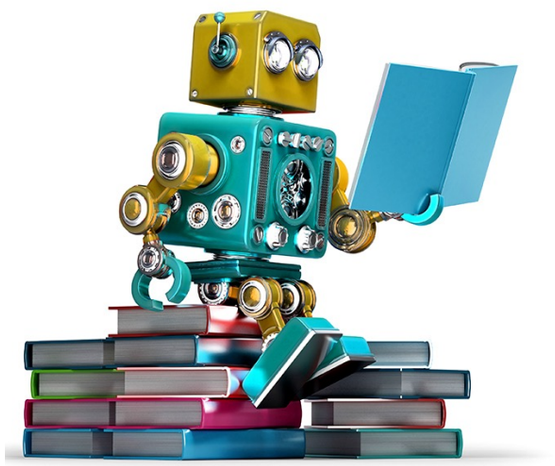
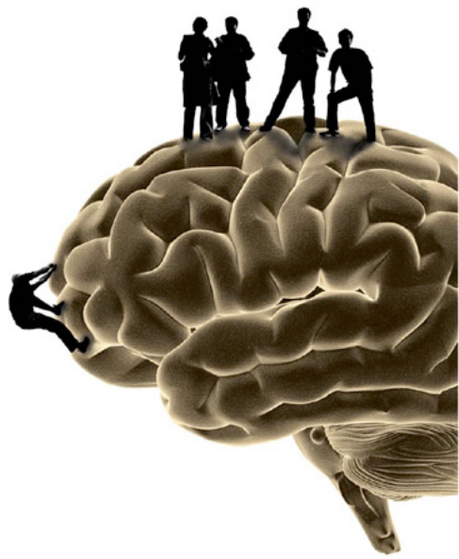
Gradient boosting

Alexandre Gramfort
http://alexandre.gramfort.net



informatics mathematics



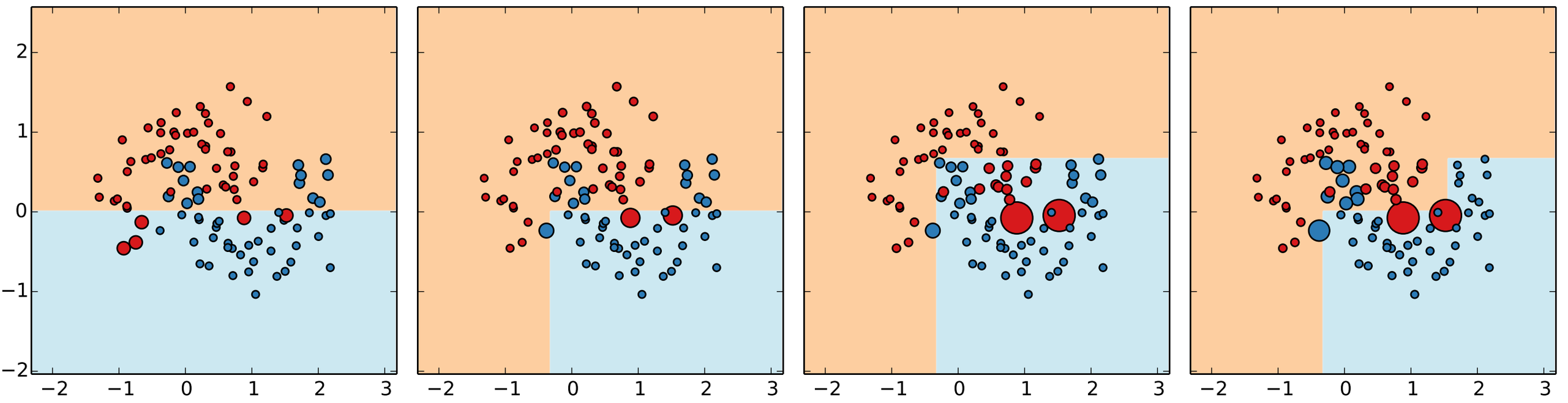
CIRM - Jan 2019

Ensemble of experts: Boosting Principle

- Each model is an expert on the errors of its predecessor
- Iteratively re-weights training examples based on errors
- ERM with weights:

$$\arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_i w_i \ell(f(x_i), y_i)$$

- Example with decision stumps (trees of depth 1):



Adaboost [Y. Freund & R. Schapire, 1995]

For binary classification

ADABOOST$(D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \text{BASE}(\cdot,\cdot), T)$

1 $\quad \mathbf{w}^{(1)} \leftarrow (1/n, \ldots, 1/n)$ $\quad \triangleright$ *initial weights*

2 $\quad$ **for** $t \leftarrow 1$ **to** T

3 $\quad\quad h^{(t)} \leftarrow \text{BASE}\left(D_n, \mathbf{w}^{(t)}\right)$ $\quad \triangleright$ *calling the base learner*

4 $\quad\quad \gamma^{(t)} \leftarrow \sum_{i=1}^{n} w_i^{(t)} h^{(t)}(\mathbf{x}_i) y_i$ $\quad \triangleright$ *edge* $= 1 - 2 \times$ *error*

5 $\quad\quad \alpha^{(t)} \leftarrow \frac{1}{2} \ln\left(\frac{1+\gamma^{(t)}}{1-\gamma^{(t)}}\right)$ $\quad \triangleright$ *coefficient of* $h^{(t)}$

6 $\quad\quad$ **for** $i \leftarrow 1$ **to** n $\quad \triangleright$ *re-weighting the points*

7 $\quad\quad\quad$ **if** $h^{(t)}(\mathbf{x}_i) \neq y_i$ **then**

8 $\quad\quad\quad\quad w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1-\gamma^{(t)}}$

9 $\quad\quad\quad$ **else**

10 $\quad\quad\quad\quad w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{1}{1+\gamma^{(t)}}$

11 $\quad$ **return** $f^{(T)}(\cdot) = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}(\cdot)$

Remark: Freund & Schapire won the Gödel prize 2003

AdaBoost Hands on

Gradient Boosting

- Gradient Boosting generalizes adaboost to any arbitrary loss
- a.k.a. GB(R)T, Gradient boosting (regression) trees
- It was originally proposed by [J. Friedman, 1999]
- Variants of the original GBT algorithm are now state-of-the-art models.
- Numerous successes in Kaggle competitions
- State-of-the-art implementations:
  - XGBoost [Chen & Guestrin, Arxiv 2016] (w. Apple, NVidia)
  - LightGBM [Ke et al., Proc. NIPS 2017] (by Microsoft)
  - CatBoost [Prokhorenkova et al. Arxiv 2017] (by Yandex)

Tree Boosting in a nutshell

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F}$$ (additive ensemble model)

where $\mathcal{F} = \{f(\mathbf{x}) = w_{q(\mathbf{x})}\}$ (set of trees with T leafs)

with $q : \mathbb{R}^m \to \{1, \ldots, T\}$ (tree structure)

and $w \in \mathbb{R}^T$ (leaf weights)

Each f_k has a different tree structure

Remark: Continuous weights even for classification

notations based on [Chen & Guestrin, Arxiv 2016]

Tree Boosting in a nutshell

Objective function:

(smooth convex loss function)

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$ (penalize complex trees)

Remark: Original GBT algo. by Friedman had no regularization

notations based on [Chen & Guestrin, Arxiv 2016]

Tree Boosting in a nutshell

Model is trained in a sequential / additive manner:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

Taylor approximation (order 2):

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ (gradient) $\quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ (hessian)

Removing constant terms:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

notations based on [Chen & Guestrin, Arxiv 2016]

Tree Boosting in a nutshell

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

which can be rewritten:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T$$

where: $I_j = \{i | q(\mathbf{x}_i) = j\}$ (samples in leaf j)

notations based on [Chen & Guestrin, Arxiv 2016]

Tree Boosting in a nutshell

Minimizing this:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T$$

leads to: $$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}$$

and: $$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T.$$

notations based on [Chen & Guestrin, Arxiv 2016]

Tree Boosting in a nutshell

Greedy optimization of:

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T.$$

leads to the following splitting criteria:

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

It corresponds to the loss reduction by splitting: $I = I_L \cup I_R$

notations based on [Chen & Guestrin, Arxiv 2016]

Tree Boosting algorithm

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i,\ H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  $G_L \leftarrow 0,\ H_L \leftarrow 0$
  for j *in sorted*$(I,$ *by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j,\ H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L,\ H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split with max score

Going beyond:

- Approximate splitting (feature binning)
- Parallel implementation (multi-thread & multi-machine)
- Sparsity aware split finding (think one-hot encoding)
- Cache-aware access

Approximate splitting

Algorithm 2: Approximate Algorithm for Split Finding

for $k = 1$ ***to*** m **do**

  Propose $S_k = \{s_{k1}, s_{k2}, \cdots s_{kl}\}$ by percentiles on feature k.

  Proposal can be done per tree (global), or per split(local).

end

for $k = 1$ ***to*** m **do**

$$G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$$

$$H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$$

end

Follow same step as in previous section to find max score only among proposed splits.

GBRT Hands on

See: https://github.com/agramfort/tinygbt

Learning curves

Scaling to Very Very Large Corpora for Natural Language Disambiguation

Michele Banko and Eric Brill
Microsoft Research
1 Microsoft Way
Redmond, WA 98052 USA
{mbanko,brill}@microsof

Abstract

The amount of readily available on-line text has reached hundreds of billions of words and continues to grow. Yet for most core natural language tasks, algorithms continue to be optimized, tested and compared after training on corpora consisting of only one million words or less. In this paper, we evaluate the performance of different learning methods on a prototypical natural language disambiguation task, confusion set disambiguation, when trained on orders of magnitude more labeled data than has previously been

potentially
those learnin

The em
substantial e
a large num
over fixed, a
since we no
data, one ha
have been d
over when t
using much

