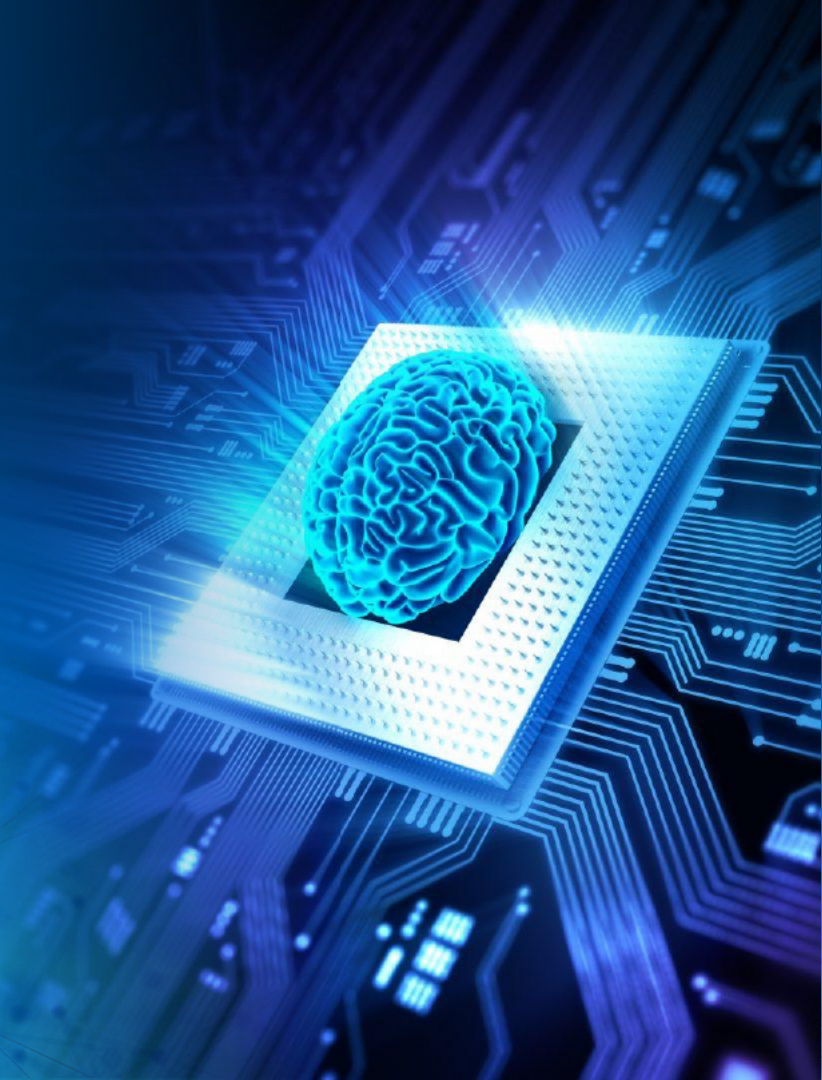
intel AI

INTEL® ARTIFICIAL INTELLIGENCE OPTIMIZED SOFTWARE

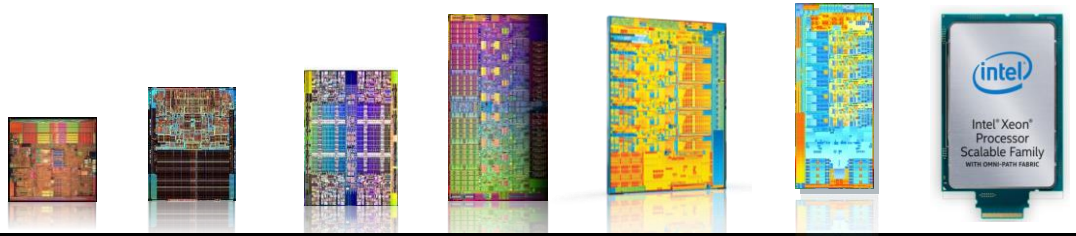
Dr. Fabio Baruffa & Shailen Sobhee

Technical Consulting Engineers, Intel IAGS



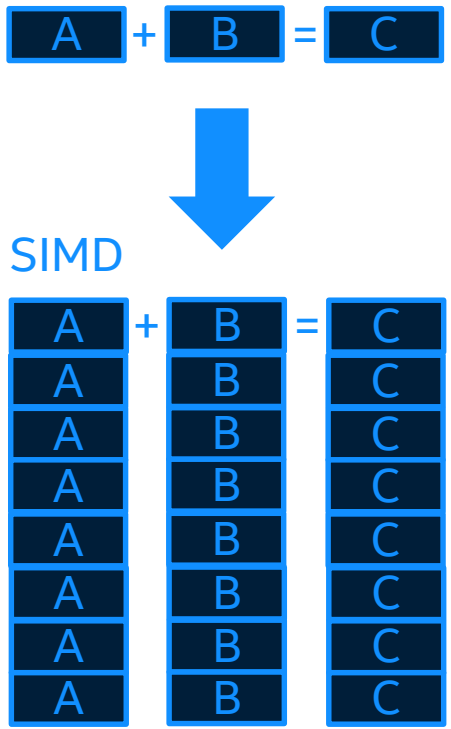
THE EVOLUTION OF MICROPROCESSOR PARALLELISM

More cores → More Threads → Wider vectors



	Intel® Xeon® Processor 64-bit	Intel® Xeon® Processor 5100 series	Intel® Xeon® Processor 5500 series	Intel® Xeon® Processor 5600 series	Intel® Xeon® Processor E5-2600 v2 series	Intel® Xeon® Processor E5-2600 v3 series v4 series	Intel® Xeon® Scalable Processor[1]
Up to Core(s)	1	2	4	6	12	18-22	28
Up to Threads	2	2	8	12	24	36-44	56
SIMD Width	128	128	128	128	256	256	512
Vector ISA	Intel® SSE3	Intel® SSE3	Intel® SSE4- 4.1	Intel® SSE 4.2	Intel® AVX	Intel® AVX2	Intel® AVX-512

1. Product specification for launched and shipped products available on ark.intel.com.



INTEL® OPTIMIZED SOFTWARE AND LIBRARIES

Optimization techniques for getting performance

Scaling

- Improve load balancing
- Reduce synchronization events, all-to-all comms

Utilize all the cores

- OpenMP, MPI
- Reduce synchronization events, serial code
- Improve load balancing

Vectorize/SIMD

- Unit strided access per SIMD lane
- High vector efficiency
- Data alignment

Efficient memory/cache use

- Blocking
- Data reuse
- Prefetching
- Memory allocation

Visit:
www.intel.ai/technology

SPEED UP DEVELOPMENT

using open AI software

MACHINE LEARNING ← → DEEP LEARNING

TOOLKITS
App developers

ANALYTICS ZOO
Open source platform for building E2E Analytics & AI applications on Apache Spark* with distributed TensorFlow*, Keras*, BigDL

OpenVINO™
Deep learning inference deployment on CPU/GPU/FPGA/VPU for Caffe*, TensorFlow*, MXNet*, ONNX*, Kaldi*

NAUTA
Open source, scalable, and extensible distributed deep learning platform built on Kubernetes (BETA)

LIBRARIES
Data scientists

Python
- Scikit-learn
- Pandas
- NumPy

R
- Cart
- Random Forest
- e1071

Distributed
- MlLib (on Spark)
- Mahout

Intel-optimized Frameworks
TensorFlow* · Caffe2* · ONNX* · mxnet* · PyTorch* · BigDL

And more framework optimizations underway including PaddlePaddle*, Chainer*, CNTK* & others

KERNELS
Library developers

Intel® Distribution for Python*
Intel distribution optimized for machine learning

Intel® Data Analytics Acceleration Library (Intel® DAAL)
High performance machine learning & data analytics library

Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN)
Open source DNN functions for CPU / integrated graphics

nGraph
Open source compiler for deep learning model computations optimized for multiple devices (CPU, GPU, NNP) from multiple frameworks (TF, MXNet, ONNX)

PRODUCTIVITY WITH PERFORMANCE VIA INTEL® PYTHON*

Intel® Distribution for Python*

Easy, out-of-the-box access to high performance Python

- Prebuilt accelerated solutions for data analytics, numerical computing, etc.
- Drop in replacement for your existing Python. No code changes required.

Learn More: software.intel.com/distribution-for-python

INSTALLING INTEL® DISTRIBUTION FOR PYTHON* 2019

Standalone Installer

Download full installer from
https://software.intel.com/en-us/intel-distribution-for-python

Anaconda.org
Anaconda.org/intel channel

```
> conda config --add channels intel
> conda install intelpython3_full
> conda install intelpython3_core
```

PyPI

```
> pip install intel-numpy
> pip install intel-scipy
> pip install mkl_fft
> pip install mkl_random
```

+ Intel library Runtime packages
+ Intel development packages

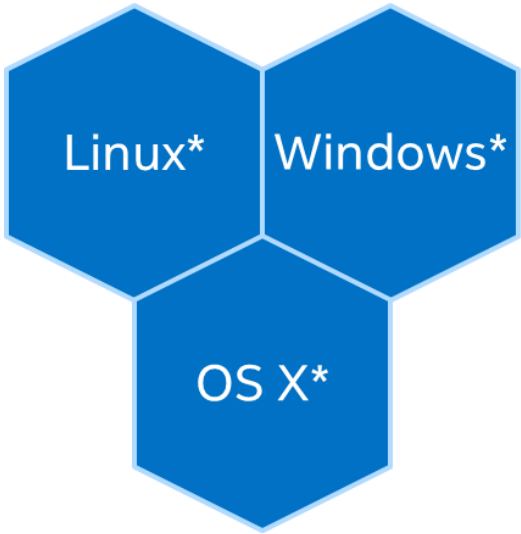
Docker Hub

```
docker pull intelpython/intelpython3_full
```

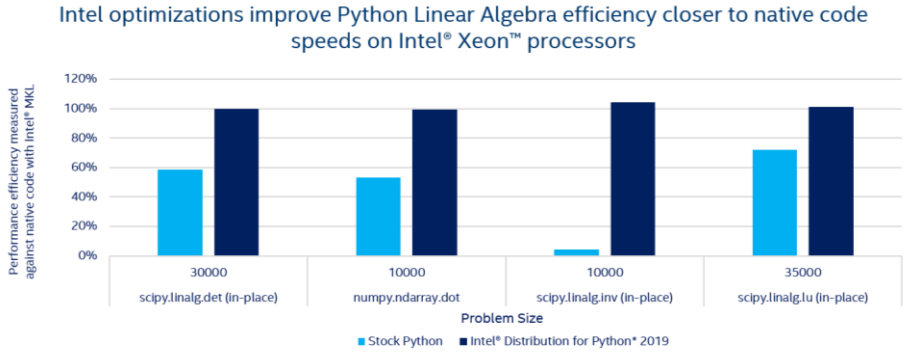
YUM/APT

Access for yum/apt:
https://software.intel.com/en-us/articles/installing-intel-free-libs-and-python

2.7 & 3.6
(3.7 coming soon)



FASTER PYTHON* WITH INTEL® DISTRIBUTION FOR PYTHON 2019



Performance results are based on testing as of **July 9, 2018** and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.
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 complete information, see Performance Benchmark Test Disclosure.

