Hyperparameter Optimization using Ranking Loss Surrogates

Master's Thesis
by

Abdus Salam Khazi

Albert-Ludwigs-University Freiburg
Representation Learning Department

July 20, 2022

Table of Contents

Motivation

Motivation

- Machine Learning models are very sensitive to their hyperparameters (HP).
- Tuning hyperparameters is essential to get the most performance out of a given ML model.
- **Aim of the thesis: Improve Hyperparameter Optimization (HPO) process.**
- Let us define HPO first!

HPO can be seen as selecting a hyperparameter setting to obtain the best (lowest) validation error during ML training. Mathematically, the HPO objective function is defined as:

$$\underset{\mathbf{x}}{\operatorname{argmin}} \; f(m_{\mathbf{x}}^{trained}(\mathrm{Data}_{\mathrm{val}})) \mapsto \mathbb{R} \;\; \forall \mathbf{x} \in \mathbb{S}$$

where

- f is the evaluation criterion. (Different for Regression, Classification, etc.)
- $\mathbf{x}$ is an HP configuration.
- $m_{\mathbf{x}}^{trained}$ is a model trained with 'x' HP configuration.
- $\mathrm{Data}_{\mathrm{val}}$ is the validation split of the data.
- $\mathbb{S}$ is the HP search space.

Related Works

- Various classes of HPO solutions have been proposed - Blackbox HPO, Online HPO, and Multi-fidelity HPO.
- **SMBO** (Sequential Model-Based Optimization) algorithm for HPO has given good results.
- Transfer learning has shown good results in the HPO domain due to the availability of metadata to train.
- **Thesis target: Propose a transfer HPO surrogate for the SMBO process using the concept of ranking.**
- The results are compared against SOTA HPO models like RGPE, TAF, TST, and FSBO for transfer case and GP, BOHAMIANN, and DNGO for the non-transfer case.

Baselines

Deep Ensembles

- Used as non-transfer surrogates in SMBO algorithm.
- It is an ensemble of neural networks.

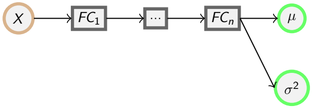


Figure: Architecture of a single neural network.

- Training can be done in parallel.
- Loss function used to train the neural networks (σ^2 is variance and μ is the mean):

$$L_{de} = -\log p(y|x) = \frac{\log \sigma^2}{2} + \frac{(y-\mu)^2}{2\sigma^2} + \text{constant}$$

FSBO

FSBO - Few Shot Bayesian Optimization.

- It is a state-of-the-art transfer HPO method.
- Formulates HPO problem as a few shot learning task.
- Uses deep kernels as surrogates which are of the form:

  $$k(\phi(\mathbf{x}, \mathbf{w}), \phi(\mathbf{x'}, \mathbf{w})|\theta)$$

  where $\mathbf{x}$ and $\mathbf{x'}$ are HP configurations. θ and $\mathbf{w}$ are parameters of kernel k and the neural network ϕ respectively.
- The loss function maximizes the following log probability

  $$\log p(\mathbf{y}^{(1)}...\mathbf{y}^{(T)}|\mathbf{X}^{(1)}...\mathbf{X}^{(T)}, \theta, \mathbf{w})$$

Background: Ranking Losses

Understanding Ranking

Consider a set:

$$\mathbb{A} = \{a_1, a_2, a_3, ..., a_n\}$$

- Ranking means ordering elements of the set $\mathbb{A}$.
- We consider that lower rank is better than higher rank, hence

$$\texttt{Rank}(a_i) < \texttt{Rank}(a_j) \iff a_i \succ a_j$$

Ranking can be divided into the following components:

- Obtaining relevance scores of objects in the set $\mathbb{A}$.
- Sorting objects based on relevance scores.

