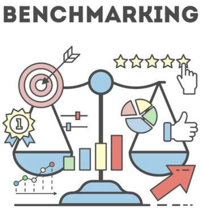
Universiteit Leiden

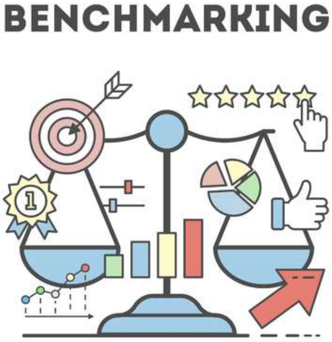
OpenML Use Cases

Jan N. van Rijn and Joaquin Vanschoren

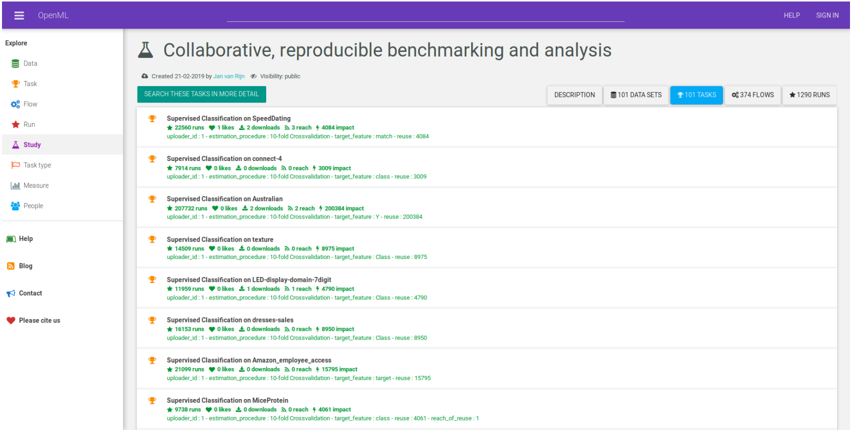
March 23th, 2019

Table of Contents





OpenML Benchmark Suites


OpenML
HELP
SIGN IN
Explore
Data
Task
Flow
Run
Study
Task type
Measure
People
Help
Blog
Contact
Please cite us
Collaborative, reproducible benchmarking and analysis
Created 21-02-2019 by Jan van Rijn
Visibility: public
SEARCH THESE TASKS IN MORE DETAIL
DESCRIPTION
101 DATA SETS
101 TASKS
374 FLOWS
1290 RUNS
Supervised Classification on SpeedDating
22560 runs 1 likes 2 downloads 3 reach 4084 impact
uploader_id : 1 - estimation_procedure : 10-fold Crossvalidation - target_feature : match - reuse : 4084
Supervised Classification on connect-4
7914 runs 0 likes 0 downloads 0 reach 3009 impact
uploader_id : 1 - estimation_procedure : 10-fold Crossvalidation - target_feature : class - reuse : 3009
Supervised Classification on Australian
207732 runs 0 likes 2 downloads 2 reach 200384 impact
uploader_id : 1 - estimation_procedure : 10-fold Crossvalidation - target_feature : Y - reuse : 200384
Supervised Classification on texture
14509 runs 0 likes 0 downloads 0 reach 8975 impact
uploader_id : 1 - estimation_procedure : 10-fold Crossvalidation - target_feature : Class - reuse : 8975
Supervised Classification on LED-display-domain-7digit
11959 runs 0 likes 1 downloads 1 reach 4790 impact
uploader_id : 1 - estimation_procedure : 10-fold Crossvalidation - target_feature : Class - reuse : 4790
Supervised Classification on dresses-sales
16153 runs 0 likes 0 downloads 0 reach 8950 impact
uploader_id : 1 - estimation_procedure : 10-fold Crossvalidation - target_feature : Class - reuse : 8950
Supervised Classification on Amazon_employee_access
21099 runs 0 likes 0 downloads 0 reach 15795 impact
uploader_id : 1 - estimation_procedure : 10-fold Crossvalidation - target_feature : target - reuse : 15795
Supervised Classification on MiceProtein
9738 runs 0 likes 0 downloads 0 reach 4061 impact
uploader_id : 1 - estimation_procedure : 10-fold Crossvalidation - target_feature : class - reuse : 4061 - reach_of_reuse : 1

