



AI ON INTEL

SCIKIT-LEARN WITH INTEL

Georg Zitzlsberger

Bayncore

MACHINE LEARNING - LAYERS

Caffe
Caffe2

TensorFlow

scikit
learn

Models & Networks



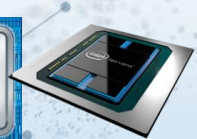
Frameworks

+Intel® DAAL

+Intel® MKL



Hardware



SCIKIT LEARN



- Via Intel® Distribution for Python*
 - Available via Anaconda*, Docker*, Linux* packages (RPM/APT) or stand-alone installation
 - **Scikit Learn** using Intel® DAAL
 - **NumPy** and **SciPy** using Intel® MKL



- For:
 - Classification
 - Regression
 - Clustering
 - Dimensionality reduction
 - Model selection
 - Preprocessing

						for SciPy
scikit-learn	0.18.2	✓		linux-64, win-64, osx-64	zlib, sqlite, tcl, tk, openssl, xz, mkl, openmp, icc_rt, numpy, scipy, tbb, daal, pydaal	A set of python modules for machine learning and data mining
scipy	0.19.1	✓	✓	linux-64, win-64	zlib, sqlite, tcl, tk, openssl, xz, mkl	SciPy: Scientific Library for Python

Replaced by daal4py

INTEL[®] DATA ANALYTICS ACCELERATION LIBRARY (INTEL[®] DAAL)



SCIKIT LEARN – INTEL® DAAL



- Directly integrated into Scikit Learn
- Currently implements (2019)
 - PCA (full SVD)
`sklearn.decomposition.PCA`
 - K-Means
`sklearn.cluster.KMeans`
 - Linear & ridge regression (not Kernel ridge regression)
`sklearn.linear_model.LinearRegression` &
`sklearn.linear_model.Ridge`
 - Pairwise distances (metrics: cosine & correlation)
`sklearn.metrics.pairwise.pairwise_distances`

SCIKIT LEARN – INTEL® DAAL



- Automatically turned on for Intel version of Scikit Learn (e.g conda module **scikit-learn**)
- Find out what is currently covered by Intel DAAL:

```
import daal4py.sklearn.monkeypatch.dispatcher as daaldisp
for k,v in daaldisp._mapping.items():
    print(k)
```
- Work in progress – not all configurations are supported yet, e.g.:
DAAL < 2019.4 PCA only optimized fit, using DAAL's SVD
(`svd_solver != 'full'`)
- Automatic fallback to Scikit Learn algorithm if not covered by Intel DAAL

SCIKIT LEARN - INTEL® DAAL



- **Enable DAAL:**

```
import daal4py.sklearn  
daal4py.sklearn.patch_sklearn()
```
- **Disable DAAL:**

```
daal4py.sklearn.unpatch_sklearn()
```

SCIKIT LEARN - INTEL® DAAL



Find implementation here:

```
.../site-packages/daal4py/sklearn $ ls
```

cluster

decomposition

ensemble

__init__.py

linear_model

monkeypatch/dispatcher.py (start here)

neighbors

__pycache__

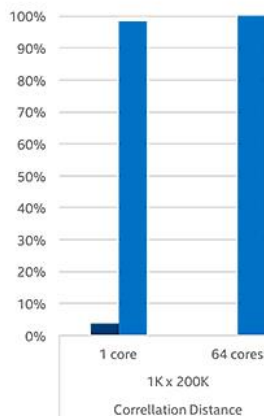
svm

utils.py

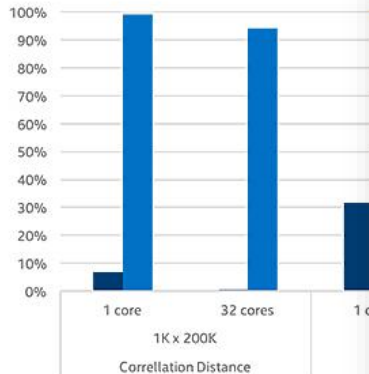
INTEL® DAAL PERFORMANCE WITH SCIKIT-LEARN



