Evolutionary Optimisation of Deep Neural Networks

Christian Kirfel

7th of October 2020

UNIVERSITÄT BONN

Initial problems

Obstacles of optimisation processes

- Grid searches are both tedious and ressource intensive
- Knowledge gained in one problem is rarely universal
- Experienced users can develop a bias towards certain hyperparamters

Making machine learning more accesible to students

- Physics students need increasingly high programming skills
- Most students actively search this challenge but appreciate the rewarded of early results
- Desirable to create a framework that is easy to use and allows to face some challenges later on

Goals

Network optimisation

- Create an algorithm more efficient than a grid search
- The algorithm should be unbiased and avoiding local minima

Accesibility

- A new user should have easy access
- The challenges for a new user should be small enough to allow a feeling of achievment

BAF compatability

- The algorithm should utilize the BAF worker node structure
- Ideally it should be efficient enough to create satisfying results on CPUs

Evolutionary optimisation of neural networks

- Combination of a grid searches with a survival of the fittest setup
- Start with a number of random hyperparamters
- Evaluate the set of hyperparameters
- Create a a new set of networks based on the previous best
- Add recombination and variation to avoid local minima and bias

Neural Networks - Processing information

Humam senses

- Extraction of relevant info
- Impossible for machines

Human brain

- Web of neuron cells
- Input from surrounding cells
- Single combination $\rightarrow$ action

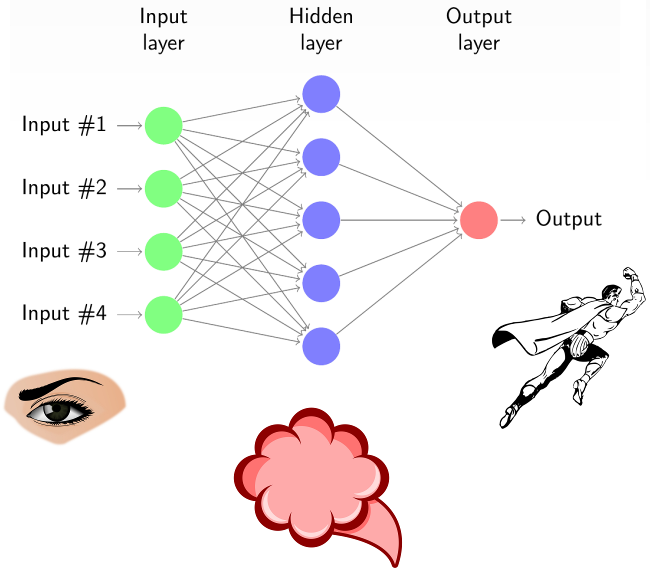
Input variables

- Preprocessed by user
- e.g. kinematic variables

Net of nodes

- Nodes = simple processors
- Connected by linear function
- Combination forms non-linear model

Neural network structure



Neural Networks - Choosing the next step

Evaluation of an action

- Simple perceptions: pain, satisfaction
- Expectation

Decision for a next step

- Trial and error
- Learning from experience

Loss function

- Supervised learning: compare to the desired outcome
- Loss = estimator for quality

Optimisation

- Back-propagation impact of parameters' on the loss
- Adjust parameters to minimise plot

Hyperparameter optimisation

What is a hyperparameter?

- During the learning process the neural network optimises its internal parameters
- Some parameters are still set by the user according to the task of the network
- These are called hyperparameters

How does one optimise the choice?

- Neural networks provide several metrics to estimate result and performance
- To optimise the hyperparameters one usually runs several configurations to find a good set of parameters

Concepts of evolutionary network optimisation

Choosing a start

- Randomly choose a set of values for each hyperparameter
- Combine the random selection to create a set of network configurations

Evaluating a start

- Run the networks for a small number of epochs
- Use networks' metrics to evaluate the performance

Choosing a next step

- Rank networks by their metrics
- Reuse, mate and mutate networks

The initial generation

Current setup

- Choose a random set of hyperparameters from a range of paramters set by the user
- Create a set of neural networks from all possible combinations

Planned

- Draw each hyperparameter from a fitting distribution
- Restrict the number of combinations based on the hyperparameter

Evaluating a generation

Current setup

- Evaluate all networks based on a metric of choice
- The metric of choice can be simply the AUC
- Save the x best configuration

Planned

- Test and combine different metrics
- Implement different metrics with regard to early stopping

Breeding the next generation

Current setup

- Always keep the best configuration
- Recombine the λ best configurations
- Recombine the λ best generations again and vary μ hyperparameters

Planned

- Include more hyperparameters
- Specify different variation probabilities and values for different hyperparameters

Summary of the optimisation process

- Survival of the fittest: Keep the best setup
- Recombination: Reuse the best hyperparameters for the next batch
- Variation: Randomly change the hyperparamters to avoid local minima and bias

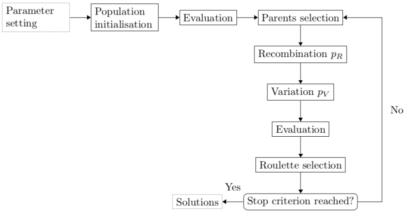


Figure 2: Breeding pipeline in EvoDeep algorithm.