OpenML Benchmark Suites

Common limitations of experimental evaluations:

- Often done on a small selection of datasets
  - Not sure about generalization to more datasets
  - 'Cherry picking'
- Publication bias
  - Every paper only reports good results
  - More useful to know WHEN (on what type of data) a new algorithm performs well
- Hard to compare conclusions across papers
- Abused datasets (example: liver-disorders)

OpenML Benchmark Suites

OpenML-100: A curated benchmark suite [Bischl et al., 2017].
Inclusion criteria:

- Basic data properties (500–100000 observations, < 5000 features, $>= 2$ classes)
- The ratio of the minority class and the majority class > 0.05
- Scientific publication introducing the dataset and learning task
- Pure classification tasks only (no data streams, multi-class tasks)
- No artificial data, binarized versions of regression datasets or sub-samples of bigger datasets
- Disadvantage: Very hard to get a good consensus about the inclusion criteria
- Next: The OpenML-CC18 (approx. 71 datasets)

Universiteit Leiden

OpenML Benchmark Suites

Benchmark suite:

- A collection of tasks
  - Terminology: A Study is a collection of runs
  - Collection of flows, datasets, ...
- Immutable (once closed)
- Standardized train-test splits are provided to ensure that results can be objectively compared
- results can be shared in a reproducible way through the APIs
- `https://docs.openml.org/benchmark/`

[B. Bischl, G. Casalicchio, M. Feurer, F. Hutter, M. Lang, R. G. Mantovani, J. N. van Rijn, and J. Vanschoren. Openml benchmarking suites and the openml100.

arXiv preprint arXiv:1708.03731, 2017]

Universiteit Leiden

Requirements

- A set of tasks
  - Upload dataset and create tasks (task creation soon available)
  - Use existing tasks
- Latest version of Python API (Development version)
  - `openml.dataset.list_datasets(...)`
  - `openml.study.create_benchmark_suite(...)`
- OpenML account and user id (needs to be configured)
- Also available for Java (see docs) and R (comming soon)

Create Benchmark Suite

```
import openml

tasks = openml.tasks.list_tasks(
  number_instances='100..500',
  number_features='4..20',
  size=20)
# task is a Dict[int, OpenMLTask]

study = openml.study.create_benchmark_suite(
  alias=None,
  name="Benchmark Example",
  description="illustrates creating benchmark suites",
  task_ids=tasks.keys()
)
study_id = study.publish()
print('Uploaded study with id=%d' % study_id)
```

Find Benchmark Suite

```python
import openml

studies = openml.study.list_studies(
  main_entity_type='task',
  creator=1,
  status='all'
)
```

Lists all studies / benchmark suites that comply to a set of filters

- Legal filters: main_entity_type, uploader, status, ...
- `studies` is now a `Dict[int, OpenMLStudy]`
- Note: Recently created studies are 'in preparation'

Common operations

Attach additional tasks

```
tasks = openml.tasks.list_tasks(data_name='letter', size=1)
# tasks_new is Dict[int, OpenMLTask]
openml.study.attach_to_study(study_id, tasks.keys())
```

Common operations

Attach additional tasks

```
tasks = openml.tasks.list_tasks(data_name='letter', size=1)
# tasks_new is Dict[int, OpenMLTask]
openml.study.attach_to_study(study_id, tasks.keys())
```

Detach tasks

```
# given a variable study_id (int)
task_id = [2, 3, 4]
openml.study.detach_from_study(study_id, task_id)
```

Common operations

Attach additional tasks

```
tasks = openml.tasks.list_tasks(data_name='letter', size=1)
# tasks_new is Dict[int, OpenMLTask]
openml.study.attach_to_study(study_id, tasks.keys())
```

