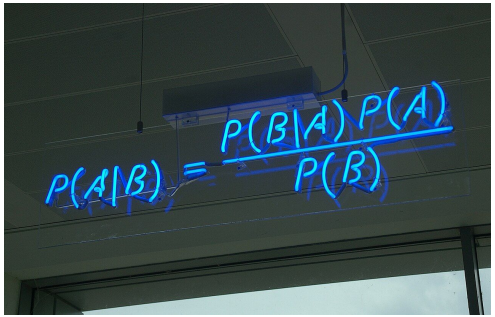
Plasma Science & Fusion Center

An introduction to Bayesian Optimization with Optuna

Enrique de Dios Zapata Cornejo



Outline

1. Optimization 101
2. Optuna
3. Hackathon (hard optimization problem)
4. Hyperparameter tuning in ML
5. Multiobjective optimization
6. Practical session

Disclaimers

1. Please, wait for questions at the end of each section
2. This not a fully Bayesian statistic tutorial
3. Gaussian processes are out of the scope of this tutorial

Optimization 101

What is optimization?

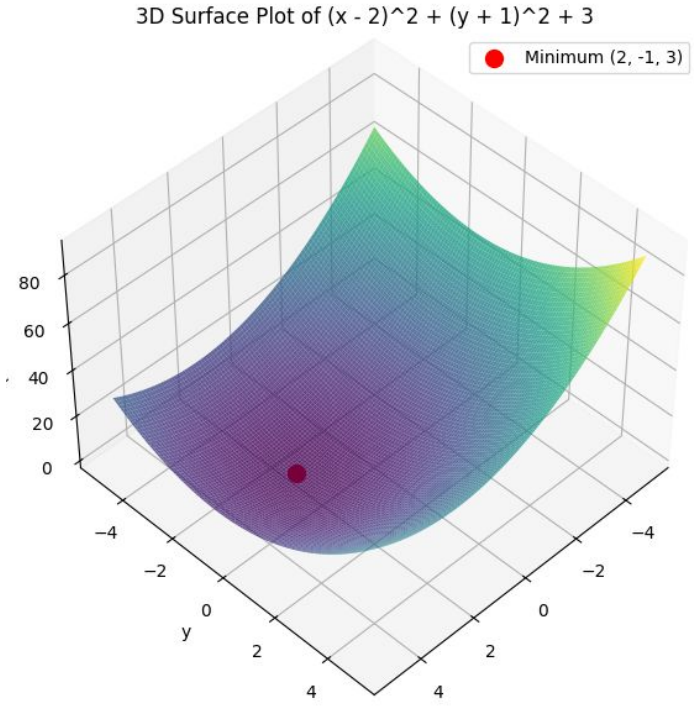
Optimization is the process of finding the solution x* for a function f(x) that maximize its output y, satisfying given constraints (optional).

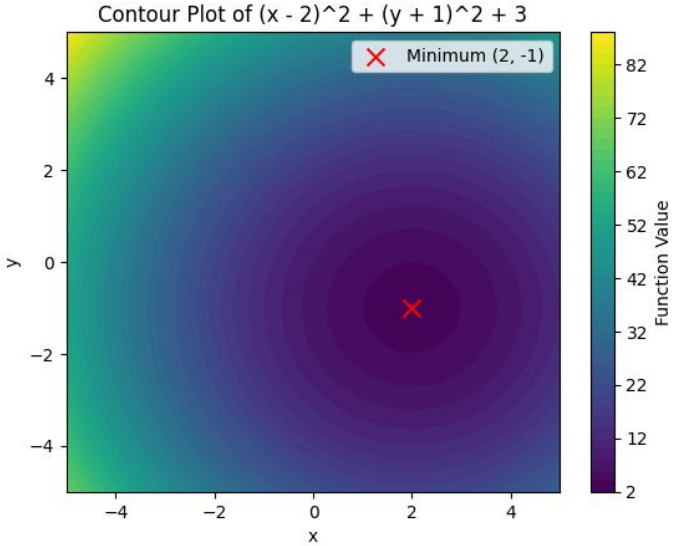
$$\mathbf{x}^\star = \arg\max_{\mathbf{x}\in\chi} f(x)$$

$$\chi = \text{design space}$$

$$\mathbf{x}^\star = \text{optimal solution}$$

Example 1





👉 python example1.py

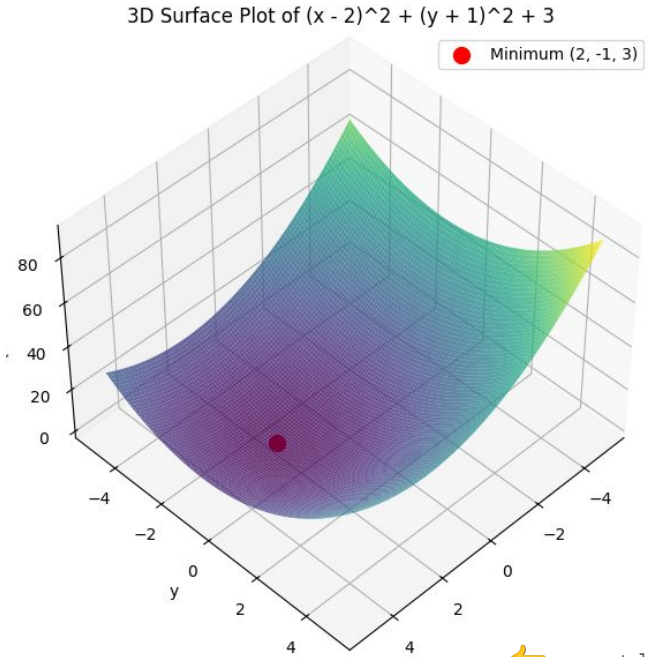
Why Bayesian optimization?

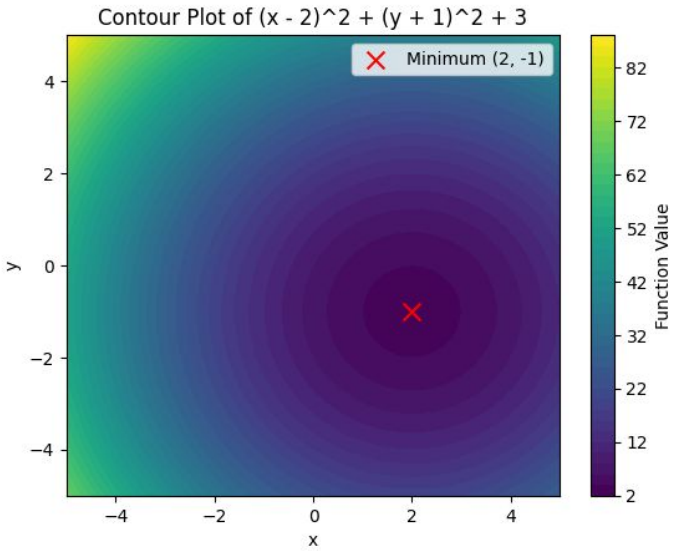
Some optimization methods.

