

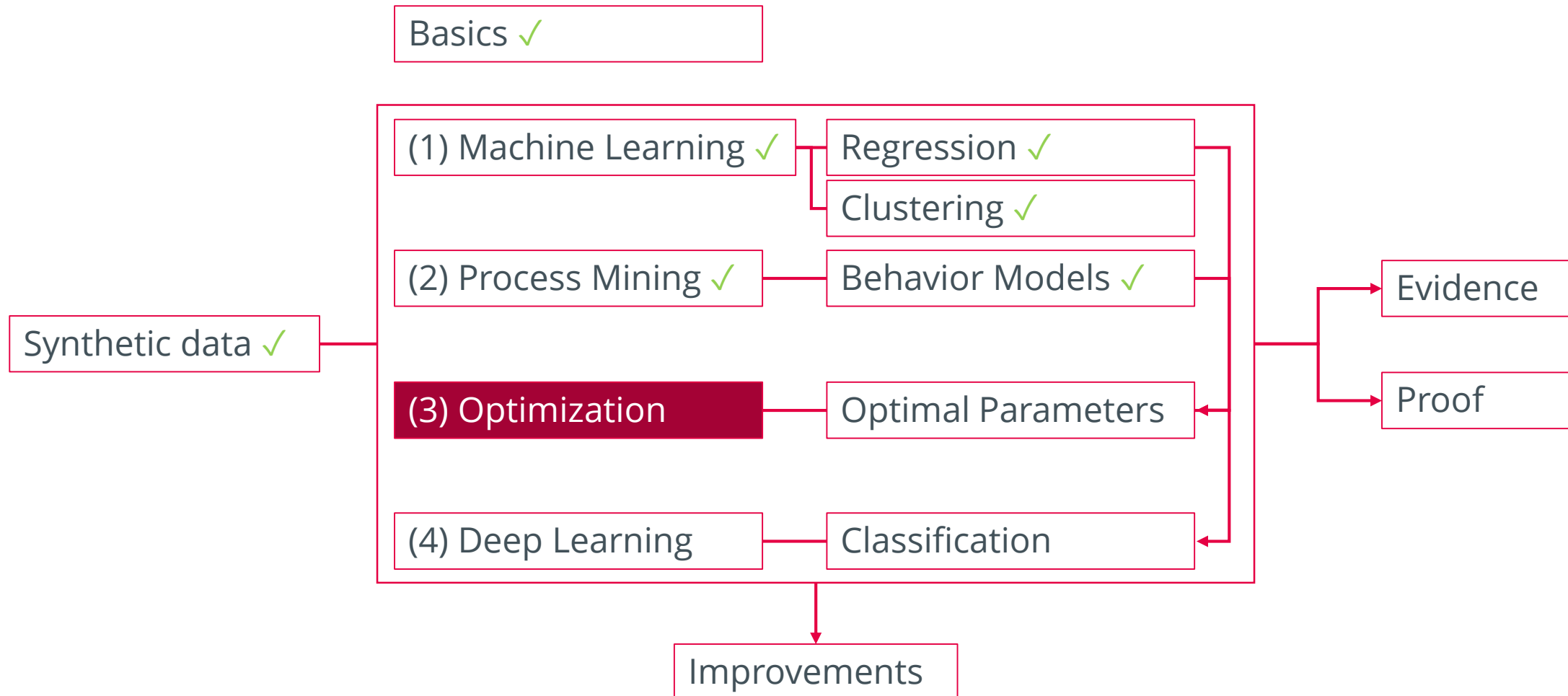


TECHNISCHE HOCHSCHULE
OSTWESTFALEN-LIPPE
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS

Welcome

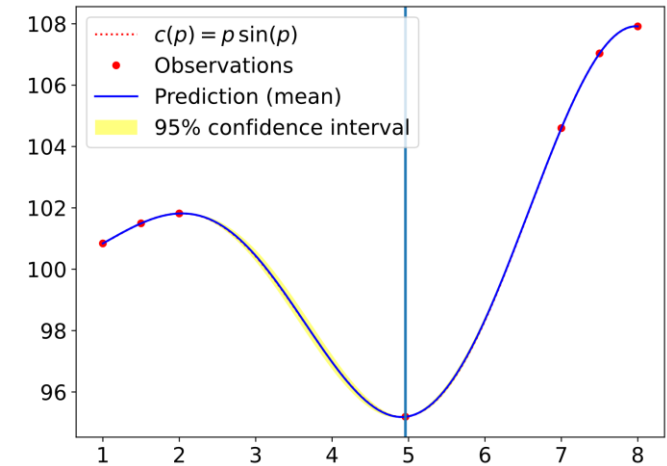
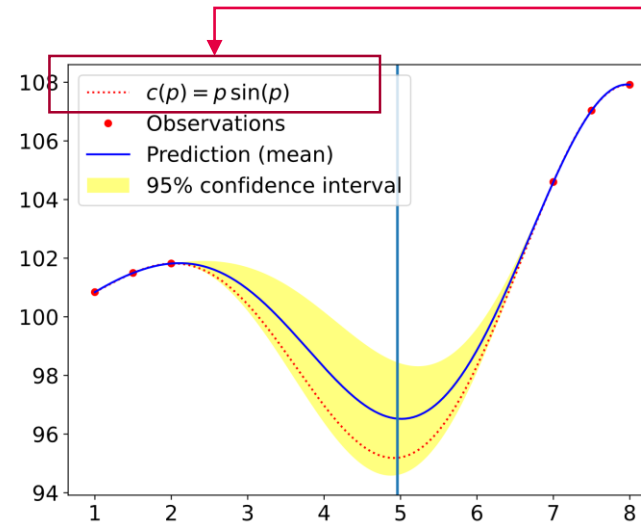
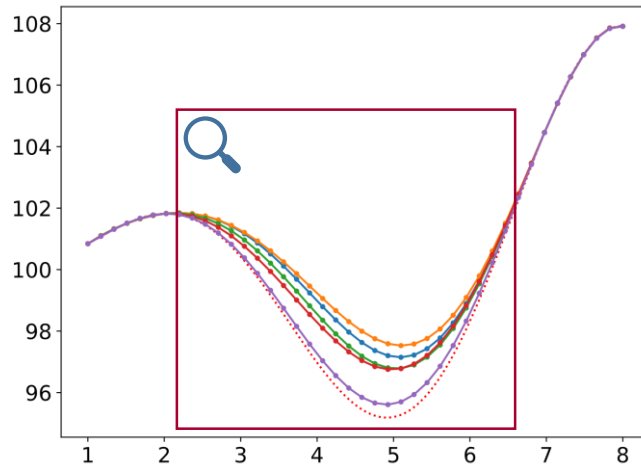
to Advanced Topics in Algorithms