In this
effects of d
natural lang
we study t
confusable
more trainin

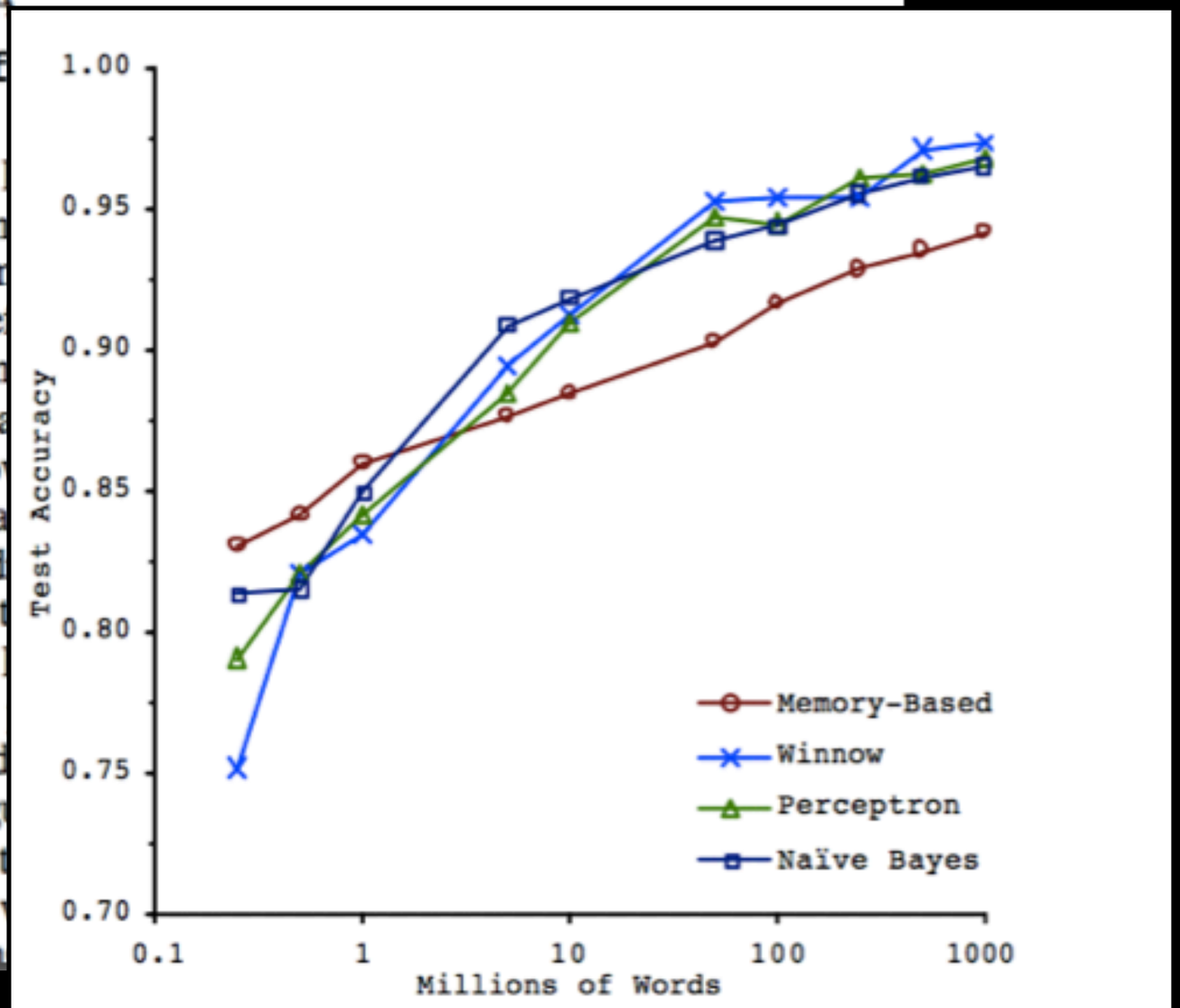
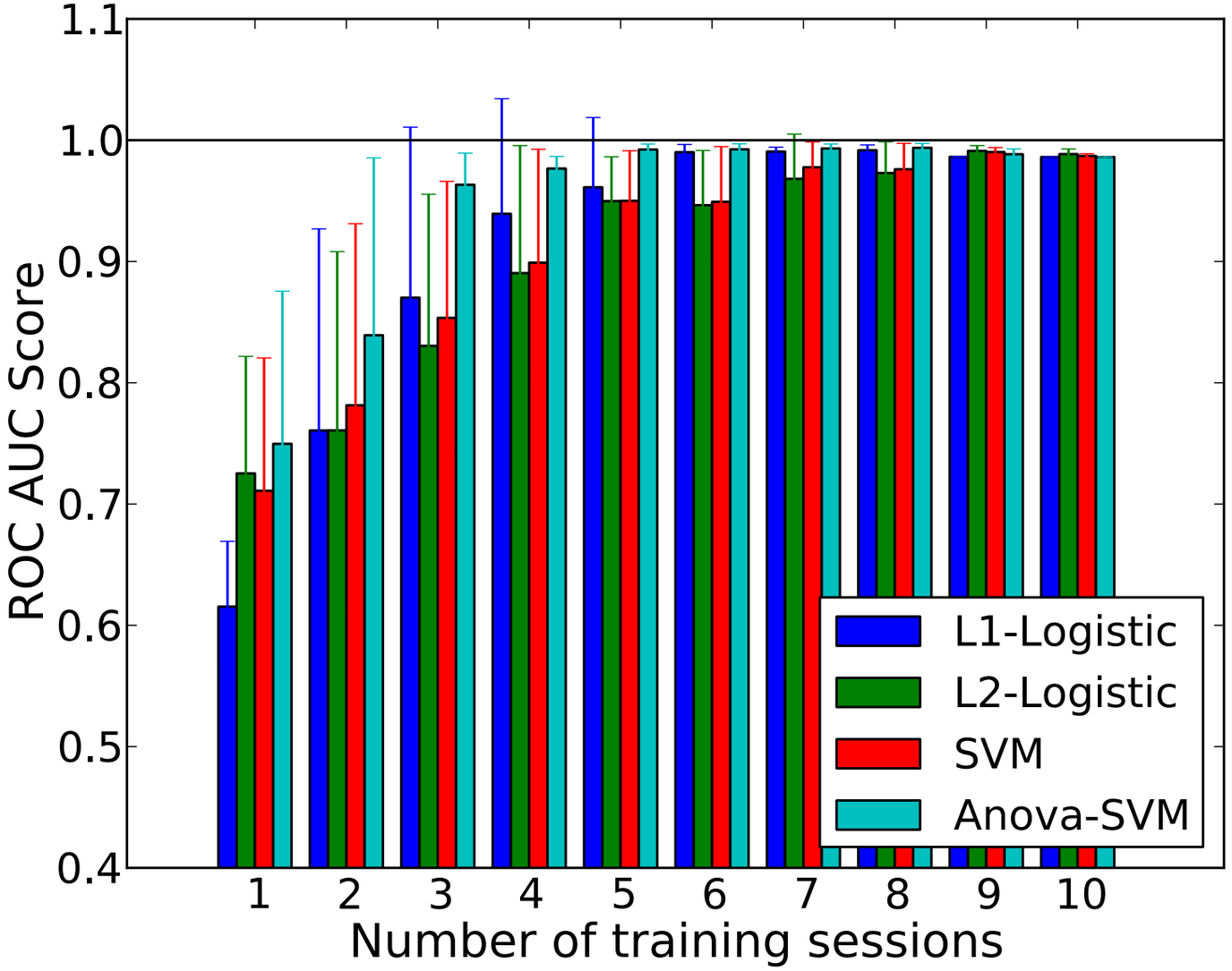


Figure 1. Learning Curves for Confusion Set Disambiguation

Why more data is better?

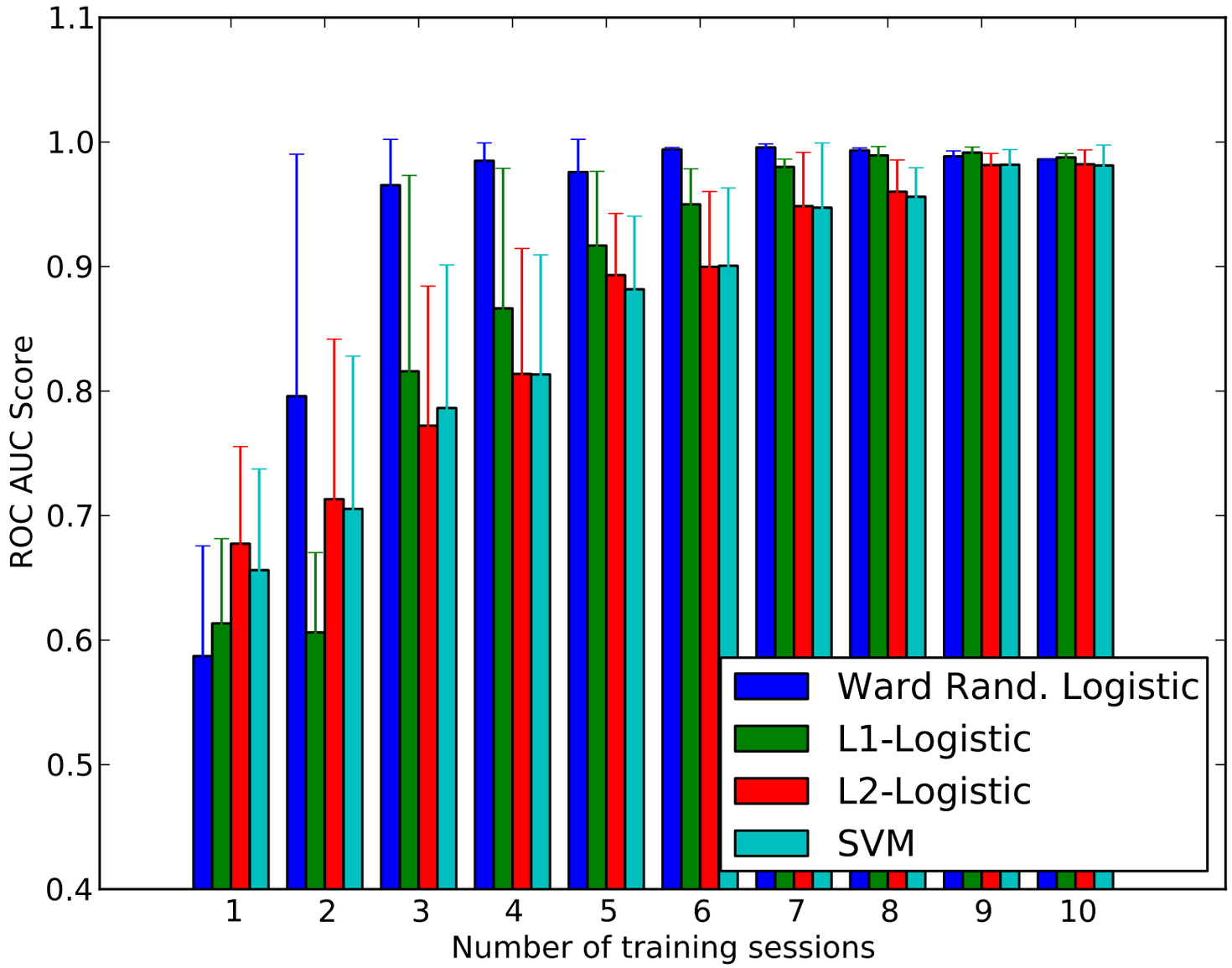


- 5 subjects
- 12 sessions (more than 1000 scans)
- Binary classification
- Test of 2 left-out sessions

- The more data the better
- Almost no noise

Data from *[Haxby et al. 2001]*
Figure from *[Gramfort et al. 2011]*

Sparsity can help



Gramfort et al. Beyond brain reading: randomized sparsity and clustering to simultaneously predict and identify, Proc. MLNI NIPS Workshop 2011

https://hal.inria.fr/hal-00704875

Hands on

Hyperparameter optimization

Related concepts

- Meta learning
- “Learning to learn”
- Automatic Machine Learning (AutoML)