Testing by Intel as of July 9, 2018. Configuration: Stock Python: python 3.6.6 hc3d631a_0 installed from conda, numpy 1.15, numba 0.39.0, llvmlite 0.24.0, scipy 1.1.0, scikit-learn 0.19.2 installed from pip; Intel Python: Intel® Distribution for Python* 2019 Gold: python 3.6.5 intel_11, numpy 1.14.3 intel_py36_5, mkl 2019.0 intel_101, mkl_fft 1.0.2 intel_np114py36_6,mkl_random 1.0.1 intel_np114py36_6, numba 0.39.0 intel_np114py36_0, llvmlite 0.24.0 intel_py36_0, scipy 1.1.0 intel_np114py36_6, scikit-learn 0.19.1 intel_np114py36_35; OS: CentOS Linux 7.3.1611, kernel 3.10.0-514.el7.x86_64; Hardware: Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz (2 sockets, 18 cores/socket, HT: off), 256 GB of DDR4 RAM, 16 DIMMs of 16 GB@2666MHz

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.
For more complete information about compiler optimizations, see our Optimization Notice.

Linear Algebra functions in Intel Distribution for Python are faster than equivalent stock Python functions

High Performance Python Distribution

- Accelerated NumPy, SciPy, scikit-learn well suited for scientific computing, machine learning & data analytics
- Drop-in replacement for existing Python. No code changes required
- Highly optimized for latest Intel processors
- Take advantage of Priority Support – connect direct to Intel engineers for technical questions[2]

What's New in 2019 version

- Faster Machine learning with Scikit-learn: Support Vector Machine (SVM) and K-means prediction, accelerated with Intel® Data Analytics Acceleration Library
- Integrated into Intel® Parallel Studio XE 2019 installer. Also available as easy command line standalone install.
- Includes XGBoost package (Linux* only)

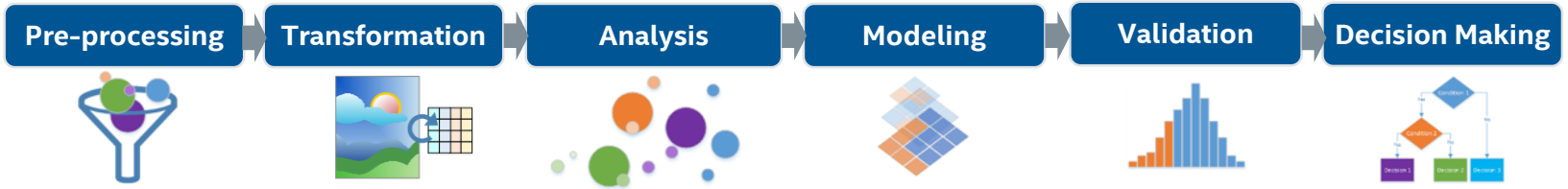
Software & workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark & MobileMark, are measured using specific computer systems, components, software, operations & functions. Any change to any of those factors may cause the results to vary. You should consult other information & 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 http://www.intel.com/performance.

[2]Paid versions only.

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.

INTEL® DATA ANALYTICS ACCELERATION LIBRARY (INTEL® DAAL)

Building blocks for all data analytics stages, including data preparation, data mining & machine learning



Common Python*, Java*, C++ APIs across all Intel hardware

Optimizes data ingestion & algorithmic compute together for highest performance

Supports offline, streaming & distributed usage models for a range of application needs

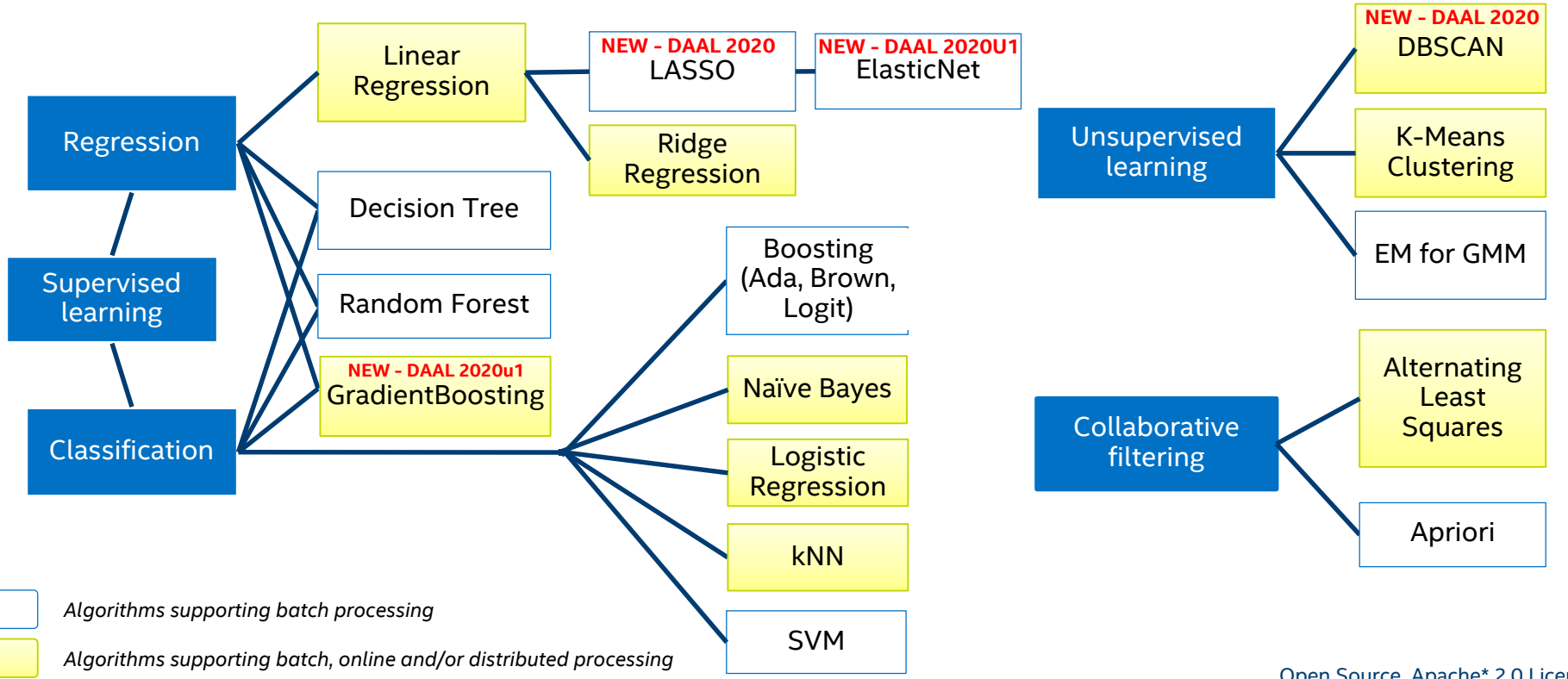
Flexible interfaces to leading big data platforms including Spark*

Split analytics workloads between edge devices & cloud to optimize overall application throughput