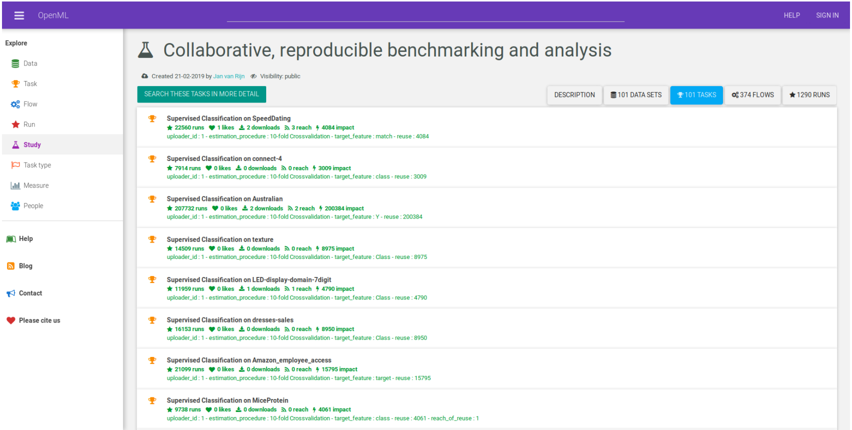
Detach tasks

```
# given a variable study_id (int)
task_id = [2, 3, 4]
openml.study.detach_from_study(study_id, task_id)
```

Activate benchmark suite (no mutations possible afterwards)

```
# given a variable study_id (int)
openml.study.status_update(study_id, 'active')
```

Benchmark Suite



Benchmark Suite

```
import openml  # development branch
import sklearn # version 0.20.0 and up

# given a variable study_id (int, str)
benchmark_suite = openml.study.get_study(study_id, 'tasks')
# build a sklearn classifier
clf = sklearn.pipeline.make_pipeline(
  sklearn.preprocessing.Imputer(),
  sklearn.tree.DecisionTreeClassifier()
)
# iterate over all tasks
for task_id in benchmark_suite.tasks:
  task = openml.tasks.get_task(task_id) # download the OpenML task
  X, y = task.get_X_and_y() # get the data (not used in this example)
  # run classifier on splits (requires API key)
  run = openml.runs.run_model_on_task(clf, task)
  # print accuracy score
  score = run.get_metric_score(sklearn.metrics.accuracy_score)
  print('Data set: %s; Accuracy: %0.2f' % (task.get_dataset().name,
                                            score.mean()))
  run.publish() # publish the experiment on OpenML (optional)
  print('URL for run: %s/run/%d' %(openml.config.server,run.run_id))
```


BUSTING THE
MYTHS

Myth Busting for Data Mining

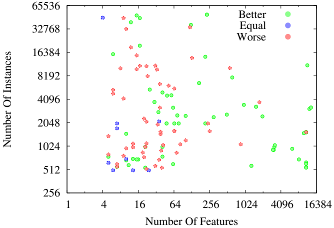
- Papers are generally build upon claims that are not well grounded, e.g.,
  - "We performed data transformation X because it is common practise."
  - "We set hyperparameter Y to value Z because the authors recommended these values."
- We can empirically analyze the validity of these claims on the meta-data from OpenML
- In this case: "The Importance of Feature Selection"

Effect of Feature Selection

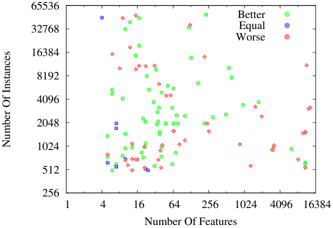
Experimental Setting by Post et al. [2016]:

- 400 binary classification datasets from OpenML
- 12 algorithms from Weka
- Correlation-based Feature Subset Selection
- We added runs that not existed on OpenML
- Recorded Area Under the ROC curve
- Limitation: Hyperparameter Optimization

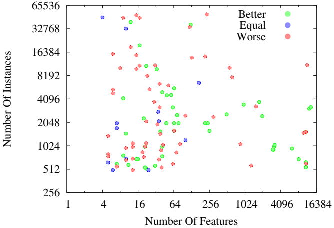
Effect of Feature Selection



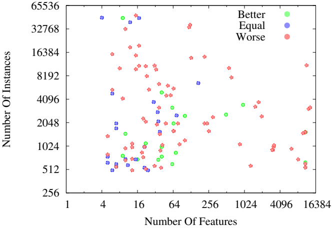
k-NN



Naive Bayes



J48



SMO

Effect of Feature Selection

Initial conclusions:

- Feature selection is often beneficial for the classifiers for which we expect it to be: k-NN and Naive Bayes
  - Surprisingly also for Decision Trees
- Feature selection is beneficial in 41% of the cases, only statistically significant in 10%
- Whether or not to use feature selection can be learned (see paper)
- Low amount of datasets on which feature selection significantly effects performance potentially indicates data bias
- Current Work: Linear vs. Non Linear classifiers

[M. J. Post, P. van der Putten, and J. N. van Rijn. Does Feature Selection Improve Classification? A Large Scale Experiment in OpenML.