Approaches

- Grid search
- Random search
- Bayesian Optimization
- Gradient based optimization
- Evolutionary optimization

```
losses = ['ls', 'lad', 'huber', 'quantile']

best_n_components_2, best_n_estimators, best_learning_rate, best_loss, best_subsample, best_mi
n_samples_split, best_min_samples_leaf, best_min_weight_fraction_leaf, best_max_depth, best_mi
n_impurity_split, best_alpha, best_error = -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 100
for n_components_2 in range(10, 15, 5):
    for n_estimators in range(100, 1500, 200):
        for learning_rate in frange(0.05, 0.25, 0.05):
            for loss in losses:
                for subsample in frange(0.9, 1, 0.2):
                    for min_samples_split in range(2, 3, 1):
                        for min_samples_leaf in range(1, 2, 1):
                            for min_weight_fraction_leaf in frange(0, 0.1, 0.1):
                                for max_depth in range(3, 5, 2):
                                    for min_impurity_split in frange(1e-7, 2e-7, 1e-7):
                                        for alpha in frange(0.9, 1.0, 0.1):
                                            error = []
                                            for t in range(5):
                                                skf = ShuffleSplit(n_splits=2, test_size=0.2,
random_state=57)

                                                skf_is = list(skf.split(X_df))[0]
                                                error.append(train_test_model(X_df, y_df, skf_
is, FeatureExtractorClf, Classifier, FeatureExtractorReg, Regressor, n_components_2, n_estimat
ors, learning_rate, loss, subsample, min_samples_split, min_samples_leaf, min_weight_fraction_
leaf, max_depth, min_impurity_split, alpha))
                                            if np.mean(error) < best_error:
                                                best_error = np.mean(error)
                                                best_n_components_2, best_n_estimators, best_l
earning_rate, best_loss, best_subsample, best_min_samples_split, best_min_samples_leaf, best_m
in_weight_fraction_leaf, best_max_depth, best_min_impurity_split, best_alpha = n_components_2,
n_estimators, learning_rate, loss, subsample, min_samples_split, min_samples_leaf, min_weight_
fraction_leaf, max_depth, min_impurity_split, alpha
print('best_n_components_2 = %s - best_n_estimator = %s - best_learning_rate = %s - best_loss
= %s - best_subsample = %s - best_min_samples_split = %s - best_min_samples_leaf = %s - best_m
in_weight_fraction_leaf = %s - best_max_depth = %s - best_min_impurity_split = %s - best_alpha
```

Grid search

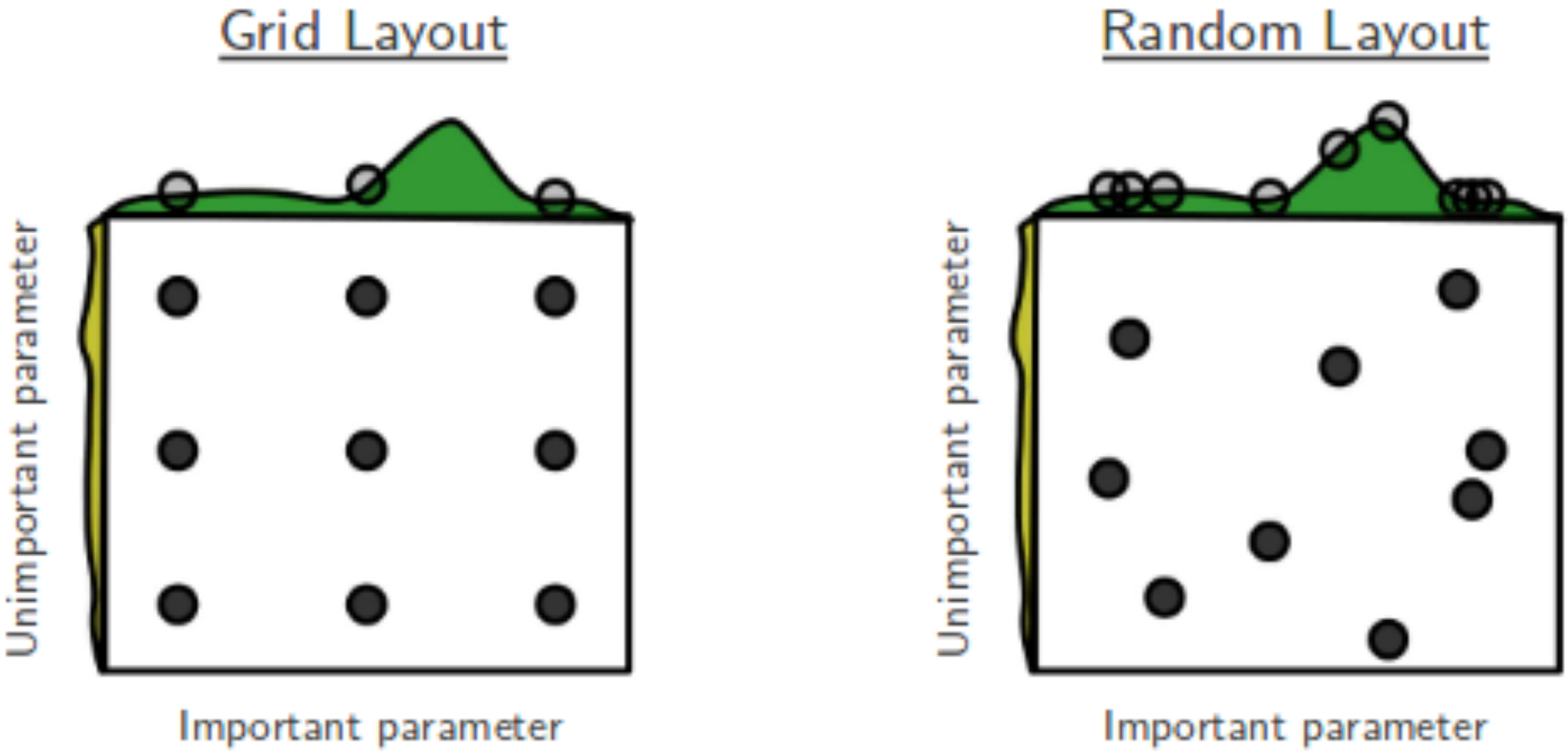
Grid search

```
>>> param_grid = {'C': [1e3, 5e3, 1e4, 5e4, 1e5],
                  'gamma': [0.0001, 0.0005, 0.001, 0.005,
                            0.01, 0.1], }
>>> clf = GridSearchCV(SVC(kernel='rbf',
                           class_weight='balanced'),
                           param_grid, cv=5)
>>> clf = clf.fit(X_train_pca, y_train)
>>> print("Best estimator found by grid search:")
>>> print(clf.best_estimator_)
```

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

https://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html

Random Search



http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

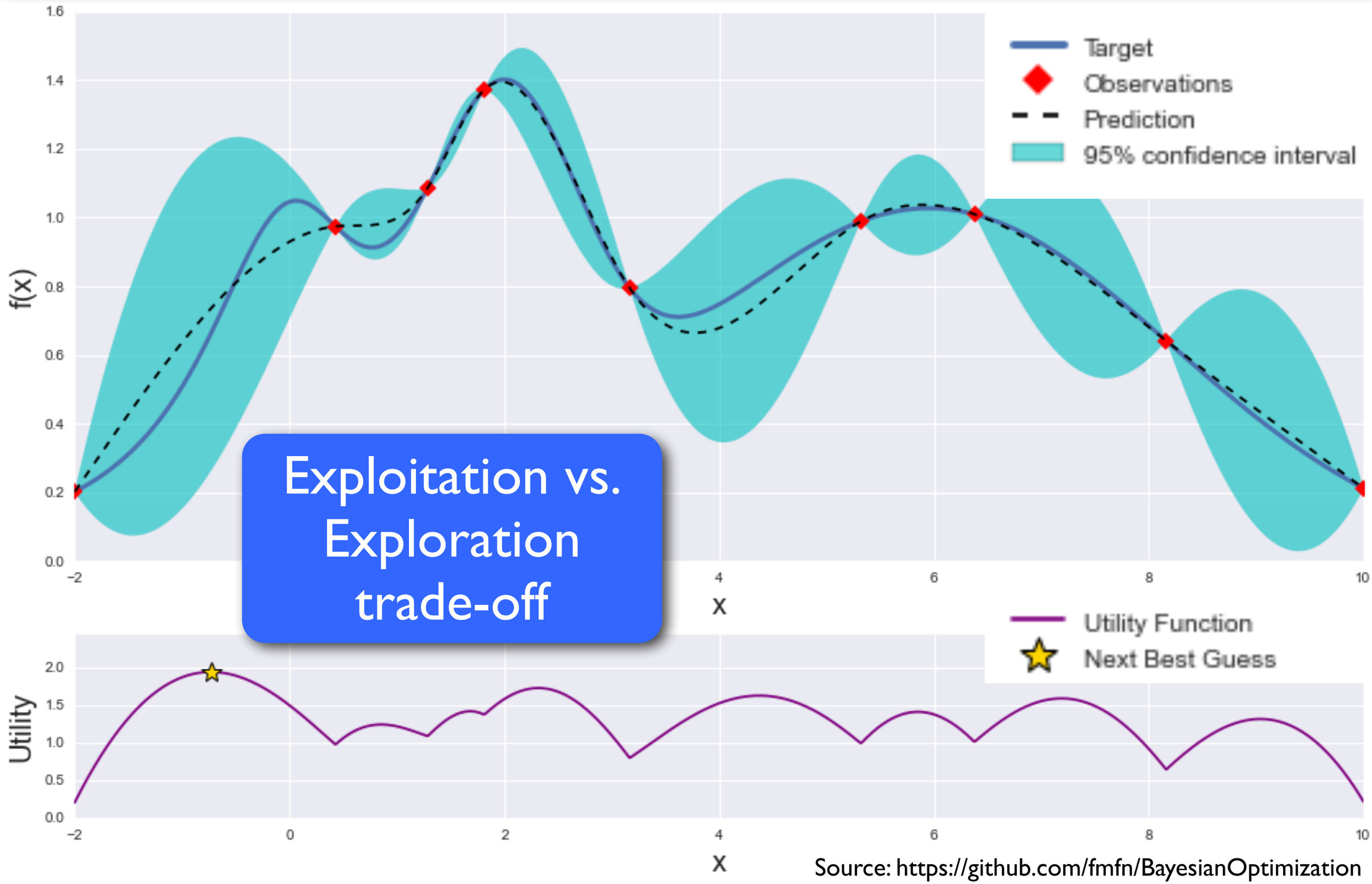
Random Search

```
# specify parameters and distributions to sample from
>>> param_dist = {"max_depth": [3, None],
                  "max_features": randint(1, 11),
                  "min_samples_split": randint(2, 11),
                  "bootstrap": [True, False],
                  "criterion": ["gini", "entropy"]}
>>> clf = RandomForestClassifier(n_estimators=20)
>>> n_iter_search = 20
>>> random_search = \
        RandomizedSearchCV(clf,
                           param_distributions=param_dist,
                            n_iter=n_iter_search, cv=5)
>>> random_search.fit(X, y)
```

