# Basic Search Algorithms – Intro

1. Hyperparam optimization recap

* The best hyperparams are those that maximizes the performance of the ML algorithm
* Hyperparam tuning – Search
* Hyperparam space (section 2)
* **Method for sample candidate hyperparams**
* Cross validation scheme (section 4)
* Performance metric to minimize/maximize (section 3)
* Hyperparam tuning – Challenges
* We can’t define a formula to find the hyperparams
* Try different combinations of hyperparam and evaluate model performance
* The critical step is to choose how many different hyperparam combinations we are going to test
* Hyperparam tuning – methods
* Low effective dimension
* Most params do not have a huge effect on model performance
* Some params affect performance a lot

1. Hyperparam nature

* Some hyperparams are discrete
* Number of estimators in ensemble models
* Some hyperparams are continuous
* Penalization coefficient
* Number of samples per split
* Some hyperparams are categorical
* Loss (deviance, exponential)
* Regularization (Lasso, Ridge)

1. Hyperparam tuning – Considerations

* Number of hyperparams of the ML model
* Low effective dimension
* Nature of params (discrete, continuous)
* Computing resources available

1. Basic hyperparam tuning methods

* Manual search
* Grid search
* Random search

# Manual search

1. Manual search of hyperparams

* Different hyperparams are set and experimented with manually

1. Uses

* Used to identify regions of promising hyperparams
* To delimit the Grid search
* Get familiar with the hyperparams and their effect on the models
* Establishing the benchmark model

1. Limitations

* Lack of reproducibility
* Time consuming
* Does not explore the entire hyperparam space
* Does not scale

# Grid Search

1. Grid Search of Hyperparams

* Exhaustive search through a specified subset of hyperparams of a learning algorithm
* Examines all possible combinations of the specified hyperparams
* Cartesian product
* Combinations: hyp1 \* hyp2 \* … \* hypn

1. Limitations

* Curse of dimensionality: possible combinations grow exponentially with the number of hyperparams
* Computationally expensive
* Hyperparam values are determined manually
* Not ideal for continuous hyperparams
* A subset of reasonable hyperparam values are set manually
* Does not explore the entire hyperparam space (not feasible)
* Performs worse than other searches (for models with complex hyperparam space)

1. Advantages

* For models with simpler hyperparam spaces works well
* It can be parallelized

1. Considerations

* Grid search is the most expensive method in terms of total computation time
* However, if run in parallel, it is fast in terms of wall clock time
* Sometimes, we run a small grid, determine where the optimum lies, and then expand the grid in that direction

# Random Search

1. Overview

* Hyperparam values are selected by independent (random) draws from a uniform distribution of the hyperparam space
* Selects the combinations of hyperparam values at random from all the possible combinations given a hyperparam space

1. Grid search of hyperparams

* Examines **all possible combinations** of the specified hyperparams

1. Random search of hyperparams

* Examines **some combinations** of the specified hyperparams, selected at random
* User determines the number of combinations to examine

1. Random search & Low effective dimension

* Some params affect performance a lot and some others don’t

1. Grid vs Random search

* Random search allows the exploration of more dimensions of the important param
* Grid search wastes time exploring non-important dimensions
* Random search selects values from a distribution of param values
* As opposed to Grid search where params are defined manually
* Random search is suitable for continuous hyperparams

|  |  |  |
| --- | --- | --- |
|  | **Grid search** | **Random search** |
| Parallelized | Yes | Yes |
| Effective in high dimension | No | Yes |
| Effective in low dimension | Yes |  |
| Suited for continuous hyperparam | No | Yes |
| Hyperparam values | Manually defined | Drawn from a distribution |

1. Advantages

* Can be parallelized
* High efficiency in high dimensional spaces
* Well suited for continuous hyperparams
* Small reduction in efficiency in low dimensional spaces

1. Considerations

* We choose a (computational) budget independently(ish) of the number of params and possible values
* Adding params that do not influence the performance does not decrease efficiency of the search (if enough iterations are allowed ~ 60 iterations)
* Important to specify a continuous distribution of the hyperparam to take full advantage of the randomization

# Hyperopt

1. Hyperopt

* Objective function to minimize -> fmin
* Hyperparam space -> hp
* Search algorithm -> **rand**, tpe, anneal

1. Search space

* The search space can be a dict, a list, or a tuple
* The search space can be nested