Since sorting is not differentiable trivially, we only model the scoring function s using an ML model.

$$s : \mathbb{A} \mapsto \mathbb{R}$$

Understanding Ranking Losses

Consider that data is given in the following format:

Instance	Object Set	Ground Truth (Scores)
1	$\{a_1, a_2, a_3, \ldots, a_{10}\}$	$\{y_1, y_2, y_3, \ldots, y_{10}\}$
2	$\{a'_1, a'_2, a'_3, \ldots, a'_{15}\}$	$\{y'_1, y'_2, y'_3, \ldots, y'_{15}\}$
3	$\{a''_1, a''_2, a''_3, \ldots, a''_7\}$	$\{y''_1, y''_2, y''_3, \ldots, y''_7\}$
...	$\{\ldots\}$	$\{\ldots\}$

Table: Data used to train a scoring function.

How to learn the scoring function s?

- Use a parametrized model s_θ.
- Use a (ranking) loss function to get the optimum θ^*.

Understanding Ranking Losses

Types of ranking:

- **Pointwise** - The model directly predicts the rank and not the score. E.g., Subset Regression.
- **Pairwise** - The model predicts which of the 2 inputs is better. E.g., Ranknet.
- **Listwise** - Deals with a full input set as 1 learning instance. E.g., ListMLE.

Types of Ranking Losses:

- $L_{\texttt{pointwise}} : s(\mathbb{A}) \times \mathbb{Y} \mapsto \mathbb{R}$
- $L_{\texttt{pairwise}} : s(\mathbb{A})_1 \times s(\mathbb{A})_2 \times \mathbb{Y}_1 \times \mathbb{Y}_2 \mapsto \mathbb{R}$
- $L_{\texttt{listwise}} : s(\mathbb{A})_1 \times s(\mathbb{A})_2 ... s(\mathbb{A})_n \times \mathbb{Y}_1 \times \mathbb{Y}_2 \times ... \mathbb{Y}_n \mapsto \mathbb{R}$

ListMLE [1] stands for List Maximum Likelihood Estimation. The basic idea is:

- Find the "distance" between the predicted score list and the ground truth.
- Reducing this "distance" amounts to optimization.

Finding Distance:
Probability of selecting an item can be written as:

$$P = \frac{\phi(s(a))}{\Sigma_i \phi(s(a_i))}$$

where ϕ is a strictly positive increasing function.

Finding Distance (Continued):
If π defines a permutation of a list, the probability of selecting the permutation is:

$$P_\pi = \prod_{j=1}^{k} \frac{\phi(s(\pi_j))}{\sum_{t=j}^{k} \phi(s(\pi_k))}$$

Applying log on both sides:

$$\log P_\pi = \sum_{j=1}^{k} \log \frac{\phi(s(\pi_j))}{\sum_{t=j}^{k} \phi(s(\pi_k))}$$

Finding Distance (Continued): Let π^* be the ground truth permutation. The probability of selecting π^* is:

$$\log P_{\pi^*} = \sum_{j=1}^{k} \log \frac{\phi(s(\pi_j^*))}{\sum_{t=j}^{k} \phi(s(\pi_k^*))}$$

The distance metric is given by

$$L_{mle} = -\log P_{\pi^*}$$

and hence

$$L_{mle} = -\sum_{j=1}^{k} \log \frac{\phi(s(\pi_j^*))}{\sum_{t=j}^{k} \phi(s(\pi_k^*))}$$

Weighted ListMLE

- In SMBO, it is more important to rank the first HP configuration than the last.
- Hence, using a weighting strategy during the calculation of distance metric makes sense.

If $c(j)$ gives the weight of the position j, then the loss becomes:

$$L_{mle} = -\sum_{j=1}^{k} c(j) \log \frac{\phi(s(\pi_j^*))}{\sum_{t=j}^{k} \phi(s(\pi_k^*))}$$

- We use inverse logarithmic weighting given by $c(j) = \frac{1}{\log(j+1)}$.

Proposed Method

Basic Ranker

A basic ranker is a Deep Neural Network that outputs a score and its corresponding rank.

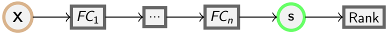


Figure: Basic ranker.