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

https://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html

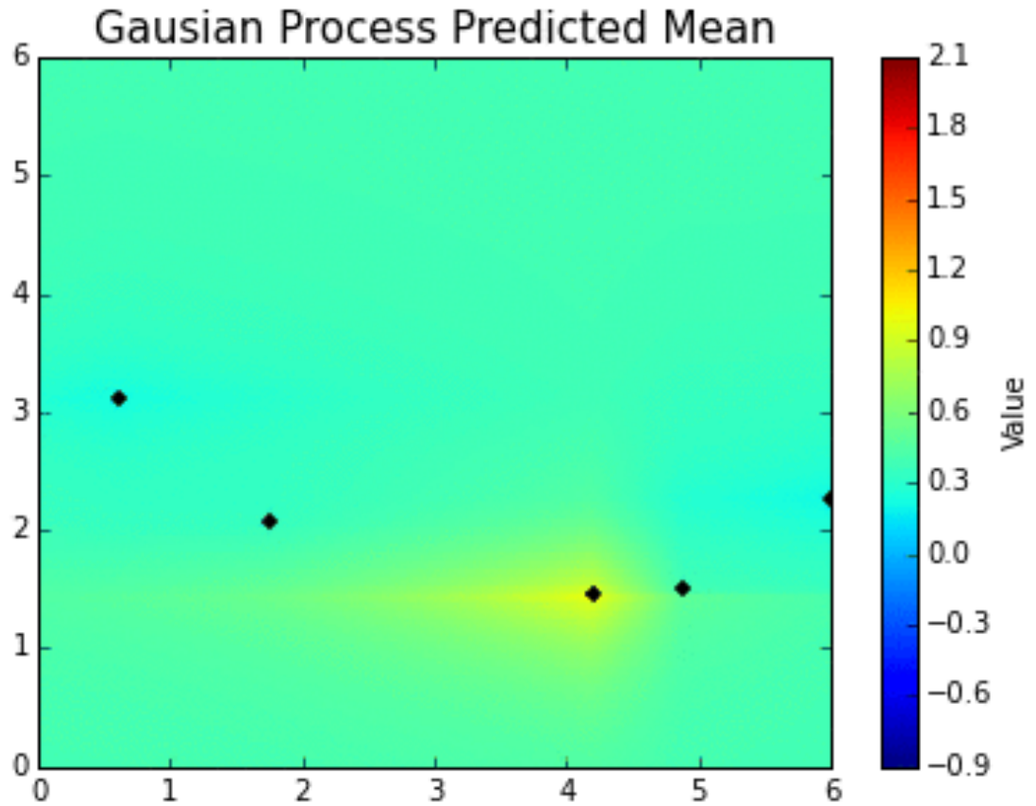
Bayesian Optimization

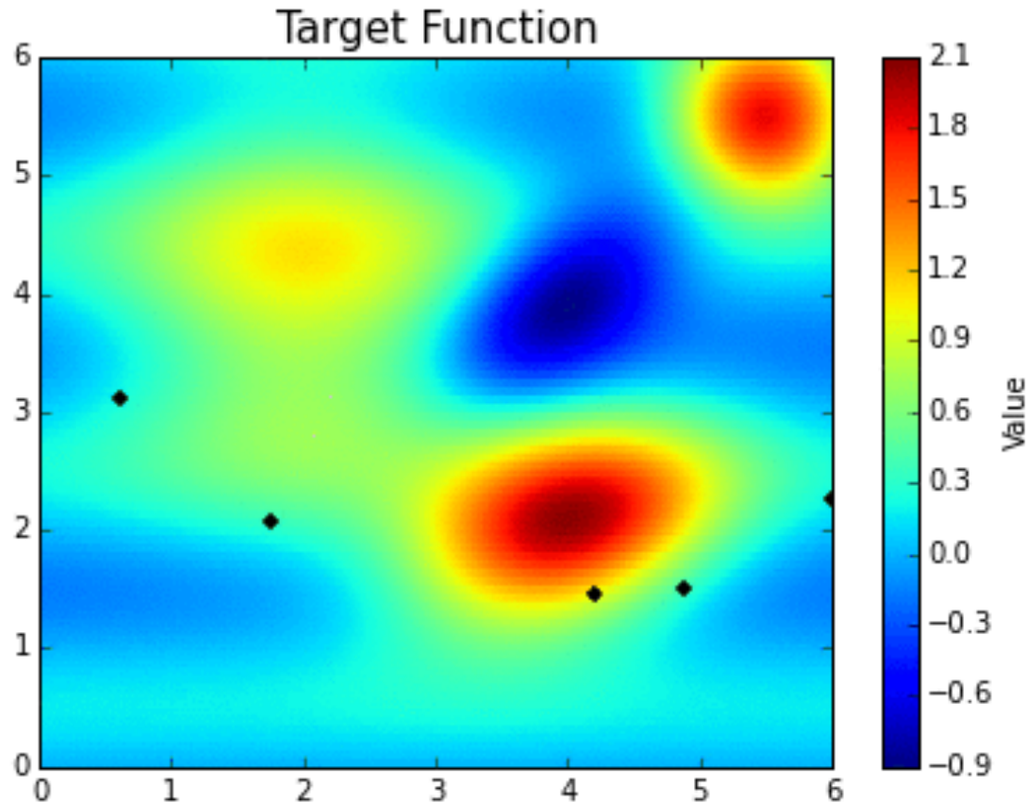


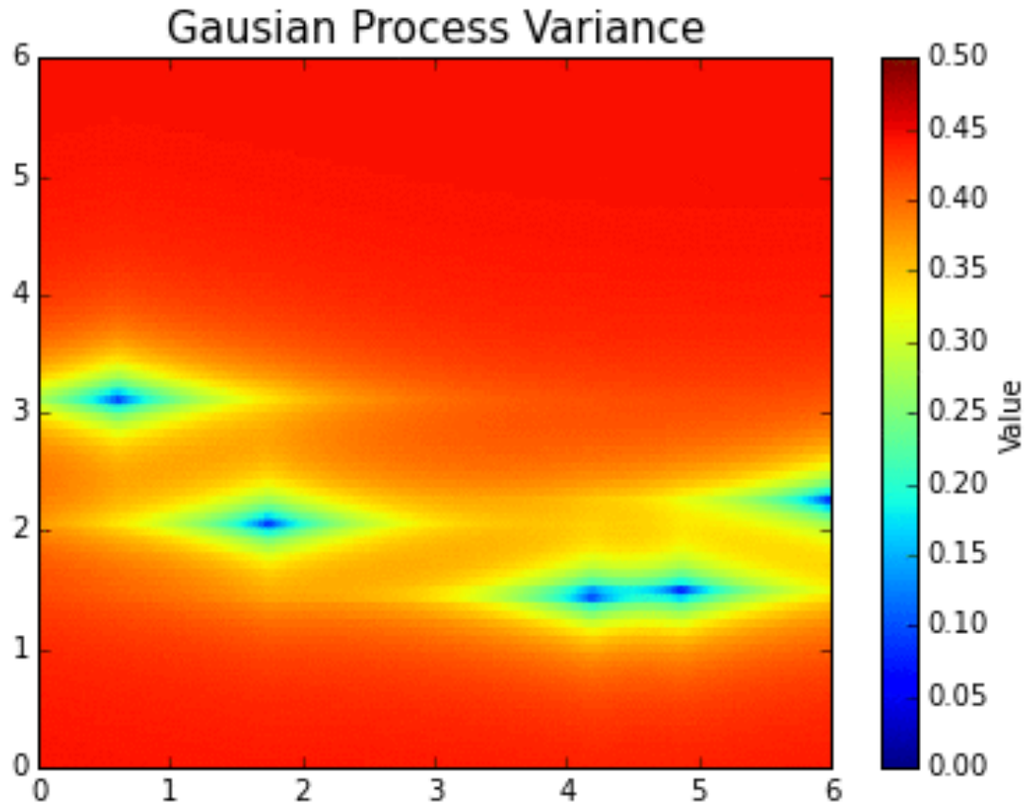
Exploitation vs. Exploration trade-off

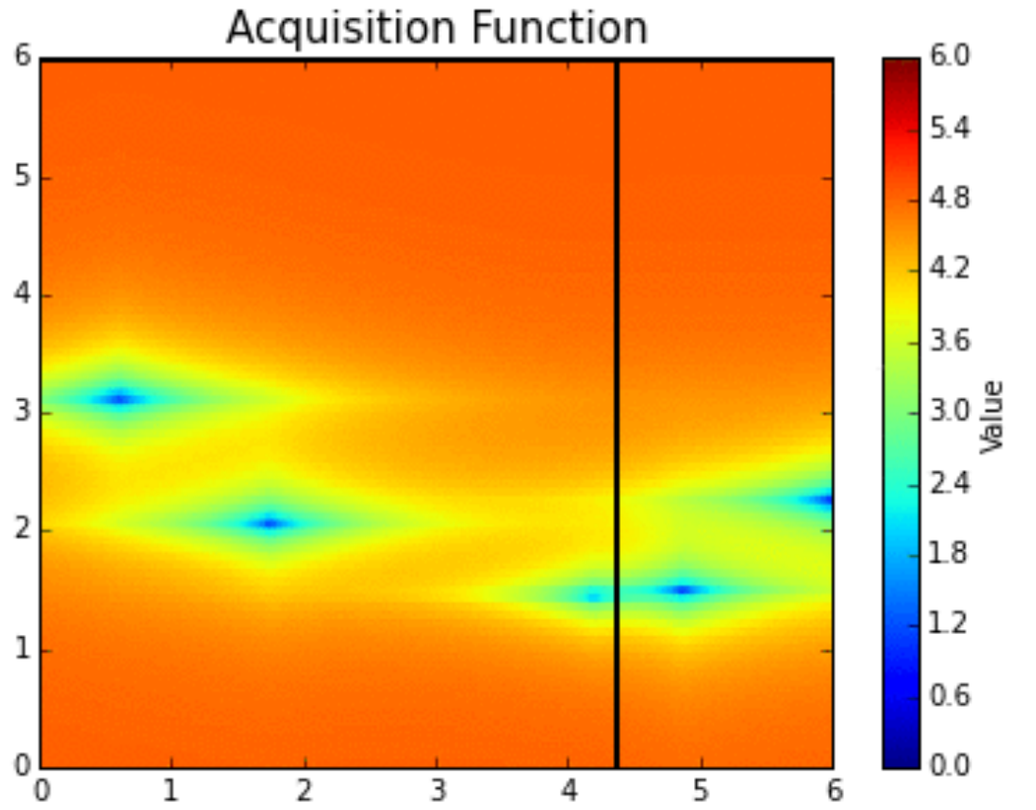
Source: https://github.com/fmfn/BayesianOptimization