Method	Pros	Cons
Grid/Random Search	Simple	Inefficient
Gradient Descent	Fast	Needs gradients (differentiable search spaces)
Bayesian Opt	**Sample-efficient**	More compute upfront

Example 1: Grid search

Minimum value of Z is 3.000100401203208 at (x, y) = (1.993987975951904, -0.9919839679358722), with 500000 function evaluations.

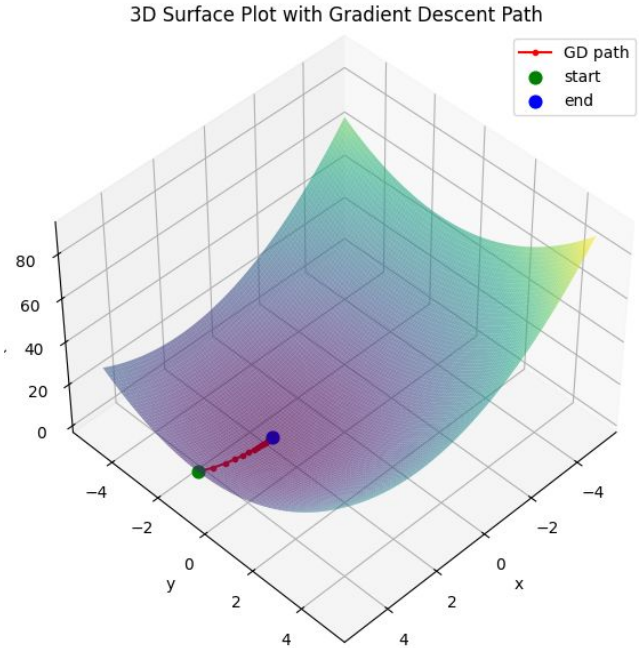


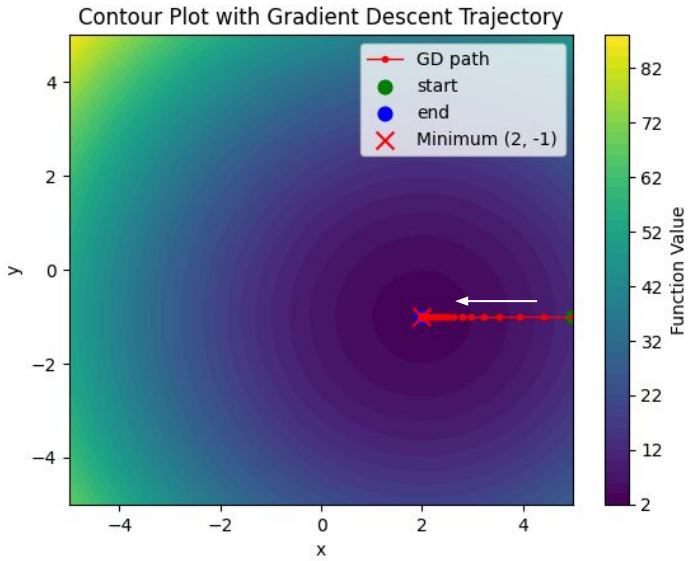


👉 python example1.py

Example 2: Gradient descent $\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta_t)$

Exact value of f(x,y) is 3.0 at (x, y) = (2.0, -1.0) reached after 85 iterations

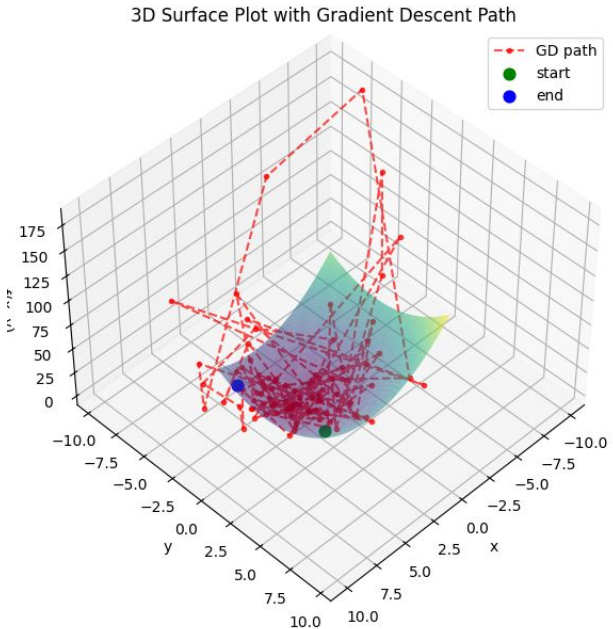


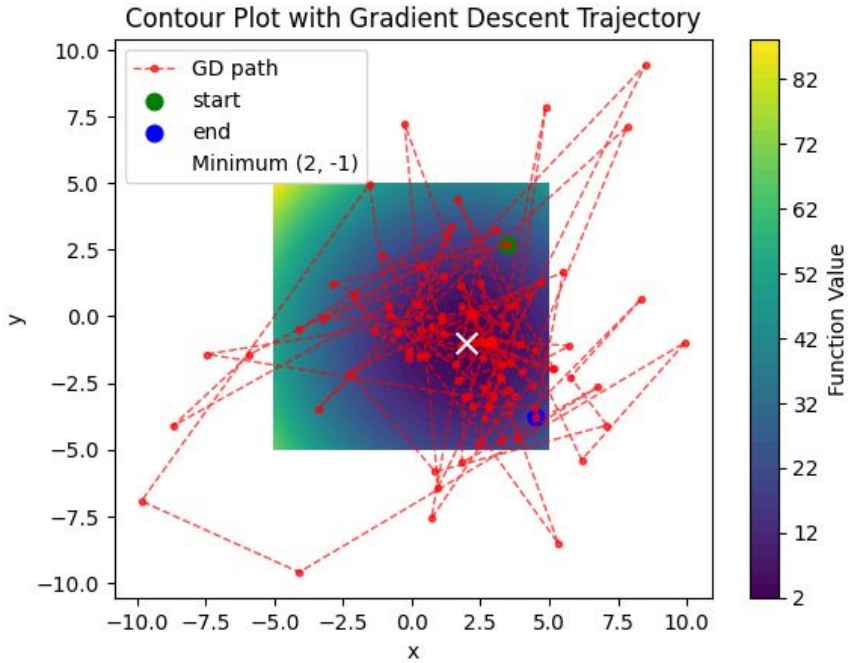


👉 python example2.py

Example 3: Bayesian Optimization (Spoiler!)