[1]

Setup on baf

- Create a random set of hyperparameters
- Submit a job to baf for each configuration
- Use dagman to wait for the jobs to finish
- Evaluate the results and create the next set of configurations
- Anji's talk
- Confluence link

Setup

- tZq as signal against t$\bar{\text{t}}$ as background
- Using basic kinematic variables
- No specific region
- Training without weights due to some recent problems
- Testing: nodes, layers, dropout
- Fixed hyperparameters:
  - Optimizer: Adam
  - Activation: relu, sigmoid
  - Batchsize: 1000
  - Epochs: 25

An example of parameter development

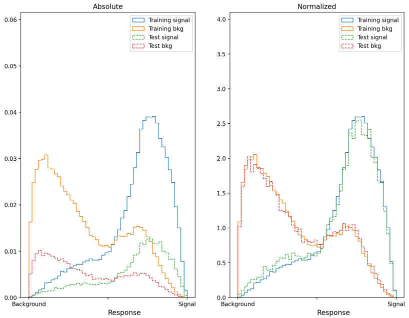
Initial parameters

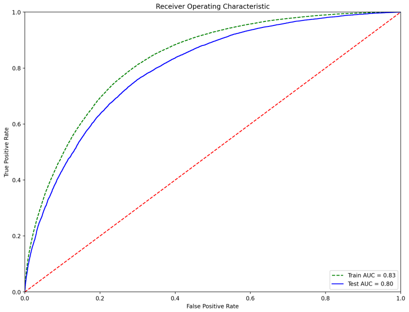
- Layers: $1-10$
- Nodes: $1-100$
- Dropout: $0-1$

Final parameters

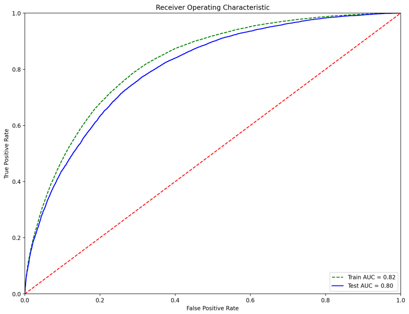
- Layers: 4 ± 2
- Nodes: 67 ± 33
- Dropout: 0.4 ± 0.3

ROC and Separation

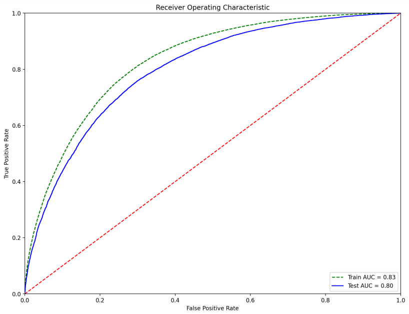




Comparing to a grid search

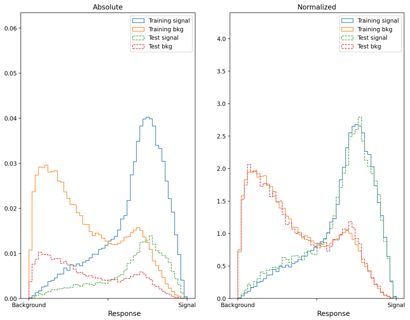


Grid search

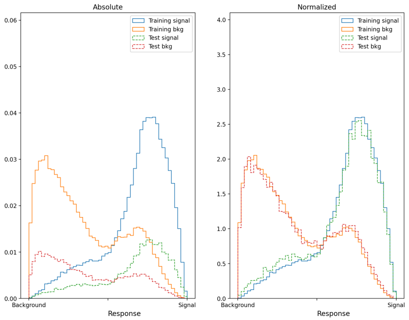


Evolutionary search

Comparing to a grid search



Grid search



Evolutionary search

Conclusion

- Now is the time for intense testing. Can the network uphold its promises?
- There is a large number of features still to be implemented
- I have to test on a proper sample with weights and a good set of variables
- For now the code is running and shows the expected behaviour
- For a different option utilizing grid ressources and ATLAS GPUs see this talk by Rui Zhang in the ATLAS ML Forum Link

Sources

Alejandro Martín García et al. "EvoDeep: a new Evolutionary approach for automatic Deep Neural Networks parametrisation". In: *Journal of Parallel and Distributed Computing* 117 (Oct. 2017). DOI: 10.1016/j.jpdc.2017.09.006.