Bayesian Optimization in Action



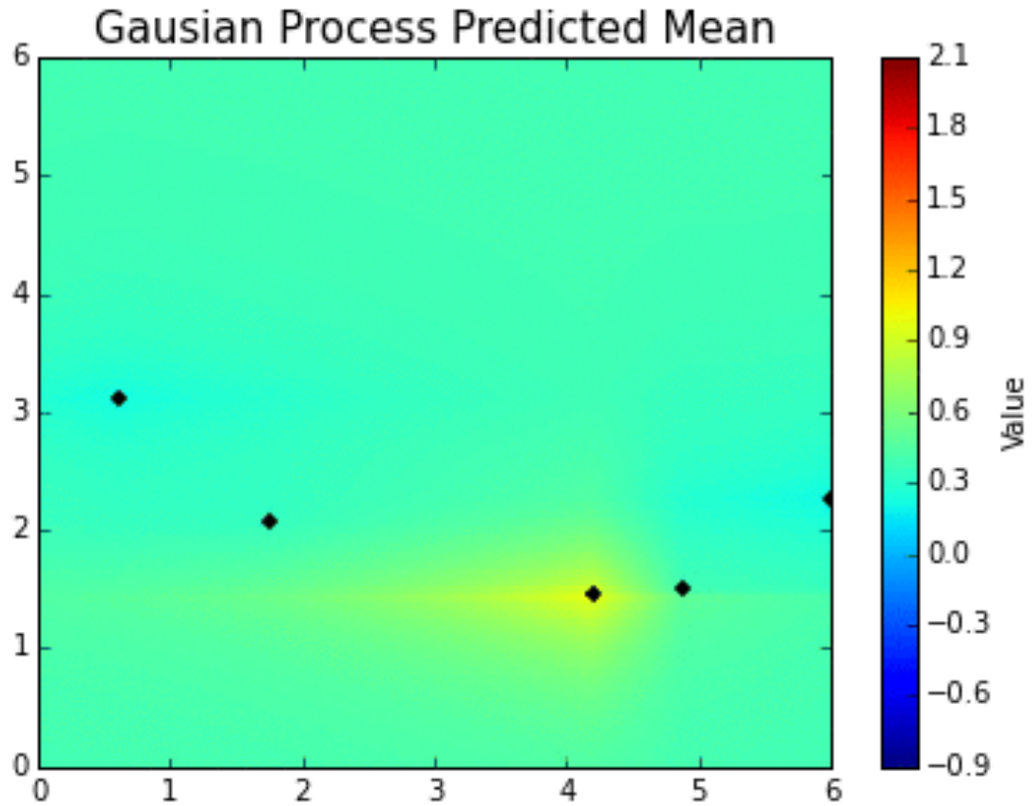


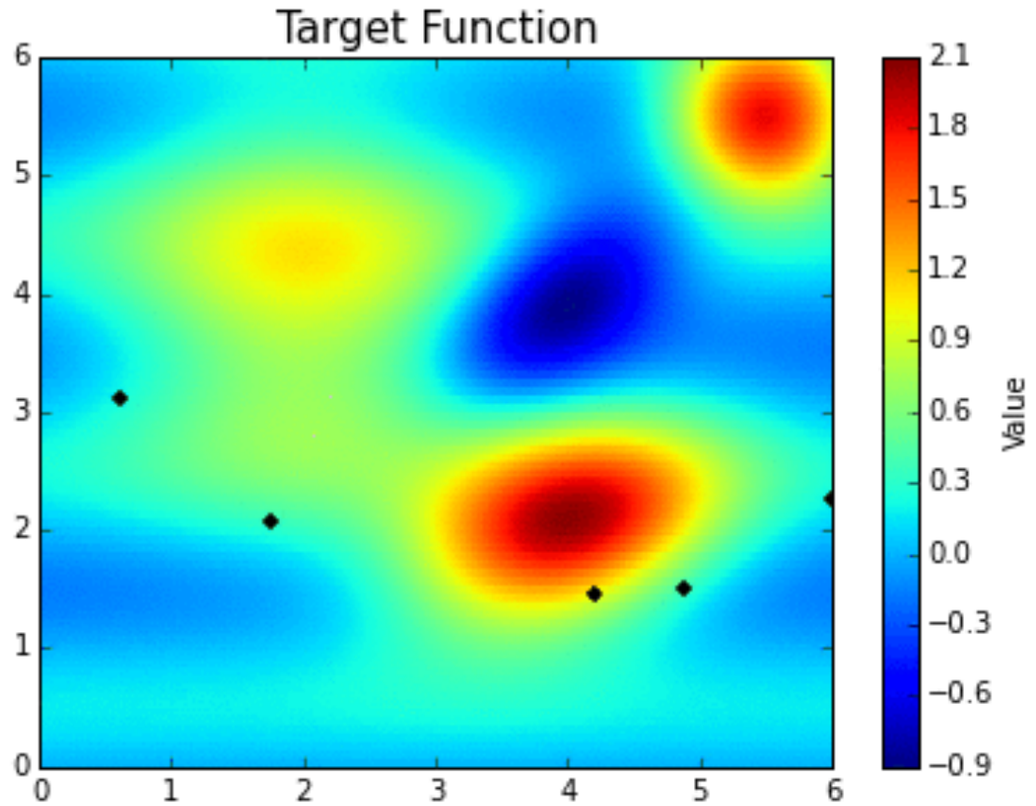


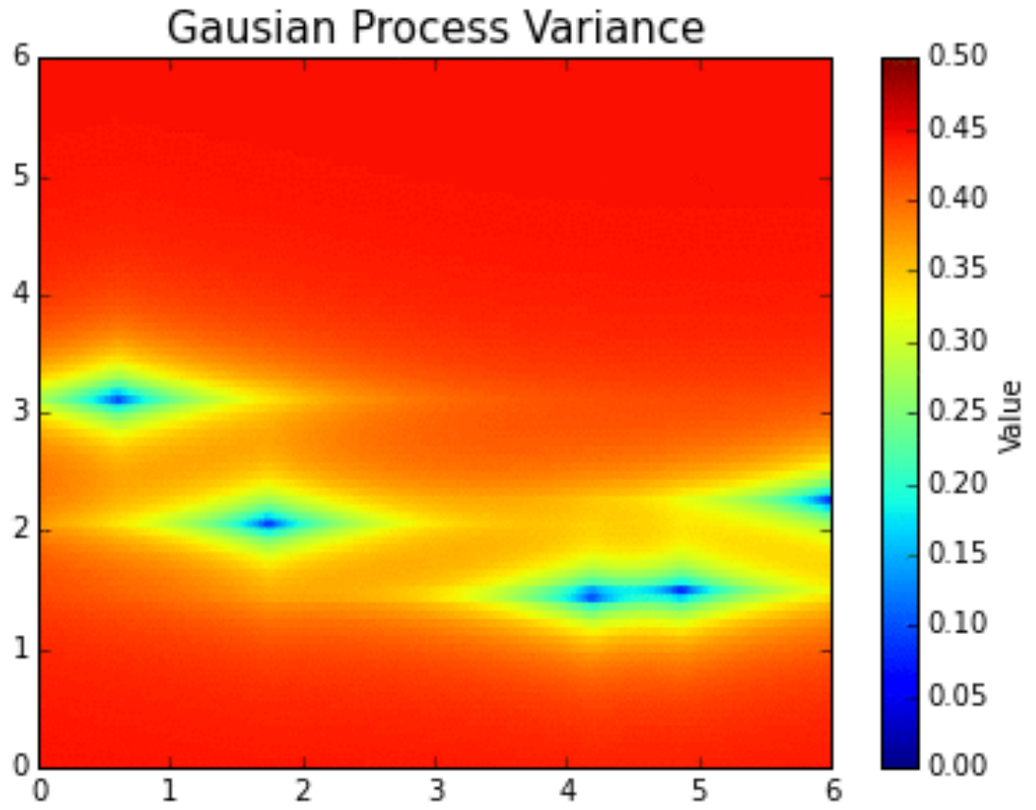


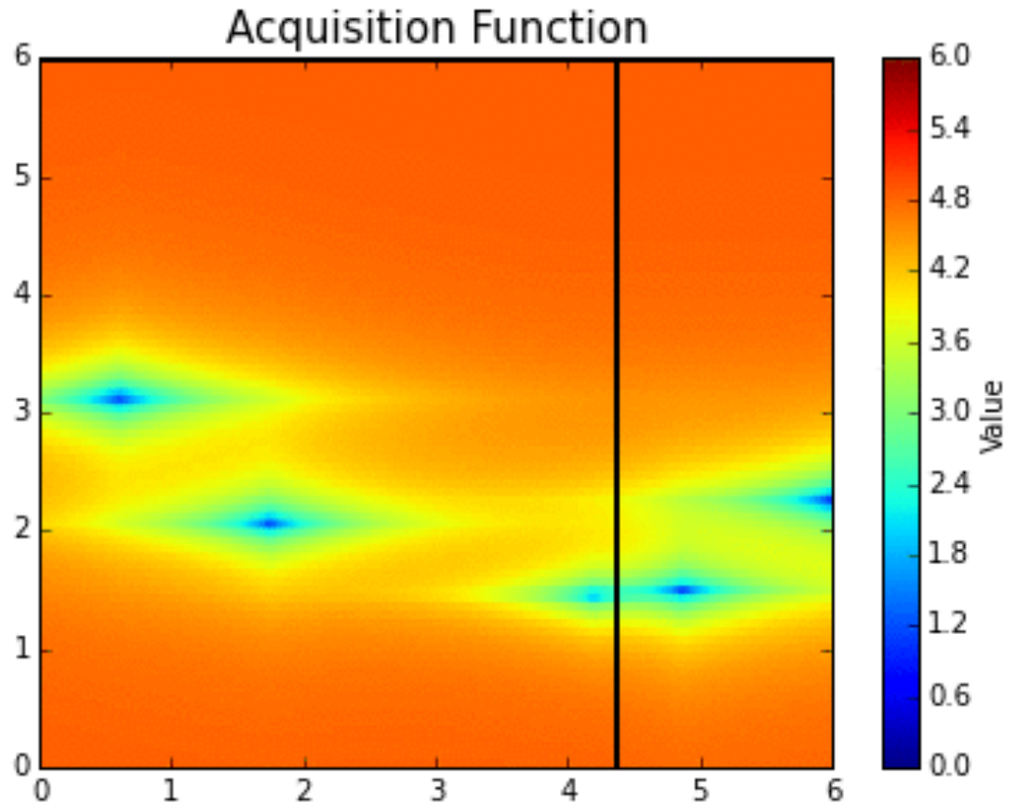
Source: https://github.com/fmfn/BayesianOptimization

Bayesian Optimization in Action









Source: https://github.com/fmfn/BayesianOptimization

Ex. scikit-optimize

```
from skopt import BayesSearchCV

# log-uniform: search over p = exp(x) by varying x
opt = BayesSearchCV(
    SVC(),
    {
        'C': (1e-6, 1e+6, 'log-uniform'),
        'gamma': (1e-6, 1e+1, 'log-uniform'),
        'degree': (1, 8),  # integer valued parameter
        'kernel': ['linear', 'poly', 'rbf'],  # categorical
    },
    n_iter=32
)

opt.fit(X_train, y_train)
```

https://scikit-optimize.github.io/#skopt.BayesSearchCV

Gradient based

- **Idea:** Compute gradient of cross-validation score w.r.t. hyper parameters

- Eg. use automatic differentiation with a smooth loss and an iterative algorithm like gradient descent

https://arxiv.org/abs/1502.03492
https://arxiv.org/abs/1602.02355
https://github.com/HIPS/hypergrad

Software

- hyperopt
  https://github.com/hyperopt/hyperopt
- hyperband
  https://github.com/zygmuntz/hyperband
- scikit-optimize
  https://scikit-optimize.github.io/
- smac
  https://github.com/automl/SMAC3
- spearmint https://github.com/HIPS/Spearmint

Hands on

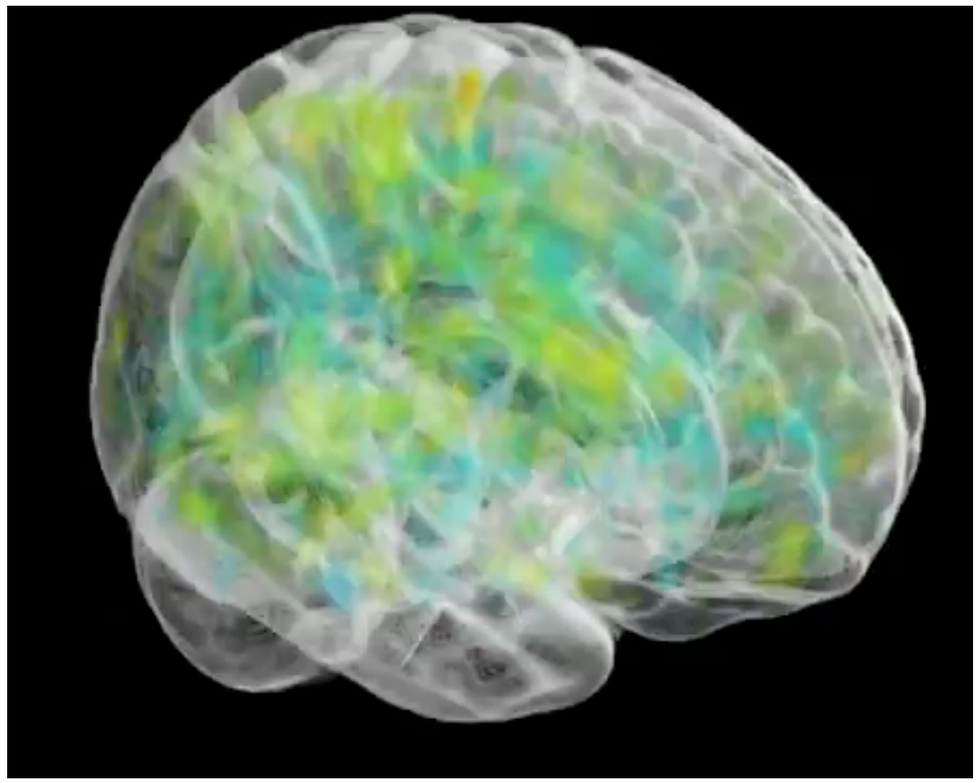
Let's take a step back…

Let's take a step back…

Food for thoughts

What is the difference?

