

Neural Network Hyperparameter Optimization with Sparse Grids

Maximilian Michallik

Technical University Munich

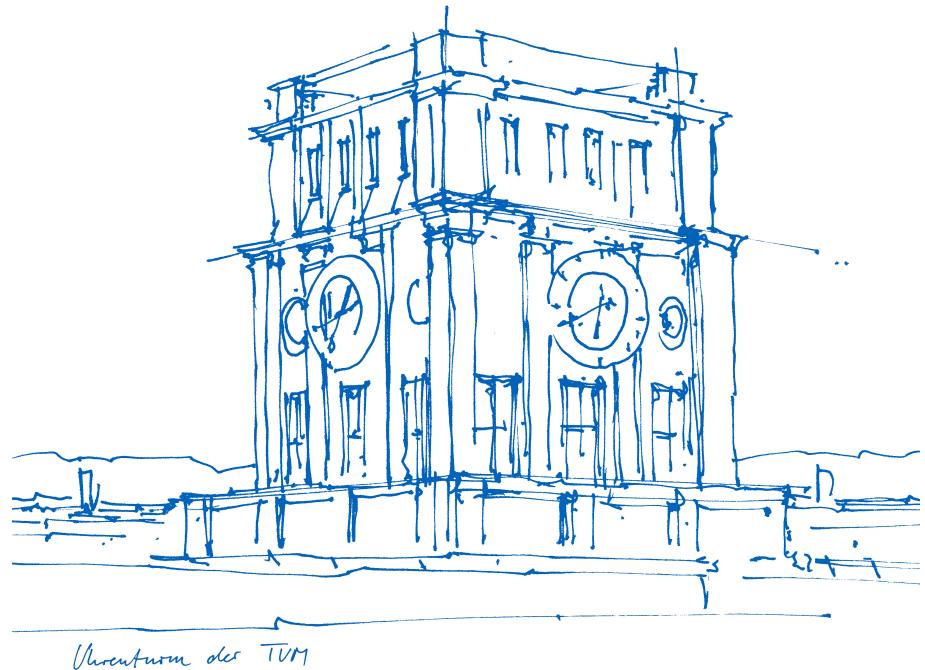
Chair of Scientific Computing

Dr. Felix Dietrich, Dr. Michael Obersteiner

Department Computer Science

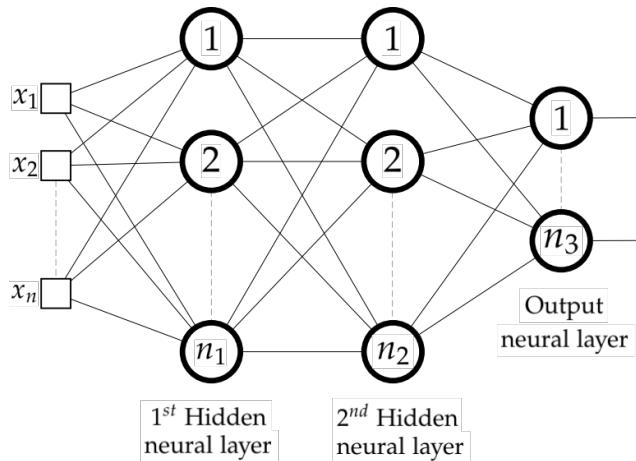
TUM School of Computation, Information
and Technology

Garching, 02. August 2023



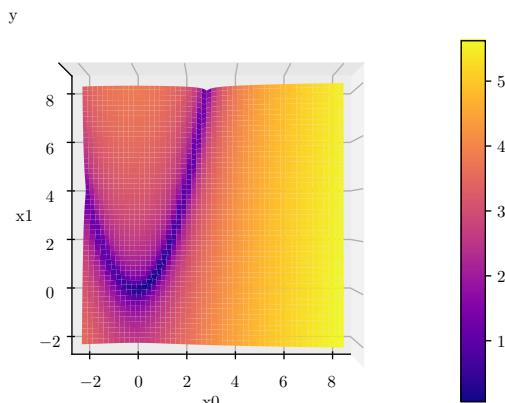
Hyperparameter Optimization

- **Hyperparameters:** Parameters of a ML model that are fixed before training



Number of layers/ neurons, epochs, learning rate, ...

- **Optimization:** finding the optimum of a function

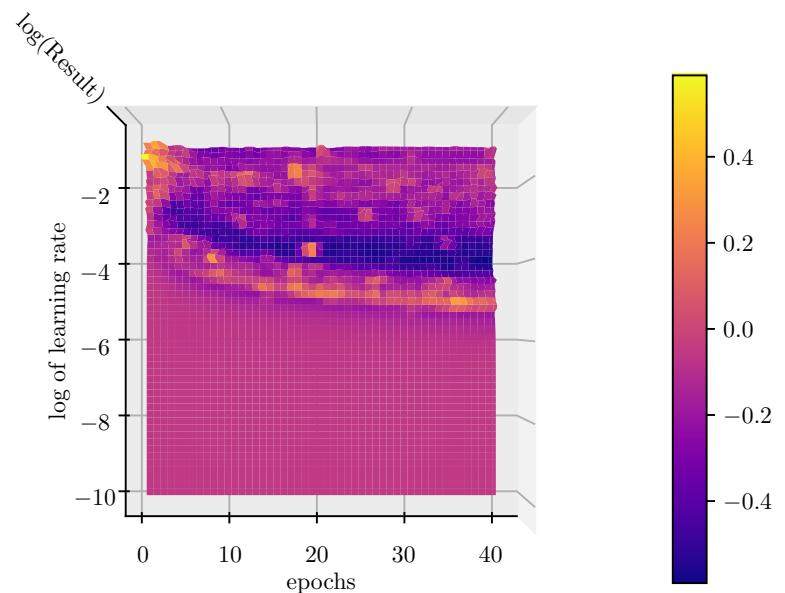


Rosenbrock: Optimum at (1, 1)

Hyperparameter Optimization

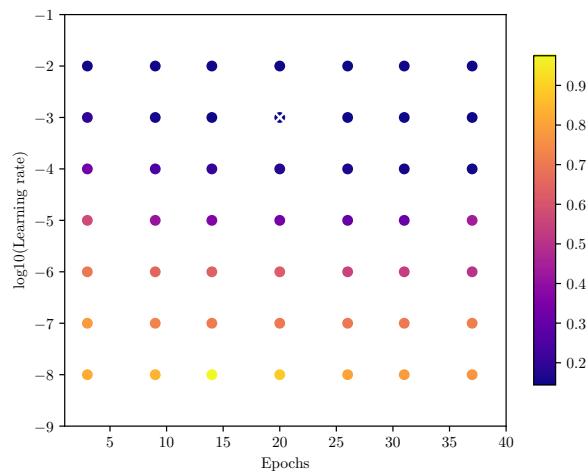
Regression with 2-layer neural network:

- Each function evaluation: 2-fold cross-validation of the network → expensive
- Final value: mean average percentage error on test set (average of 2-fold cross-validation)