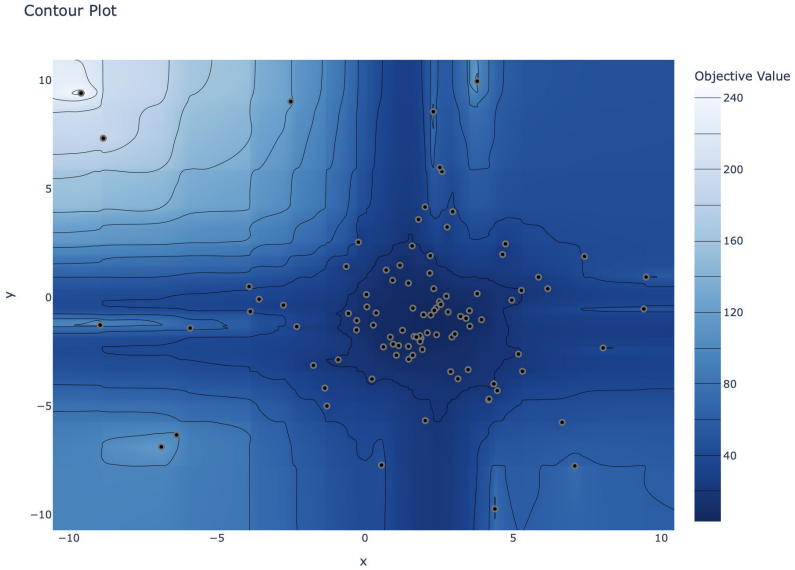
Minimum value of f(x,y) is 3.0156737909138682 at (x,y) = (1.893938637086194,-1.0665190063871524), with 100 trials.

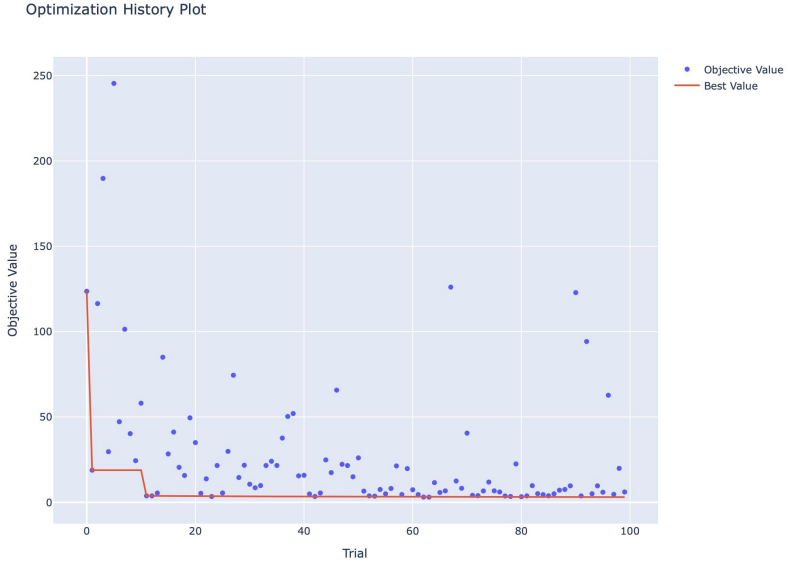




👉 python example3.py

Example 3: Bayesian Optimization (Spoiler!)





Best parameters: {'x': 1.893938637086194, 'y': -1.0665190063871524}
Best value: 3.0156737909138682
Number of trials: 100 (good value after ~12 trials)

Bayesian Optimization with OPTUNA

What is Bayesian optimization?

Bayesian optimization aims to find an optimal solution by building a probability model of the objective function and use it to chose the most proximing direction.

$$P(\text{score}|\text{hyperparameters})$$

This **probabilistic model** is a ***surrogate of the objective function***.

$$P(\text{y}|\text{x})$$

Based on Koehrsen. Medium. 2018

Bayes theorem rules conditional probabilities

Prior P(y) (the probability before the evidence is considered)

Posterior P(y|x) (updated probability after the evidence is considered)

Likelihood P(y|x) (probability of the evidence, given the prior belief is true)

Marginal P(x) (probability of the evidence, under any circumstance)

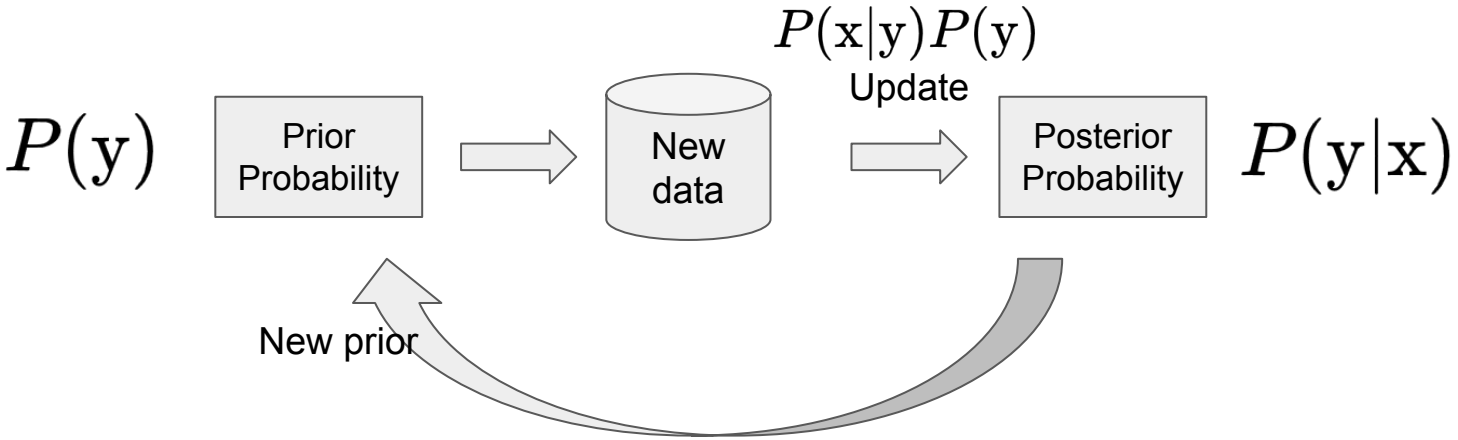
$$P(\mathrm{y}|\mathrm{x}) = \frac{P(\mathrm{x}|\mathrm{y})P(\mathrm{y})}{P(\mathrm{x})}$$

$$P(\mathrm{score}|\mathrm{parameters}) = \frac{P(\mathrm{parameters}|\mathrm{score})P(\mathrm{score})}{P(\mathrm{parameters})}$$

! ⚠️ Often likelihood integration is intractable, then other approaches/approximations are developed.