Self-driving cars



Predicting autism from MRI?

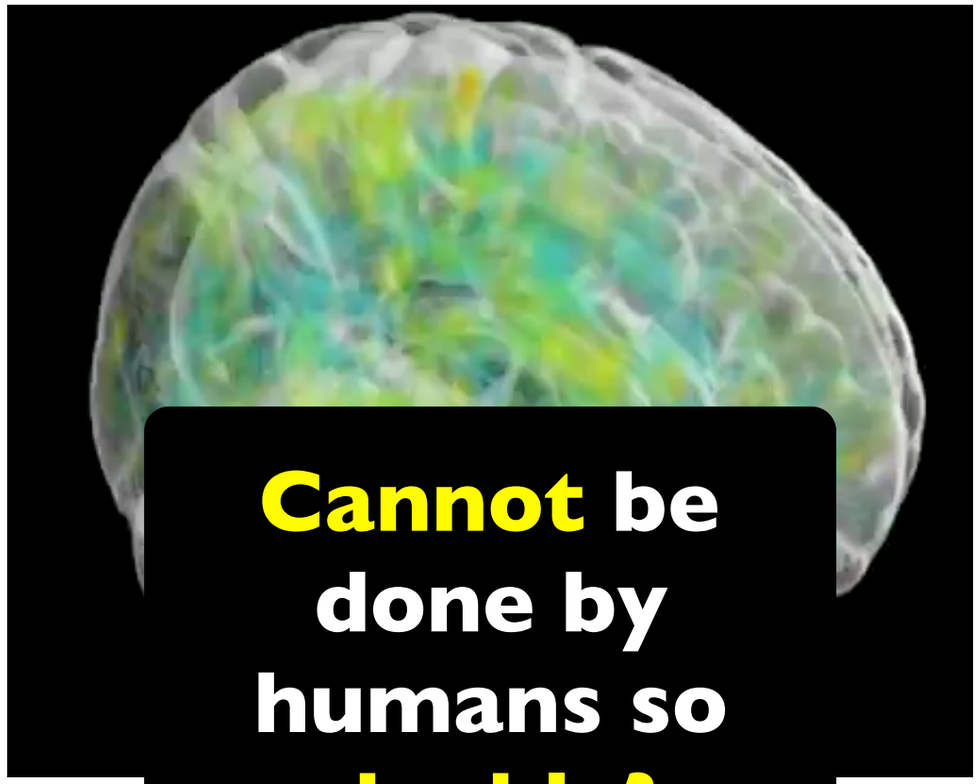
Reconstruction movies from MRI?

Controlling a robotic arm from EEG signals?

Food for thoughts

What is the difference?

Self-driving cars



Predicting autism from MRI?

Reconstruction movies from MRI?

Controlling a robotic arm from EEG signals?

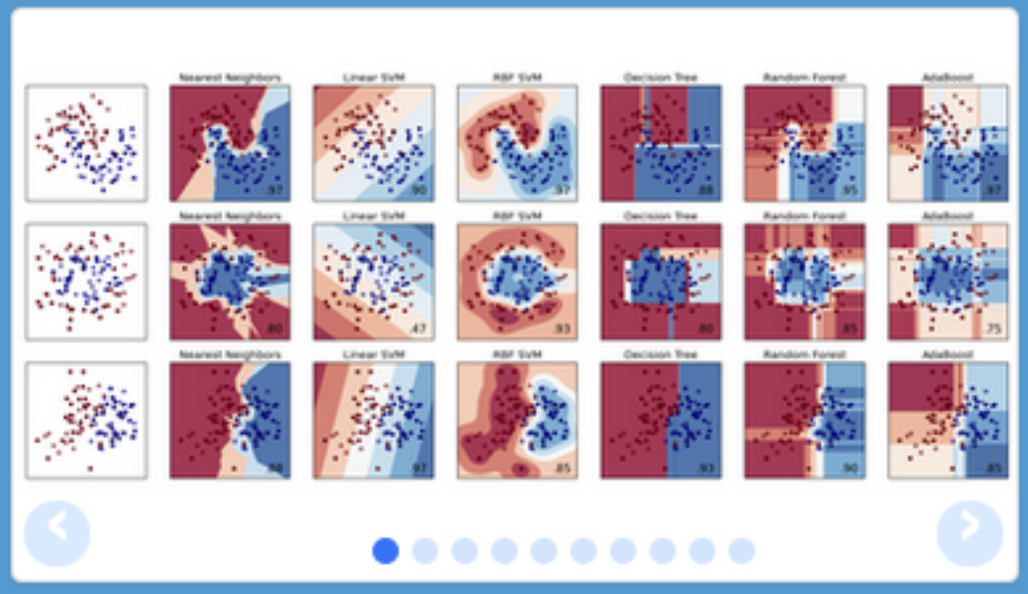
The human learning problem

Observations

Supervised model,
ReLU, Conv Net,
ADAM, GBRT,
AdaBoost etc...

brain
imaging
people
(doctors,
neuro-
scientists,
…)

How do you solve
this learning problem?



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: *SVM*, *nearest neighbors*, *random forest*, ... — *Examples*

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: *SVR*, *ridge regression*, *Lasso*, ... — *Examples*

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: *k-Means*, *spectral clustering*, *mean-shift*, ... — *Examples*

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: *PCA*, *feature selection*, *non-*

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: *grid search*, *cross validation*

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: *preprocessing*, *feature extraction*.

MNE Tutorials Gallery Manual API FAQ Cite Site Page Search

MNE

MEG + EEG ANALYSIS & VISUALIZATION

http://www.martinos.org/mne

MNE is a community-driven software package designed for for **processing electroencephalography (EEG) and magnetoencephalography (MEG) data** providing comprehensive tools and workflows for:

1. Preprocessing
2. Source estimation
3. Time–frequency analysis
4. Statistical testing
5. Estimation of functional connectivity
6. Applying machine learning algorithms
7. Visualization of sensor- and source-space data

MNE includes a comprehensive Python package (provided under the simplified BSD license), supplemented by tools compiled from C code for the LINUX and Mac OSX operating systems, as well as a MATLAB toolbox.

Documentation

- Getting Started
- What's new
- Cite MNE
- Related publications
- Tutorials
- Examples Gallery
- Manual
- API Reference
- Frequently Asked Questions
- Advanced installation and setup
- MNE with CPP

MNE software for processing MEG and EEG data, A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, M. Hämäläinen, Neuroimage 2013

Celer

This is a library to run the Constraint Elimination for the Lasso with Extrapolated Residuals (Celer) algorithm [1]. This algorithm uses an extrapolated dual point which enables a tight control of optimality and quick feature identification.

Installation

First clone the repository available at https://github.com/mathurinm/celer:

```
$ git clone https://github.com/mathurinm/celer.git
$ cd celer/
```

We recommend to use the Anaconda Python distribution, and create a conda environment with:

```
$ conda env
```

Then, you can co

```
$ source act
$ pip instal
```

To check if every

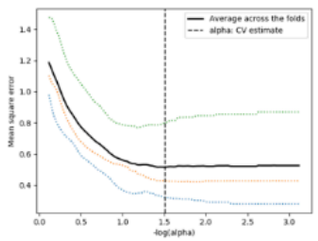
Celer: a Fast Solver for the Lasso with Dual Extrapolation

Mathurin Massias [1] **Alexandre Gramfort** [1] **Joseph Salmon** [2]

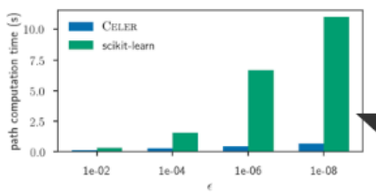
Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

https://github.com/mathurinm/celer

Examples Gallery



Run LassoCV for cross-validation on Leukemia dataset



The example runs the Celer algorithm on the Leukemia dataset which is dense dataset.

Lasso path computation on Leukemia dataset

Lasso path computation on Finance/log1p dataset

off the shelf examples

Download all examples in Python source code: auto_examples_python.zip

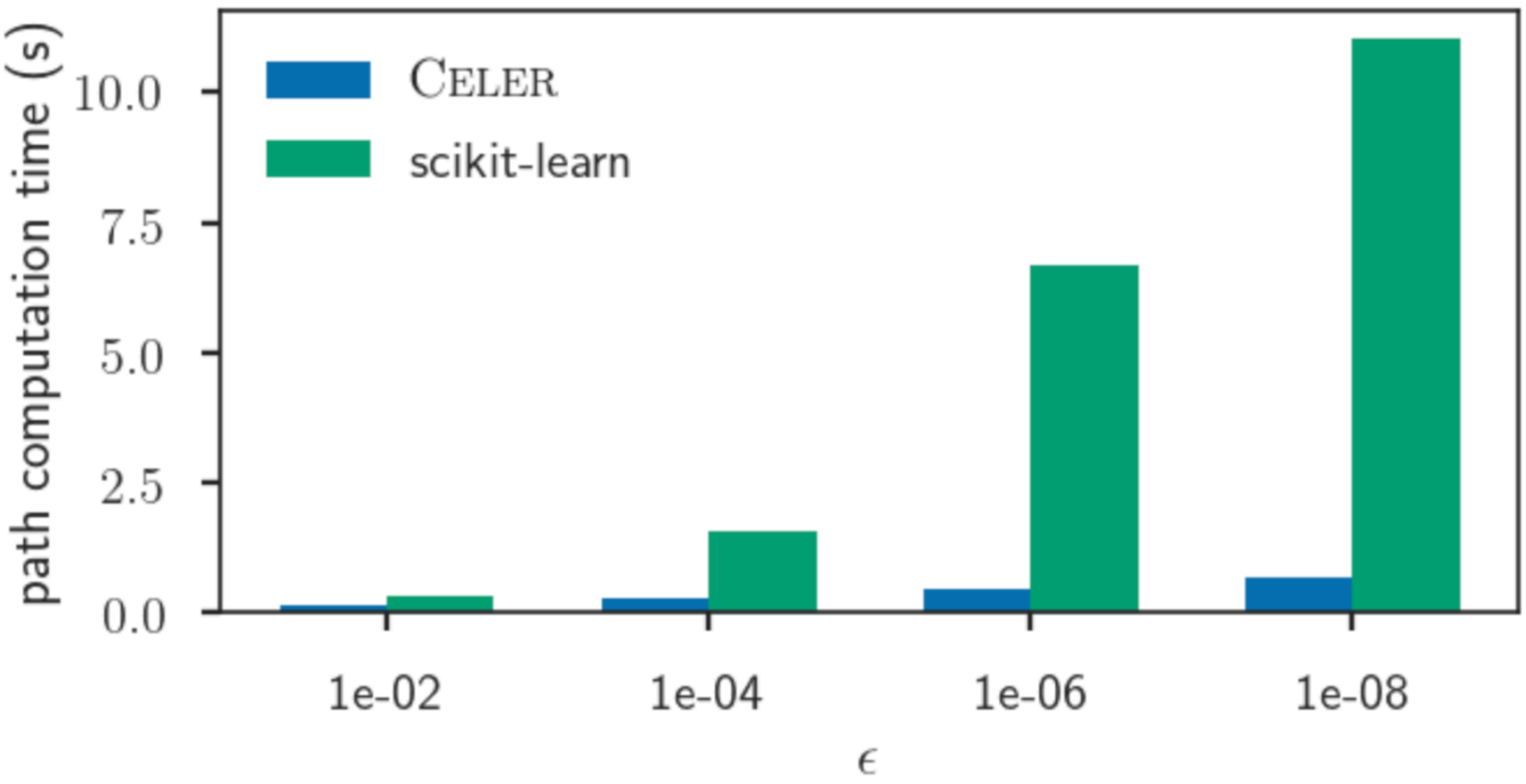
Download all examples in Jupyter notebooks: auto_examples_jupyter.zip