Summary: Advanced Topics in Algorithms

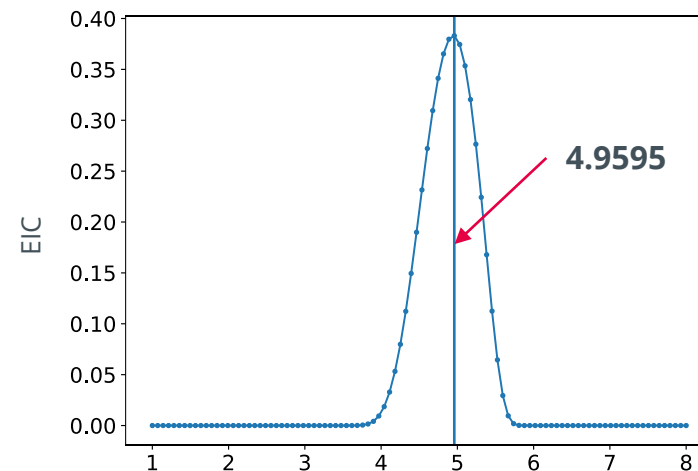


Revision: Bayesian Optimization

Sample possible values



- Use Gaussian Process Regression
- Calculate values using the Expected Improvement Criterion (EIC)
- Choose parameter configuration with the highest EIC



