3,3)\}, H_{prev} = \{\},$
- $n_{min} = 2, n_{max} = 4$



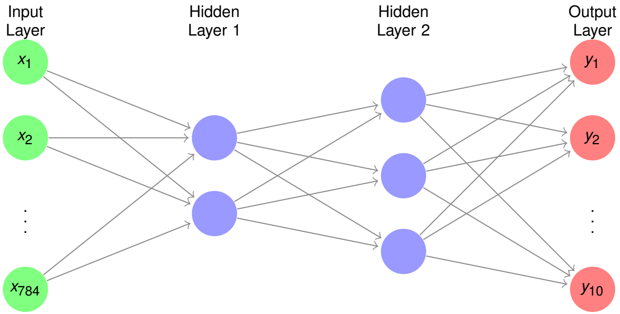
The Algorithm (2-Layer Example)

- Now at iteration: **1**, $\Delta > 0.10$
- $H^{(1)} = \{(2,2),(2,3),(3,2),(3,3),(3,4),(4,4)\}, H_{prev} = \{\}$
- $n_{min} = 2, n_{max} = 4$



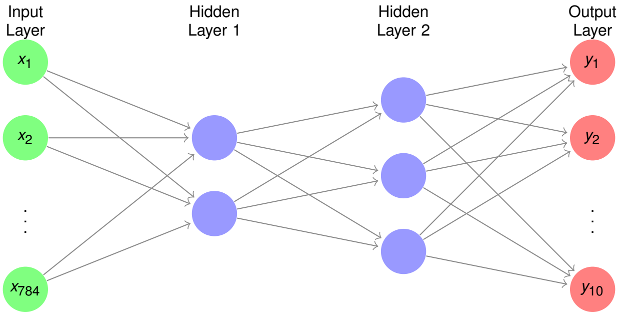
The Algorithm (2-Layer Example)

- Now at iteration: **1**, $\Delta > 0.10$
- $H^{(1)} = \{(2,2),(2,3),(3,2),(3,3),(3,4),(4,4)\}$, $H_{prev} = H_{prev} \cup H^{(1)} \setminus H^{(1)} \cap H_{prev}$
- $n_{min} = 2, n_{max} = 4, H_{curr,best} = \{\}$



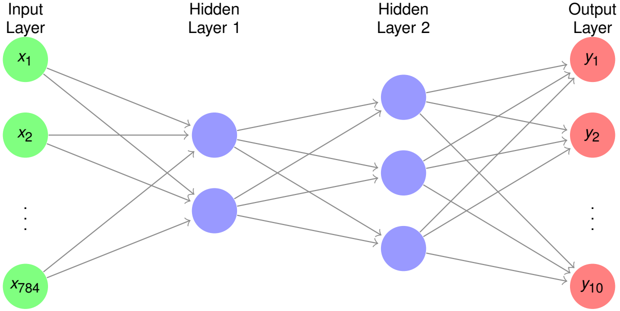
The Algorithm (2-Layer Example)

- Now at iteration: **1**, $\Delta > 0.10$
- $H^{(1)} = \{(2,2),(2,3),(3,2),(3,3),(3,4),(4,4)\}, H_{prev} = H_{prev} \cup H^{(1)} \setminus H^{(1)} \cap H_{prev}$
- $n_{min} = 2, n_{max} = 4, H_{curr,best} = \{(2,3)\}$



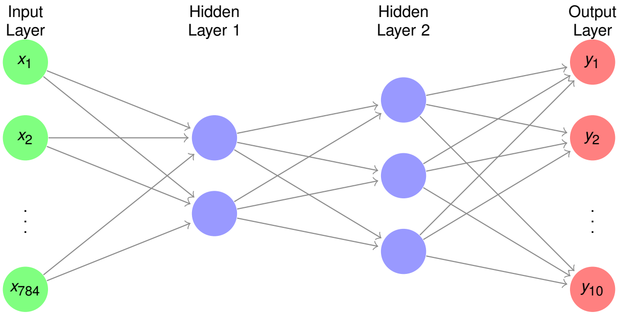
The Algorithm (2-Layer Example)