High Performance Machine Learning & Data Analytics Library

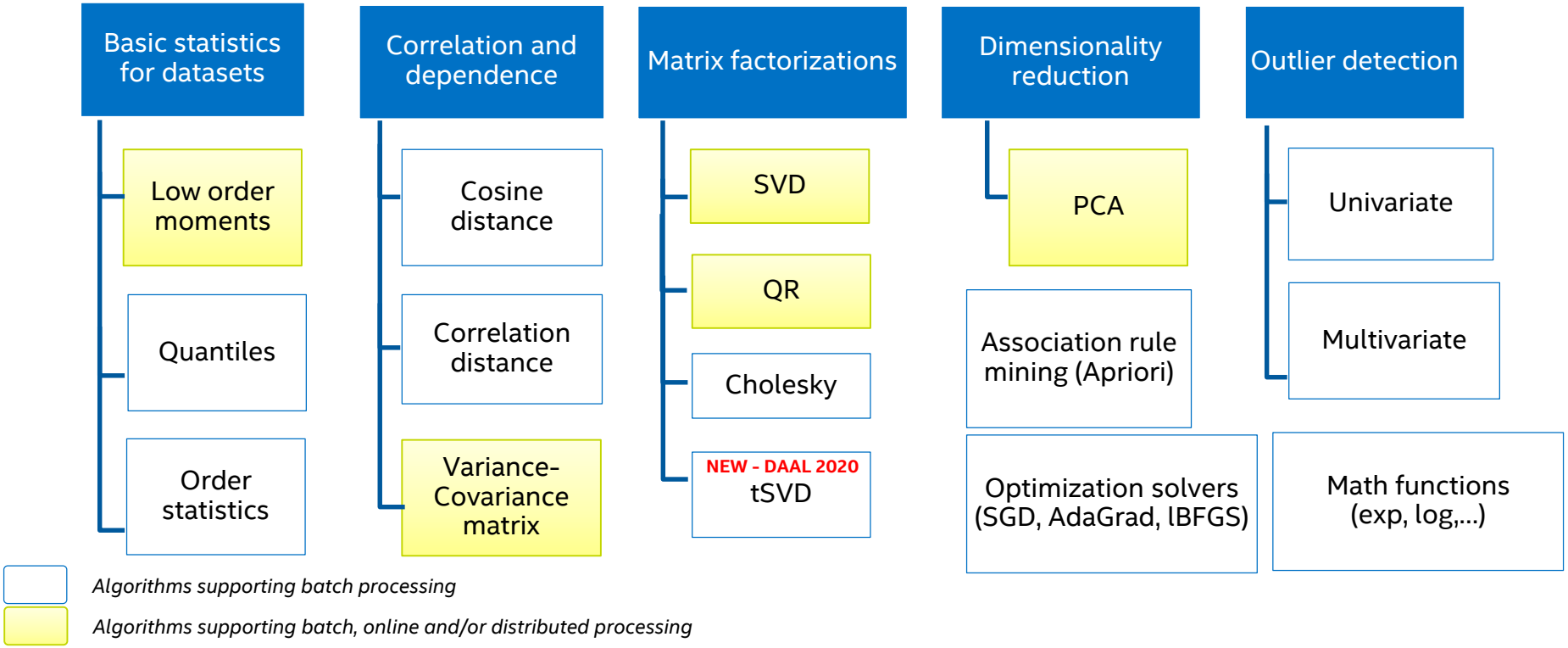
Open Source, Apache* 2.0 License

MACHINE LEARNING ALGORITHMS

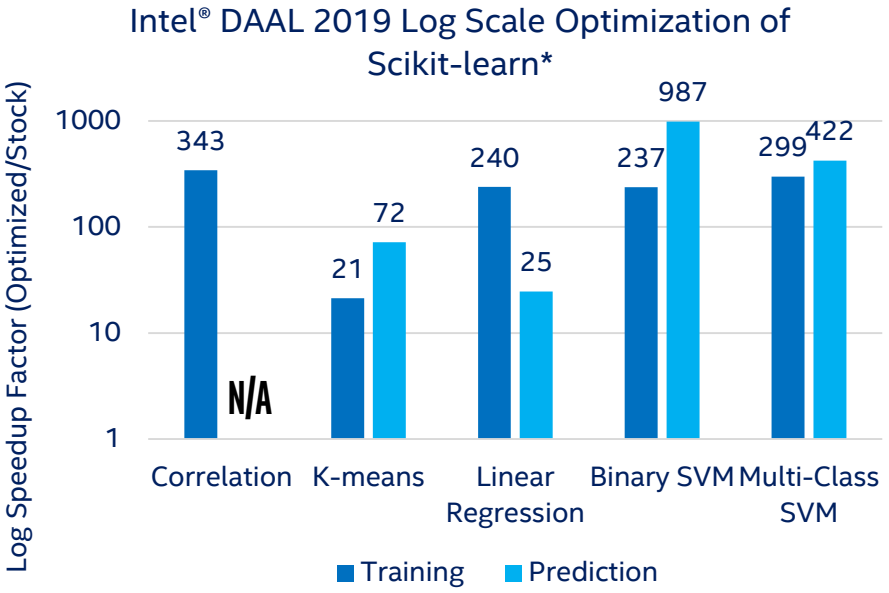


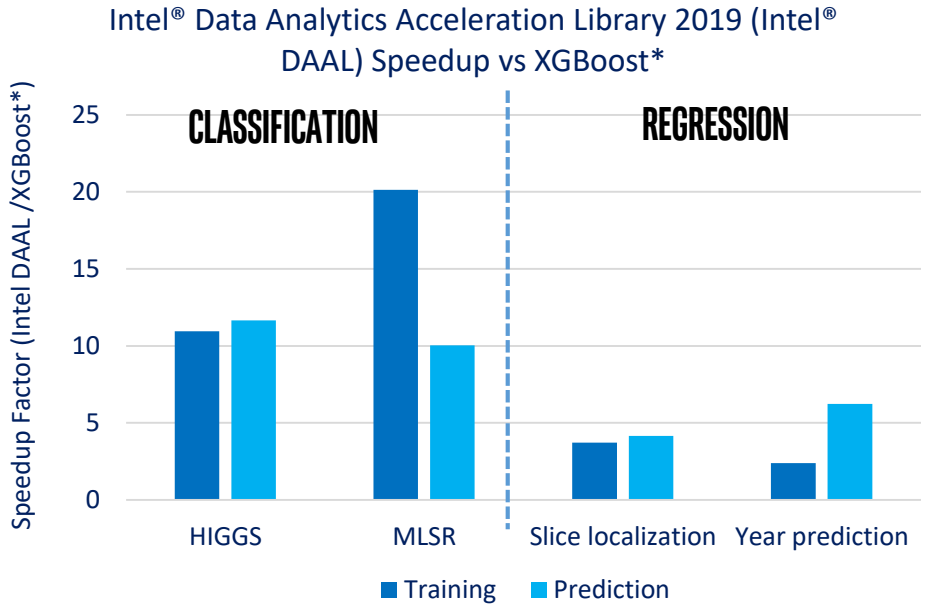
Open Source, Apache* 2.0 License

DATA TRANSFORMATION & ANALYSIS ALGORITHMS



INTEL® DAAL - PERFORMANCE





Performance results are based on testing as of **July 9, 2018** and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.
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 proc when combined with other products. For more complete information, see Performance Benchmark Test Disclosure.

Testing by Intel as of July 9, 2018. Configuration: Intel® Xeon® Platinum 8180 H0 205W, 2x28@2.50GHz, 192GB, 12x16gb DDR4-2666, Intel® Data Analytics Acceleration Library (Intel® DAAL 2019), RHEL 7.2

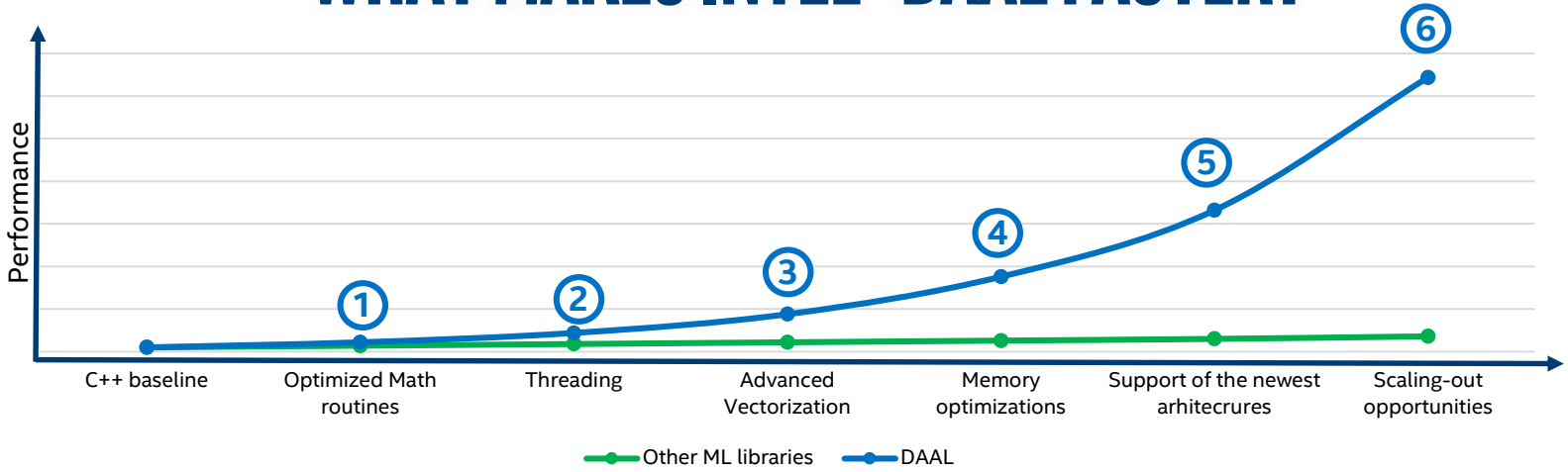
Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Inte not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizatior 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.
For more complete information about compiler optimizations, see our Optimization Notice.

Performance results are based on testing as of **July 9, 2018** and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.
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 complete information, see Performance Benchmark Test Disclosure.

Testing by Intel as of July 9, 2018. Configuration: Intel® Xeon® Gold 6140 CPU, 2x18@2.30GHz, 256GB, 16x16gb DDR4-2666, Intel® Data Analytics Acceleration Library (Intel® DAAL 2019), Optimized: Scikit-learn*_intel 0.19.1, Numpy*_intel 1.14.3 Stock: Scikit-learn* 0.19.2, Numpy* 1.15.0, CentOS Linux 7.3.1611

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.
For more complete information about compiler optimizations, see our Optimization Notice.

WHAT MAKES INTEL® DAAL FASTER?



1. The best performance on Intel Architectures with Intel® MKL vs. less performance OS BLAS/LAPACK libs
2. Intel® DAAL targets to many-core systems to achieve the best scalability on Intel® Xeon, other libs mostly target to client versions with small amount of cores
3. Intel® DAAL uses the latest available vector-instructions on each architecture, enables them by compiler options, intrinsics. Usually other ML libs build application without vector-instructions support or sse4.2 only.
4. Intel® DAAL's uses the most efficient memory optimization practices: minimally access memory, cache access optimizations, SW memory prefetching. Usually Other ML libs don't make low-level optimizations.
5. Intel® DAAL enables new instruction sets and other HW features even before official HW lunch. Usually other ML libs do this with long delay.
6. Intel® DAAL provides distributed algorithms which scale on many nodes

FASTER, SCALABLE CODE WITH INTEL® MATH KERNEL LIBRARY

LINEAR ALGEBRA

- BLAS
- LAPACK
- ScaLAPACK
- Sparse BLAS
- Iterative sparse solvers
- PARDISO*
- Cluster Sparse Solver

FFTS

- Multidimensional
- FFTW interfaces
- Cluster FFT

VECTOR RNGS

- Congruential
- Wichmann-Hill
- Mersenne Twister
- Sobol
- Neirderreiter
- Non-deterministic

SUMMARY STATISTICS

- Kurtosis
- Variation coefficient
- Order statistics
- Min/max
- Variance-covariance

VECTOR MATH

- Trigonometric
- Hyperbolic
- Exponential
- Log
- Power
- Root

AND MORE

- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver

INTEL® MATH KERNEL LIBRARY FOR DEEP NEURAL NETWORKS (INTEL® MKL-DNN)

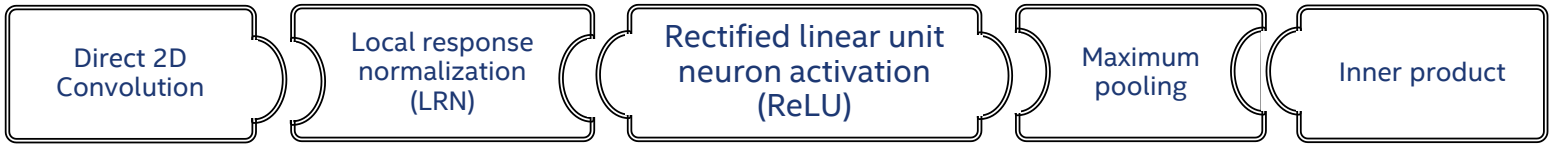
For developers of deep learning frameworks featuring optimized performance on Intel hardware

Distribution Details

- Open Source
- Apache* 2.0 License
- Common DNN APIs across all Intel hardware.
- Rapid release cycles, iterated with the DL community, to best support industry framework integration.
- Highly vectorized & threaded for maximal performance, based on the popular Intel® Math Kernel Library.

github.com/01org/mkl-dnn

Examples:



Accelerate Performance of Deep Learning Models

INTEL® AI OPTIMIZED FRAMEWORKS

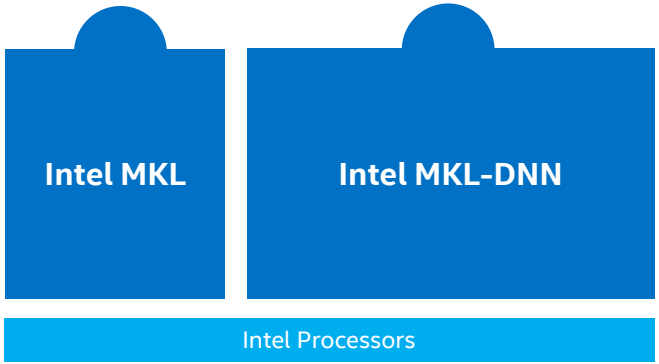
Popular DL Frameworks are now optimized for CPU!