Given a set of inputs **X**, the ranker outputs the following values

- Output (or) relevance scores of input elements. These values are in the **Output Space**.
- Ranks of the input elements. These values are in the **Ranking Space**.

Note:

- Normal Loss functions work in the Output Space.
- Ranking Loss functions work in the Ranking Space.

Basic Ranker

A basic ranker is a Deep Neural Network that outputs a score and its corresponding rank.

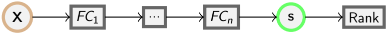


Figure: Basic ranker.

Advantages of working in the ranking space:

- Ranking is agnostic to affine transformations of score i.e $\alpha * s + \beta$ where $\alpha, \beta \in \mathbb{R}$.
- Larger target output space leads to easier convergence/learning.
- Ranks from different rankers can be combined easily.

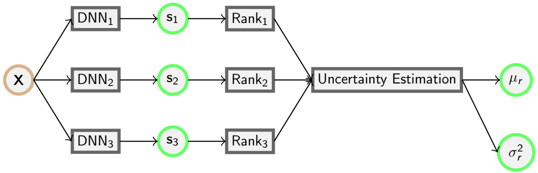


Figure: Ensemble of Basic Rankers.

- Ensemble of Basic rankers gives a list of ranks for every input element.
- Gaussian uncertainty is calculated in the ranking space. The mean and standard deviation are calculated using the following formulae:

$$\mu_r = \frac{\sum_{i=1}^{m} y_i}{m} \qquad \sigma_r = \frac{\sum_{i=1}^{m} (y_i - y_{mean})^2}{m}$$

Making model context aware

The model is made context-aware by using Deep Sets.

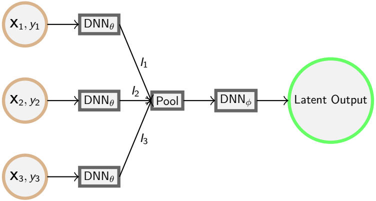


Figure: Deep Set Architecture.

- The architecture consists of 3 separate components: DDN$_\theta$, Pool, and DNN$_\phi$.
- Pooling operator $\frac{l_1+l_2+l_3}{3}$ takes care of:
  - Permutation-Invariance constraint.
  - Set cardinality invariance constraint.

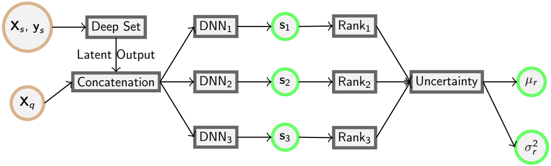


Figure: Skeleton of the proposed model with Deep Sets.

- Support set, i.e., known HP configurations and their evaluations are passed through Deep Set to get a latent output.
- The query HP configurations are concatenated with the latent output.
- This concatenation is passed through the ensemble of deep rankers.
- During training/finetuning, losses are aggregated in the ranking space.

Meta-training the surrogate

Algorithm Meta-training Surrogate with Deep Set.

Input : epochs $\in \mathbb{I}$

Input : batch size $\in \mathbb{I}$

Input : X_{train}, y_{train} ▷ Metadata to train from.

Input : DS_ϕ ▷ Deep set.

Input : $R_{\theta_1}, \ldots, R_{\theta_m}$ ▷ Basic rankers.

procedure MetaTrain(X_{train}, y_{train}, epochs, DS_ϕ, $R_{\theta_1}, \ldots, R_{\theta_m}$)
    for $i <$ epochs * batch size **do**
        $B_X, B_y \leftarrow$ DoubleSample(X_{train}, y_{train}) ▷ Training batch.
        $S_X, S_y, Q_X, Q_y \leftarrow$ Split(B_X, B_y)
        $LO \leftarrow DS_\phi(S_X, S_y)$ ▷ Latent output from deep set.
        $y_1 \leftarrow R_{\theta_1}(Q_X : LO)$
        ...
        $y_m \leftarrow R_{\theta_m}(Q_X : LO)$
        $l_1 \leftarrow$ ListMLEWeighted(y_1, Q_y)
        ...
        $l_m \leftarrow$ ListMLEWeighted(y_1, Q_y)
        loss $\leftarrow$ Mean(l_1, ..., l_m)
        Backpropogate the loss through $R_{\theta_1}, \ldots, R_{\theta_m}$ and DS_ϕ.
    end for
end procedure

Experiments and Results

Q1: Which loss function is better?