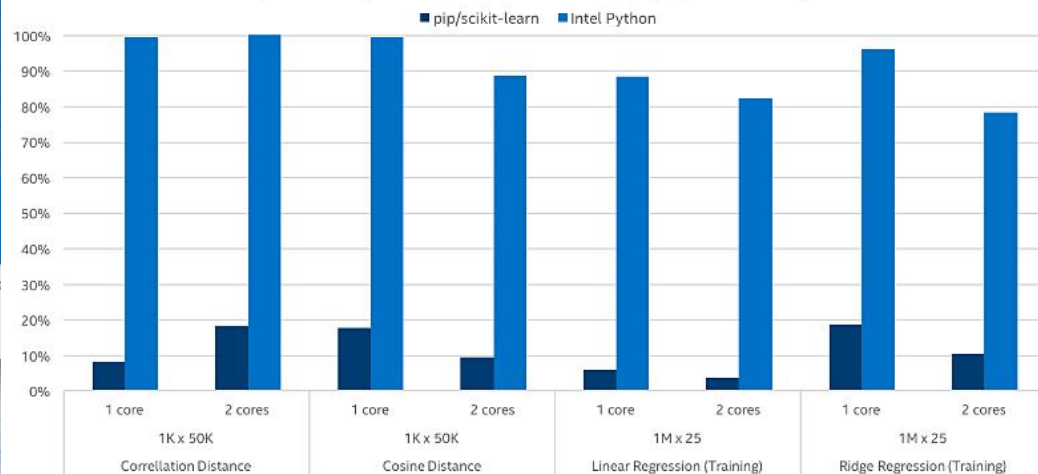
Python* Performance as a Percentage of C++ Intel® Data Analytics Acceleration Library
(Intel® DAAL) for Intel® Xeon Phi™ Product Family (Higher is Better)



Python* Performance as a Percentage of C++ Intel® Data Analytics Acceleration Library
(Intel® DAAL) on Intel® Xeon® Processors (Higher is Better)



Python* Performance as a Percentage of C++ Intel® Data Analytics Acceleration Library
(Intel® DAAL) on Intel® Core™ i5 Processors (Higher is Better)