See installation guides at ai.intel.com/framework-optimizations/

More under optimization: Caffe2* PYTORCH* PaddlePaddle*

SEE ALSO: Machine Learning Libraries for Python (Scikit-learn, Pandas, NumPy), R (Cart, randomForest, e1071), Distributed (MLlib on Spark, Mahout)
**Limited availability today*
Other names and brands may be claimed as the property of others.

AI (ML & DL) SOFTWARE STACK FOR INTEL® PROCESSORS



Deep learning and AI ecosystem includes edge and datacenter applications.

- Open source frameworks (Tensorflow*, MXNet*, PyTorch*, PaddlePaddle*)
- Intel deep learning products (, BigDL, OpenVINO™ toolkit)
- In-house user applications

Intel® MKL and Intel® MKL-DNN optimize deep learning and machine learning applications for Intel® processors :

- Through the collaboration with framework maintainers to upstream changes (Tensorflow*, MXNet*, PyTorch, PaddlePaddle*)
- Through Intel-optimized forks (Caffe*)
- By partnering to enable proprietary solutions

Intel® MKL-DNN is an open source performance library for deep learning applications (available at https://github.com/intel/mkl-dnn)

- Fast open source implementations for wide range of DNN functions
- Early access to new and experimental functionality
- Open for community contributions

Intel® MKL is a proprietary performance library for wide range of math and science applications
Distribution: Intel Registration Center, package repositories (apt, yum, conda, pip), Intel® Parallel Studio XE, Intel® System Studio

INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT

OpenVINO™

DEEP LEARNING

Caffe TensorFlow ONNX mxnet KALDI

- Model Optimizer
- Inference Engine
- Supports 100+ public models, incl. 30+ pretrained models

COMPUTER VISION

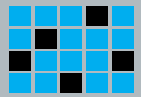
OpenCV OpenCL™ OpenVX™

- Computer vision library (kernel & graphic APIs)
- Optimized media encode/decode functions

SUPPORTS MAJOR AI FRAMEWORKS

Rapid adoption by developers

CROSS-PLATFORM FLEXIBILITY



Multiple products launched based on this toolkit

HIGH PERFORMANCE, HIGH EFFICIENCY

Breadth of product portfolio

Strong Adoption + Rapidly Expanding Capability

software.intel.com/openvino-toolkit

Obtain open source version at 01.org/openvinotoolkit

INTEL® DEEP LEARNING DEPLOYMENT TOOLKIT

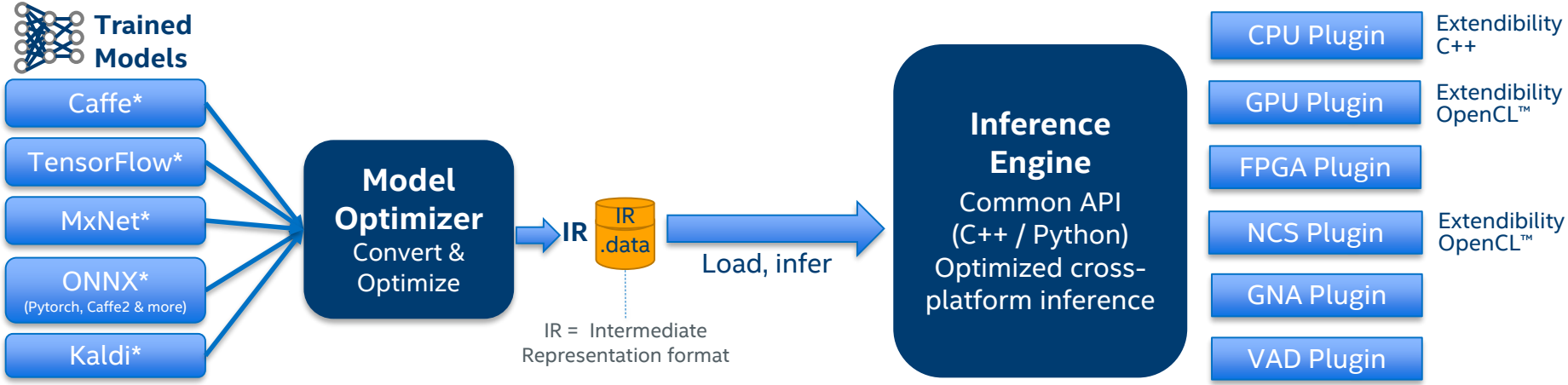
For Deep Learning Inference – Part of Intel® Distribution of OpenVINO toolkit

Model Optimizer

- **What it is**: A Python*-based tool to import trained models and convert them to Intermediate representation.
- **Why important**: Optimizes for performance/space with conservative topology transformations; biggest boost is from conversion to data types matching hardware.

Inference Engine

- **What it is**: High-level inference API
- **Why important**: Interface is implemented as dynamically loaded plugins for each hardware type. Delivers best performance for each type without requiring users to implement and maintain multiple code pathways.

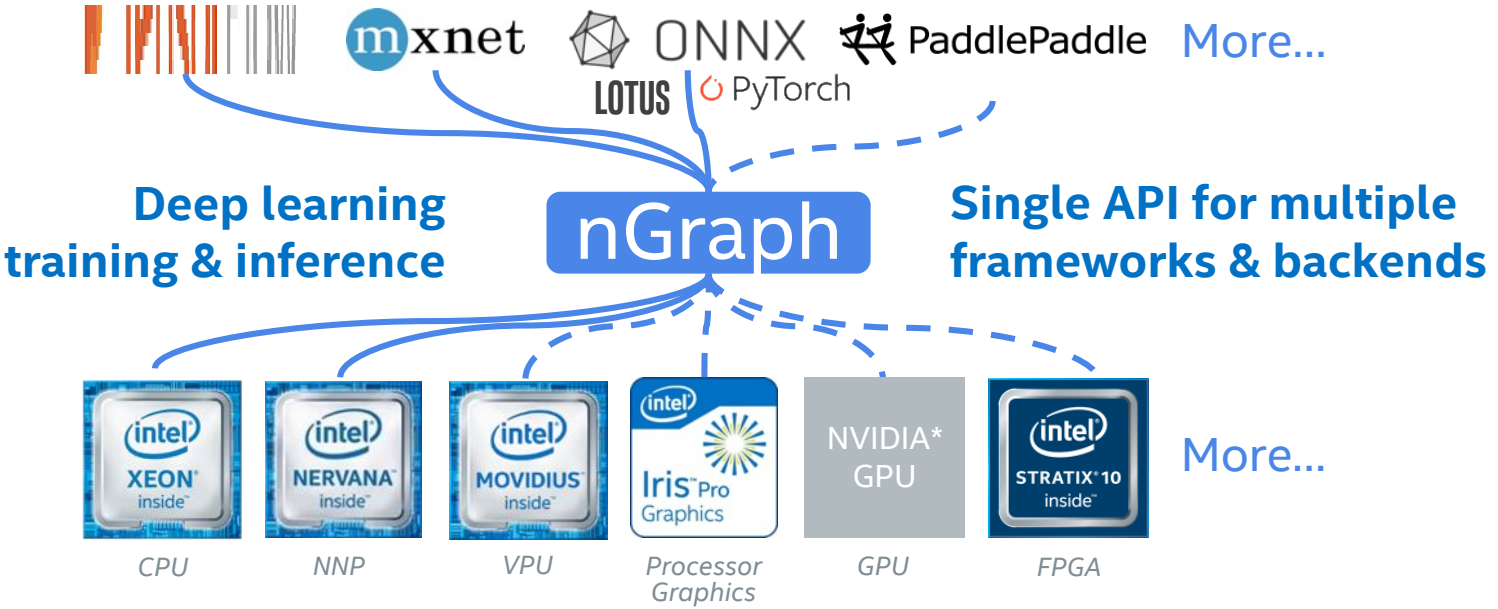


GPU = Intel CPU with integrated graphics processing unit/Intel® Processor Graphics

Work in progress

INTEL® NGRAPH™ COMPILER

NOW IN BETA!



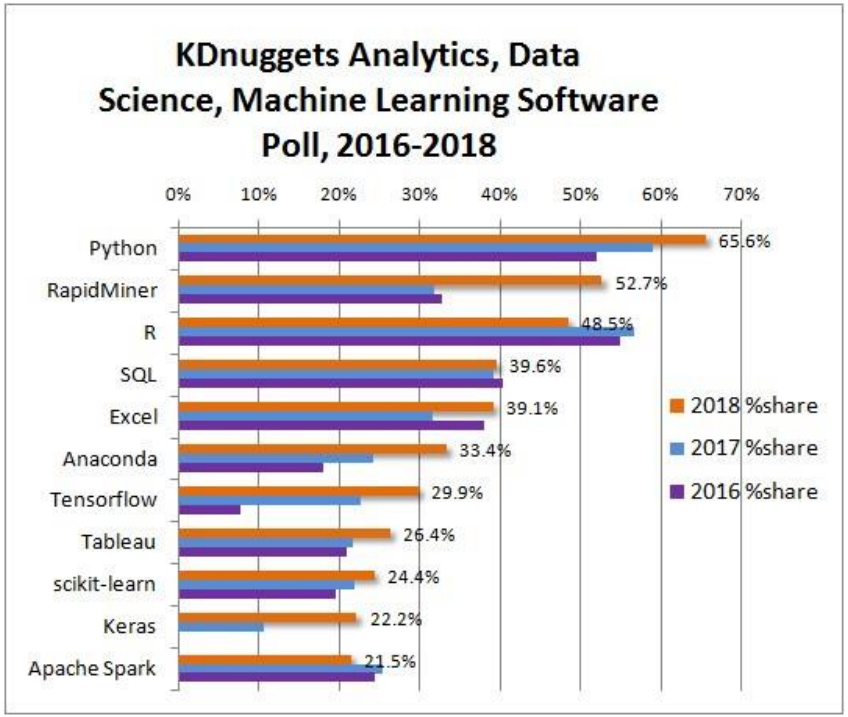
Open-source C++ library, compiler & runtime for deep learning enabling flexibility to run models across a variety of frameworks & hardware

*Other names and brands may be claimed as the property of others.
All products, computer systems, dates, and figures are preliminary based on current expectations, and are subject to change without notice.