In *Advances in Intelligent Data Analysis XV*, pages 158–170. Springer, 2016]

Linear vs. Non-linear

Reoccuring topic in literature, see [Strang et al., 2018] for several examples

	Linear	Non-linear
ease of tuning		
interpretability		
fit risk		
performance		

Linear vs. Non-linear

Reoccuring topic in literature, see [Strang et al., 2018] for several examples

	Linear	Non-linear
ease of tuning	+	-
interpretability		
fit risk		
performance		

Linear vs. Non-linear

Reoccuring topic in literature, see [Strang et al., 2018] for several examples

	Linear	Non-linear
ease of tuning	+	-
interpretability	+	-
fit risk		
performance		

Linear vs. Non-linear

Reoccuring topic in literature, see [Strang et al., 2018] for several examples

	Linear	Non-linear
ease of tuning	+	-
interpretability	+	-
fit risk	underfit	overfit
performance		

Linear vs. Non-linear

Reoccuring topic in literature, see [Strang et al., 2018] for several examples

	Linear	Non-linear
ease of tuning	+	-
interpretability	+	-
fit risk	underfit	overfit
performance	-	+

Linear vs. Non-linear

Reoccuring topic in literature, see [Strang et al., 2018] for several examples

	Linear	Non-linear
ease of tuning	+	-
interpretability	+	-
fit risk	underfit	overfit
performance	-	+
Tree	Decision Stump	Decision Tree

Linear vs. Non-linear

Reoccuring topic in literature, see [Strang et al., 2018] for several examples

	Linear	Non-linear
ease of tuning	+	-
interpretability	+	-
fit risk	underfit	overfit
performance	-	+
Tree	Decision Stump	Decision Tree
SVM	Linear Kernel	Gaussian Kernel

Linear vs. Non-linear

Reoccuring topic in literature, see [Strang et al., 2018] for several examples

	Linear	Non-linear
ease of tuning	+	-
interpretability	+	-
fit risk	underfit	overfit
performance	-	+
Tree	Decision Stump	Decision Tree
SVM	Linear Kernel	Gaussian Kernel
Neural Network	Perceptron	MLP

Universiteit Leiden

Requirements

- A Benchmark suite
  - This case: the OpenML-100
- Latest version of Python API (Development version)
- Run results of the linear and non-linear classifier
  - We will generate them
- Listing function `openml.evaluation.list_evaluations(...)`
- Plotting library (matplotlib)

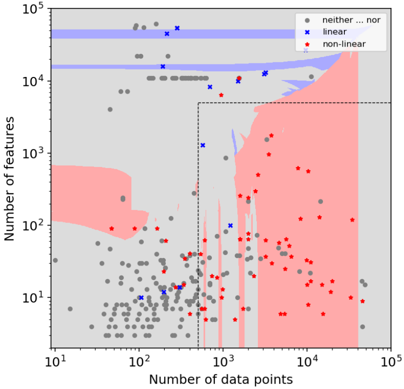
Generate Results

```python
import openml  # development branch
import sklearn # version 0.20.0 and up

# given a variable study_id (int, str)
benchmark_suite = openml.study.get_study('OpenML100', 'tasks')
# build a sklearn classifier
clfs = [
  sklearn.pipeline.make_pipeline( # non-linear
    sklearn.preprocessing.Imputer(),
    sklearn.svm.SVC()
  ),
  sklearn.pipeline.make_pipeline( # linear
    sklearn.preprocessing.Imputer(),
    sklearn.svm.LinearSVC()
  ),
]
run_ids = list()
for task_id in benchmark_suite.tasks:
  task = openml.tasks.get_task(task_id)
  for clf in clfs:
    run = openml.runs.run_model_on_task(clf, task)
    run.publish()
    print('URL for run: %s/run/%d' % (openml.config.server,
                                       run.run_id))
    run.push_tag('linear_vs_nonlinear')
```