We compare the following loss function types:

- Regression Loss (Ref. RMSE)
- Pointwise ranking loss (Ref. Subset Regression)
- Pairwise ranking loss (Ref. RankNet)
- Listwise ranking loss (Ref. ListMLE)

Deep Ensemble baseline with a legend, "DE-M10-RS", is used as a reference. 2 cases have to be compared separately:

- Non-transfer HPO
- Transfer HPO

Q1: Which loss function is better? - Non-transfer HPO

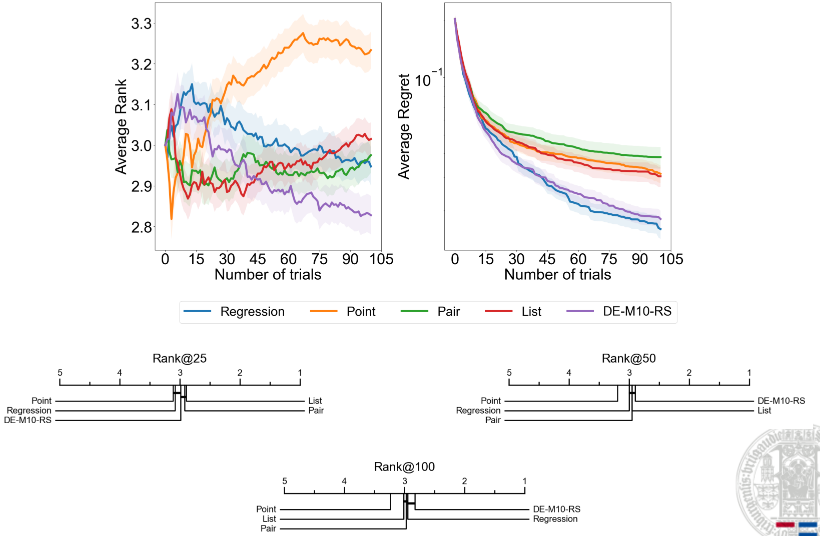


Figure: Loss comparisons for non-transfer HPO.

Q1: Which loss function is better? - Transfer HPO

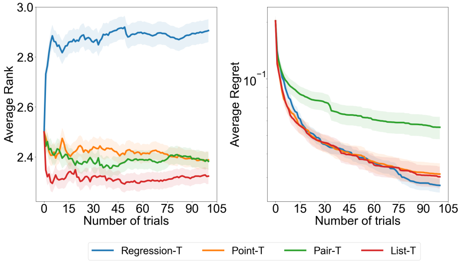


Figure: Loss comparison for transfer HPO. "T" stands for transfer.

Conclusion: Listwise ranking loss on average performs better than other losses.

Q2: Does weighting Listwise loss improve results?

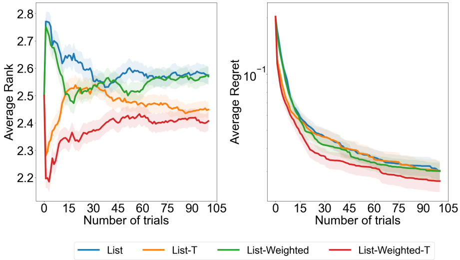


Figure: Listwise losses with and without weighting.

- Inverse logarithmic weighting was used for both the transfer and non-transfer cases.
- **Conclusion: Weighting improves the results for both transfer and non-transfer HPO.**

Q3: Does adding Deep Sets improve performance?