Add observation:
 $p = 4.9595$
 $c(p) = 95.1911$

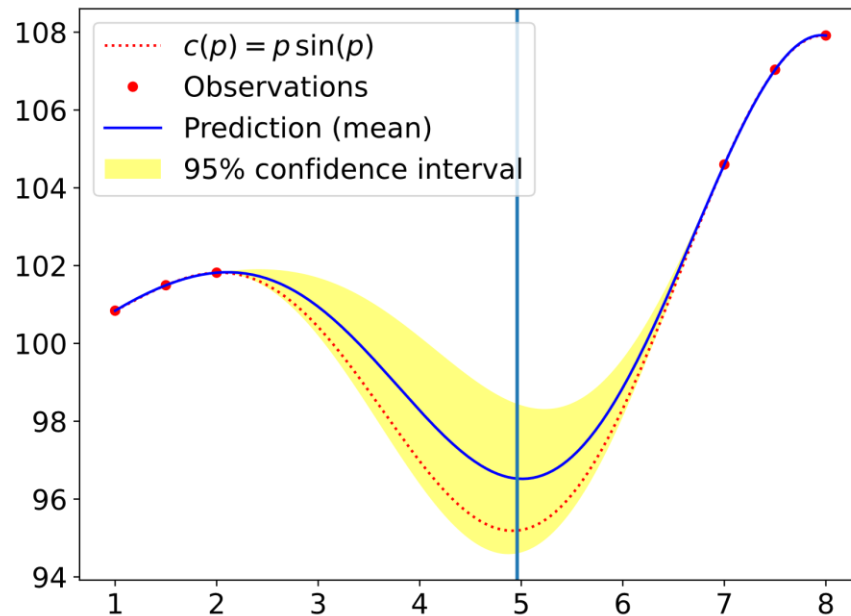
Function value

Function

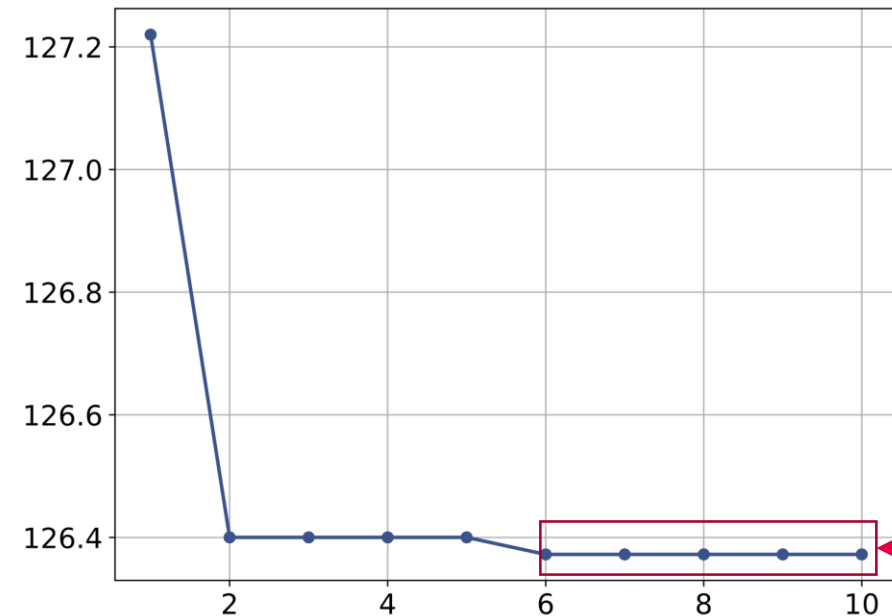
Exercise 1: Bayesian Optimization (scikit-optimize)

Find Optimal Parameters for Regression with Polynomial Features (scikit-optimize)

Implement “Exercises 06: Bayesian Optimization: Exercise 1: Find Optimal Parameters for Regression with Polynomial Features” with scikit-optimize (<https://scikit-optimize.github.io/stable/index.html>). Use “skopt.plots: Plotting functions” (<https://scikit-optimize.github.io/stable/modules/classes.html>) to visualize results of the algorithm. Compare the “scikit-optimize solution” with your implementation.

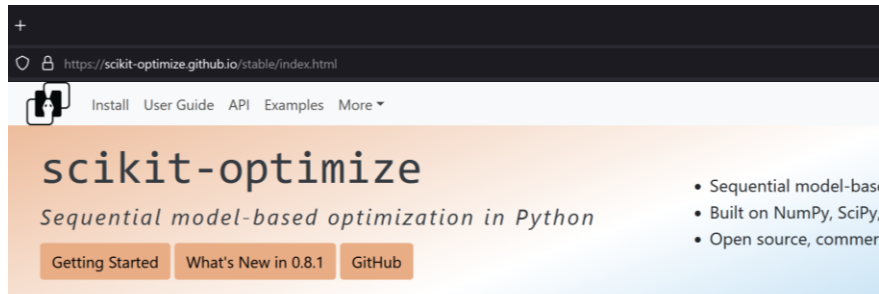


Convergence



Exercise 1: Bayesian Optimization (scikit-optimize)