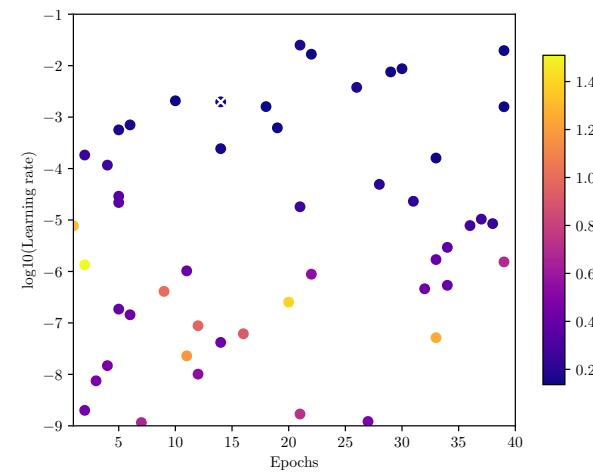


Hyperparameter Optimization – State of the Art

Grid search:



Random search:



Hyperparameter Optimization – State of the Art

Bayesian optimization:

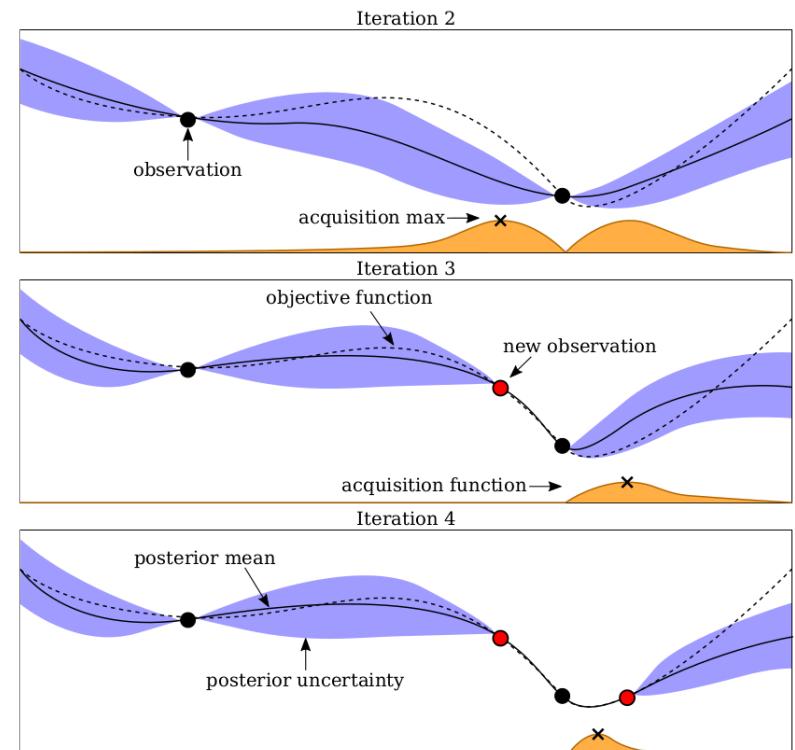
- **Surrogate model:** approximates the function, e.g. gaussian process, random forest, ...
- **Acquisition function:** finds new candidates for the optimum, e.g. expected improvement:

$$E[I(\lambda)] = E[\max(f_{\min} - y, 0)]$$

λ : configuration

f_{\min} : best value so far

y: model prediction



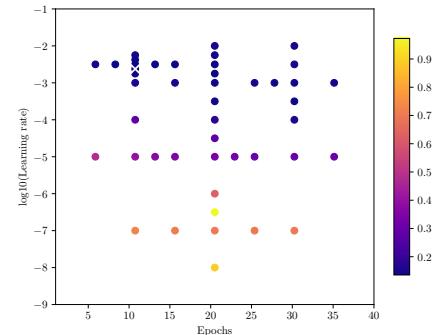
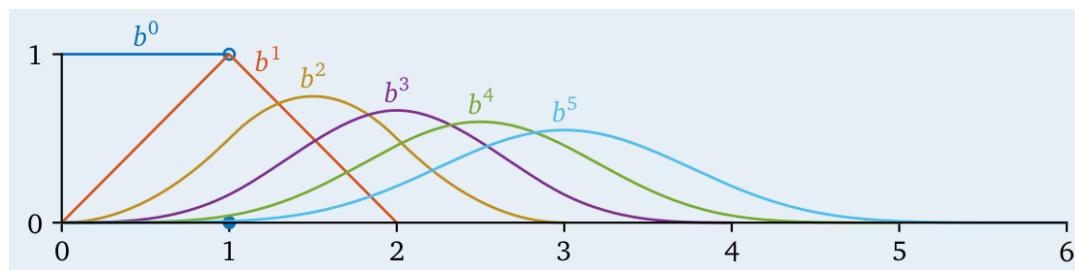
Hyperparameter Optimization with Sparse Grids

Background:

- Efficient way for representing functions:

Grid	Number grid points	Error
Full grid	$\mathcal{O}(2^{nd})$	$\mathcal{O}(h_n^2)$
Sparse grid	$\mathcal{O}\left(h_n^{-1} (\log h_n^{-1})^{d-1}\right)$	$\mathcal{O}\left(h_n^2 (\log h_n^{-1})^{d-1}\right)$

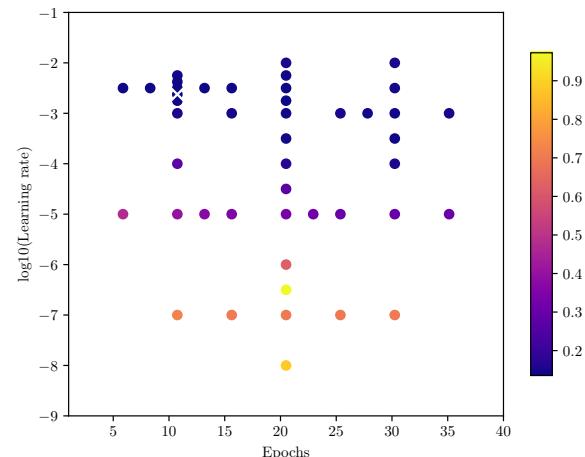
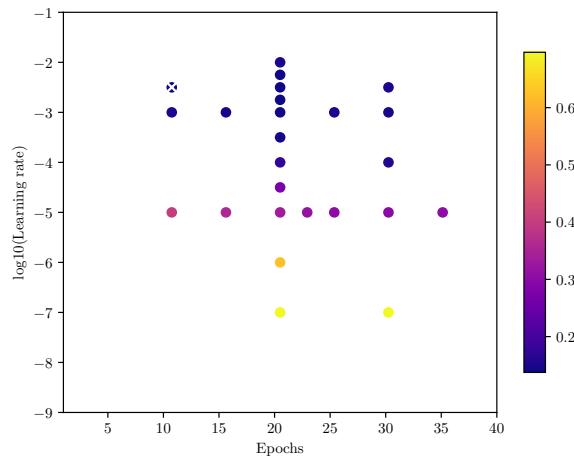
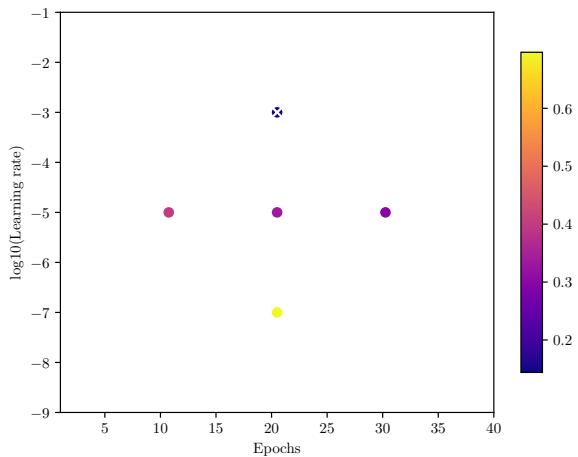
- Interpolation with B-splines as basis function:



Example for a 2-dimensional, adaptive sparse grid

Sparse Grid Hyperparameter Optimization

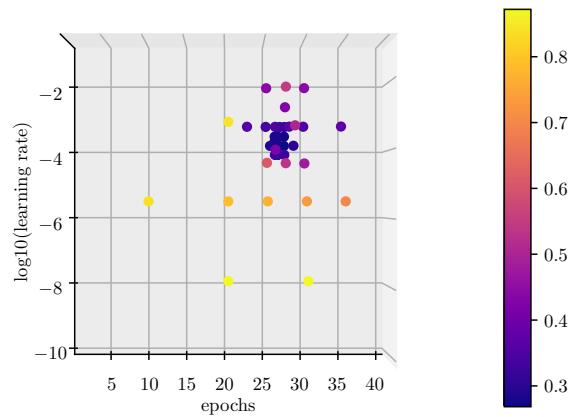
Iterative adaptive grid generation:



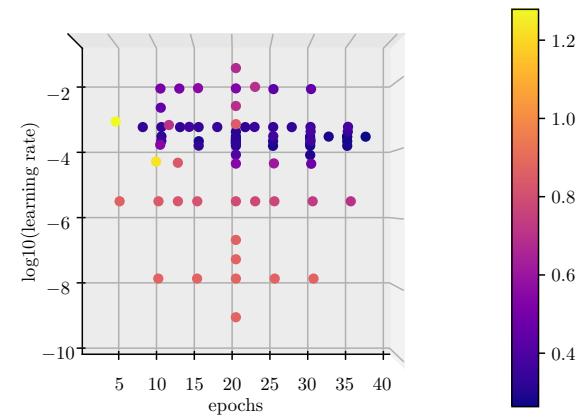
With Novak-Ritter refinement criterion: $(r_{l,i}+1)^{1-\gamma} \cdot (\|l_1\| + d_{l,i} + 1)^\gamma$

Adaptivity Parameter γ

$$\gamma = 0.5$$



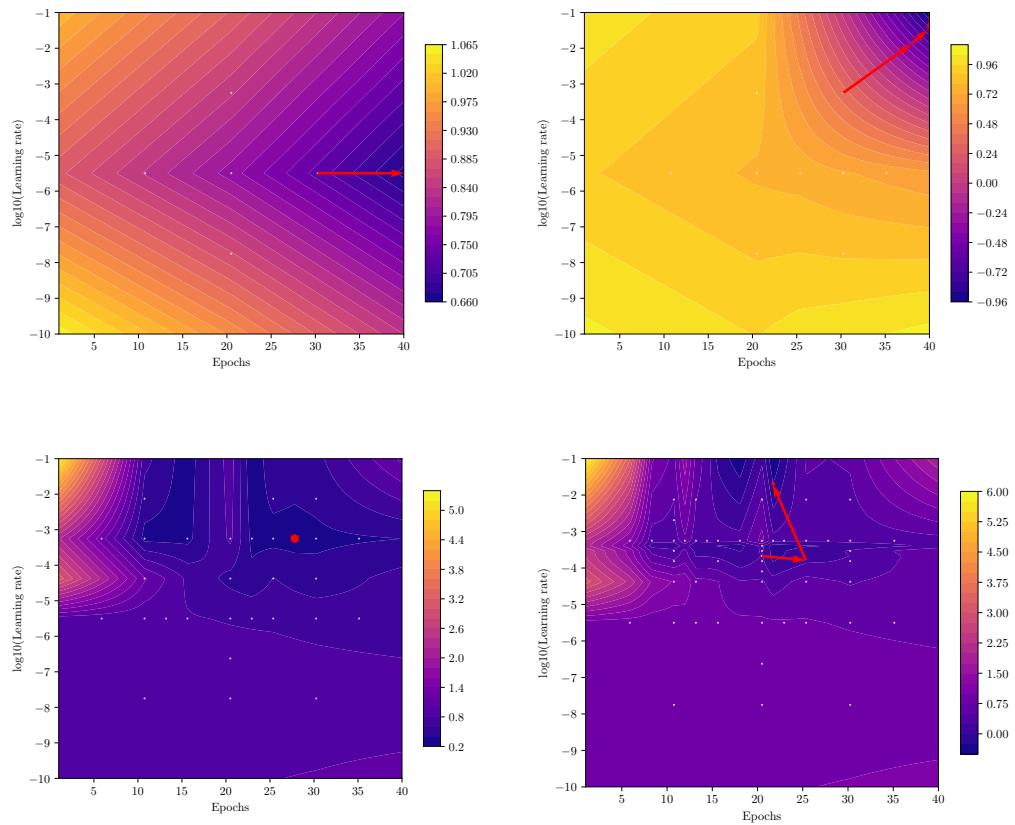
$$\gamma = 0.85$$



With Novak-Ritter refinement criterion: $(r_{l,i}+1)^{1-\gamma} \cdot (\|l_1\| + d_{l,i} + 1)^\gamma$

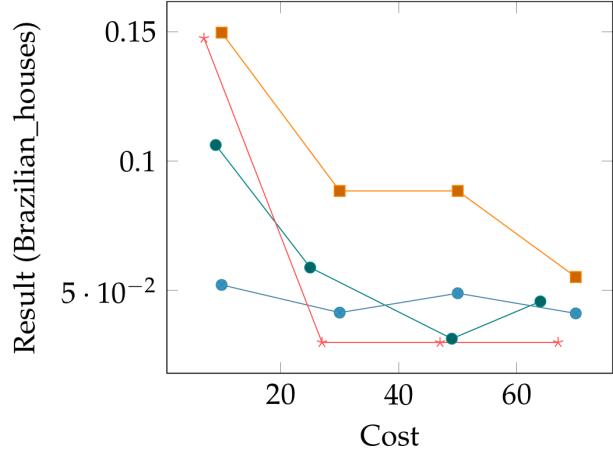
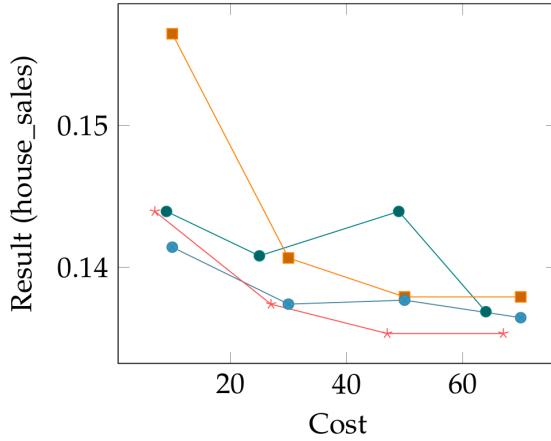
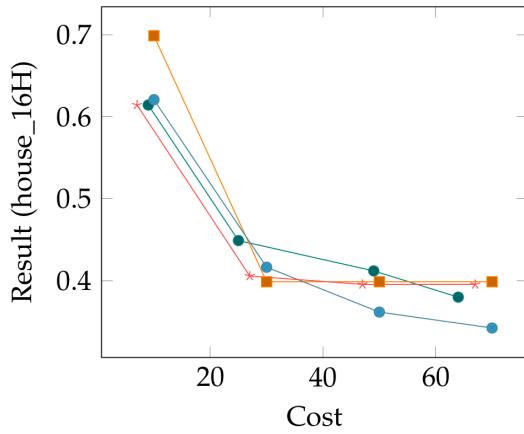
Sparse Grid Hyperparameter Optimization