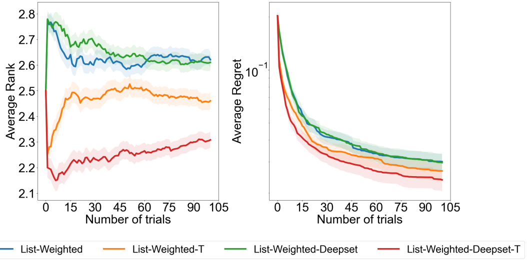


Figure: Effect of adding deep sets into the architecture.

- 20% of data is used as support set for both training and finetuning.
- **Conclusion**:
  - **Non-transfer HPO results become worse. Reason is the high complexity of the model.**
  - **Transfer HPO results become very good in the initial steps.**

Q4: Does finetuning improve performance?

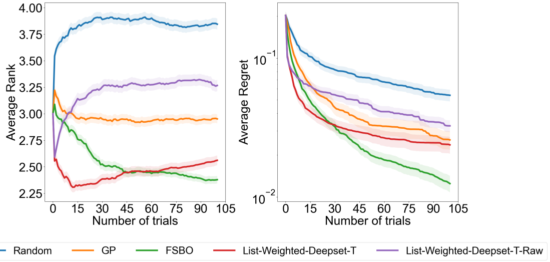


Figure: Effect of using finetuning.

- Here, "Raw" means a meta-trained model without finetuning.
- Not doing finetuning amounts to ranking the complete list of HP configuration and evaluating it.
- Using finetuning means learning the new context.
- **Conclusion: Finetuning is required for better long-term results.**

Non-transfer HPO

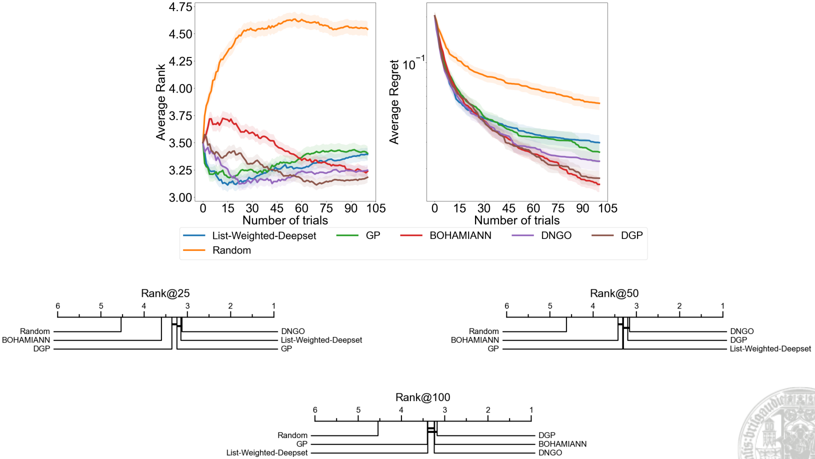


Figure: Graphs of the proposed model with other SOTA non-transfer HPO surrogates.

Comparing with SOTA HPO models

Transfer HPO

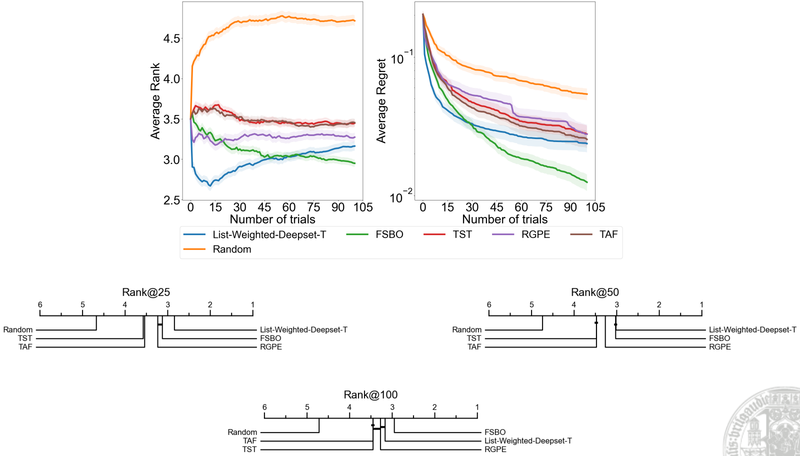


Figure: Graphs of the proposed model and SOTA transfer HPO surrogates.