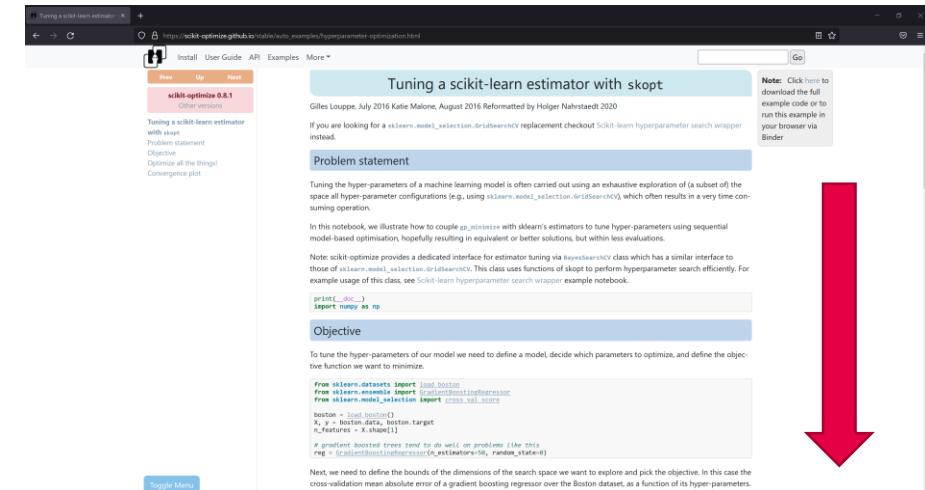
(1) <https://scikit-optimize.github.io/stable/index.html>



scikit-optimize
Sequential model-based optimization in Python

- Sequential model-based
- Built on NumPy, SciPy,
- Open source, commercial

Getting Started What's New in 0.8.1 GitHub

Tuning a scikit-learn estimator with skopt

Gilles Louppe, July 2016 Katie Malone, August 2016 Reformatted by Holger Nehstade 2020

If you are looking for a `sklearn.model_selection.GridSearchCV` replacement checkout `Scikit-learn hyperparameter search wrapper` instead.

Problem statement

Tuning the hyper-parameters of a machine learning model is often carried out using an exhaustive exploration of (a subset of) the space all hyper-parameter configurations (e.g., using `sklearn.model_selection.GridSearchCV`), which often results in a very time consuming operation.

In this notebook, we illustrate how to couple `gp_minimize` with `sklearn`'s estimators to tune hyper-parameters using sequential model-based optimisation, hopefully resulting in equivalent or better solutions, but within less evaluations.

Note: scikit-optimize provides a dedicated interface for estimator tuning via `BayesSearchCV` class which has a similar interface to those of `sklearn.model_selection.GridSearchCV`. This class uses functions of `skopt` to perform hyperparameter search efficiently. For example usage of this class, see `Scikit-learn hyperparameter search wrapper example notebook`.

```
print(__doc__)
import numpy as np
```

Objective

To tune the hyper-parameters of our model we need to define a model, decide which parameters to optimize, and define the objective function we want to minimize.

```
from sklearn.datasets import load_boston
from sklearn ensemble import GradientBoostingRegressor
from sklearn model_selection import cross_val_score

boston = load_boston()
X, y = boston.data, boston.target
n_features = X.shape[1]

# gradient boosted trees tend to do well on problems like this
reg = GradientBoostingRegressor(random_state=0)
```

Next, we need to define the bounds of the dimensions of the search space we want to explore and pick the objective. In this case the cross-validation mean absolute error of a gradient boosting regressor over the Boston dataset, as a function of its hyper-parameters.

scroll



(2)