- **Gradient-free optimizers:**
Nelder Mead, differential evolution, CMA-ES
- **Gradient-based optimizers:**
Gradient descent, NLCG, Newton, Rprop



Numerical Results

2D Optimization: Epochs, Learning rate

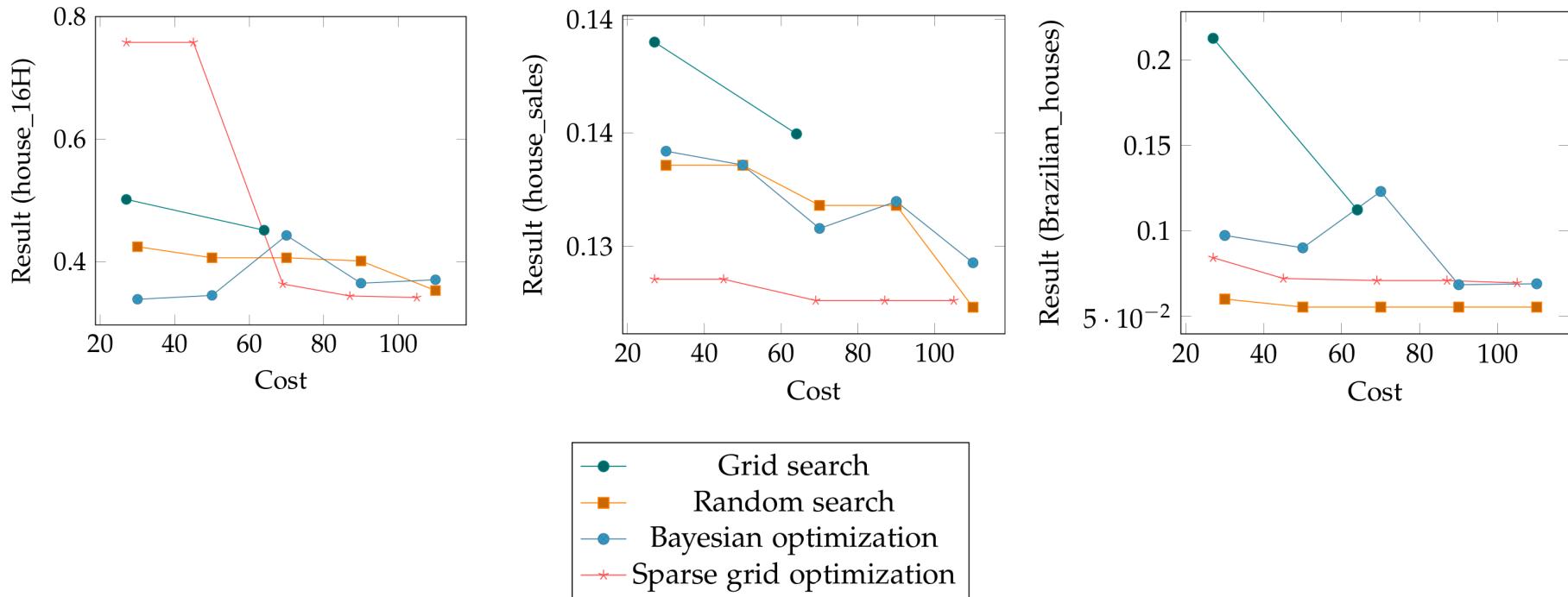


Legend:

- Grid search
- Random search
- Bayesian optimization
- ★ Sparse grid optimization