github.com/NervanaSystems/ngraph

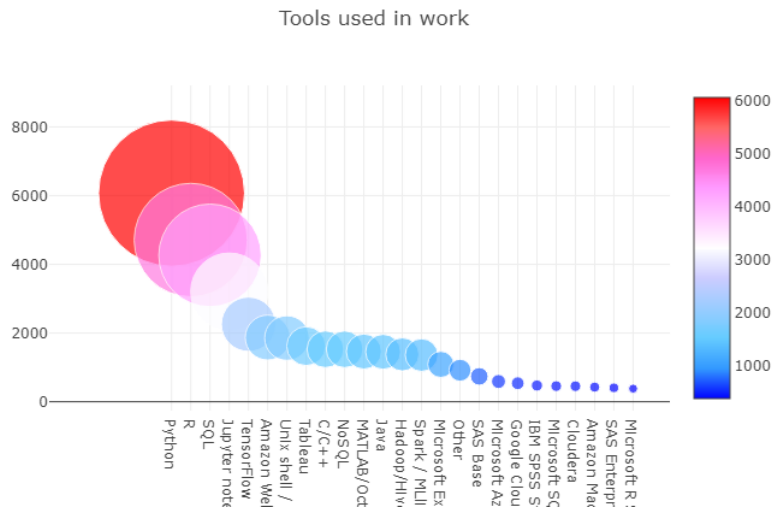
INTEL® DISTRIBUTION FOR PYTHON 2019

THE MOST POPULAR LANGUAGES FOR DATA SCIENCE



https://www.kdnuggets.com/2018/05/poll-tools-analytics-data-science-machine-learning-results.html

Kaggle ML and Data Science Survey, 2017

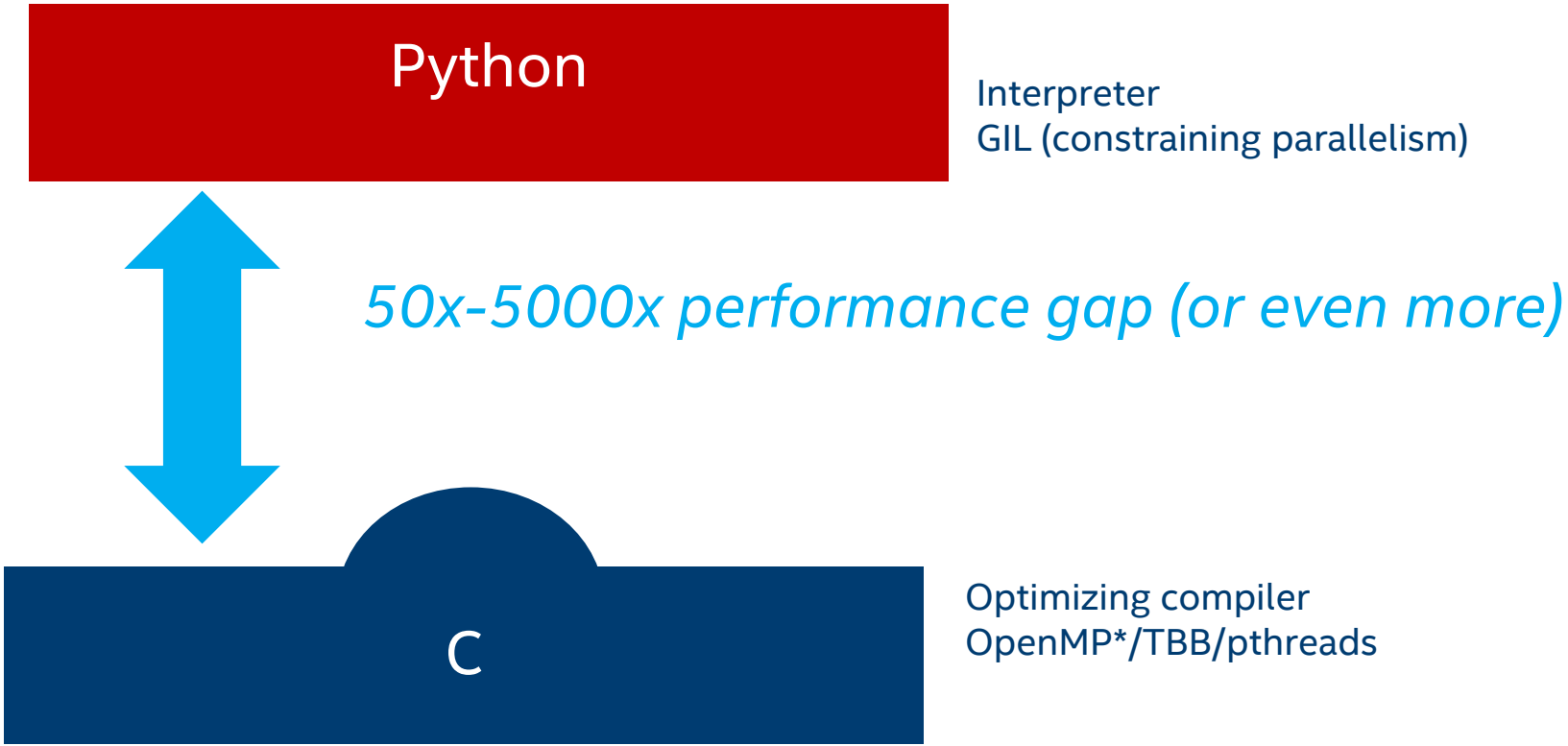


THE REALITY OF "DATA CENTRIC COMPUTING"

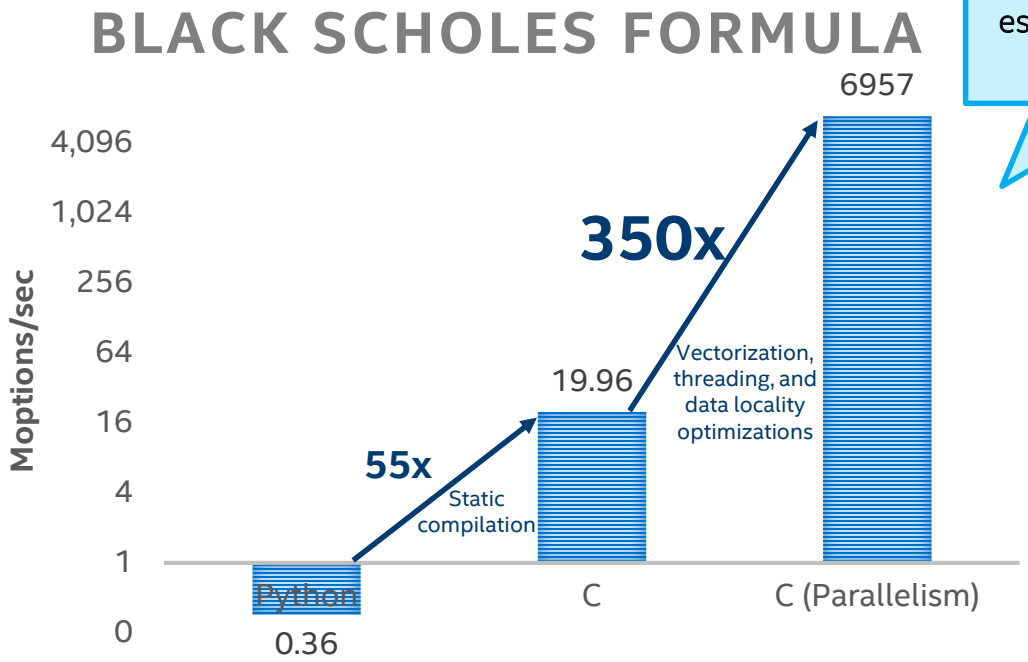
Software Challenges:

Performance Limited	• Software is slow and single-node for many organizations • Only sample a small portion of the data
Productivity Limited	• More performant/scalable implementations require significantly more development & deployment skills & time
Compute Limited	• Performance bottleneck often in compute, not storage/memory

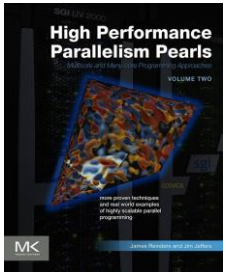
PERFORMANCE OF PYTHON



PARALLELISM MATTERS MOST



Unlocking parallelism is essential to make Python useful in production



Chapter 19: Performance Optimization of **Black—Scholes** Pricing

$$V_{call} = S_0 \cdot \mathrm{CDF}(d_1) - e^{-rT} \cdot X \cdot \mathrm{CDF}(d_2)$$
$$V_{put} = e^{-rT} \cdot X \cdot \mathrm{CDF}(-d_2) - S_0 \cdot \mathrm{CDF}(-d_1)$$
$$d_1 = \frac{\ln\left(S_0/X\right) + \left(r + \sigma^2/2\right)T}{\sigma\sqrt{T}}$$
$$d_2 = \frac{\ln\left(S_0/X\right) + \left(r - \sigma^2/2\right)T}{\sigma\sqrt{T}}$$

Configuration info: - Versions: Intel® Distribution for Python 2.7.10 Technical Preview 1 (Aug 03, 2015), icc 15.0; Hardware: Intel® Xeon® CPU E5-2698 v3 @ 2.30GHz (2 sockets, 16 cores each, HT=OFF), 64 GB of RAM, 8 DIMMS of 8GB@2133MHz; Operating System: Ubuntu 14.04 LTS.

TWO INGREDIENTS TO GET CLOSE-TO-NATIVE PERFORMANCE

Pure Python

Serial
Interpreted

Python + Libraries

Partially Ninja-level
Partially Interpreted

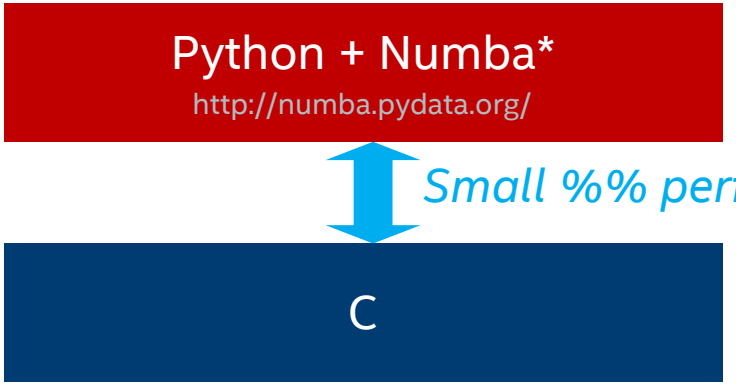
Libraries + JITC

Largely Ninja-level
100% native

C++

100% Ninja-level
100% native

PERFORMANCE OF PYTHON




```
9  @numba.jit(nopython=True, parallel=True)
10 def logistic_regression(Y, X, w0, step, iterations):
11     """SGD solver for binary logistic regression."""
12     w = w0.copy()
13     for i in range(iterations):
14         w += step * np.dot((1.0/(1.0 + np.exp(Y * np.dot(X, w)))) * Y, X)
15     return w
16
```

https://www.anaconda.com/blog/developer-blog/parallel-python-with-numba-and-parallelaccelerator/