Download Python source code: `hyperparameter-optimization.py`

Download Jupyter notebook: `hyperparameter-optimization.ipynb`

Exercise 1: Bayesian Optimization (scikit-optimize)

```
hyperparameter-optimization.py X
hyperparameter-optimization.py > ...
1 """
2 =====
3 Tuning a scikit-learn estimator with 'skopt'
4 =====
5
6 Gilles Louppe, July 2016
7 Katie Malone, August 2016
8 Reformatted by Holger Nahrstaedt 2020
9
10 .. currentmodule:: skopt
11
12 If you are looking for a :obj:`sklearn.model_selection.GridSearchCV` replacement checkout
13 :ref:`sphx_glr_auto_examples_sklearn-gridsearchcv-replacement.py` instead.
14
15 Problem statement
16 =====
17
18 Tuning the hyper-parameters of a machine learning model is often carried out
19 using an exhaustive exploration of (a subset of) the space all hyper-parameter
20 configurations (e.g., using :obj:`sklearn.model_selection.GridSearchCV`), which
21 often results in a very time consuming operation.
22
23 In this notebook, we illustrate how to couple :class:`gp_minimize` with sklearn's
24 estimators to tune hyper-parameters using sequential model-based optimisation,
25 hopefully resulting in equivalent or better solutions, but within less
26 evaluations.
27
28 Note: scikit-optimize provides a dedicated interface for estimator tuning via
29 :class:`BayesSearchCV` class which has a similar interface to those of
30 :obj:`sklearn.model_selection.GridSearchCV`. This class uses functions of skopt to perform hyperparameter
31 search efficiently. For example usage of this class, see
32 :ref:`sphx_glr_auto_examples_sklearn-gridsearchcv-replacement.py`
33 example notebook.
34 """
35 print(__doc__)
36 import numpy as np
37
38 =====
39 # Objective
40 # =====
41 # To tune the hyper-parameters of our model we need to define a model,
42 # decide which parameters to optimize, and define the objective function
43 # we want to minimize.
44
45 from sklearn.datasets import load_boston
46 from sklearn.ensemble import GradientBoostingRegressor
47 from sklearn.model_selection import cross_val_score
48
49 boston = load_boston()
```

Remove
unnecessary
code



```
hyperparameter-optimization.py > ...
Run Cell | Run Below | Debug Cell
1 #%%
2 import numpy as np
3
4 from skopt import gp_minimize
5 from skopt.space import Real, Integer
6 from skopt.utils import use_named_args
7 from skopt.plots import plot_convergence
8
9
10 space = [Integer(1, 20, name='degree')]
11
12 @use_named_args(space)
13 def objective(**params):
14     print(params)
15     return 1
16
17 res_gp = gp_minimize(objective, space, n_calls=50, random_state=0)
18
19 "Best score=%.4f" % res_gp.fun
20
21 print("best score: ", res_gp.fun)
22 print("best degree: ", res_gp.x[0])
23
24 plot_convergence(res_gp)
25
```

Exercise 2

Benchmark the Knapsack Problem

Open “Knapsack.py” and implement a generator for “v” and “w”. Increase items and measure the time to solve the problem. Repeat each benchmark (n-times) and calculate mean and variance time. Create a table and a plot to visualize your results.

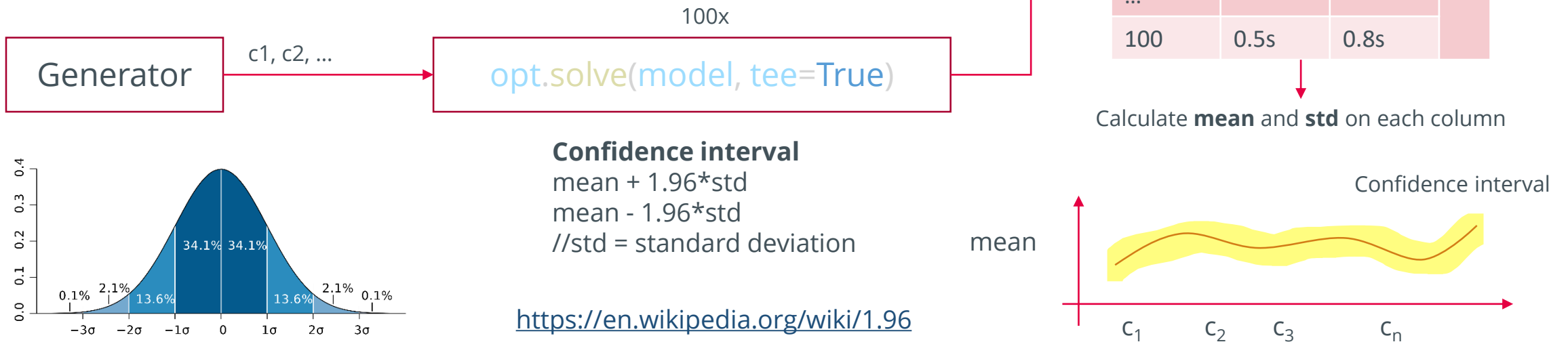
<https://realpython.com/python-timer/>

<https://docs.python.org/3/tutorial/datastructures.html#dictionaries>

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.mean.html>

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.std.html>

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



Revision: Knapsack Problem (1/2)

Which items should be chosen to maximize the amount of money while keeping the overall weight under or equal to 6 kg?