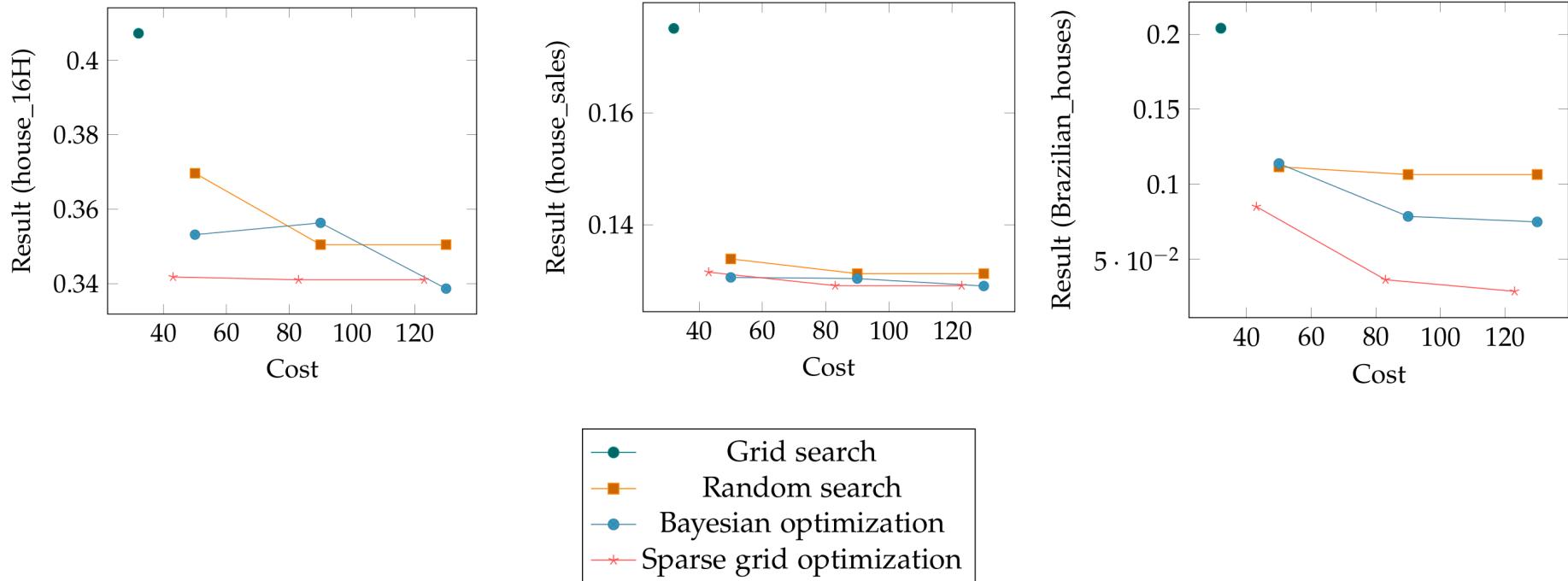
Numerical Results

3D Optimization: Epochs, Learning rate, Batch size



Numerical Results

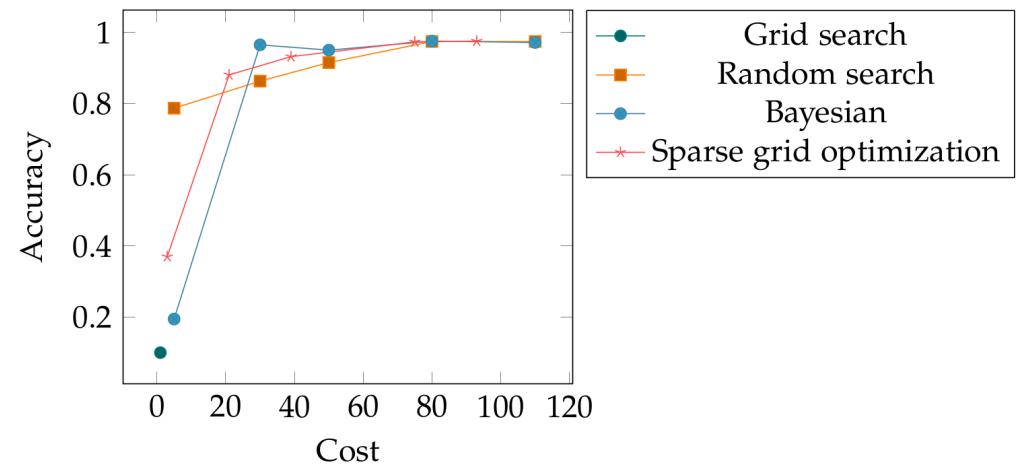
5D Optimization: Epochs, Learning rate, Batch size, Number of layers & neurons per layer



Application: MNIST

9D Optimization:

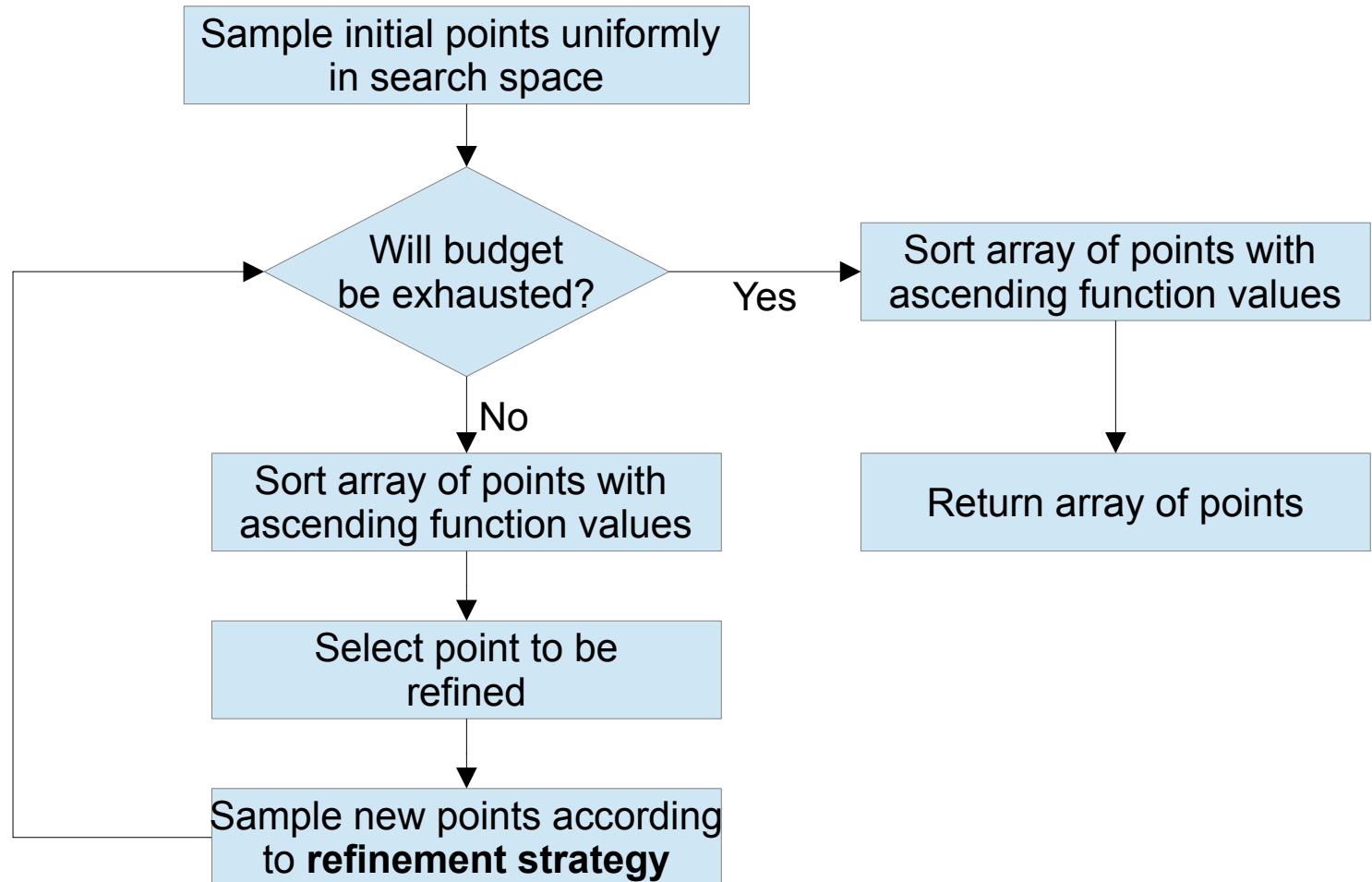
- Epochs
- Batch size
- Learning rate
- Number of convolutional Layers
- Number of fully connected layers
- Kernel size
- Pool size
- Neurons per fully connected
- Dropout probability



Algorithm	Configuration	Accuracy	Cost
GS	(5, 600, 10^{-6} , 2, 2, 2, 2, 4, 0.5)	10.1%	1
RS	(9, 975, 0.0173, 2, 1, 3, 1, 6, 0.619)	97.4%	80
BO	(6, 584, $10^{-2.17}$, 2, 1, 3, 1, 5, 0.281)	97.5%	80
SG	(7, 400, 10^{-2} , 2, 2, 2, 2, 5, 0.5)	97.5%	93

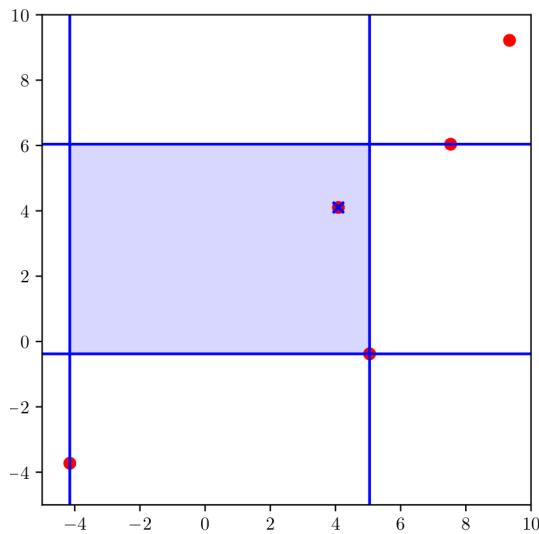
Iterative Adaptive Random Search

Algorithm:

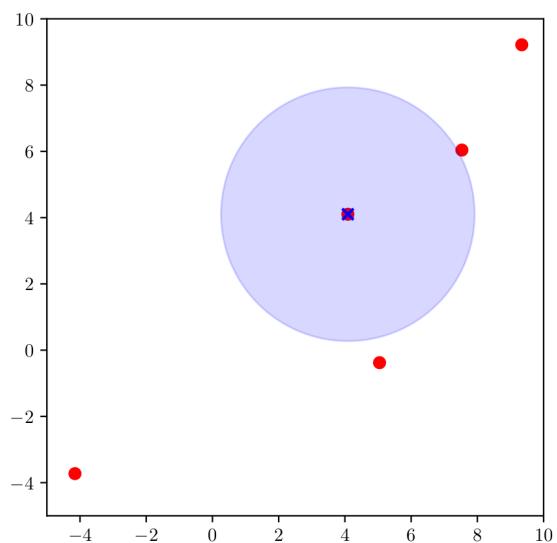


Refinement Strategies

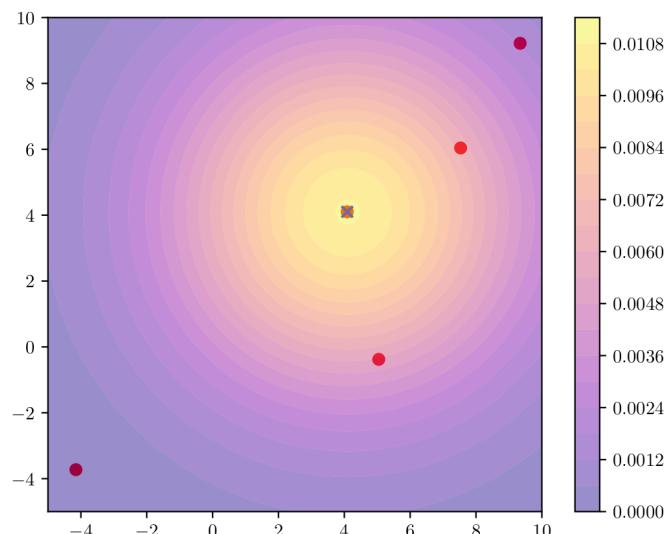
Interval based refinement



Uniform d-ball sampling



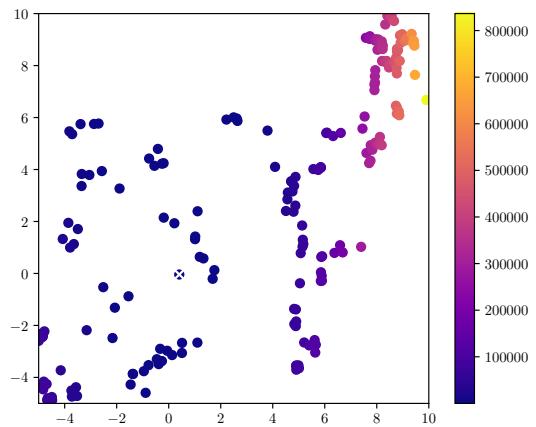
Normal distribution sampling



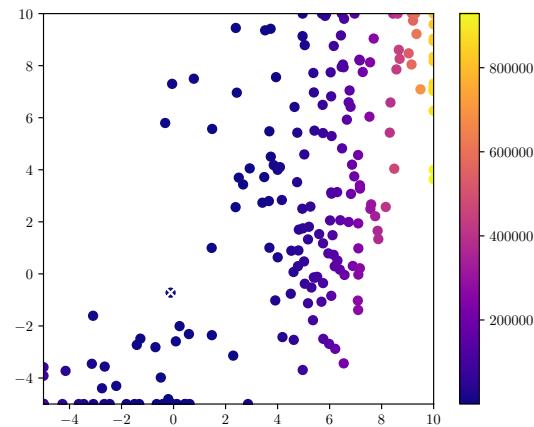
Refinement criterion: $(rank_i + 1)^{1-\gamma} \cdot (level_i + refinements_i + 1)^\gamma$

Adaptivity parameter $\gamma = 1.0$

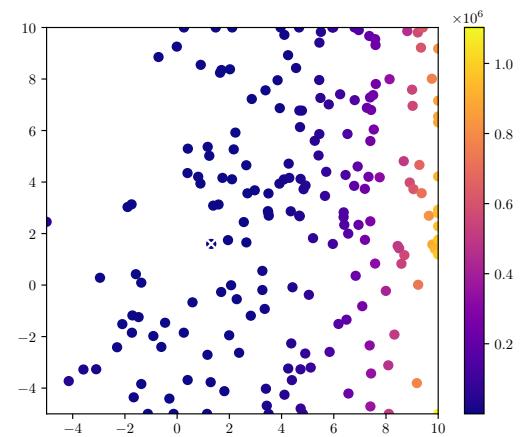
Interval based refinement



Uniform d-ball sampling



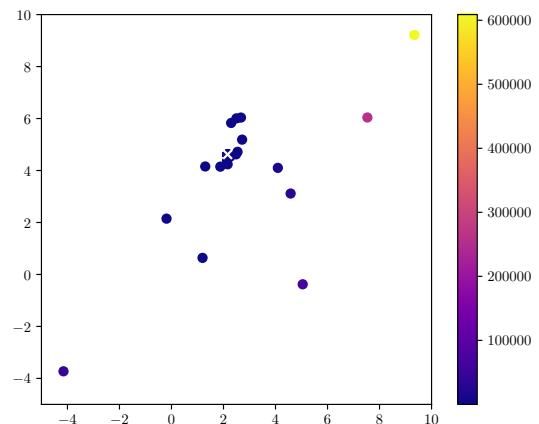
Normal distribution sampling



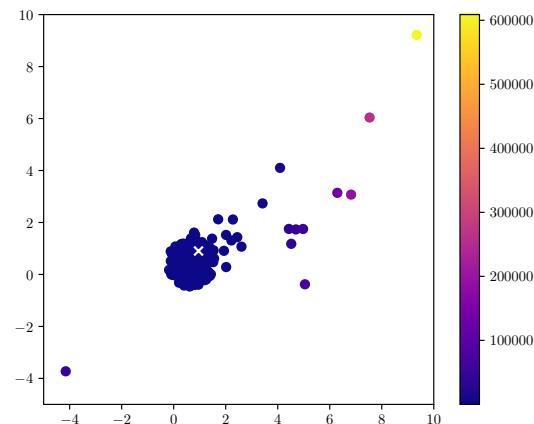
Refinement criterion: $\underbrace{(rank_i + 1)^{1-\gamma} \cdot (level_i + refinements_i + 1)^\gamma}_1$