Plot Results

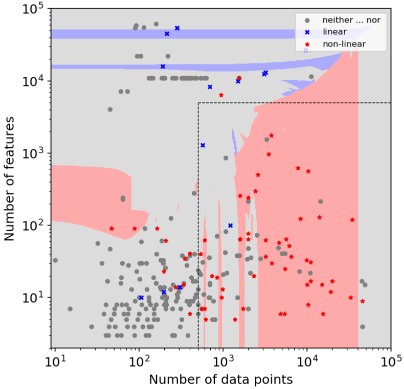
```
import openml  # development branch

# given a variable study_id (int, str)
suite = openml.study.get_study('OpenML100', 'tasks')
tasks = openml.tasks.list_tasks(task_ids=suite.tasks)
# given a variable uploader_id (int)
evals = openml.evaluations.list_evaluations(
  'predictive_accuracy',
  uploader=uploader_id,
  tag='linear_vs_nonlinear'
)

# organization
results = collections.defaultdict(dict)
for evaluation in evals.values():
    results[evaluation.setup_id][evaluation.task_id] = evaluation.value
# plot
for setup_id in results.keys():
  # find the tasks on which this evaluation is best
  res_x = []
  res_y = []
  for task_id in results[setup_id].keys():
    if results[setup_id][task_id] == [results[sid][task_id] for sid in results.keys()]:
      res_x.append(tasks[task_id]['NumberOfInstances'])
      res_y.append(tasks[task_id]['NumberOfFeatures'])
      plt.scatter(res_x, res_y)
# set labels, titles and (log-)scales
```

Linear vs. Non-Linear

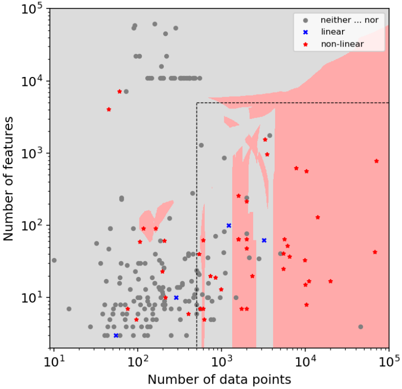


SVM

Linear vs. Non-Linear



SVM



Neural Networks

Linear vs. Non-Linear

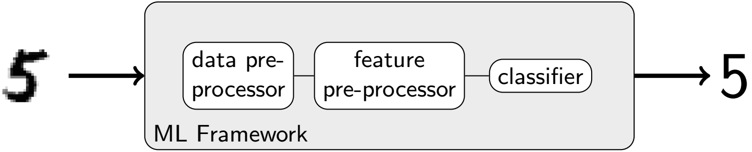
- Non-linear models are almost exclusively better than linear models (as expected)
- Statistical equivalent for low data regimes
- Limitation: Conservative statistical test
- benchmark suites such as OpenML-100 have limitations

[B. Strang, P. van der Putten, J. N. van Rijn, and F. Hutter. Don't rule out simple models prematurely: A large scale benchmark comparing linear and non-linear classifiers in openml.

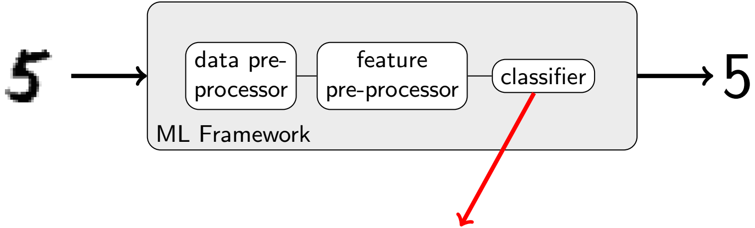
In *International Symposium on Intelligent Data Analysis*, pages 303–315. Springer, 2018]

Warm Starting Bayesian Optimization

The Machine Learning Pipeline

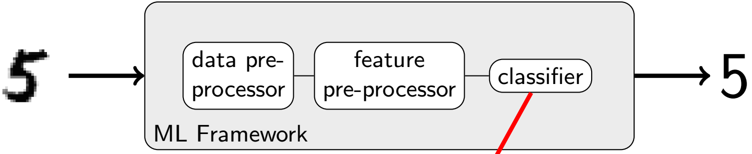


The Machine Learning Pipeline



classifier	#λ
Adaboost	4
Bernoulli Naive Bayes	2
Decision Tree	4
Extra Trees	5
Gradient Boosting	6
k-NN	3
LDA	4
...	