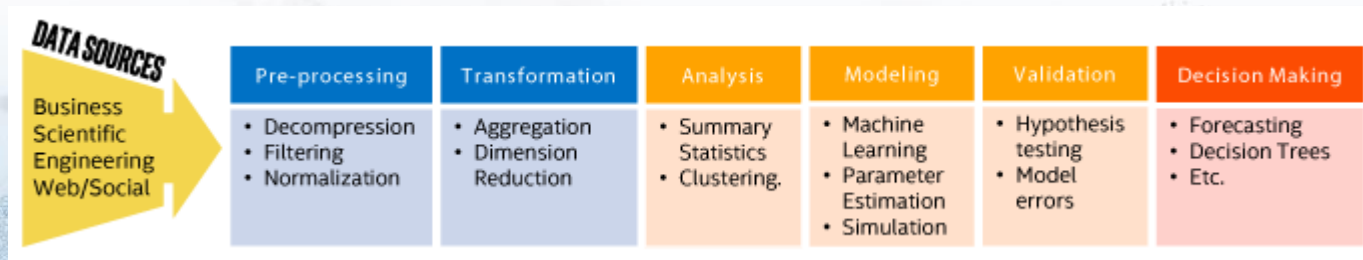


<https://software.intel.com/en-us/distribution-for-python/features>

DAAL4PY – THE PYTHONIC DAAL



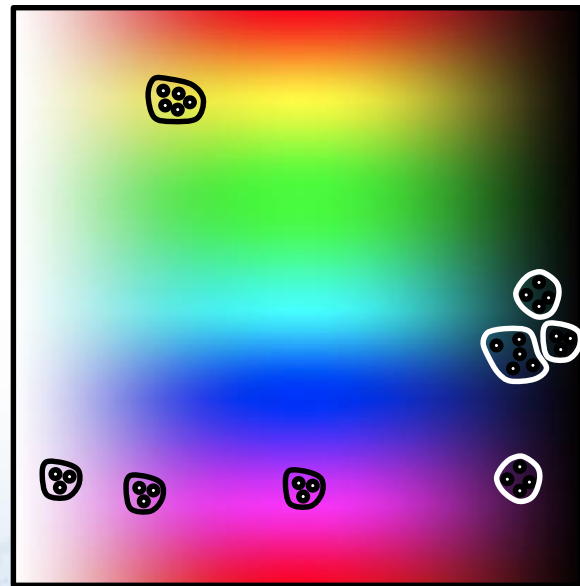
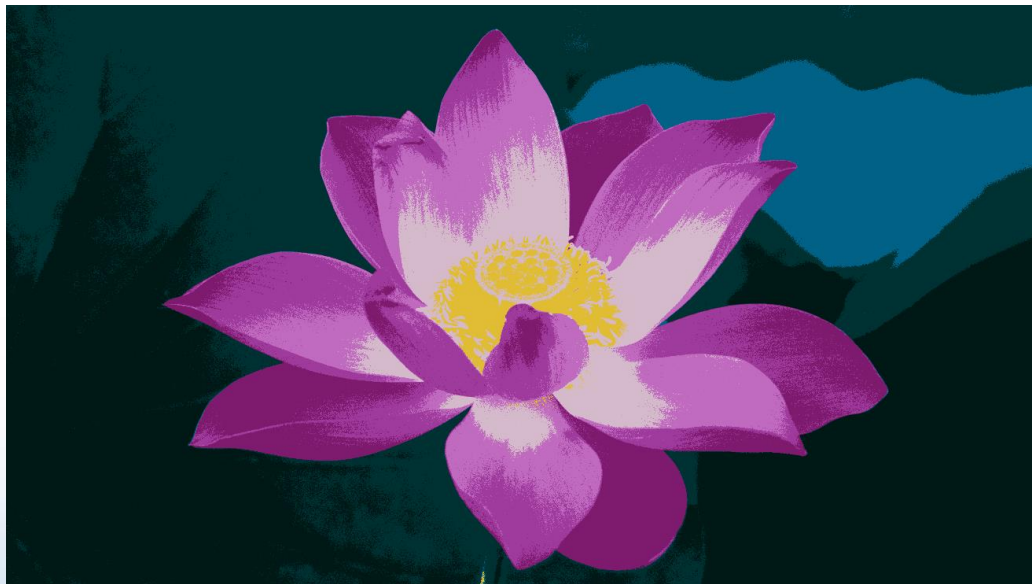
- Higher Abstraction layer
- Use Intel® DAAL
- Documentation: <https://intelpython.github.io/daal4py/>
- For:
 - PCA
 - SVM
 - Naive Bayes
 - SVD
 - KMEANS
 - Linear Regression
 - Multivariate/Univariate Outlier Detection



The background is a solid blue color with a complex, abstract pattern of white dots and thin white lines. These lines and dots form a network that resembles a molecular structure or a data visualization. A single, thin white horizontal line runs across the center of the image, passing behind the text.

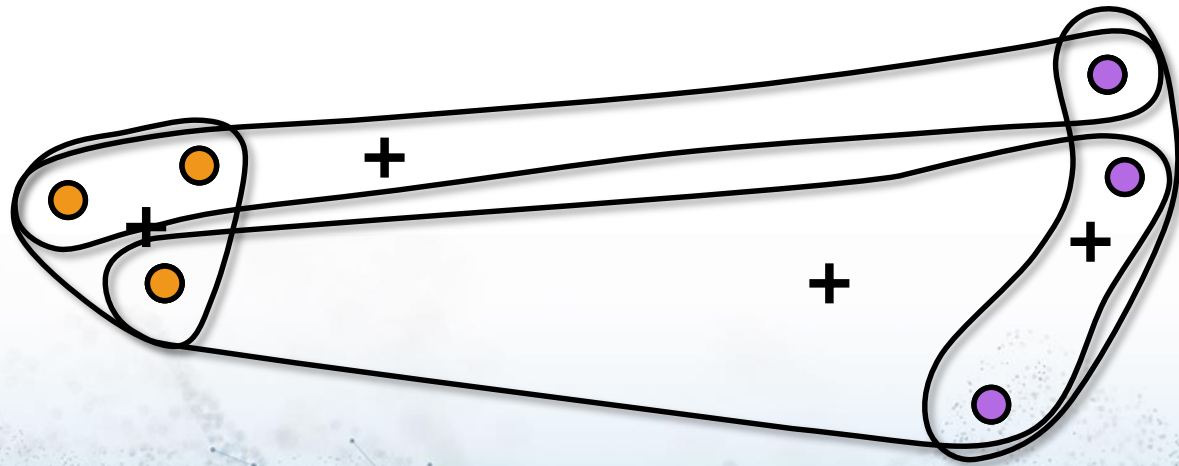
DEMO – K-MEANS (COLOR QUANTIZATION)

EXCURSION: COLOR QUANTIZATION



14 745 600 points (pixels) in dimension 3(RGB)
8 clusters (final colors)