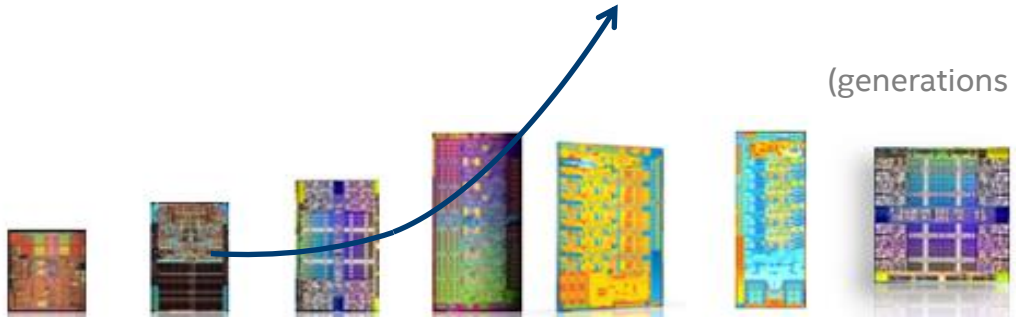
HIGH PERFORMANCE PYTHON

Python Libraries

Thin layer in Python or Cython

Intel® Performance Libraries

Native highly optimized libraries
(Intel® MKL, Intel® DAAL, Intel® IPP)

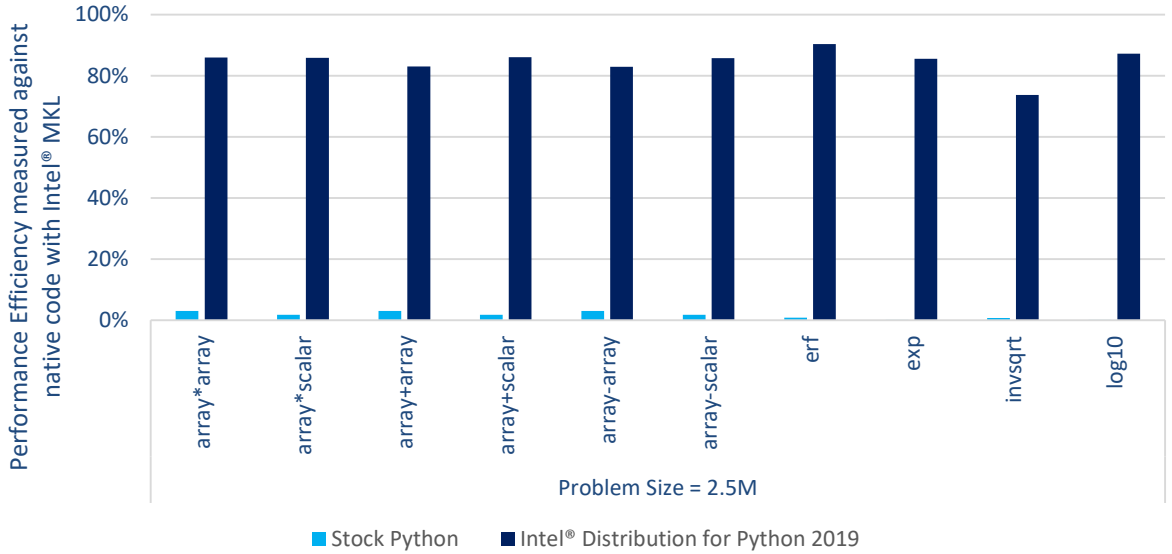


(generations of processors)

more nodes,
more cores,
more threads,
wider vectors, …

CLOSE TO NATIVE CODE UMATH PERFORMANCE WITH INTEL PYTHON 2019

COMPARED TO STOCK PYTHON PACKAGES ON INTEL® XEON PROCESSORS



87%

*native efficiency on a full **Black-Scholes** code with Intel **numpy + numba**.*

Configuration: Stock Python: python 3.6.6 hc3d631a_0 installed from conda, numpy 1.15, numba 0.39.0, llvmlite 0.24.0, scipy 1.1.0, scikit-learn 0.19.2 installed from pip;Intel Python: Intel Distribution for Python 2019 Gold: python 3.6.5 intel_11, numpy 1.14.3 intel_py36_5, mkl 2019.0 intel_101, mkl_fft 1.0.2 intel_np114py36_6,mkl_random 1.0.1 intel_np114py36_6, numba 0.39.0 intel_np114py36_0, llvmlite 0.24.0 intel_py36_0, scipy 1.1.0 intel_np114py36_6, scikit-learn 0.19.1 intel_np114py36_35; OS: CentOS Linux 7.3.1611, kernel 3.10.0-514.el7.x86_64; Hardware: Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz (2 sockets, 18 cores/socket, HT:off), 256 GB of DDR4 RAM, 16 DIMMs of 16 GB@2666MHz
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 complete information visit www.intel.com/benchmarks. Source: Intel Corporation - performance measured in Intel labs by Intel employees. 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.

Intel® Distribution for Python*

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

conda create –c intel intelpython3_full
docker pull intelpython/intelpython3_full
pip install intel-numpy intel-scipy intel-scikit-learn

Accelerated NumPy, SciPy
- Intel® MKL
- Intel® C and Fortran compilers
  - Linear algebra, universal functions, FFT

Accelerated Scikit-Learn
- Intel® MKL
- Intel® C and Fortran compilers

} via NumPy/Scipy

- Intel® Data Analytics Acceleration Library (DAAL)

Solutions for efficient parallelism
- TBB4py
- github.com/IntelPython/smp
- Intel® MPI library

Python APIs for Intel® MKL functions
- github.com/IntelPython/mkl_fft
- github.com/IntelPython/mkl_random
- github.com/IntelPython/mkl-service [*]

Python APIs for Intel® DAAL
- github.com/IntelPython/daal4py

Numba with upstreamed Intel contributions
- Parallel Accelerator
- support for SVML
- support for TBB/OpenMP threading runtimes

https://software.intel.com/en-us/distribution-for-python/benchmarks

HANDS-ON PREPARATION

Hands-On Sessions are for You!

Take your time to understand the Python code samples – don't just execute Jupyter cells 1by1

Also… there are solution files available, while it is in your own interest trying to find a solution yourself …

INSTALL ON YOUR LOCAL MACHINE

- Go to the following url to download slides and samples:

  https://tinyurl.com/intel-ai-workshop

- For installing the software on your own system:

  1. You need to have Conda on your system:

     https://docs.conda.io/projects/conda/en/latest/user-guide/getting-started.html

- Install Intel optimized software

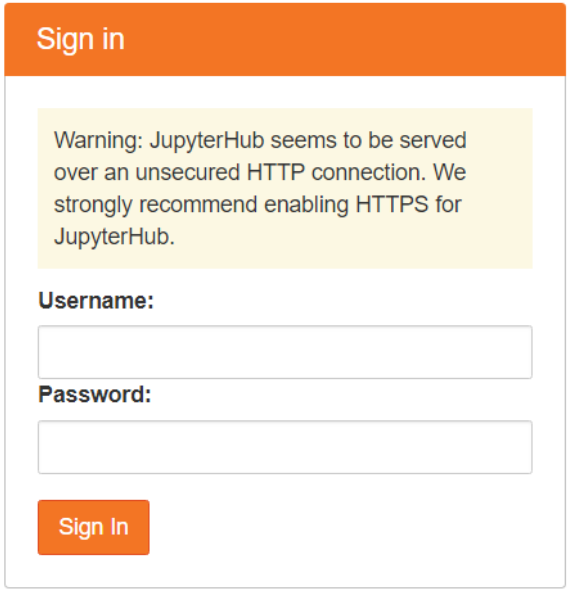
```
conda create -n intel-py -c intel intelpython3_core==2019.4
```

ACCESS THE JUPYTER SERVER

- Go to the following url:

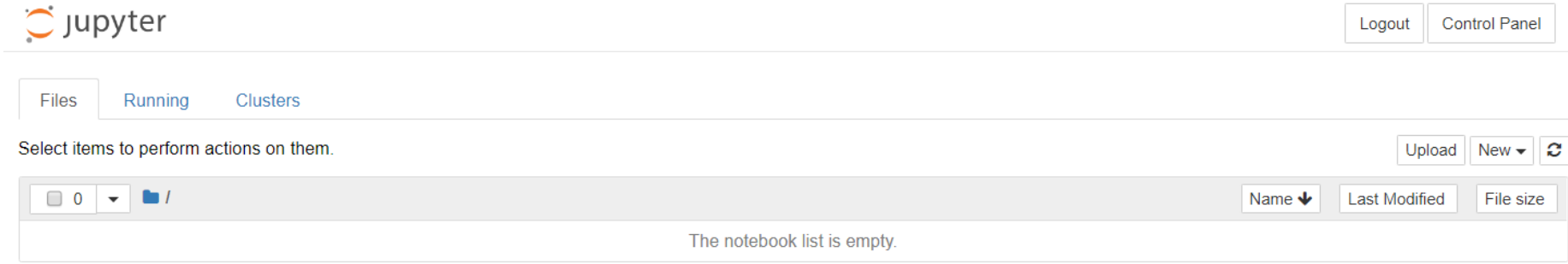
  https://tinyurl.com/intel-ai-workshop

- Look into the file **GCP_IP_ADDRESSES** for your assigned **IP_ADDRESS** associated to your unique integer number
- Take note of your **IP_ADDRESS**
- Open your browser and go to the following url: **http://IP_ADDRESS**