!! ⚠️ Bayesian ~ Probabilistic model

What is Bayesian optimization (BO)?



BO steps:

1. Build a surrogate *probability model* ⚠️ of the objective function
2. Find the hyperparameters that perform best on the surrogate (*Define acquisition function* ⚠️)
3. Try these hyperparameters in the true objective function
4. Update the surrogate model incorporating the new results
5. Repeat steps 2–4 until max iterations or time is reached

BO steps from Koehrsen. Medium. 2018

⚠️ Many options!

What is Optuna

OPTUNA

Open-source Python lib for automated ML (hyperparam tuning, etc.)

Key Features:

- Define search space declaratively with Python (suggest_float, suggest_int, etc.)
- State of the Art algorithms with pruning (Early-stop bad trials)
- ML framework agnostic: Integrates with scikit-learn, XGBoost, PyTorch…
- Parallelization
- Visualizations: Basic plotting functions using Plotly and Matplotlib

Easy Install: `pip install optuna`

Optuna has an excellent documentation!

👉 https://optuna.readthedocs.io/en/stable/reference/index.html

Optuna syntax

```python
import numpy as np
import optuna

SEED = 42
np.random.seed(SEED)

# Define the function to optimize
def function_to_optimize(x, y):
    # Quadratic function: f(x,y) = (x-2)^2 + (y+1)^2 + 3
    # Minimum is at (2, -1) with value 3
    return (x - 2) ** 2 + (y + 1) ** 2 + 3

# Define the objective function for Optuna
def objective(trial):
    # Define parameters to optimize
    x = trial.suggest_float('x', -10.0, 10.0)
    y = trial.suggest_float('y', -10.0, 10.0)
    return function_to_optimize(x, y)

# Create study object and optimize
study = optuna.create_study(direction='minimize',
                            sampler=optuna.samplers.TPESampler(seed=SEED))
study.optimize(objective, n_trials=100)

# Print results
print(f"Best parameters: {study.best_params}")
print(f"Best value: {study.best_value}")
print(f"Number of trials: {len(study.trials)}")
```

👉 🤖 `python example3.py`

} Seed

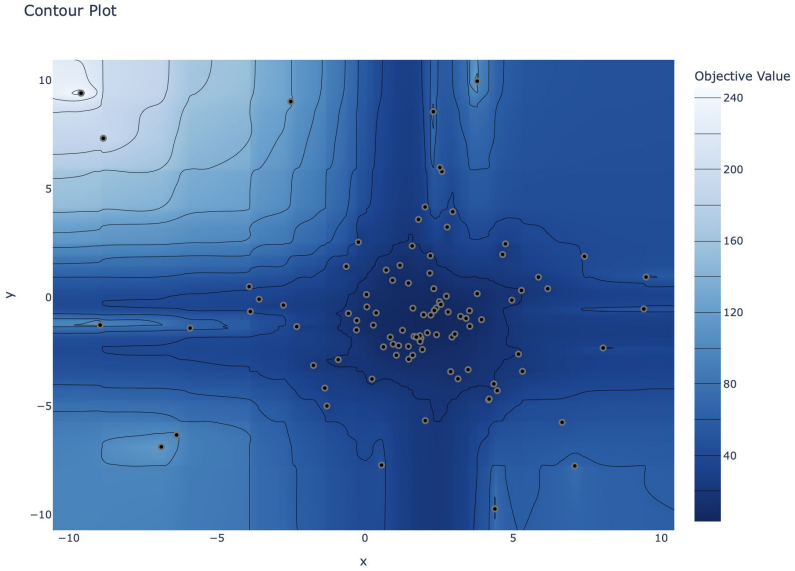
} Function to optimize (usually a metric)

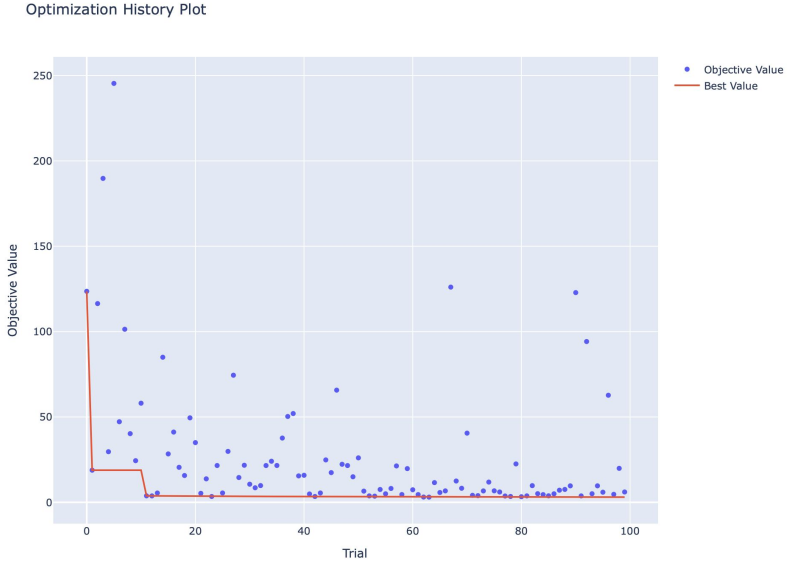
} Search space definition

} Objective function executes on each trial

} Optimization algorithm (sampler)