EXCURSION: K-MEANS - AN ITERATIVE ALGORITHM

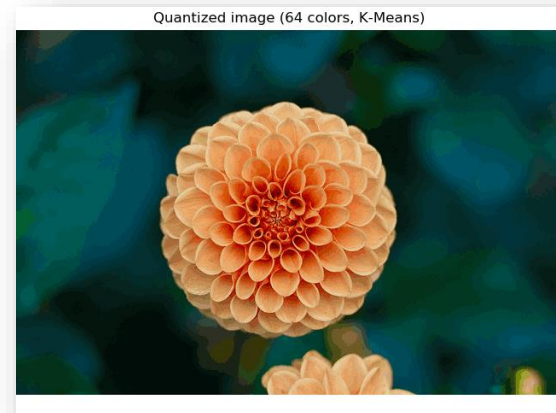


COLOR QUANTIZATION WITH K-MEANS

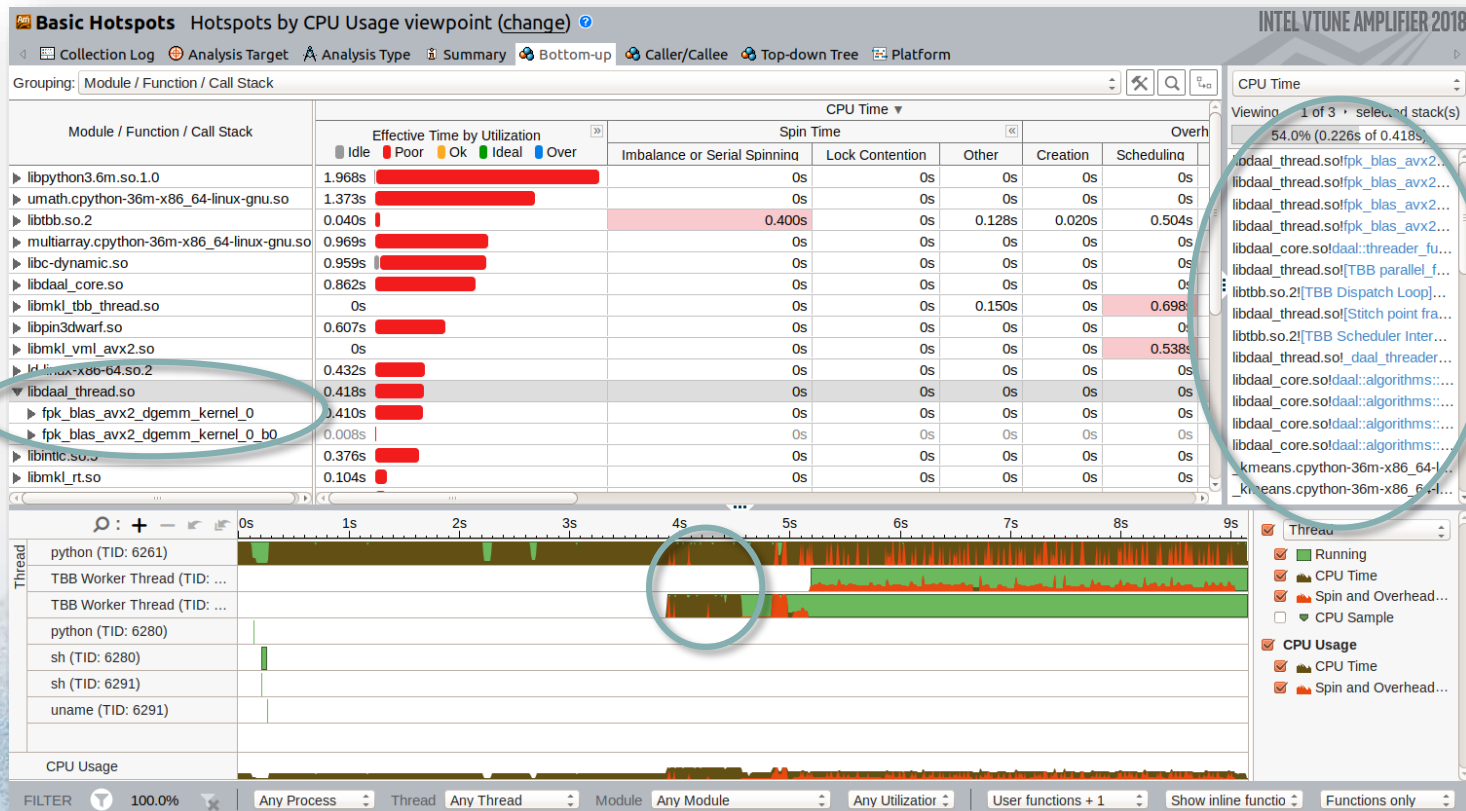
- Group colors into clusters (`n_clusters`)
- RGB yields 3D feature vectors
- Select a smaller but representative sample (for fitting):
ca. 1000 random pixel
- Centroids define color palette

```
print("Fitting model on a small sub-sample of the data")
t0 = time()
image_array_sample = shuffle(image_array, random_state=0)[:1000]
kmeans = KMeans(n_clusters=n_colors, random_state=0).fit(image_array_sample)
print("done in %.3fs." % (time() - t0))
```