Adaptivity parameter $\gamma = 0.0$

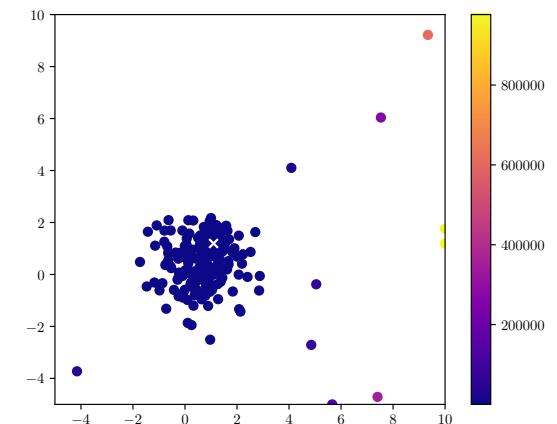
Interval based refinement



Uniform d-ball sampling



Normal distribution sampling

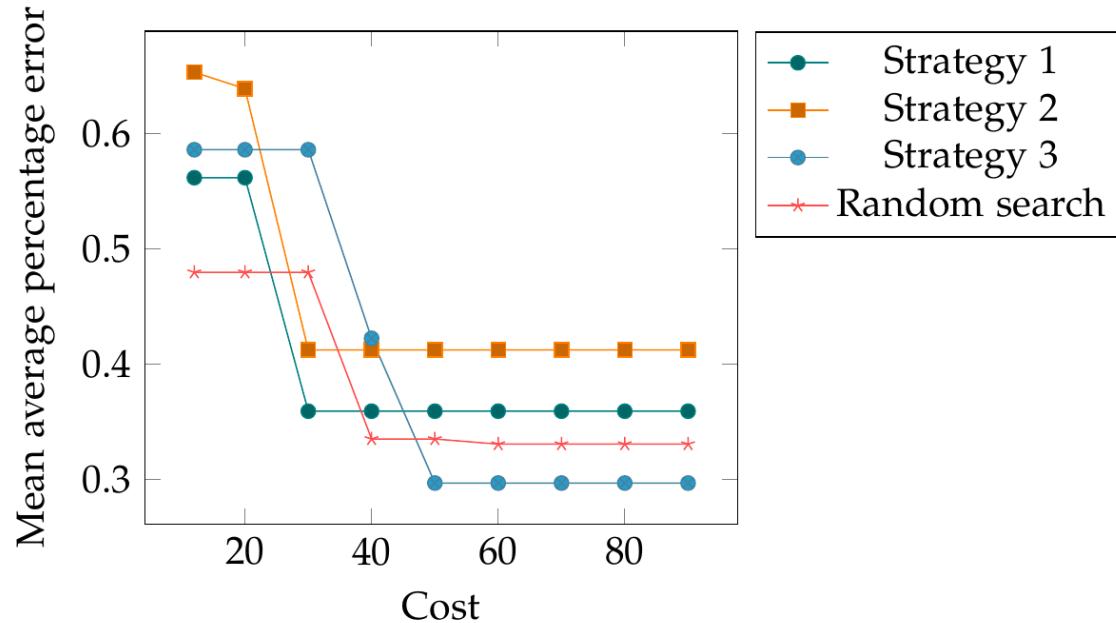


Refinement criterion: $(rank_i + 1)^{1-\gamma} \cdot \underbrace{(level_i + refinements_i + 1)^\gamma}_1$