Example 3: Optuna

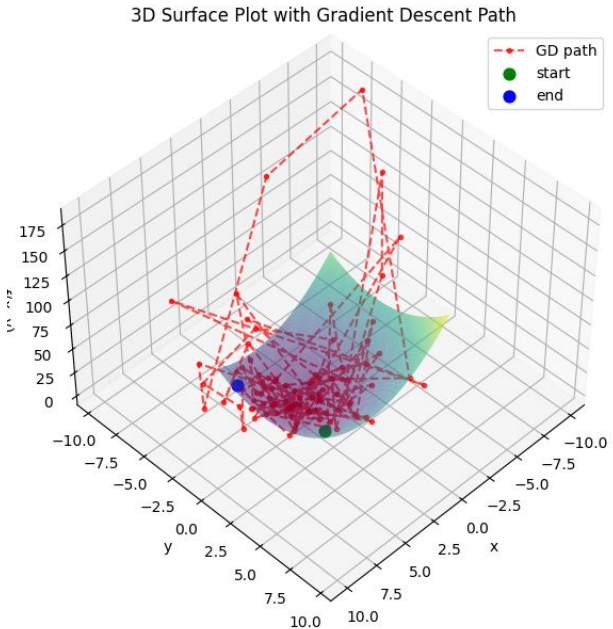


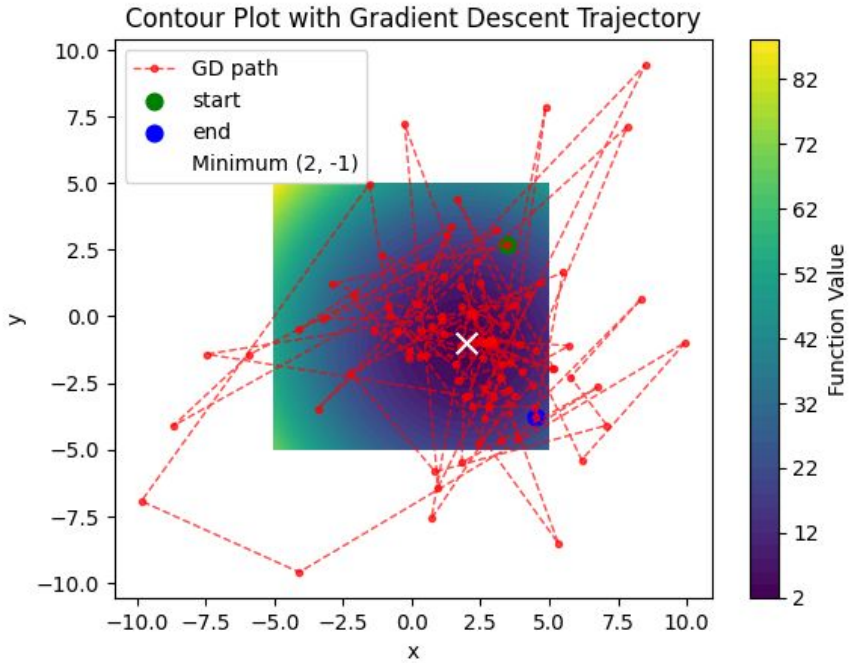


Best parameters: {'x': 1.893938637086194, 'y': -1.0665190063871524}
Best value: 3.0156737909138682
Number of trials: 100 (good value after ~12 trials)

Example 3: Optuna

Minimum value of f(x,y) is 3.0156737909138682 at (x,y) = (1.893938637086194,-1.0665190063871524), with 100 trials.





👉 python example3.py

Pythonic search spaces

```python
import optuna


def objective(trial):
    # Categorical parameter
    optimizer = trial.suggest_categorical("optimizer", ["MomentumSGD", "Adam"])

    # Integer parameter
    num_layers = trial.suggest_int("num_layers", 1, 3)

    # Integer parameter (log)
    num_channels = trial.suggest_int("num_channels", 32, 512, log=True)

    # Integer parameter (discretized)
    num_units = trial.suggest_int("num_units", 10, 100, step=5)

    # Floating point parameter
    dropout_rate = trial.suggest_float("dropout_rate", 0.0, 1.0)

    # Floating point parameter (log)
    learning_rate = trial.suggest_float("learning_rate", 1e-5, 1e-2, log=True)

    # Floating point parameter (discretized)
    drop_path_rate = trial.suggest_float("drop_path_rate", 0.0, 1.0, step=0.1)
```

Categories

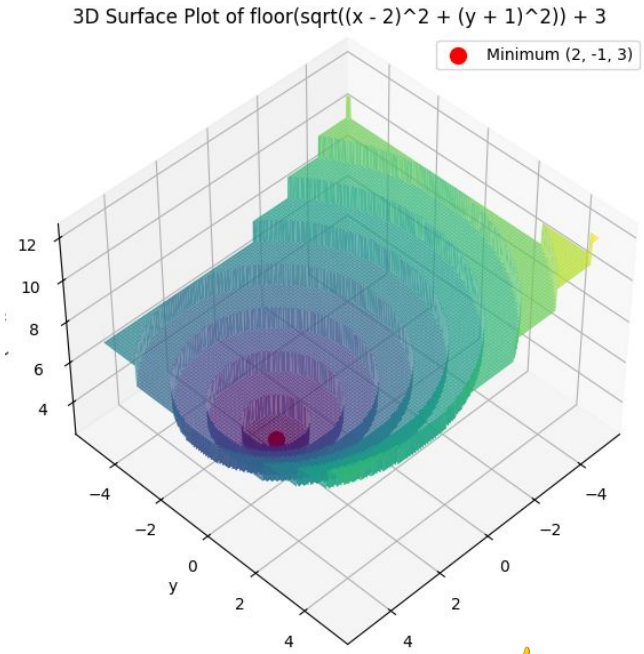
Integers

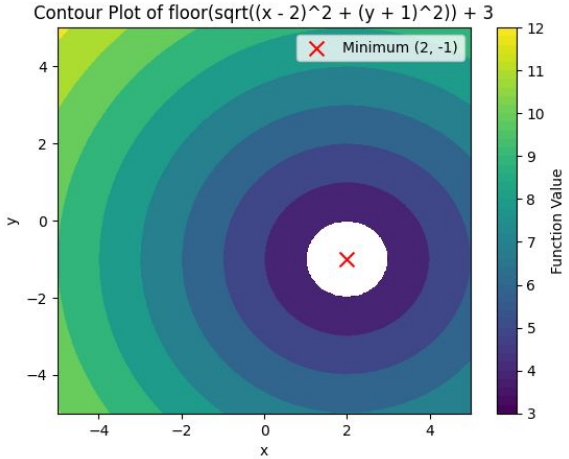
Floats

👉 see the docs Pythonic Search Spaces

Why is optimization so hard?