Revision: Knapsack Problem (2/2)

Knapsack Problem (KP)


$$\text{maximize: } \sum_{i=1}^k v_i b_i$$

$$\text{s.t.: } \sum_{i=1}^k w_i b_i \leq \text{obj},$$

$$b_i \in \{0,1\}, i = 1, \dots, k$$

Pyomo is a Python-based, open-source optimization modeling language with a diverse set of optimization capabilities.

<http://www.pyomo.org/>



```
10
11 model = ConcreteModel()
12 model.elements = Set(initialize = v.keys())
13 model.x = Var(model.elements, within = Binary)
14 model.value = Objective(expr=sum(v[i]*model.x[i] for i in model.elements), sense=maximize)
15 model.weight = Constraint(expr=sum(w[i]*model.x[i] for i in model.elements) <= limit)
16
```

- <https://www.gams.com/products/gams/gams-language/>
- <https://zimpl.zib.de/>
- ...

Revision: Multiple-Choice Knapsack Problem

Multiple-Choice Knapsack Problem (MCKP)

$$\text{maximize: } \sum_{i=1}^k \sum_{j \in N_i} v_{ij} b_{ij}$$

$$\text{s.t.: } \sum_{i=1}^k \sum_{j \in N_i} w_{ij} b_{ij} \leq \text{obj},$$

$$\sum_{j \in N_i} b_{ij} = 1, i = 1, \dots, k,$$

$$b_{ij} \in \{0,1\}, i = 1, \dots, k, j \in N_i$$



Exercise 4: Implement a Decision Support System

Knapsack?



Goal

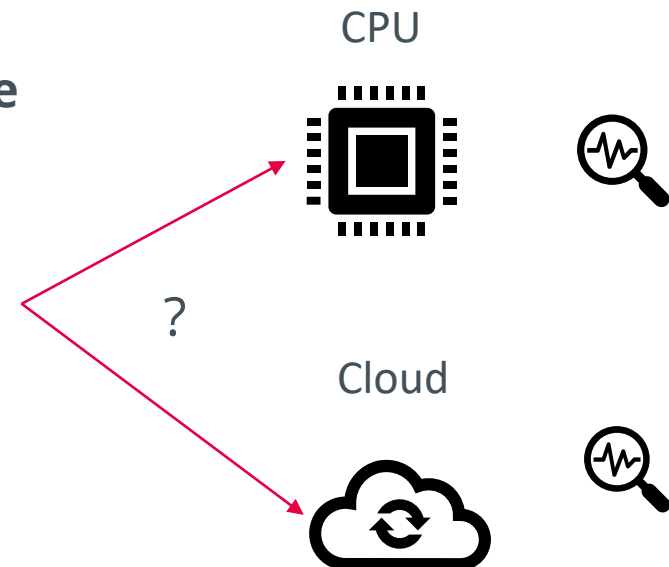
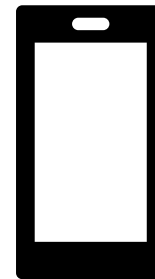
Original image (96,615 colors)