[Massias, Gramfort, Salmon (2018), Celer: a Fast Solver for the Lasso with Dual Extrapolation, ICML]

Lasso path computation on Leukemia dataset

The example runs the Celer algorithm on the Leukemia dataset which is dense dataset.

Running time is compared with the scikit-learn implementation.



Reproducible benchmarks

[Massias, Gramfort, Salmon (2018), Celer: a Fast Solver for the Lasso with Dual Extrapolation, ICML]

Picard

This is a library to run the Preconditioned ICA for Real Data (PICARD) algorithm [1] and its orthogonal version (PICARD-O) [2]. These algorithms show fast convergence even on real data for which sources independence do not perfectly hold.

Installation

We recommend the Anaconda Python distribution. Otherwise, to install `picard`, you first need to install its dependencies:

```
$ pip install numpy matplotlib numexpr scipy
```

Then install Picard:

```
$ pip install python-picard
```

If you do not have admin privileges on the computer, use the `--user` flag with *pip*. To upgrade, use the `--upgrade` flag provided by *pip*.

To check if everything worked fine, you can do:

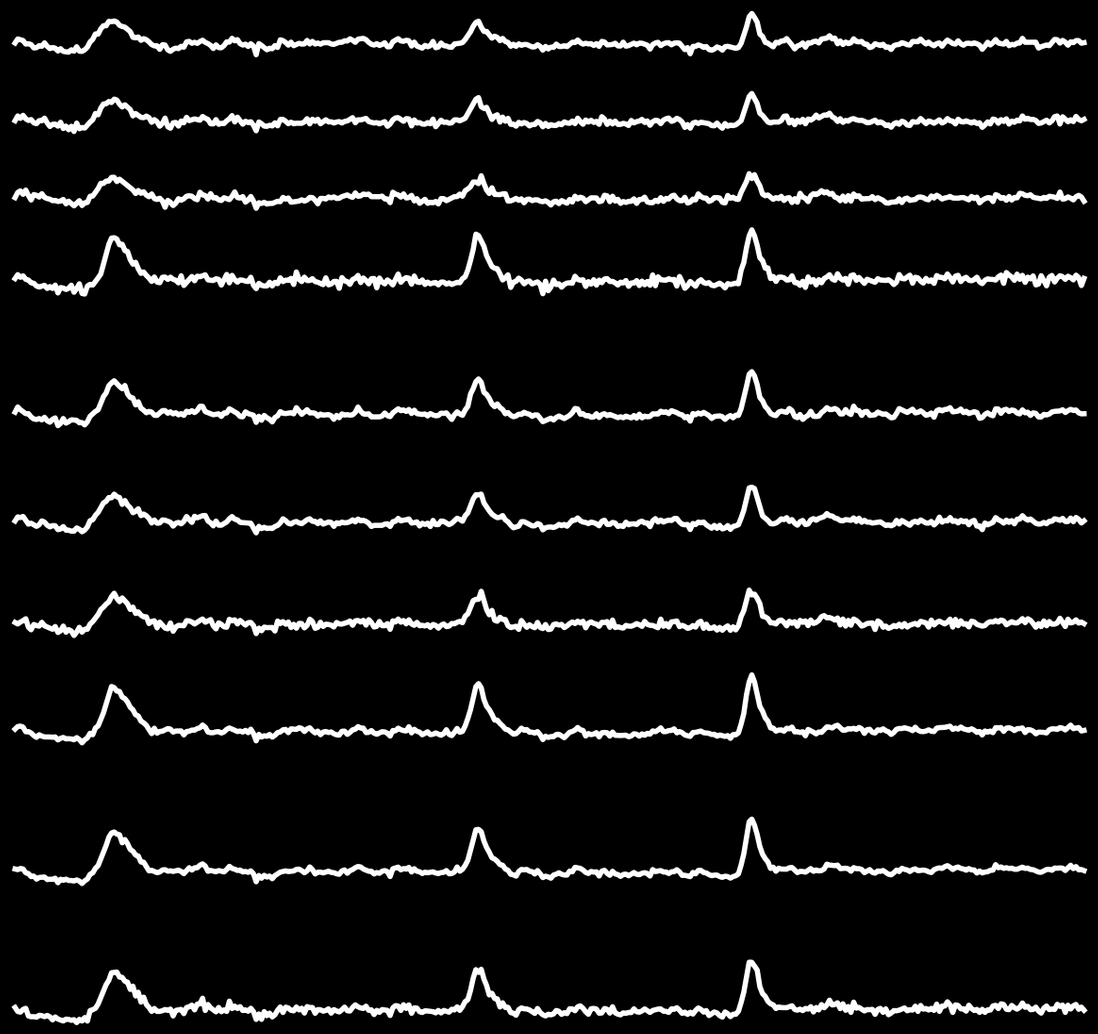
```
$ python -c 'import picard'
```

and it should not give any error message.

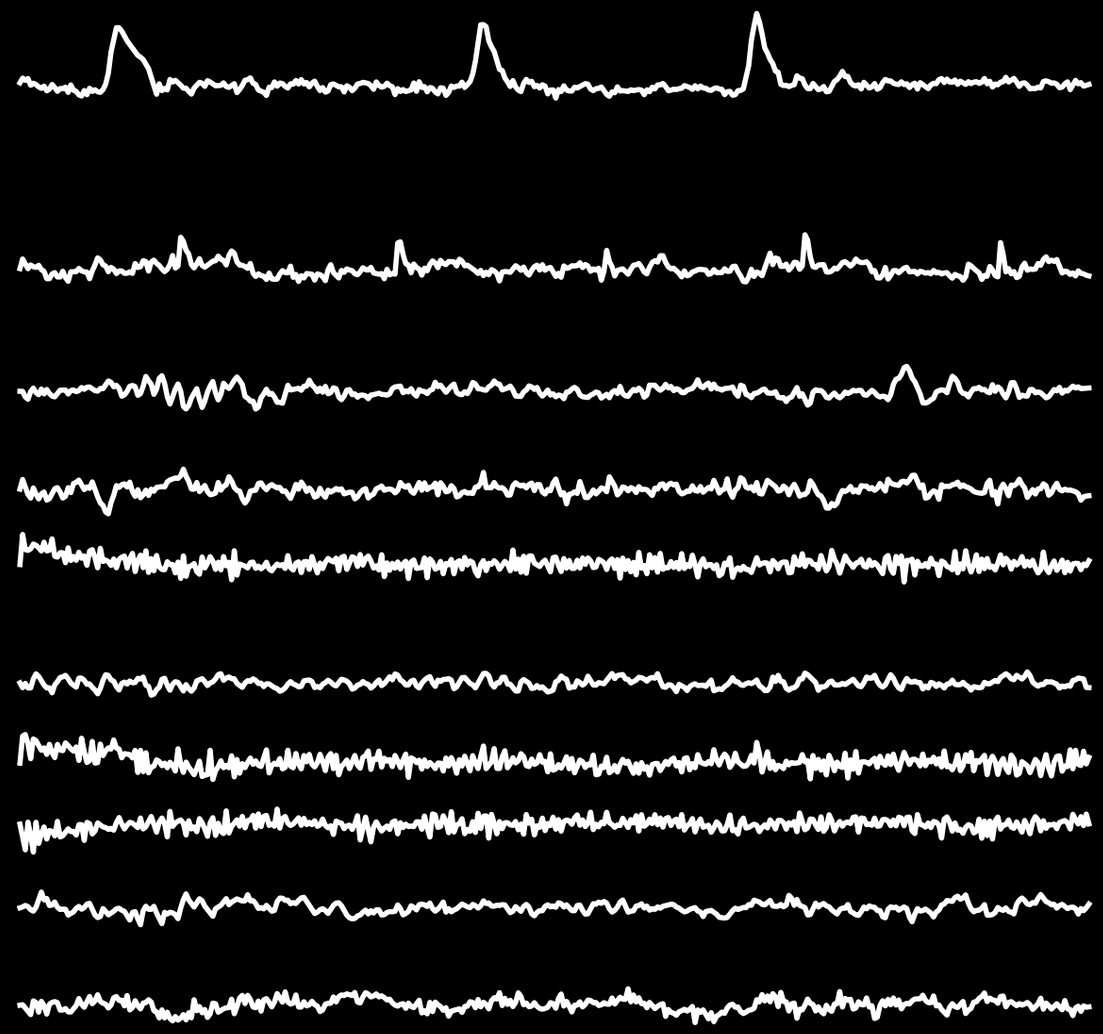
https://pierreablin.github.io/picard/

[Ablin, Cardoso, Gramfort, (2018). Faster independent component analysis by preconditioning with Hessian approximations. IEEE Trans. on Signal Processing]