- Example by courtesy of:
http://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html



INTEL® DAAL'S CONTRIBUTION





INTEL[®] MATH KERNEL LIBRARY (INTEL[®] MKL)

SCIKIT LEARN – INTEL[®] MATH KERNEL LIBRARY (INTEL[®] MKL)



- Not directly integrated into Scikit Learn but
 - **NumPy** (BLAS level1-3, LAPACK, FFT, random number generators)
 - **SciPy** (BLAS level 1-3, LAPACK)
- Intel MKL used indirectly by Scikit Learn
→ Use it directly
- Intel MKL directly used by NumPy & SciPy
→ Combine Scikit Learn with using NumPy & SciPy

SCIKIT LEARN - INTEL® MKL FOR NUMPY



```
>>> import numpy
>>> numpy.show_config()
blas_mkl_info:
  libraries = ['mkl_rt', 'pthread']
  library_dirs = ['../envs/intel/lib']
  define_macros = [('SCIPY_MKL_H',
None), ('HAVE_CBLAS', None)]
  include_dirs =
['../envs/intel/include']
blas_opt_info:
  libraries = ['mkl_rt', 'pthread']
  library_dirs = ['../envs/intel/lib']
  define_macros = [('SCIPY_MKL_H',
None), ('HAVE_CBLAS', None)]
  include_dirs =
['../envs/intel/include']
```

```
mkl_info:
  libraries = ['mkl_rt', 'pthread']
  library_dirs = ['../envs/intel/lib']
  define_macros = [('SCIPY_MKL_H', None),
('HAVE_CBLAS', None)]
  include_dirs = ['../envs/intel/include']
lapack_mkl_info:
  libraries = ['mkl_rt', 'pthread']
  library_dirs = ['../envs/intel/lib']
  define_macros = [('SCIPY_MKL_H', None),
('HAVE_CBLAS', None)]
  include_dirs = ['../envs/intel/include']
lapack_opt_info:
  libraries = ['mkl_rt', 'pthread']
  library_dirs = ['../envs/intel/lib']
  define_macros = [('SCIPY_MKL_H', None),
('HAVE_CBLAS', None)]
  include_dirs = ['../envs/intel/include']
```


SCIKIT LEARN - INTEL® MKL FOR SCIPY



```
>>> import scipy
>>> scipy.show_config()
lapack_mkl_info:
  libraries = ['mkl_rt', 'pthread']
  library_dirs = ['.../envs/intel/lib']
  define_macros = [('SCIPY_MKL_H',
None), ('HAVE_CBLAS', None)]
  include_dirs =
['.../envs/intel/include']
lapack_opt_info:
  libraries = ['mkl_rt', 'pthread']
  library_dirs = ['.../envs/intel/lib']
  define_macros = [('SCIPY_MKL_H',
None), ('HAVE_CBLAS', None)]
  include_dirs =
['.../envs/intel/include']
```

```
blas_mkl_info:
  libraries = ['mkl_rt', 'pthread']
  library_dirs = ['.../envs/intel/lib']
  define_macros = [('SCIPY_MKL_H',
None), ('HAVE_CBLAS', None)]
  include_dirs =
['.../envs/intel/include']
blas_opt_info:
  libraries = ['mkl_rt', 'pthread']
  library_dirs = ['.../envs/intel/lib']
  define_macros = [('SCIPY_MKL_H',
None), ('HAVE_CBLAS', None)]
  include_dirs =
['.../envs/intel/include']
```