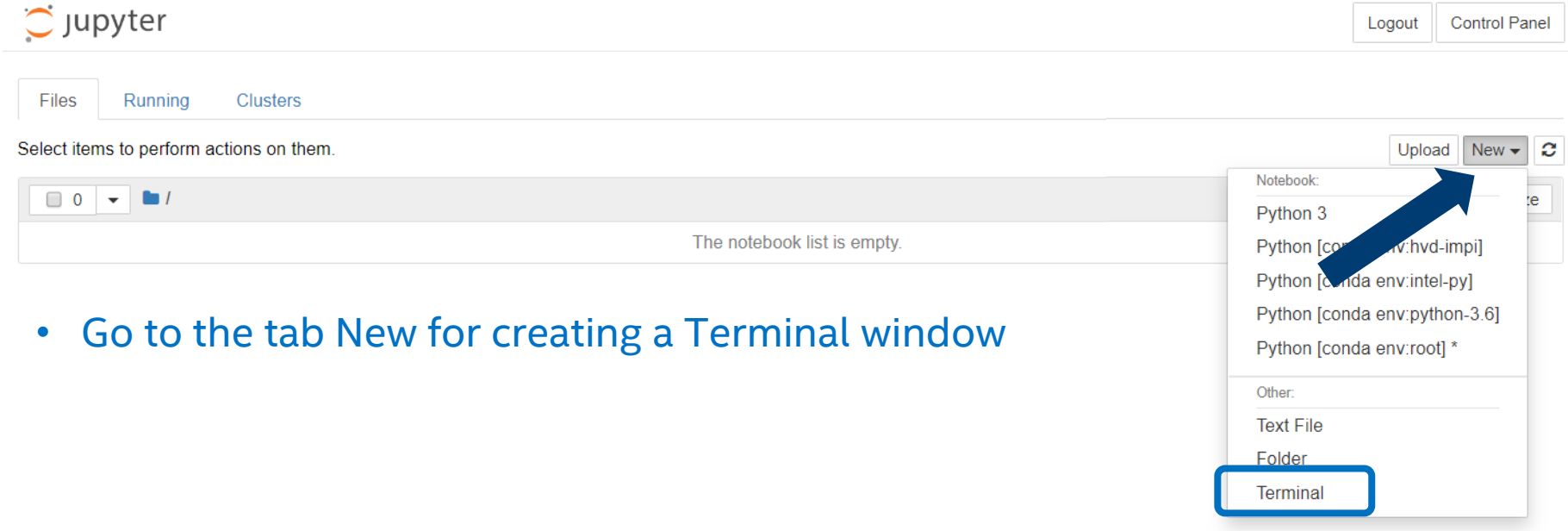


- ➢ Choose a Username and Password for your access
- ➢ Then Sign in

ACCESS THE JUPYTER SERVER

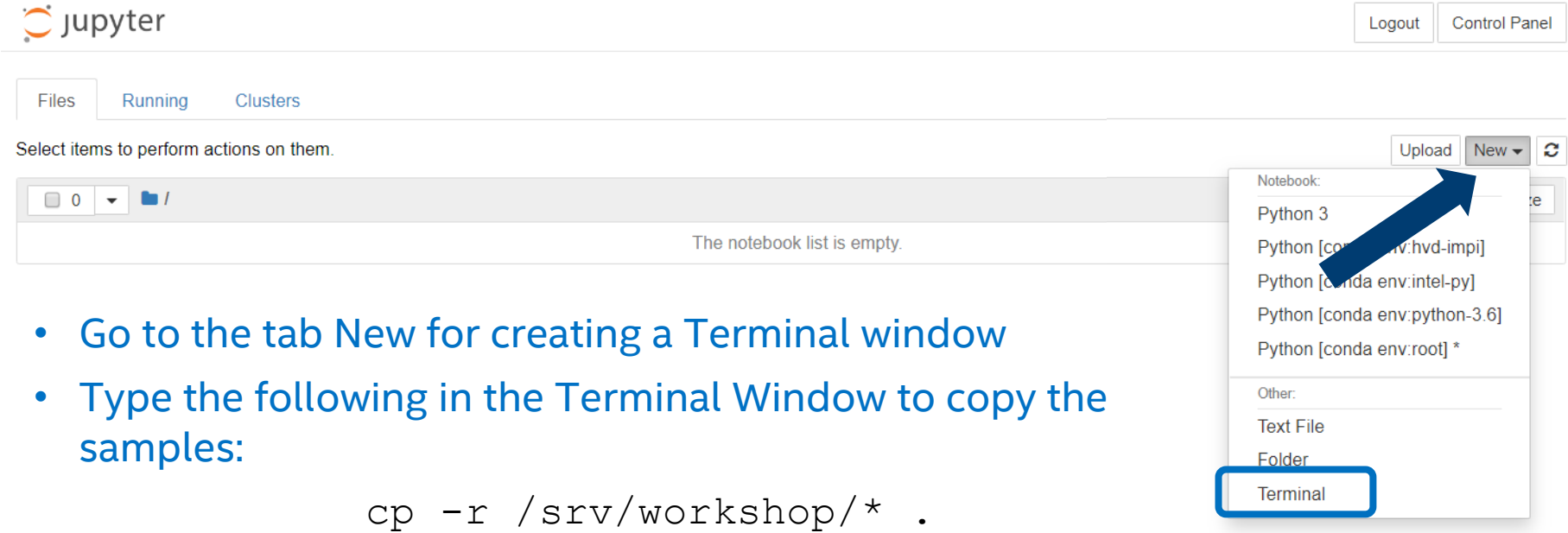


ACCESS THE JUPYTER SERVER



- Go to the tab New for creating a Terminal window

ACCESS THE JUPYTER SERVER



- Go to the tab New for creating a Terminal window
- Type the following in the Terminal Window to copy the samples:

```
cp -r /srv/workshop/* .
```

```
jupyter-usertest@ai-workshop:~$ cp -r /srv/workshop/* .
```

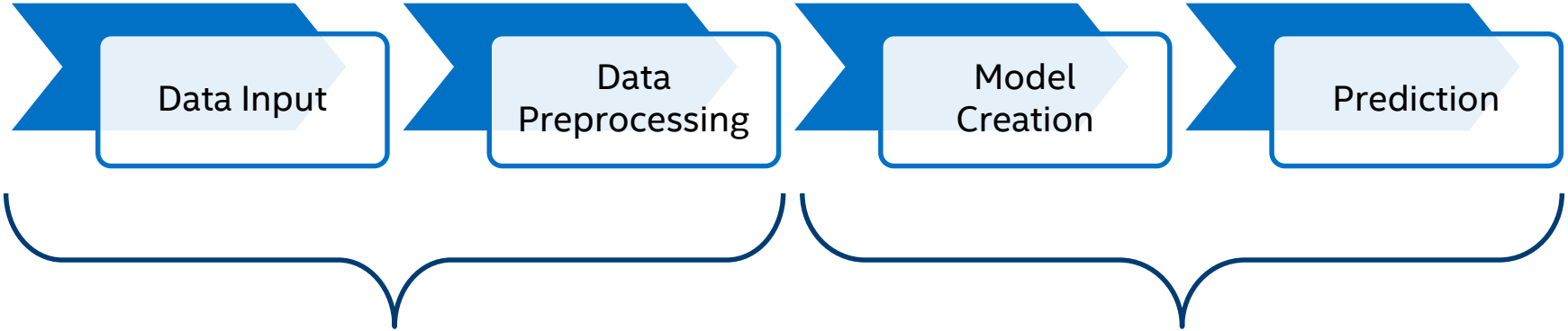
NUMPY-NUMBA:

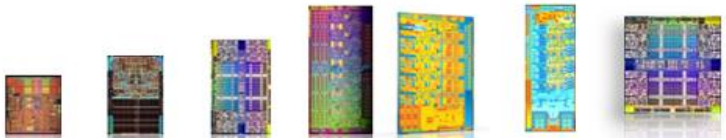
black-scholes.ipynb

1) Why is the performance using the NumPy functions is lower than expected?
2) Implement the black_scholes function in a NumPy like fashion
3) Measure the speedup and explain where exactly it is coming from
4) Implement the black_scholes function using Numba

INTEL® DATA ANALYTICS ACCELERATION LIBRARY

DATA ANALYSIS AND MACHINE LEARNING





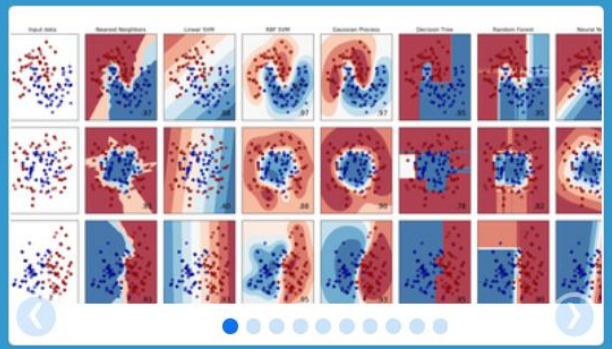
more nodes, more cores, more threads, wider vectors, …

THE MOST POPULAR ML PACKAGE FOR PYTHON*

scikit learn

Home | Installation | Documentation | Examples

Google Custom Search | Search



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

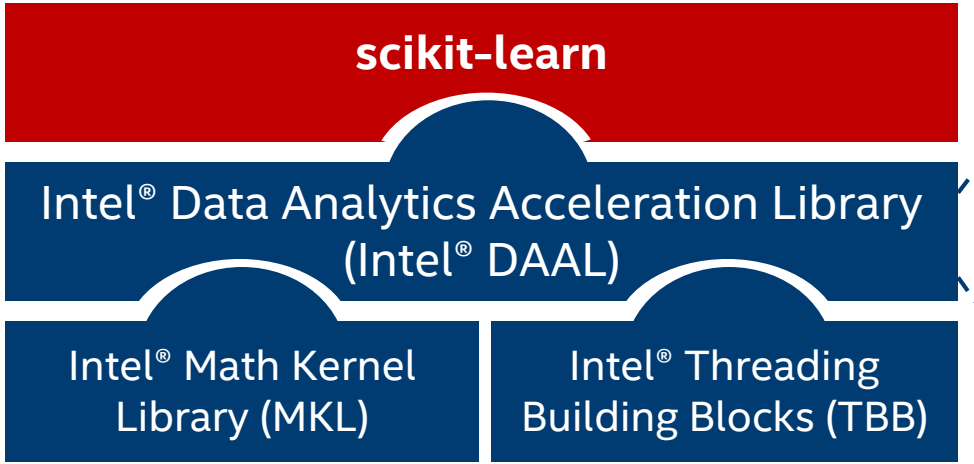
Automatic grouping of similar objects into sets.