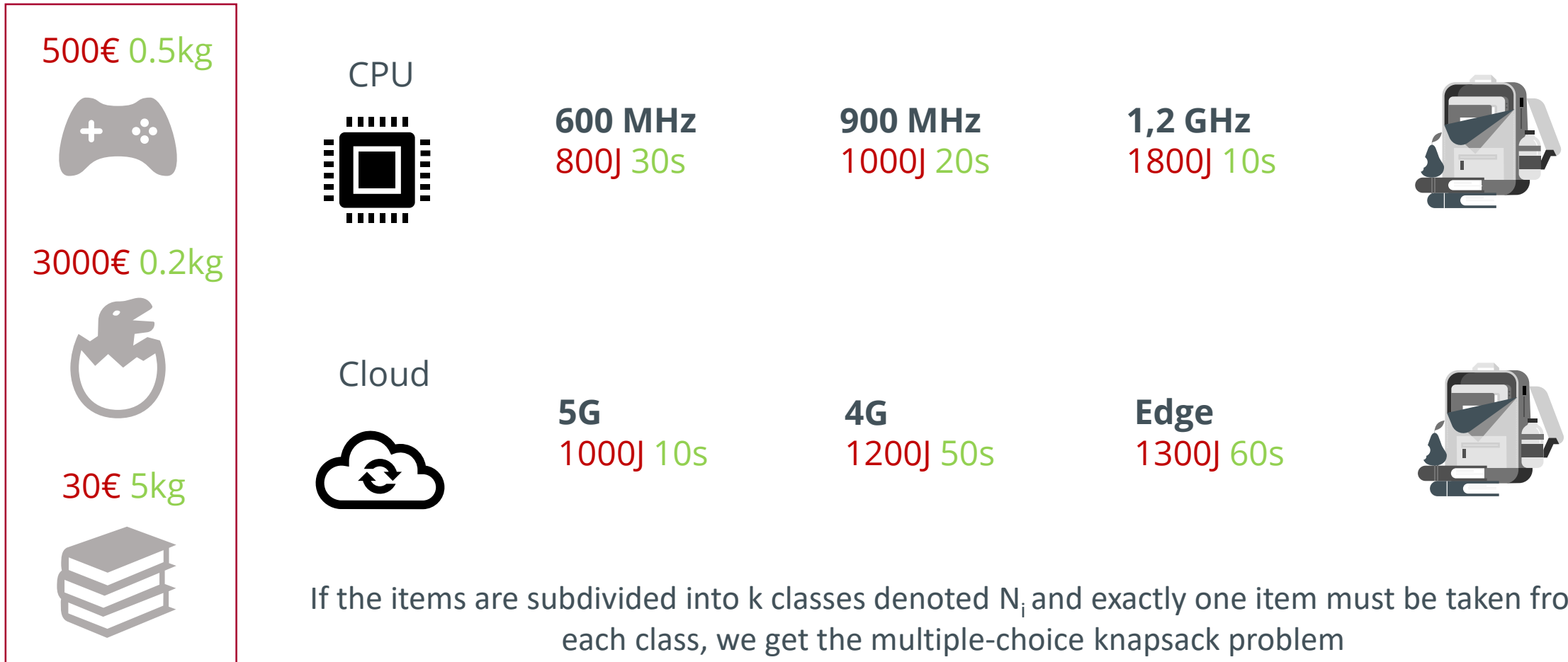
Quantized image (16 colors, K-Means)