- Now at iteration: **2**, $\Delta > 0.10$
- $H^{(1)} = \{(2,2),(2,3),(3,2),(3,3),(3,4),(4,4)\}, H_{prev} = H_{prev} \cup H^{(1)} \setminus H^{(1)} \cap H_{prev}$
- $n_{min} = 2, n_{max} = 4, H_{curr,best} = \{(2,3)\}$



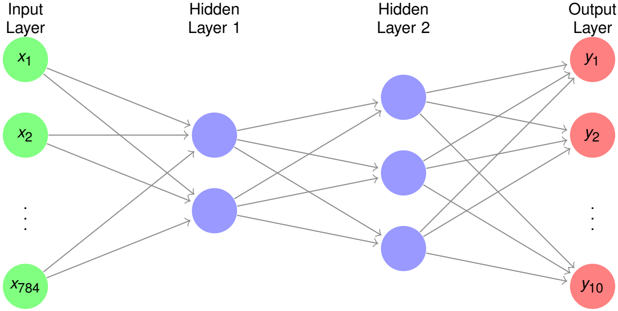
The Algorithm (2-Layer Example)

- Now at iteration: **2**, $\Delta > 0.10$
- $H^{(1)} = \{(2,2),(2,3),(3,2),(3,3),(3,4),(4,4)\}, H_{prev} = H_{prev} \cup H^{(1)} \setminus H^{(1)} \cap H_{prev}$
- $n_{min} = 1, n_{max} = 5, H_{curr,best} = \{(2,3)\}$



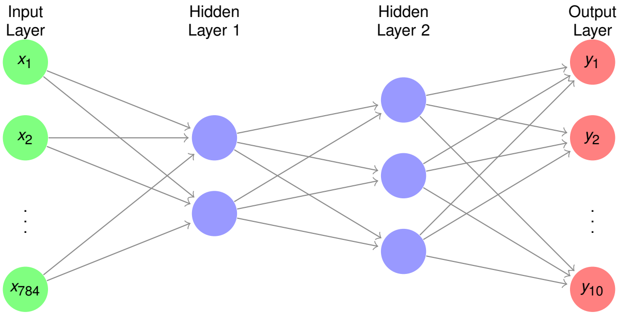
The Algorithm (2-Layer Example)

- Now at iteration: **2**, $\Delta > 0.10$
- $H^{(2)} = \{(1,1),(1,2),(2,2),\ldots,(5,5)\}, H_{prev} = H_{prev} \cup H^{(1)} \setminus H^{(1)} \cap H_{prev}$
- $n_{min} = 1, n_{max} = 5, H_{curr,best} = \{(2,3)\}$



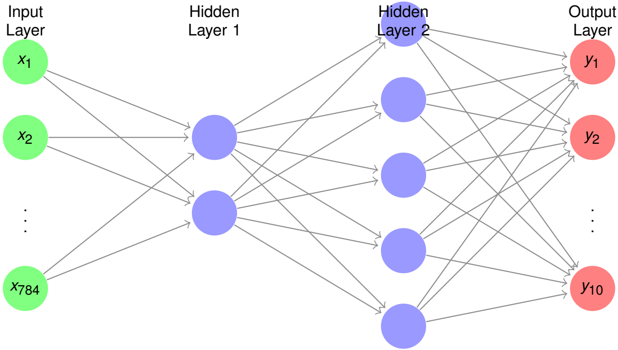
The Algorithm (2-Layer Example)

- Now at iteration: **2**, $\Delta > 0.10$
- $H^{(2)} = \{(1,1),(1,2),(2,2),\ldots,(5,5)\}$, $H_{prev} = H_{prev} \cup H^{(2)} \setminus H^{(2)} \cap H_{prev}$
- $n_{min} = 1, n_{max} = 5, H_{curr,best} = \{(2,3)\}$