Noise and discontinuities in real data make harder to find a good solution.

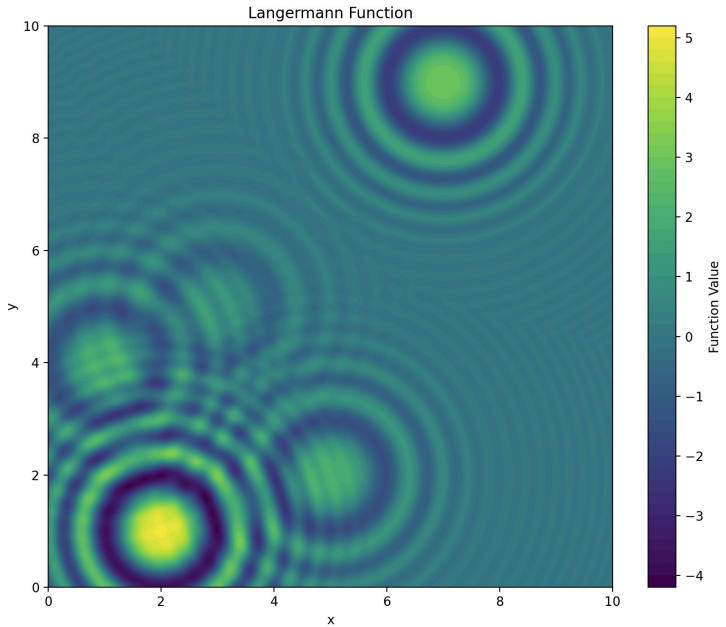


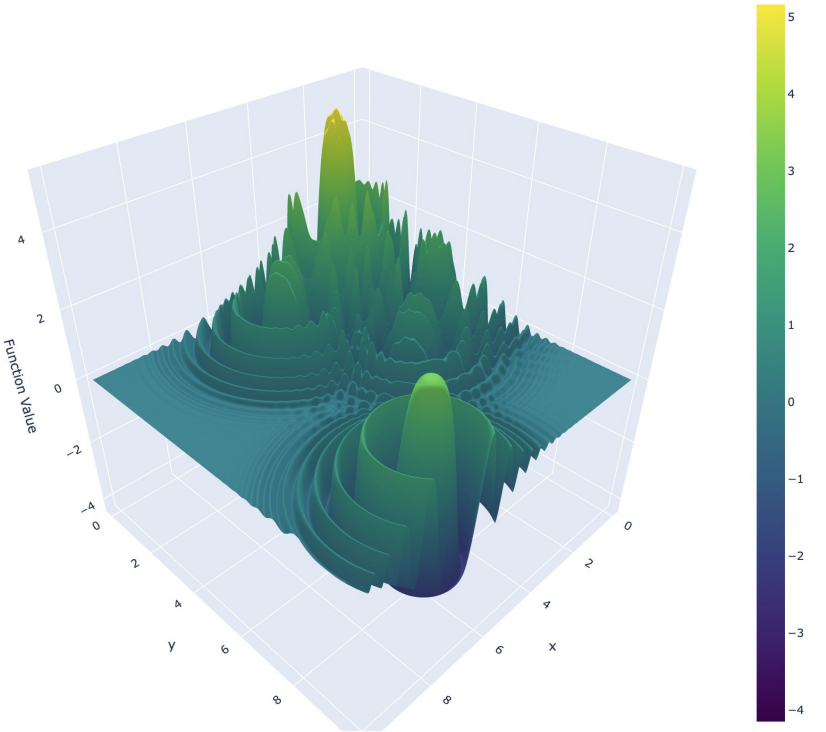


👉 python example4.py

Why is optimization so hard?

Algorithms tend to get stuck in local minima





👉 python example5.py

Search algorithms

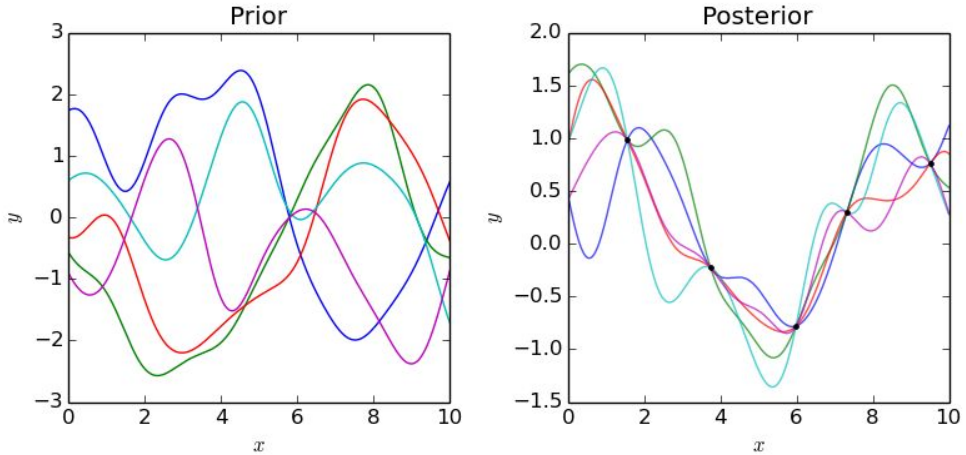
Optuna provides the following sampling algorithms:

- Grid Search implemented in GridSampler
- Random Search implemented in RandomSampler
- **Tree-structured Parzen Estimator** algorithm implemented in TPESampler (default)
- CMA-ES based algorithm implemented in CmaEsSampler
- **Gaussian process-based algorithm** implemented in GPSampler
- Algorithm to enable partial fixed parameters implemented in PartialFixedSampler
- Nondominated Sorting Genetic Algorithm II implemented in NSGAIISampler
- A Quasi Monte Carlo sampling algorithm implemented in QMCSampler

👉 Efficient_optimization_algorithms

Gaussian Processes

A Gaussian process is a probability distribution over possible functions that fit a set of points. We use it as surrogate P(y|x) of the objective function y=f(x).