Comparing with SOTA HPO models

Conclusion

- Non-transfer surrogate shows performance similar to the GP surrogate.
- Transfer surrogate performance is on par with the state-of-the-art FSBO model.
- Both transfer and non-transfer version of the model perform very well in the initial evaluation steps.

UCB and EI in the ranking space

- All previous experiments in our thesis used the average rank acquisition function.
- UCB (Upper Confidence Bound) is calculated using the mean and standard deviations returned from the ensemble of rankers.
- EI (Expected Improvement) is calculated by passing the incumbent HP in both the query and the support set.
- **Conclusion**:
  - UCB and EI acquisition functions are superior even in the ranking space.
  - The proposed surrogate model with EI acquisition function performs better than FSBO in all optimization steps.

UCB and EI in the Ranking space

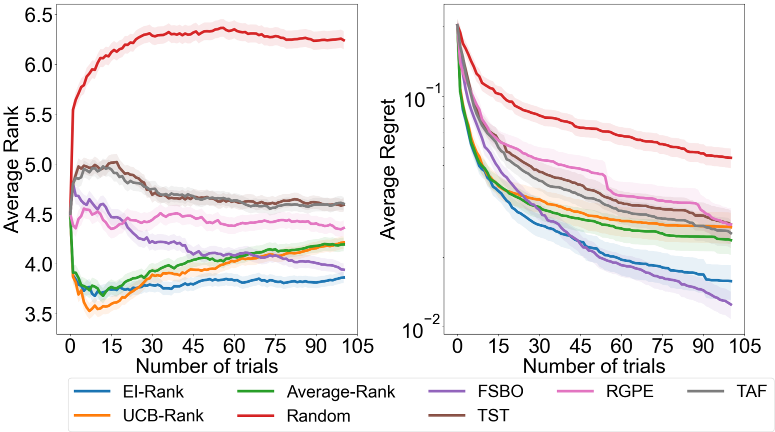


Figure: Ablation of Average-Rank, UCB-Rank, and EI-Rank with other SOTA transfer HPO methods.

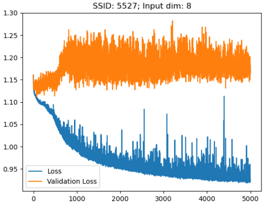
Conclusion

Advantages:

- Being a transfer HPO surrogate, it utilises already existing metadata.
- Sampling mechanism of listwise loss is superior. For example, from a set of 100 known HP configurations, there are $\binom{100}{15}$ ways to select a set of 15 items.
- Working in ranking space dampens uncertainty. Slight noise in the HP evaluations does not change the ranks of the configurations.
- Output ranges of HP evaluations across tasks do not affect the HP rankings.

Limitations:

- We ignore the sorting functionality during the ranking loss formulation. Hence the complete ranking problem is not optimized.
- We use exponentiation as the strictly increasing positive function. Hence there is a potential for overflow or underflow during optimization.
- The learned surrogate may be biased towards tasks with lesser data due to the double sampling mechanism.
- Negative transfer learning was seen in some search spaces.



Further Research Directions

- More research needs to be done for acquisition functions in the ranking space.
- Study needs to be conducted to incorporate sorting functionality into the ranking losses.
- One could explore how to incorporate learning the mean and the variance of the rank directly in the ranking loss functions instead of explicit calculation.
- The proposed ranking surrogate only works in the discrete HP search space. We need to study how to incorporate this in the continuous HP Search Space.