The Machine Learning Pipeline



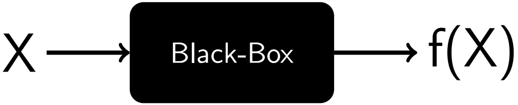
classifier	$\#\lambda$
Adaboost	4
Bernoulli Naive Bayes	2
Decision Tree	4
Extra Trees	5
Gradient Boosting	6
k-NN	3
LDA	4
...	

The Problem Definition

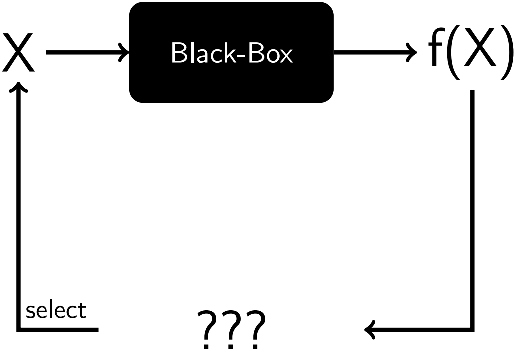
$$A^*, \lambda_* \in \operatorname*{arg\,min}_{A \in \mathbf{A}, \lambda \in \Lambda} L(A_{\lambda, D_{train}, D_{valid}})$$

Given a dataset D, find an algorithm A^* and its hyperparameters λ_* that minimizes a given loss function L

Black-Box Optimization



The Loop



Universiteit Leiden

The Human in the Loop

X → Black-Box → f(X)

+

. . .

select

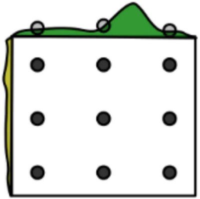
−

Not very efficient
Painful
Requires domain knowledge

Grid Search

X → Black-Box → f(X)

select



\+

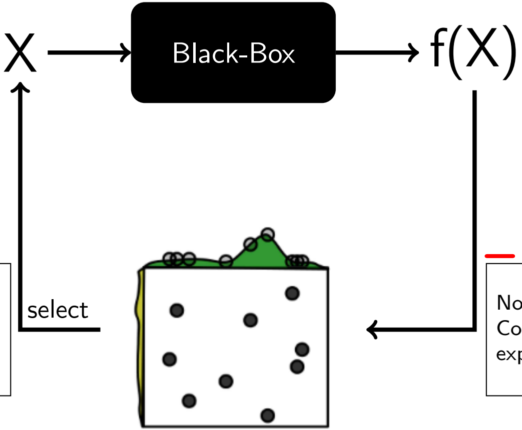
Very simple approach
Can be used to study the problem

−

Does not scale to high dimensions
Grid needs to be defined

Universiteit Leiden

Random Search



\+

Even more simple
Easily parallelizable
Eventually converges to optimum

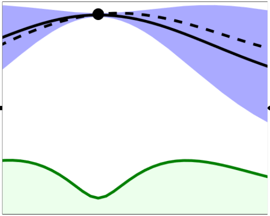
−

Not data efficient
Computationally expensive

Bayesian Optimization

X → Black-Box → f(X)