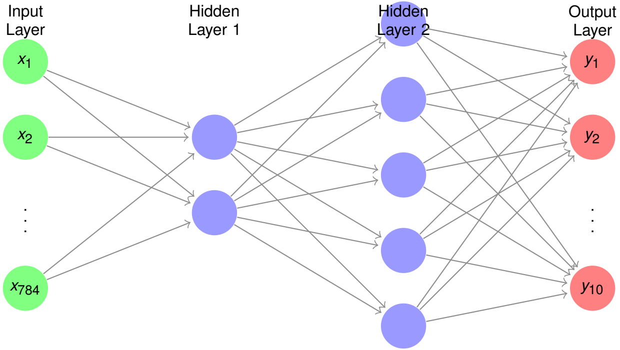
The Algorithm (2-Layer Example)

- Now at iteration: **2.** $\Delta < 0.10$
- $H^{(2)} = \{(1,1),(1,2),(2,2),\ldots,(5,5)\}, H_{prev} = H_{prev} \cup H^{(2)} \setminus H^{(2)} \cap H_{prev}$
- $n_{min} = 1, n_{max} = 5, H_{curr,best} = \{(2,5)\}$



The Algorithm (2-Layer Example)

- But now $\Delta < \alpha = 0.10$ (i.e. less than 10% improvement in RMSE)
- So algorithm terminates and returns the resulting architecture:



Test Cases (Recap)

1. Method 1 - Benchmark
   - Initializaion L_0 is the set of all possible hidden-layer combinations
   - `GridSearchCV` takes in this L_0 and returns the ‘best’ result
2. Method 2 - Heuristic Algorithm
   - Initializaion L_0 is the set of hidden-layer combinations with equal number of neurons in each hidden-layer
   - The algorithm described in previous slides takes in this L_0 and returns the‘best’ result

Results

	Benchmark	Heuristic	Benchmark	Heuristic	Benchmark	Heuristic
Number of Hidden-Layers	1		2		3	
Median Score	0.82	0.82	0.85	0.84	0.86	0.83
Median RMSE	3.76	3.50	3.68	3.56	3.63	3.57
Median Time Elapsed (s)	2.92	**2.84**	12.63	**5.07**	56.00	**7.03**

Table: Boston Housing Prices dataset with `MLPRegressor`

	Benchmark	Heuristic	Benchmark	Heuristic	Benchmark	Heuristic
Number of Hidden-Layers	1		2		3	
Median Score	0.92	0.92	0.93	0.92	0.93	0.93
Median RMSE	1.08	1.08	1.05	1.08	1.09	1.09
Median Time Elapsed (s)	1014.20	**1011.10**	9498.86	**2763.15**	36336.83	**2556.78**

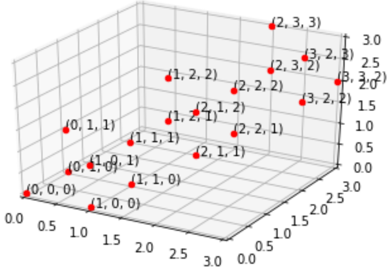
Table: MNIST dataset with `MLPClassifier`

Conclusion

- For test cases with the heuristic, as number of hidden-layers increases:
  - run-time decreases significantly
  - accuracy/score about the same as benchmark
- Heuristic appears effective in reducing model overfitting in the regression case
  - median validation RMSE for heuristic case lower than the benchmark case

Future Work

- Extend heuristic algorithm to more hyperparameters (as coordinates)
- Parameterize number of top performing starting candidates from grid search at initialization



Future Work

- Use various metrics in-place of RMSE in threshold Δ
- Selection algorithms to obtain inputs for L_0 of grid search initialization
- Open-source library (Python 3) with the above implementations in Scikit-learn, PyTorch, Tensorflow

Acknowledgements

- NUS:
  - High Performance Computing (HPC) for computing resources
  - Unrestricted-access to most journals

Selected References

R. Fletcher.
Practical Methods of Optimization; (2Nd Ed.).
Wiley-Interscience, New York, NY, USA, 1987.

Marc Claesen and Bart De Moor.
Hyperparameter search in machine learning.
CoRR, abs/1502.02127, 2015.

Ivan Jordanov and Antoniya Georgieva.
Neural network learning with global heuristic search.
Neural Networks, IEEE Transactions on, 18:937 – 942, 06 2007.

Contact

- Email: khoongweihao@u.nus.edu
- LinkedIn: https://www.linkedin.com/in/wei-hao-khoong-6b94b1101