1

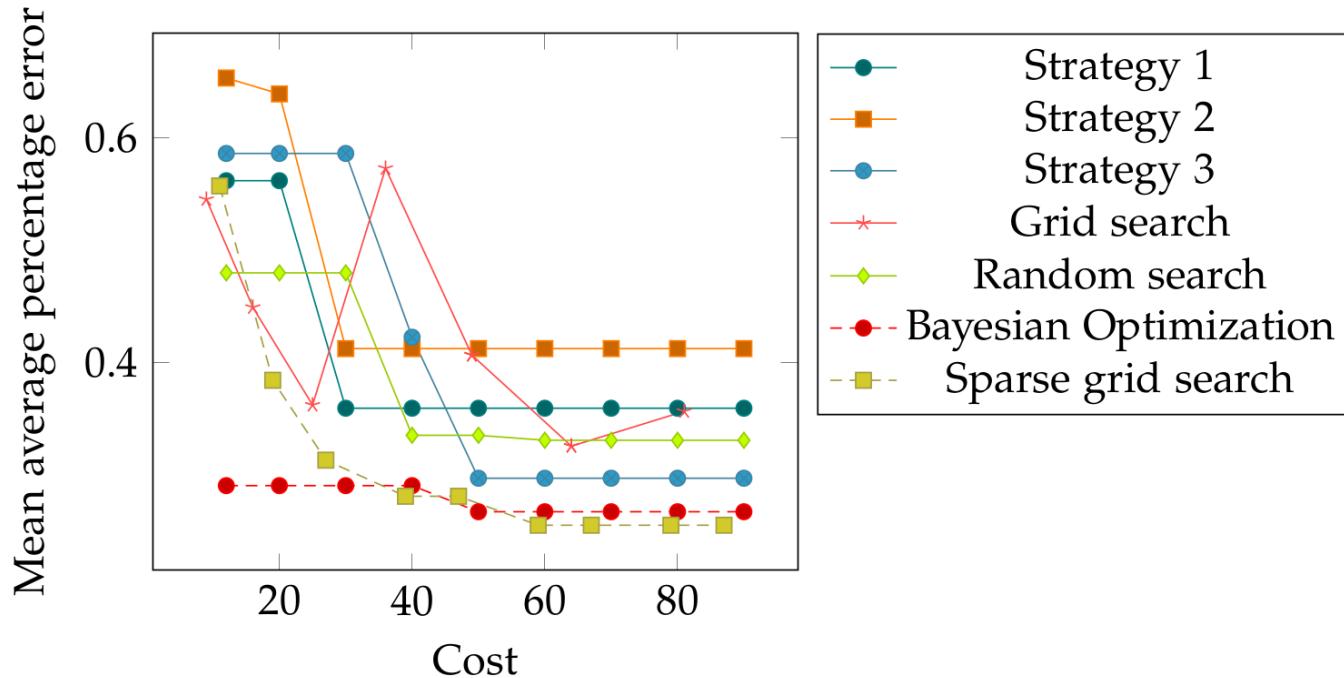
Comparison of Optimization Algorithms

2D Optimization: Epochs & Learning rate



Comparison of Optimization Algorithms

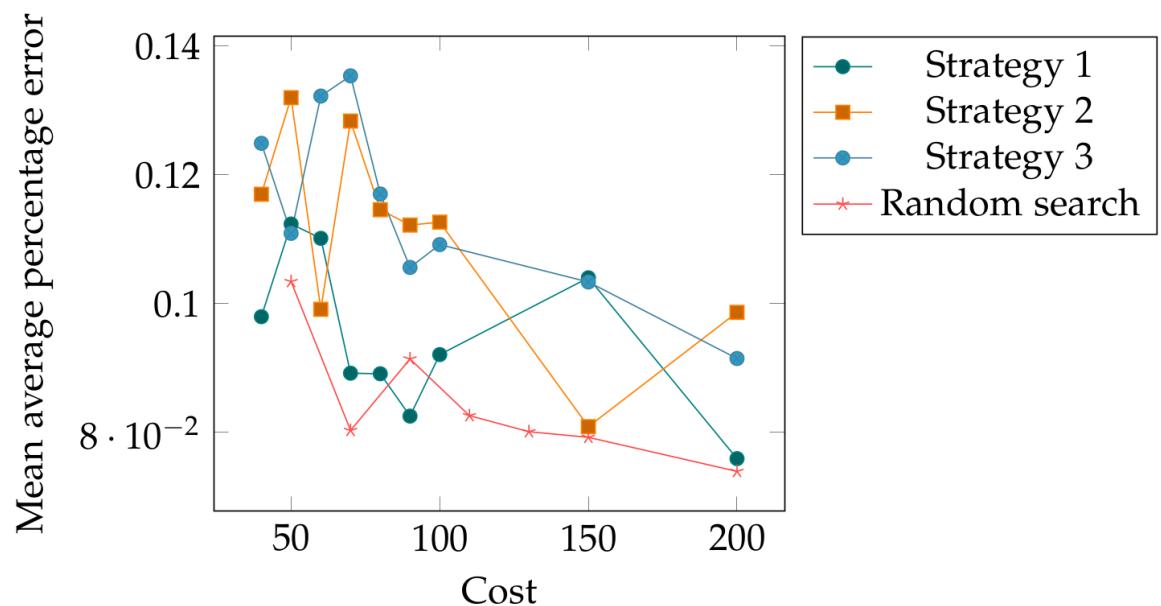
2D Optimization: Epochs & Learning rate



Comparison of Optimization Algorithms

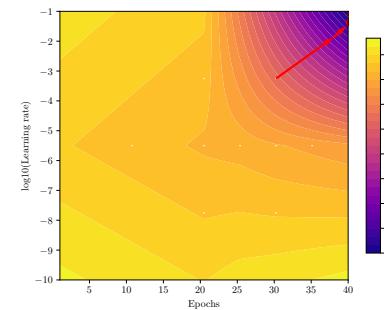
6D Optimization:

- Epochs
- Batch size
- Learning rate
- Number of layers
- Neurons per layer
- Dropout probability

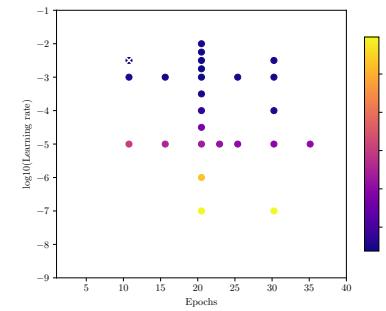


Discussion – Sparse Grid Hyperparameter Optimization

- **Often not enough grid points for optimizer**
→ apply global and local optimizer and compare with the best grid point



- **Problem if optimum at the boundary**
→ optimizer can sometimes help or use sparse grid with boundary points



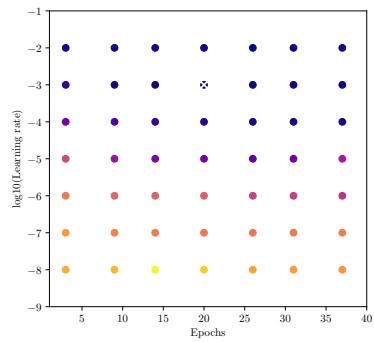
- **BUT: time to solution most important**
- Good results especially in higher dimensions

Discussion – Iterative Adaptive Random Search

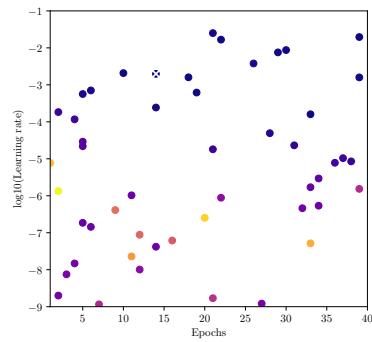
- Promising results in small dimension
- **Problem:** Volume of d-ball compared to d-hypercube decreases rapidly for increasing d
→ worse performance than conventional random search for higher dimensions
- More analysis:
 - Convergence?
 - Adapt algorithm?
 - Other search radius?
 - Change refinement criterion (e.g. include function value)?
 - Other refinement strategy?

Conclusion

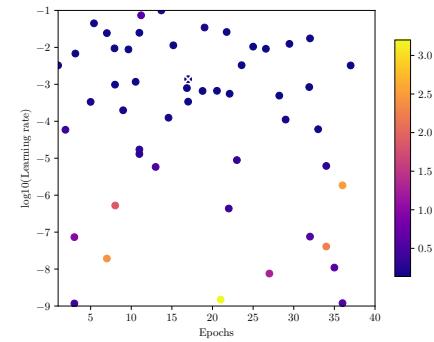
Grid search:



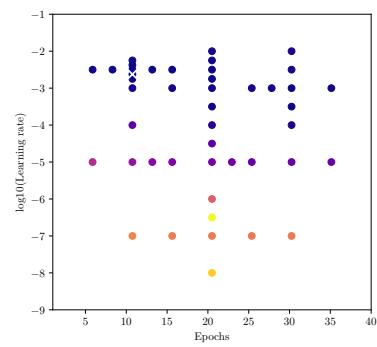
Random search:



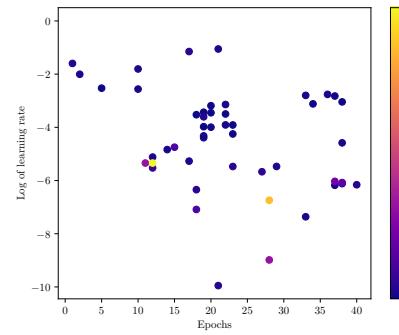
Bayesian optimization:



Sparse grid optimization:

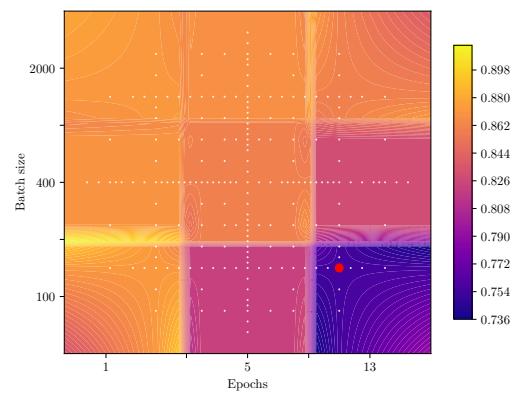
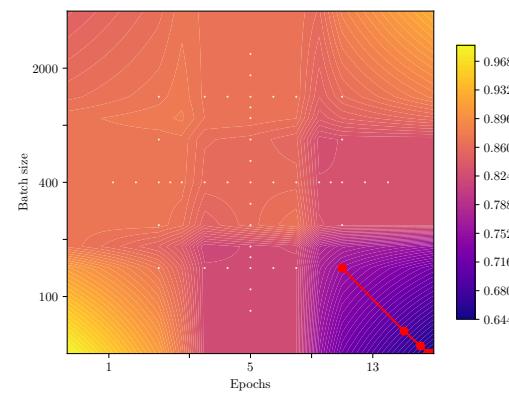
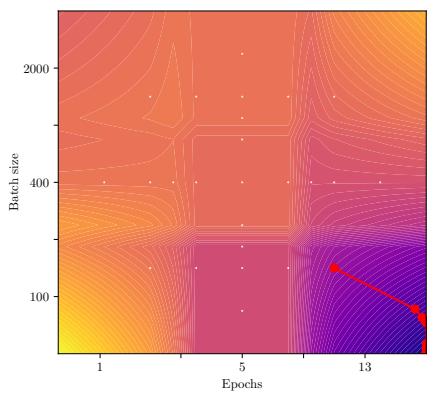


Iterative adaptive grid search:



→ best choice depends on the problem setting

Optimization – Categorical Hyperparameters



Examples: choice of optimizer as hyperparameter, rounding to integers