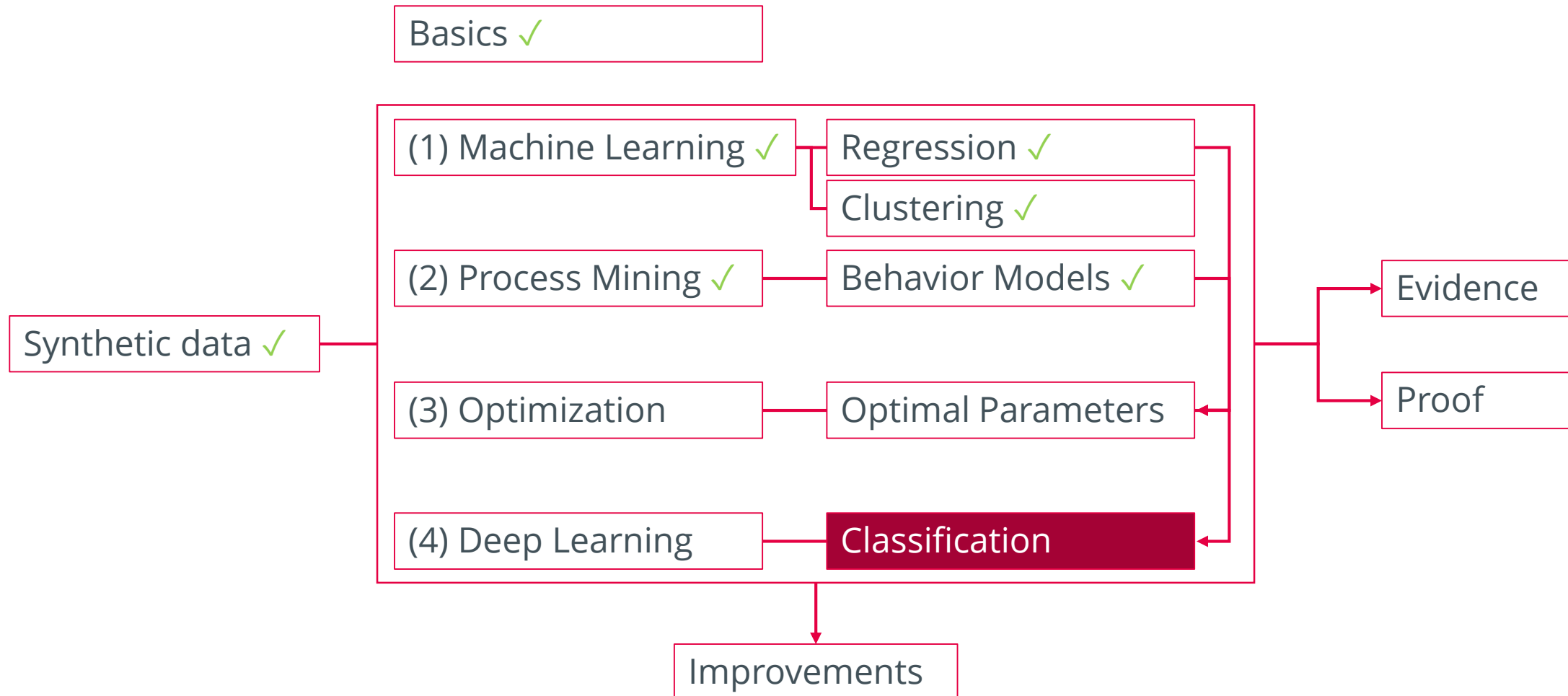
Smartphone



Exercise 4: General Idea

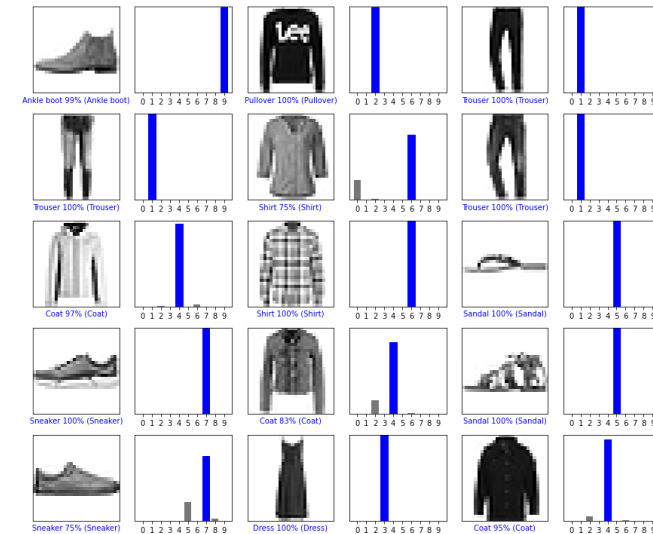
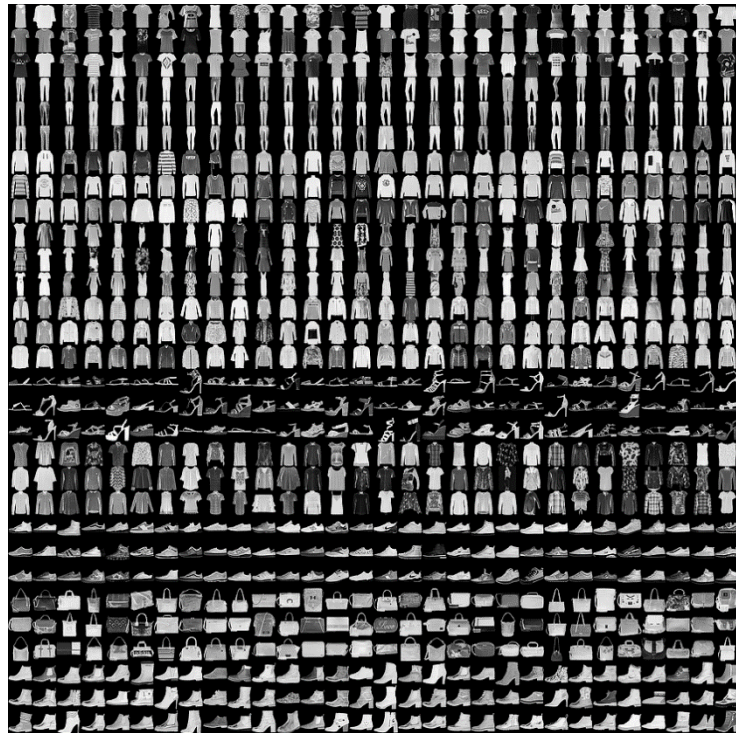


Summary: Advanced Topics in Algorithms



[Deep] Learning: Classification

Fashion-MNIST is a dataset of Zalando's article images



<https://arxiv.org/pdf/1708.07747.pdf>

XIAO, Han; RASUL, Kashif; VOLLGRAF, Roland. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.



TECHNISCHE HOCHSCHULE
OSTWESTFALEN-LIPPE
UNIVERSITY OF
APPLIED SCIENCES
AND ARTS

Thank you!