$$P(\text{score}|\text{hyperparameters})$$

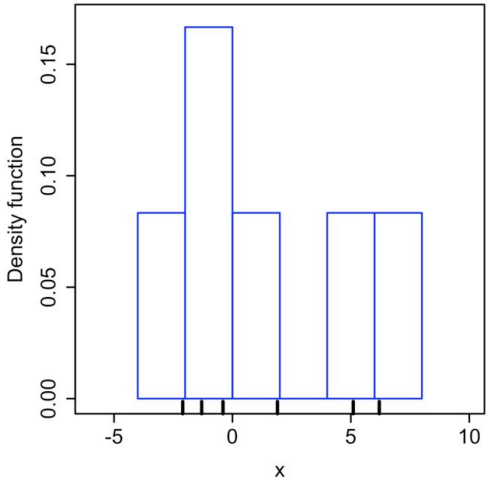
$$P(\text{y}|\text{x})$$

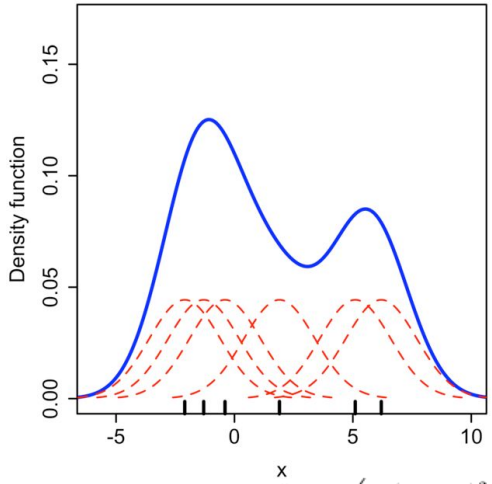
👉 https://en.wikipedia.org/wiki/Gaussian_process

Tree Parzen Estimators

Tree ~ Decision ~ Good and Bad trials

Parzen = Kernel density estimation





$$\hat{f}_h(x) = \frac{1}{nh\sigma}\frac{1}{\sqrt{2\pi}}\sum_{i=1}^{n}\exp\left(\frac{-(x-x_i)^2}{2h^2\sigma^2}\right),$$

👉 https://en.wikipedia.org/wiki/Kernel_density_estimation

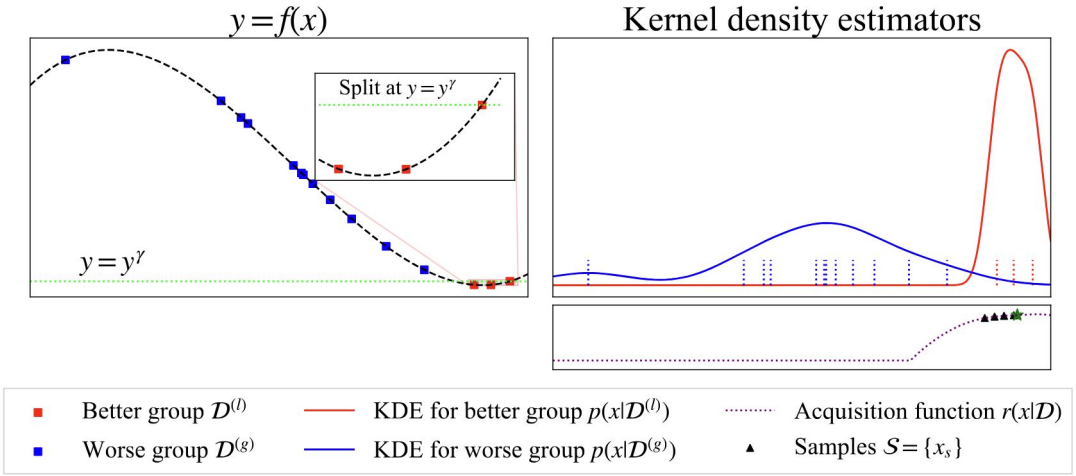
Tree Parzen Estimators

Tree = Decision tree ~ Good and Bad trials

Parzen = Kernel density estimation

It models likelihood with two distributions P(bad trial) and P(good trial)

y < y* represents a lower value of the objective function than the threshold



$$P(\mathrm{x}|\mathrm{y}) = \begin{cases} l(x) & \text{if } y < y^* \quad \mathrm{P(bad\ trial)} \\ g(x) & \text{if } y \geq y^* \quad \mathrm{P(good\ trial)} \end{cases}$$

👉 https://arxiv.org/abs/2304.11127

Search algorithms

	`TPESampler`	`GPSampler`	`CmaEsSampler`	`NSGAIISampler`	`NSGAIIISampler`	`GridSampler`
Float parameters	✓	✓	✓	▲	▲	✓
Integer parameters	✓	✓	✓	▲	▲	✓
Categorical parameters	✓	✓	▲	✓	✓	✓
Pruning	✓	▲	▲	× (▲ for single-objective)	× (▲ for single-objective)	✓
Multivariate optimization	✓	✓	✓	▲	▲	▲
Conditional search space	✓	▲	▲	▲	▲	▲
Multi-objective optimization	✓	✓	×	✓ (▲ for single-objective)	✓ (▲ for single-objective)	✓
Batch optimization	✓	▲	✓	✓	✓	✓
Distributed optimization	✓	▲	✓	✓	✓	✓
Constrained optimization	✓	✓	×	✓	✓	×
Time complexity (per trial) (*)	$O(dn \log n)$	$O(n^3)$	$O(d^3)$	$O(mp^2)$ (***)	$O(mp^2)$ (***)	$O(dn)$
Recommended budgets (#trials) (**)	100–1000	–500	1000–10000	100–10000	100–10000	number of combinations

Note

✓: Supports this feature. ▲: Works, but inefficiently. ×: Causes an error, or has no interface.

https://optuna.readthedocs.io/en/stable/reference/samplers/index.html

Hackathon

To be revealed at the seminar

Hyperparameter tuning in ML

Hyperparameter tuning in ML

```python
### Optuna hyperparameter optimization
import optuna
import numpy as np
from sklearn.model_selection import cross_val_score, StratifiedKFold

SEED = 42
np.random.seed(SEED)

def objective(trial):
    scale = trial.suggest_categorical("scale", [True, False])
    n_layers = trial.suggest_int("n_layers", 1, 10)
    hidden = tuple(trial.suggest_int(f"n_units_l{i}", 4, 128, log=True) for i in range(n_layers))
    activation = trial.suggest_categorical("activation", ["relu", "tanh", "logistic"])
    solver = trial.suggest_categorical("solver", ["adam", "sgd"])
    alpha = trial.suggest_float("alpha", 1e-6, 1e-2, log=True)
    learning_rate_init = trial.suggest_float("learning_rate_init", 1e-5, 1e-1, log=True)
    batch_size = trial.suggest_categorical("batch_size", [16, 32, 64])

    clf_trial = (make_pipeline(
        StandardScaler() if scale else None,
        MLPClassifier(
            hidden_layer_sizes=hidden,
            activation=activation,
            solver=solver,
            alpha=alpha,
            learning_rate_init=learning_rate_init,
            batch_size=batch_size,
            max_iter=2000,
            early_stopping=True,
            random_state=SEED
        )
    ))

    score = np.mean(cross_val_score(clf_trial, X_train, y_train, cv=3, scoring='accuracy'))
    return score

sampler = optuna.samplers.TPESampler(seed=SEED)
study = optuna.create_study(direction="maximize", sampler=sampler)
study.optimize(objective, n_trials=500)
```