Conclusion

- The proposed model is an SMBO surrogate in the transfer HPO domain.
- The study consisted of 2 canonical parts:
  - Ranking loss functions
  - Surrogate design
- Our results showed that listwise ranking losses perform the best compared to other losses.
- After adding weighting & Deep sets, the surrogate performs comparably to FSBO (SOTA transfer HPO model).
- **Key takeaway - For HPO, working in the ranking space is better than the output space!**

Q & A

References

- Xia et al.; Listwise approach to learning to rank: theory and algorithm (2008)
- Burges et. al; Learning to rank using gradient descent (2005)
- Chen et. al; Top-Rank Enhanced Listwise Optimization for Statistical Machine Translation (2017)
- Cossock et. al ; Statistical Analysis of Bayes Optimal Subset Ranking (2008)
- Wistuba et. al; Few-Shot Bayesian Optimization with Deep Kernel Surrogates (2021)
- Jamieson et. al; Non-stochastic Best Arm Identification and Hyperparameter Optimization (2015)

References

- Lakshminarayanan et al.; Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles (2017)
- Swezey et al.; PiRank: Scalable Learning To Rank via Differentiable Sorting (2020).

Appendix

HPO-B Benchmark

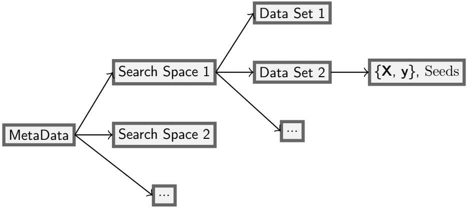


Figure: Structure of the metadata in the HPO-B benchmark.

- HPO-B is used for benchmarking BlackBox HPO.
- It consists of multiple Search Spaces (SS). An SS signifies a single ML model.
- Each SS consists of different Data Sets.
- A Data Set (task) contains HP configurations & their evaluations for a single training instance of the model.
- Seeds specify different initial configurations to start the SMBO process.

- The proposed surrogate is a transfer HPO surrogate used in SMBO.
- There are 2 stages of using it:
  - Meta training before SMBO optimization cycle.
  - Finetuning during the SMBO optimization cycle.

Meta-training

- Goal: Learn the common characteristics of all tasks and transfer knowledge to the new task.
- Separate surrogate is learned for each SS.
- Concept of double sampling is used, i.e., sample the task then from the task sample the metadata.
- Learning done using stochastic gradient descent for 5000 epochs.
- Used ADAM optimizer with a learning rate of 0.0001.

Surrogate usage

Finetuning

- Finetuning helps improve the HPO results.
- A full batch gradient is used with the ADAM optimizer to finetune for 1000 epochs.
- The finetuning is restarted at every SMBO evaluation step to get better performance.
- We use cosine annealing during the finetuning process.

Restarting is necessary during finetuning

This idea is counter-intuitive because an already trained model should converge faster to a local optimum. Reasons for this observation are:

- Model is biased towards already seen points.

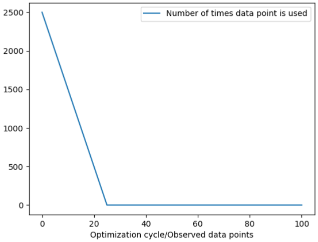


Figure: Bias at 25^{th} optimization cycle.

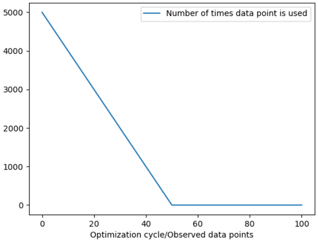
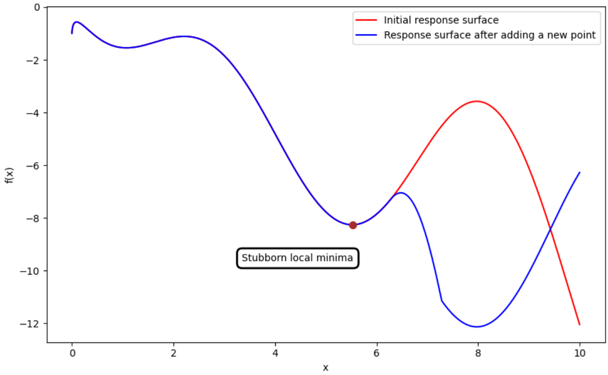


Figure: Bias at 50^{th} optimization cycle.