SCIKIT LEARN - INTEL[®] MATH KERNEL LIBRARY (INTEL[®] MKL)



Control the number of threads:

- Environment variable (static):
\$MKL_NUM_THREADS=2

- Dynamically in Python script:

```
import ctypes
mkl_rt = ctypes.CDLL('libmkl_rt.so')
mkl_rt.MKL_Set_Num_Threads(2) # Set the amount
print("# threads: %s\n" % mkl_rt.MKL_Get_Max_Threads())
```

SCIKIT LEARN - INTEL® MATH KERNEL LIBRARY (INTEL® MKL)



- More control over the threads:
 - Set/get number of threads
 - Set by MKL domain (FFT, BLAS, VML, ...)
 - Allow dynamic change of threads
 - Set/get number of stripes (only ?GEMM)
- Allows changes during runtime
- Threading default is OpenMP*
- Intel Threading Building Blocks* (Intel TBB) also possible using `-m tbb`

The screenshot shows the Intel Developer Zone page for "Threading Control" in the Intel MKL Developer Reference. The page has a blue header with the Intel logo, "Developer Zone", and navigation links for "Development", "Tools", and "Resources". A search bar is present in the top right. The main content area is titled "Threading Control" and includes an "IMPORTANT" section. The "IMPORTANT" section states that Intel MKL operates within the Intel® Threading Building Blocks (Intel® TBB) execution environment and that environment variables for OpenMP* threading control, such as `OMP_NUM_THREADS`, and Intel MKL functions discussed in this section have no effect. It also mentions that the `tbb::task_scheduler_init` class at <https://www.threadingbuildingblocks.org/documentation> can be used to specify the number of Intel TBB threads. Below this, there is a paragraph explaining that if Intel® MKL operates within an OpenMP* execution environment, you can control the number of threads for Intel MKL using OpenMP* run-time library routines and environment variables. It also mentions that Intel MKL provides optional threading control functions and environment variables that enable you to specify the number of threads for Intel MKL and to control dynamic adjustment of the number of threads independently of the OpenMP* settings. The settings made with the Intel MKL threading control functions and environment variables do not affect OpenMP* settings but take precedence over them. Another paragraph states that if none of the threading control functions is used, Intel MKL environment variables may control Intel MKL threading. For details of those environment variables, see the Intel MKL Developer Guide. A table titled "Intel MKL Function Domains" is also shown, listing function domains and their corresponding named constants.

Intel MKL Function Domains	
Function Domain	Named Constant
Basic Linear Algebra Subroutines (BLAS)	MKL_DOMAIN_BLAS

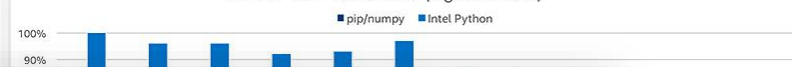
<https://software.intel.com/en-us/mkl-developer-reference-c-threading-control>

SCIKIT LEARN - INTEL® MKL PERFORMANCE

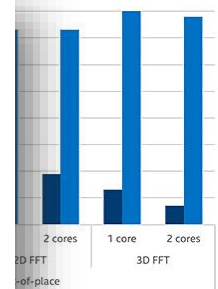
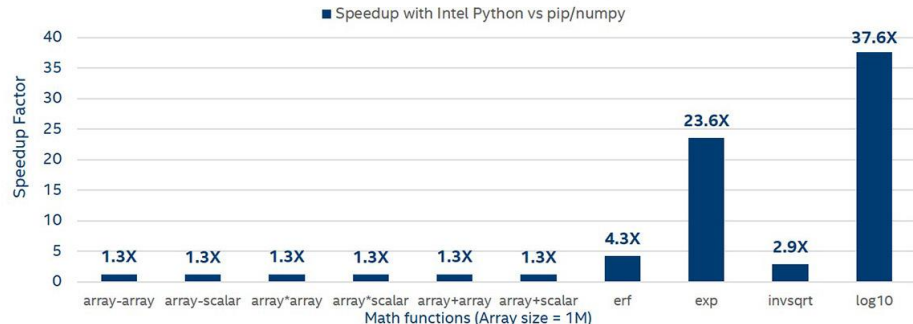


Distribution	Seconds (numpy.random)	Seconds (numpy.random_intel)
uniform (-1, 1)	0.357	0.034
normal (0, 1)	0.834	0.081
gamma (5.2, 1)	1.399	0.267
beta (0.7, 2.5)	3.677	0.556
randint (0, 100)	0.228	0.053
poisson (7.6)	2.990	0.052
hypergeometric (214, 97, 83)	11.353	0.517