Decide preprocessing options

Decide architecture and training options

👉 `python example7.py` 🤖

👉Visit https://optuna.org/#code_examples to check your ML library

Model selection

```python
def objective(trial):

    classifier_name = trial.suggest_categorical(
        "classifier",
        [
            "SVC",
            "NaiveBayes",
        ],
    )

    scale = trial.suggest_categorical("scale", [True, False])
    if classifier_name == "SVC":
        svc_c = trial.suggest_float("svc_c", 1e-10, 1e10, log=True)
        classifier_obj = sklearn.svm.SVC(C=svc_c, gamma="auto")

    elif classifier_name == "NaiveBayes":
        classifier_obj = sklearn.naive_bayes.GaussianNB()

    clf_pipeline = make_pipeline(StandardScaler() if scale else None, classifier_obj)

    score = cross_val_score(clf_pipeline, X_train, y_train, n_jobs=-1, cv=3)
    accuracy = score.mean()
    return accuracy
```

Suggest model type as category

👉 python example8.py 🤖

Multiobjective optimization

Multiobjective optimization

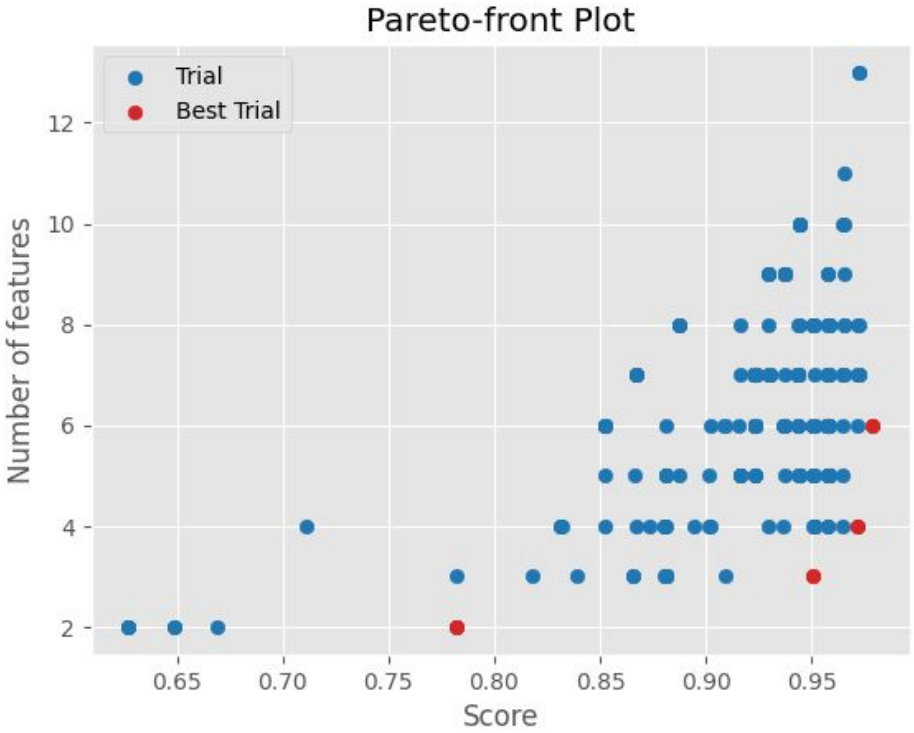
```python
def objective(trial: Trial):
    # Suggest feature subset
    feature_mask = []
    for i in range(X.shape[1]):
        feature_mask.append(trial.suggest_categorical(f"feature_{i}", [True, False]))
    X_selected = X_train[:, feature_mask]
    number_of_features = sum(feature_mask)
    if X_selected.shape[1] == 0:
        return 0.0, number_of_features

    # Train and evaluate model
    clf = GaussianNB()
    score = cross_val_score(clf, X_selected, y_train, cv=5, scoring="accuracy")
    return score.mean(), number_of_features


# Create study (single-objective maximizing the combined metric)
sampler = optuna.samplers.TPESampler(seed=SEED)
study = optuna.create_study(directions=["maximize", "minimize"], sampler=sampler)

# Enqueue the first trial with ALL features selected
all_features_params = {f"feature_{i}": True for i in range(X.shape[1])}
study.enqueue_trial(all_features_params)

# Now optimize (this will run the enqueued trial first, then 99 more)
study.optimize(objective, n_trials=300)
```

Mask a feature subset

Important to try all features!!

👉 python example9.py 🤖

Pareto front



👉 `python example9.py` 🤖

Practical session

Practical session

Use BO to get a PERFECT! accuracy score in IRIS dataset. Try different models!

👉 https://scikit-learn.org/stable/supervised_learning.html

👉 `python practical_session.py`
🤖

To conclude…

More advanced features…

Parallelization (Problem with reproducibility)

Early stopping “Pruners”

Human-in-the-loop to validate trials using dashboard

Save models during optimization

Add constraints to the optimization

Save and resume studies

Ignore duplicate samples

Handle exceptions/errors during trials

References

Webpapers & papers

A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning. Will Koehrsen. Medium. 2018

1. Algorithms for Hyper-Parameter Optimization [Link]
2. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures [Link]
3. Bayesian Optimization Primer [Link]
4. Taking the Human Out of the Loop: A Review of Bayesian Optimization [Link]

Optimization benchmarks

……

OPTUNA

https://optuna.readthedocs.io/en/stable/index.html

https://optuna.readthedocs.io/en/stable/tutorial/index.html

Gaussian Processes

https://distill.pub/2019/visual-exploration-gaussian-processes/

https://thegradient.pub/gaussian-process-not-quite-for-dummies/

TPEs

https://en.wikipedia.org/wiki/Kernel_density_estimation

https://arxiv.org/abs/2304.11127

Multiobjective optimization

https://medium.com/optuna/optuna-v4-5-81e78d8e077a