select



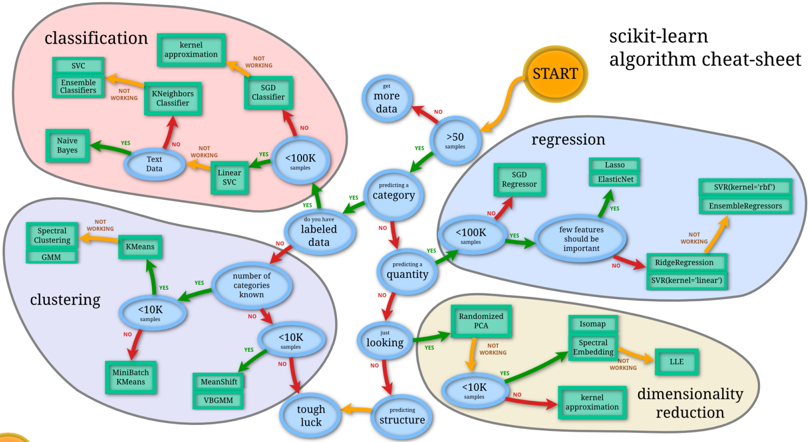
\+

Data efficient
State of the art

−

Not easy to parallelize

Flow chart

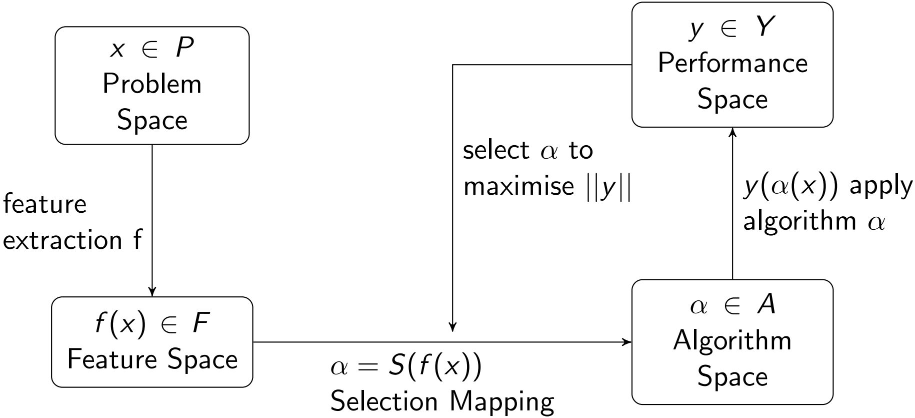


Back

Algorithm Selection Framework

Algorithm Selection Framework (Rice, 1976)

- Algorithm Selection Problem can be casted as a learning problem
- One of the most used application of meta-learning



Problem Space

- datasets
- tasks (what was learned, how was it learned)
- should be similar (see NFL)

Universiteit Leiden

Feature Space

- Characteristics of datasets (also called meta-features)
- Various types (commonly: SIL)
- Should be computationally cheap
- No good library for calculation

Feature Space

- Simple:
  - # Instances, # Attributes, # Classes, Dimensionality, Default Accuracy, # Observations with Missing Values, # Missing Values, % Observations With Missing Values, % Missing Values, # Numeric Attributes, # Nominal Attributes, # Binary Attributes, % Numeric Attributes, % Nominal Attributes, % Binary Attributes, Majority Class Size, % Majority Class, Minority Class Size, % Minority Class
- Statistical:
  - Mean of Means of Numeric Attributes, Mean Standard Deviation of Numeric Attributes, Mean Kurtosis of Numeric Attributes, Mean Skewness of Numeric Attributes
- Information Theoretic:
  - Class Entropy, Mean Attribute Entropy, Mean Mutual Information, Equivalent Number Of Attributes, Noise to Signal Ratio
- Landmarkers:
  - Accuracy of Decision Stump, Kappa of Decision Stump, Area under the ROC Curve of Decision Stump, Accuracy of Naive Bayes, Kappa of Naive Bayes, Area under the ROC Curve of Naive Bayes, Accuracy of k-NN, Kappa of k-NN, Area under the ROC Curve of k-NN, ...

Feature Space

Criticism for meta-feature based approaches:

- It is hard to construct a meta-feature set that adequately characterizes the problem space
- The most successful meta-features, landmarkers, can be computationally expensive, limiting the options
- Because not all classifiers run on all datasets, or take prohibitively long to do so, the meta-dataset usually contains many missing values, complicating the classification task.

Universiteit Leiden

Algorithm Space

- Chose a small set of algorithms (arguably an out-dated opinion)
  - Computational expenses
  - Data sparsity
- Ensure good coverage
- Different biases by choosing representatives from varied model classes

Performance Space

Predictive accuracy dominates the literature as performance measure for algorithm selection.