Python* FFT Performance as a Percentage of C/Intel® Math Kernel Library (Intel® MKL) for Intel® Core™ i5 Processor (Higher is Better)



Intel® Distribution for Python* Performance Speedups for Select Math Functions on Intel® Core™ i5 Processors



Configuration: Intel® Core™ i7-7567U CPU @ 3.50GHz (1 socket, 2 cores per socket, 2 threads per core), 32GB DDR4 @ 2133MHz
Software: Stock CentOS Linux release 7.3.1611 (Core), python 3.6.2, pip 9.0.1, numpy 1.13.1, scipy 0.19.1, scikit-learn 0.18.0, Intel® Distribution for Python* 2018 Gold: mkl 2018.0.0 intel_4, daal 2018.0.0.20170814, numpy 1.13.1 py36, intel_15, openmp 2018.0.0 intel_7, scipy 0.19.1 np113py36, intel_11, scikit-learn 0.18.2 np113py36, intel_3

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to <https://www.intel.com/performance>. Benchmark Source: Intel Corporation.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE4.3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804. *Other brands and names are the property of their respective owners.

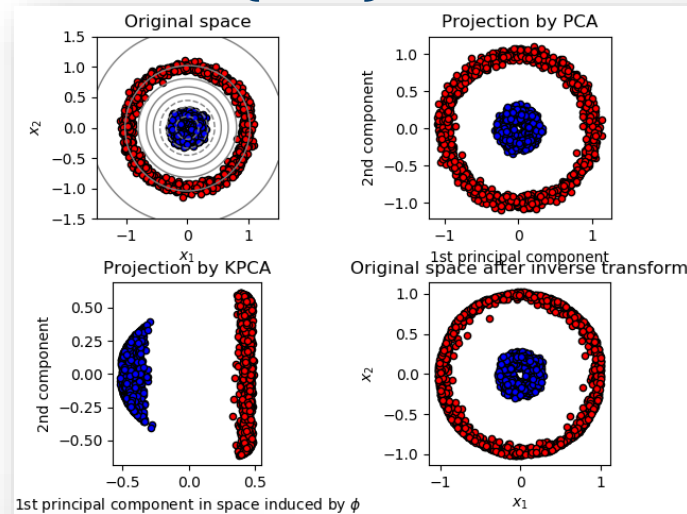
<https://software.intel.com/en-us/distribution-for-python/features>



DEMO – KERNEL PRINCIPAL COMPONENT ANALYSIS (KERNEL PCA)

KERNEL PRINCIPAL COMPONENT ANALYSIS (KPCA)