- Stubborn local minima
  - Response surface does not change significantly with the addition of only a single data point.
  - Restarting the training can increase chances of reaching good optima.



Subset Regression: Pointwise Loss

Basic idea:

- Learn the rank of the objects directly.
- Avoid intermediate relevance score.
- Similar to RMSE.

$$L_{\texttt{SubsetRegression}} = \frac{1}{N}\sum_{i=1}^{N}(s(a_i) - \texttt{rank}(y_i))^2$$

For example, ranking for the following ground truth values is,

$$\{y_1 = 0.8, y_2 = 0.9, y_3 = 0.1\}$$

$$\texttt{rank}(y_1) = 2, \texttt{rank}(y_2) = 1, \texttt{rank}(y_3) = 3$$

RankNet: Pairwise Loss

- Classify the 2 input objects.
- Answer the question: which object is better?
- From all possible pairs of inputs $\{s(a_1), s(a_2), y_1, y_2\}$, calculate loss.

$$L_{\texttt{RankNet}} = \texttt{C.E.Loss} = -P^* \log P - (1 - P^*) \log(1 - P)$$

where P and P^* are given by:

$$P(a_1 \succ a_2) = \frac{e^{s(a_1)-s(a_2)}}{1 + e^{s(a_1)-s(a_2)}}$$

$$P^*(a_1 \succ a_2) = \begin{cases} 1 & \text{if} \quad y_1 \geq y_2 \\ 0 & \text{otherwise} \end{cases}$$

Basically, RankNet loss is the binary cross entropy loss.

3 different weighting strategies were considered:

- Inverse linear weighting given by $c(j) = \frac{1}{j}$.
- Inverse logarithmic weighting given by $c(j) = \frac{1}{\log(j+1)}$.
- Position dependent ranking (scaled by 50 here) given by $c(j) = \frac{k-j+1}{\sum_{t=1}^{k} t}$ (Chen et. al).

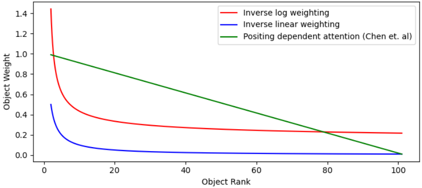


Figure: Different Weighting Functions.

HPO Constraints

HPO problem has the following unique constraints:

- It is a non-convex optimization problem.
- The HP evaluation and hence HPO objective is stochastic.
- Variables (dimensions) may be conditionally dependent on others. E.g., neurons in a layer are conditioned on the existence of the layer.
- Variables (dimensions) maybe discrete or continuous.

HPO Solutions

Various types of solutions to the HPO problem are proposed:

- **Blackbox HPO**: f is treated as a blackbox function. E.g. Random Search, Grid search.
- **Online HPO**: Hyperparameters and parameters of an ML model are trained together. E.g., Interleaved updating of the HP and parameters.
- **Multi-fidelity HPO**: Cheap approximations of the HPO objective function f called surrogates/fidelities are used. E.g., Successive halving [6].
- **Transfer Learning HPO**: Cheap Surrogates are meta learned using existing metadata. The meta-knowledge is transferred to the target HPO task. E.g. Few Shot Bayesian Optimization (FSBO) [5].

SMBO - Sequential Model based Optimization. The following steps are performed in a chronologically:

- From known data $D = (x_1, f(x_1)), (x_2, f(x_2)), (x_3, f(x_3))....,$ build a probabilistic surrogate model of the objective function.
- Use the surrogate and an acquisition function to sample the next best HP configuration x'. Evaluate $f(x')$.
- Append $(x', f(x'))$ to D and repeat the process.

Our Target:

- Design a surrogate that uses the concept of ranking to learn the objective function.
- We propose a context-aware surrogate design.
- Our proposed method is a Transfer HPO method.