Arguments in favor:

- Objective measure
- Hard to predict
- Natural solution

Arguments against:

- Problematic for imbalanced problems (use other measures)
- Does not take into account costs (no clear solution)
- Does not take into account run time

Requirements

- `sklearnbot` library
  - `https://github.com/openml/sklearn-bot`
- `openmlcontrib` library
  - Connects `openml` library with `pandas` and `ConfigSpace` library
  - `https://github.com/openml/openml-python-contrib`
- Both not on PyPI, installable from git
- Under active development (contributors welcome)

Plot Results

```python
import sklearnbot # requires openml and openmlcontib and ConfigSpace

# given a study_id (int)
tasks = openml.study.get_study(study_id, 'tasks').tasks

configuration_space = sklearnbot.config_spaces.get_config_space('svc', None)
output_dir = os.path.join(args.output_dir, args.classifier_name)

n_executions = 10
for i in range(n_executions):
    task_id = random.choice(tasks)
    success, run_id, folder = sklearnbot.bot.run_bot_on_task(task_id,
                                                             configuration_space,
                                                             output_dir,
                                                             args.upload_result)
```

Obtain Results

```
import openmlcontrib
import sklearnbot # requires openml and openmlcontib and ConfigSpace

# given a study_id (int)
suite = openml.study.get_study(study_id, 'tasks')
configuration_space = sklearnbot.config_spaces.get_config_space('svc', None)
# given a flow_id (int)
performance_data = openmlcontrib.meta.get_tasks_result_as_dataframe(
  task_ids=suite.tasks,
  flow_id=flow_id,
  num_runs=100,
  per_fold=False,
  configuration_space=config_space,
  evaluation_measures='predictive_accuracy',
  normalize=False
)

meta_features = openmlcontrib.meta.get_tasks_qualities_as_dataframe(
    study.tasks, False, -1, True)
setup_data_with_meta_features = performance_data.join(
    meta_features, on='task_id', how='inner')
```

Auto-sklearn

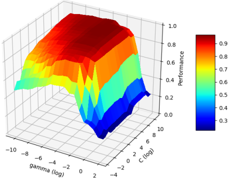
```python
import autosklearn.classification
import sklearn.model_selection
import sklearn.datasets
import sklearn.metrics

# Load data and split into train and test data
X, y = sklearn.datasets.load_digits(return_X_y=True)
X_train, X_test, y_train, y_test = \
  sklearn.model_selection.train_test_split(X, y)
# Let auto-sklearn run for 1h
autosklearn.classification.AutoSklearnClassifier(
  time_left_for_this_task=3600)
automl.fit(X_train, y_train)
y_hat = automl.predict(X_test)
print("Accuracy score",
      sklearn.metrics.accuracy_score(y_test, y_hat))
```

[M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning.

In *Advances in neural information processing systems*, pages 2962–2970, 2015]

Hyperparameter Importance

- Typically unanswered questions:
  - What is a good Configuration Space?
  - What are important hyperparameters?
  - How to sample over these?
- Hyperparameter Importance using OpenML:
  `https://www.youtube.com/watch?v=mS4vL7_rSWQ`



[J. N. van Rijn and F. Hutter. Hyperparameter importance across datasets.

In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2367–2376. ACM, 2018]

What have we learned

suites | effect of a component | warm starting AutoML

- `openml`, `openmlcontrib` and `sklearnbot`
- Also available on Java and R
- Under active development, contributors welcome :)
- This afternoon: Hands on session

References

B. Bischl, G. Casalicchio, M. Feurer, F. Hutter, M. Lang, R. G. Mantovani, J. N. van Rijn, and J. Vanschoren. Openml benchmarking suites and the openml100. *arXiv preprint arXiv:1708.03731*, 2017.

M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *Advances in neural information processing systems*, pages 2962–2970, 2015.

M. J. Post, P. van der Putten, and J. N. van Rijn. Does Feature Selection Improve Classification? A Large Scale Experiment in OpenML. In *Advances in Intelligent Data Analysis XV*, pages 158–170. Springer, 2016.

B. Strang, P. van der Putten, J. N. van Rijn, and F. Hutter. Don't rule out simple models prematurely: A large scale benchmark comparing linear and non-linear classifiers in openml. In *International Symposium on Intelligent Data Analysis*, pages 303–315. Springer, 2018.

J. N. van Rijn and F. Hutter. Hyperparameter importance across datasets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2367–2376. ACM, 2018.