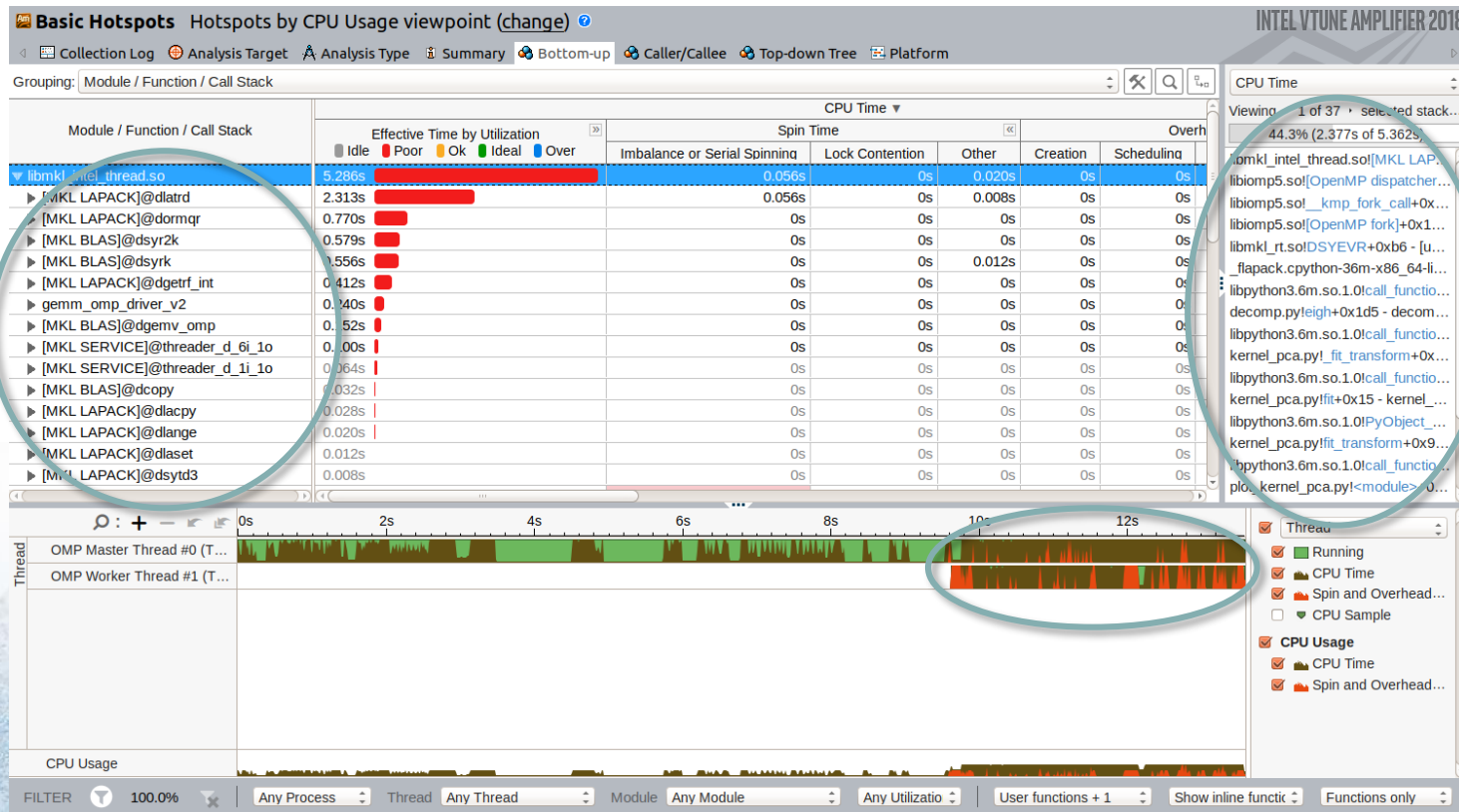
- Kernel used: Radial Basis Function (RBF)
- Project data points to kernel space (non-linear to linear transformation)
- Kernel space allows linear separation (e.g. via linear classification, linear SVM, etc.)



```
kpca = KernelPCA(kernel="rbf", fit_inverse_transform=True, gamma=10)
X_kpca = kpca.fit_transform(X)
X_back = kpca.inverse_transform(X_kpca)
pca = PCA()
X_pca = pca.fit_transform(X)
```

- Example by courtesy of:
http://scikit-learn.org/stable/auto_examples/decomposition/plot_kernel_pca.html

INTEL® MKL'S CONTRIBUTION



The background is a solid blue color with a complex, abstract pattern of white lines and dots. These lines form a network of interconnected paths, resembling particle tracks or a molecular structure. The pattern is denser in some areas and sparser in others, creating a sense of depth and movement. A thin, horizontal white band runs across the center of the image, providing a clear space for the text.

SUMMARY

SCIKIT LEARN WITH INTEL PERFORMANCE LIBRARIES

Guidelines for performance:

- Always use the **latest** Intel® Distribution for Python* (e.g. via Anaconda*)
 - Other sources **can** have Intel MKL enabled NumPy or SciPy, too
 - But **quality** of optimization varies (e.g. missing functions)
 - **Integration is in flux** - Intel engineers keep adding new extensions/improvements
- Characteristics of performance libraries (Intel MKL & Intel DAAL):
 - **Larger data set** needed, esp. large number of features and samples (not always visible with toy data sets)
 - Intel MKL heavily **used in NumPy/SciPy**, Intel DAAL can **add** additional performance

SCIKIT LEARN WITH INTEL PERFORMANCE LIBRARIES

Guidelines for performance – for **advanced users**:

- Enable/disable Intel DAAL:
 - Fallback might use Intel MKL **with different implementation**
 - Intel DAAL might have **optimizations for special cases**
- Evaluate multi-core scalability with using Intel MKL:
 - **Vary number of threads** to be used by Intel MKL
 - Consider using `-m TBB` for alternative threading model



LEGAL DISCLAIMER & OPTIMIZATION NOTICE

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © 2019, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804