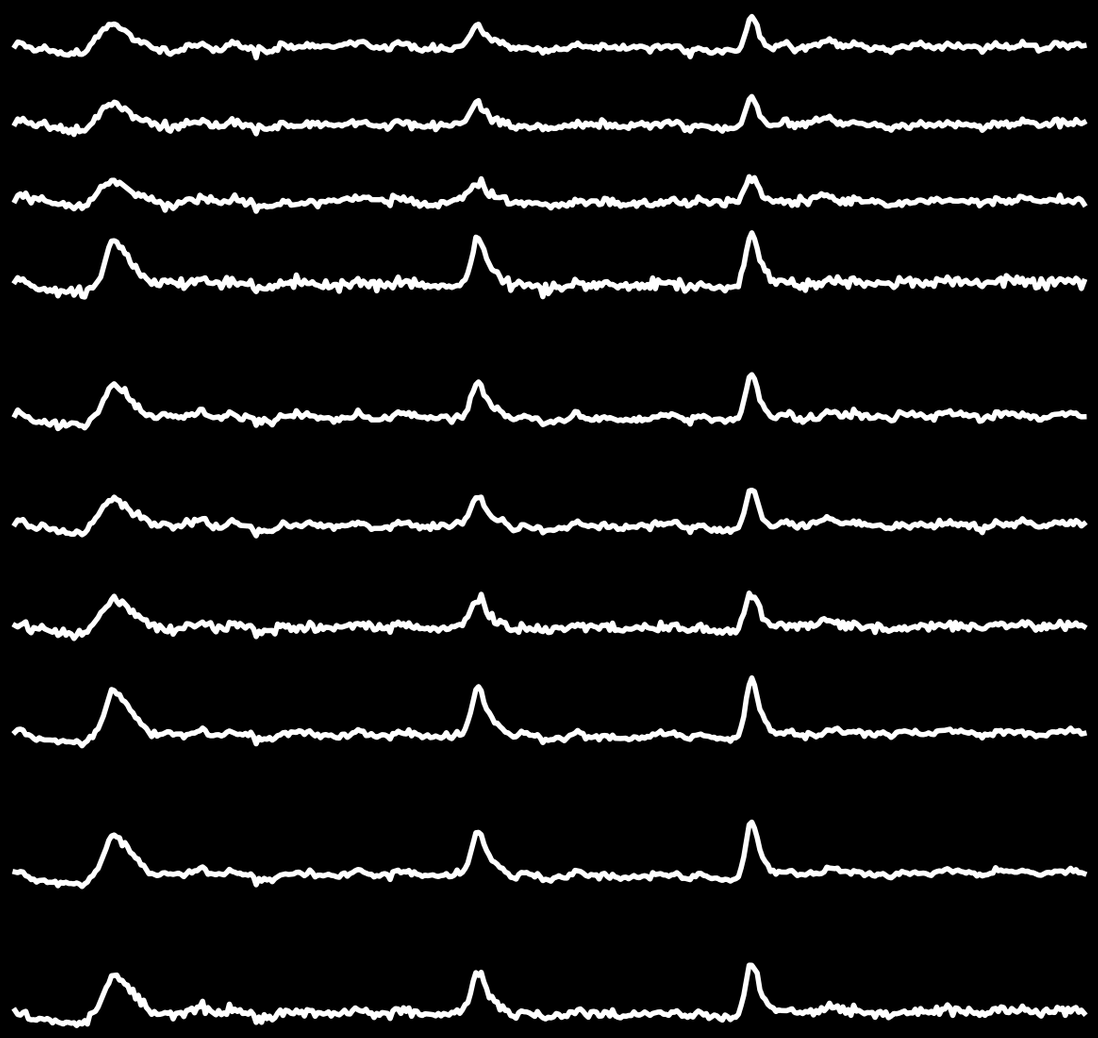
Observations (raw EEG)



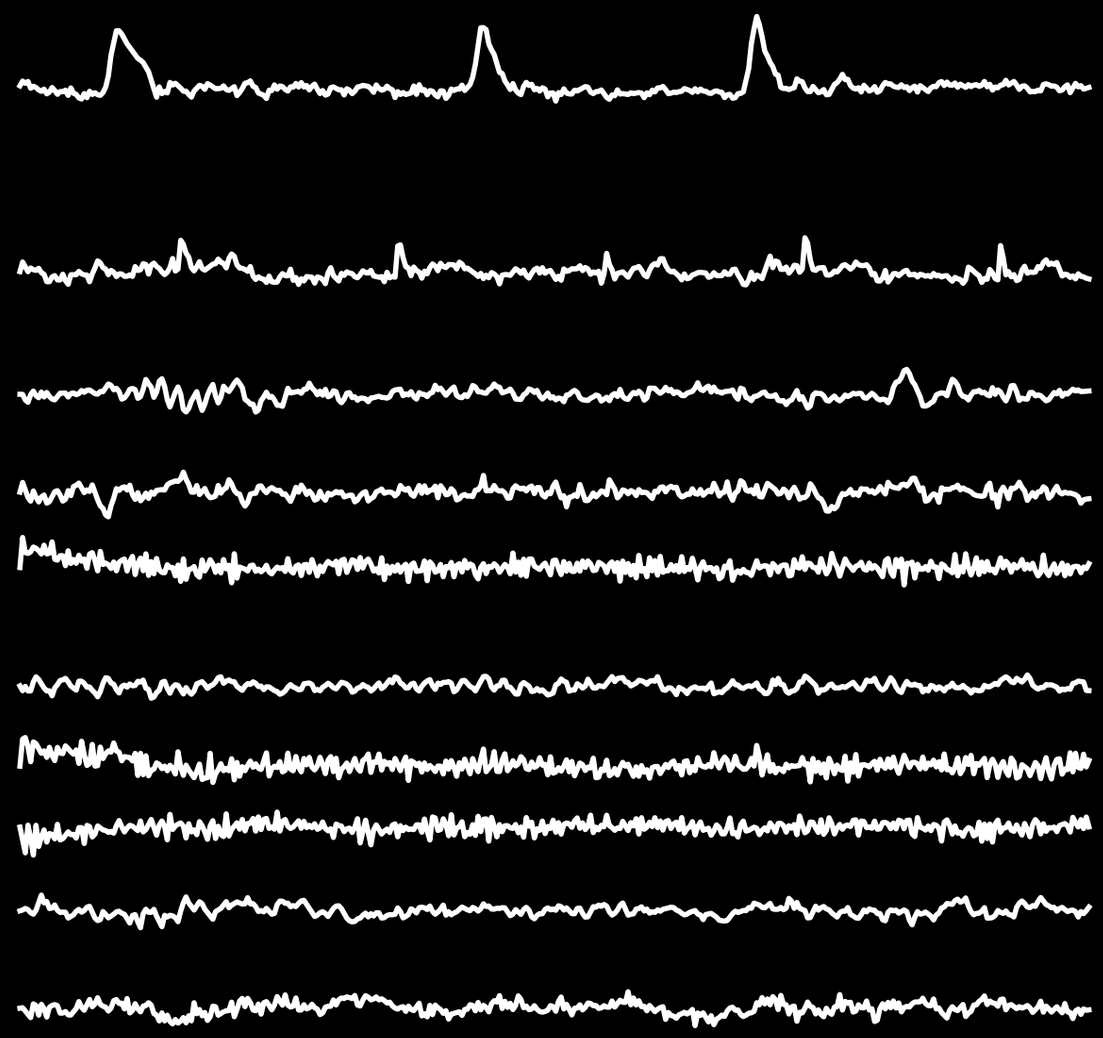
ICA recovered sources



Observations (raw EEG)



ICA recovered sources



Independent Component Analysis (ICA) is a leading approach in neuroscience

Comparison of Picard-O and FastICA on faces data

This example compares FastICA and Picard-O:

Pierre Ablin, Jean-François Cardoso, Alexandre Gramfort "Faster ICA under orthogonal constraint" ICASSP, 2018 https://arxiv.org/abs/1711.10873

On the figure, the number above each bar corresponds to the final gradient norm.

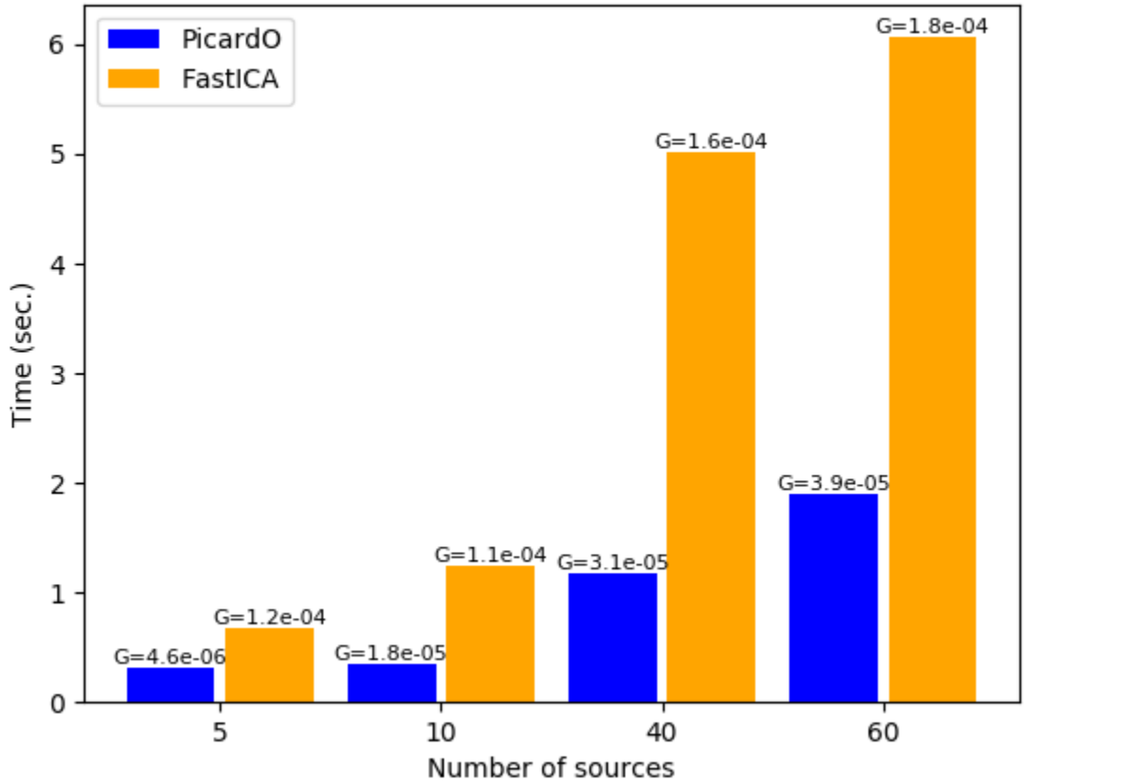
```
# Author: Pierre Ablin <pierre.ablin@inria.fr>
#         Alexandre Gramfort <alexandre.gramfort@inri
# License: BSD 3 clause


import numpy as np
from time import time
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_olivetti_faces
from sklearn.decomposition import fastica

from picard import picard


print(__doc__)

image_shape = (64, 64)
rng = np.random.RandomState(0)
```



https://pierreablin.github.io/picard/

[Ablin, Cardoso, Gramfort, (2018). Faster independent component analysis by preconditioning with Hessian approximations. IEEE Trans. on Signal Processing]

Welcome to sklearn-template's documentation!

This project is a reference implementation to anyone who wishes to develop scikit-learn compatible classes.

Getting started

Information regarding this template and how to modify it for your own project.

User Guide

An example of narrative documentation.

API Documentation

An example of API documentation.

Examples

A set of examples. It complements the User Guide.

https://github.com/scikit-learn-contrib/project-template

https://sklearn-template.readthedocs.io/en/latest/

Welcome to sklearn-template's documentation!

This project is a reference implementation to anyone who wishes to develop scikit-learn compatible classes.

Getting started

Information regarding this template and how to modify it for your own project.

User Guide

An example of narrative documentation.

API Documentation

An example of API documentation.

Examples

A set of examples. It complements the User Guide.

Make your own package but copying this template project

https://github.com/scikit-learn-contrib/project-template

Thanks !

Contact

http://alexandre.gramfort.net

GitHub : @agramfort *Twitter : @agramfort*

Support

Center for Data Science / IDEX Paris-Saclay
ANR-NSF Grant Thalameeg
European Research Council (ERC SLAB-YStG-676943)
Industrial contracts: Wendelin-IA & MEDATALAB

"An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem. ~ John Tukey"