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

DAAL4PY: ACCELERATED ANALYTICS TOOLS FOR DATA SCIENTISTS

- Package created to address the needs of **Data Scientists and Framework Designers** to harness the **Intel® Data Analytics Acceleration Library (DAAL)** with a **Pythonic API**
- Pandas compatible, **one-liner API** for accessing many hardware accelerated **Machine Learning and Analytics functions**
- Powers our ***Scikit-Learn* accelerations*** in our shipped version of the package
- **Extends capabilities** past Scikit-learn by providing **scaling and distributed modes**

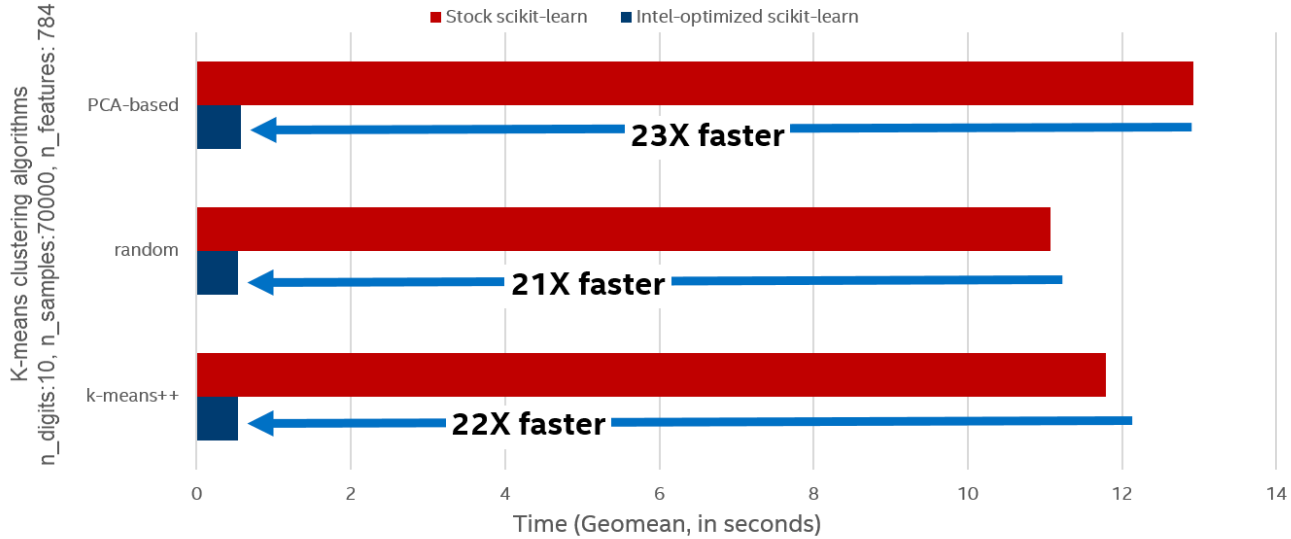
ACCELERATING MACHINE LEARNING



- Efficient memory layout via `Numeric Tables`
- Blocking for optimal cache performance
- Computation mapped to most efficient matrix operations (in MKL)
- Parallelization via TBB
- Vectorization

Try it out! `conda install -c intel scikit-learn`

ACCELERATING K-MEANS



System Configuration: GCP VM, zone us-central1-c; 96 vCPU, Intel Skylake; 360 GB memory. Ubuntu 16.04.3 LTS; Linux instance-1 4.10.0-38-generic #42~16.04.1-Ubuntu SMP Tue Oct 10 16:32:20 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux; Intel® Distribution for Python* from Docker image intelpython/intelpython3_full:latest (created 2017-09-12T20:10:42.862965559Z); Stock Python*: pip install scikit-learn

https://cloudplatform.googleblog.com/2017/11/Intel-performance-libraries-and-python-distribution-enhance-performance-and-scaling-of-Intel-Xeon-Scalable-processors-on-GCP.html

DAAL4PY

Fast & Scalable	• Close to native performance through Intel® DAAL • Efficient MPI scale-out • Streaming
Easy to use	• Known usage model • Picklable
Flexible	• Object model separating concerns • Plugs into scikit-learn • Plugs into HPAT
Open	• Open source: https://github.com/IntelPython/daal4py

https://intelpython.github.io/daal4py/

ACCELERATING SCIKIT-LEARN THROUGH DAAL4PY

Scikit-Learn **Equivalents**

Scikit-Learn **API** Compatible

daal4py

Intel® DAAL

```
> python -m daal4py <your-scikit-learn-script>
```

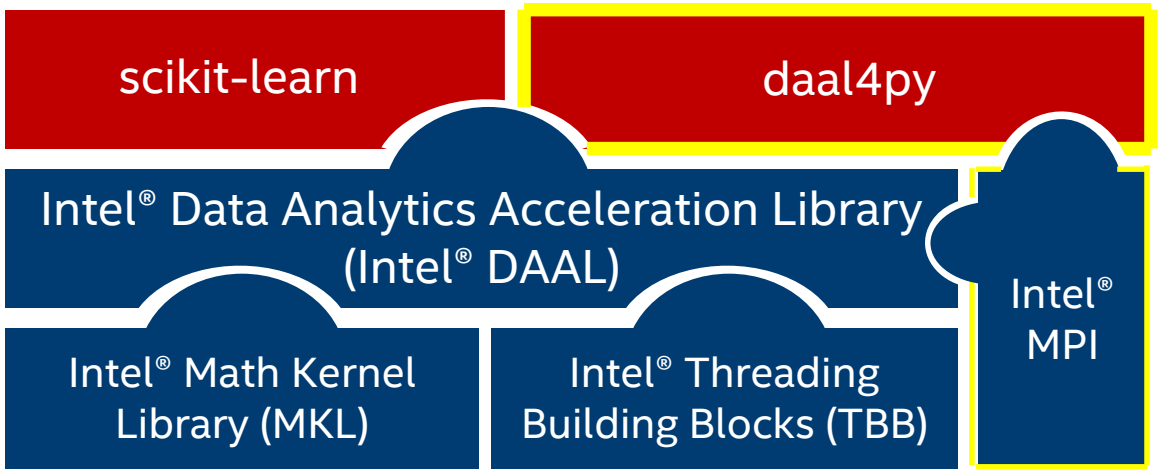
Monkey-patch any scikit-learn on the command-line

```
import daal4py.sklearn
daal4py.sklearn.patch_sklearn()
```

Monkey-patch any scikit-learn programmatically

https://intelpython.github.io/daal4py/

SCALING MACHINE LEARNING BEYOND A SINGLE NODE



Simple Python API
Powers scikit-learn

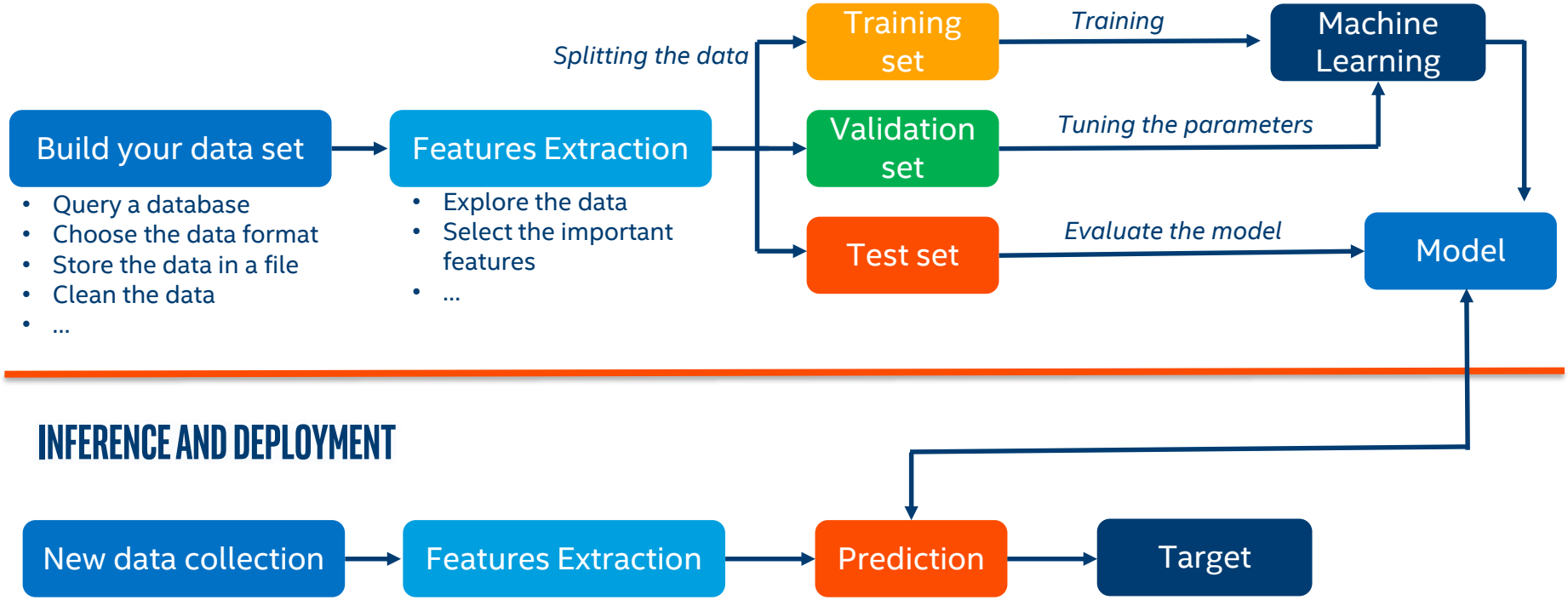
Powered by DAAL

Scalable to multiple nodes

Try it out! `conda install -c intel daal4py`

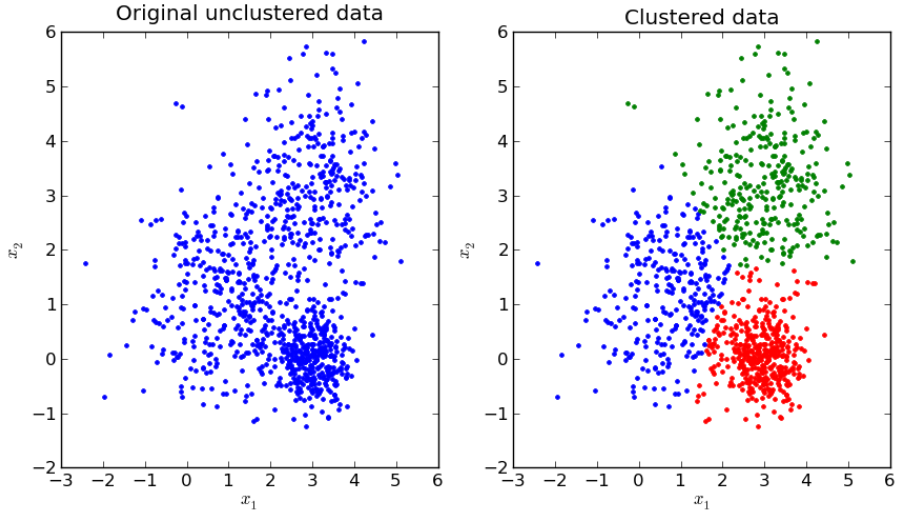
MACHINE LEARNING ALGORITHM: K-MEANS

MACHINE LEARNING WORKFLOW



K-MEANS ALGORITHM

- K-means is a clustering method
- It is used to group objects in a (high-dimensional) sample
- K-means is based on centroids



Interesting use cases for k-means

- Document classification
  - Similarity identification
- Customer segmentation
  - Identifying patters of interest areas
- Insurance fraud detection
  - Isolate new claims based on proximity to past fraud

source: https://dzone.com/articles/10-interesting-use-cases-for-the-k-means-algorithm

K-MEANS ALGORITHM DESCRIPTION

1. k (number of clusters in the dataset) is an external free parameter
2. k random objects are associated with initial centroids
3. each object of the dataset is assigned to form a cluster with its closest centroid
4. new centroids are computed by taking the average position of the objects in a given cluster
5. Evaluate convergence condition
6. Output: centroids location and association of the objects

Pros and cons:

- Simple and relatively robust

- Finding the best k is not trivial
- Results might depend on initial centroids
- Sensitive to outliers and to features with different dynamical range → data preparation

K-MEANS EXAMPLE: COLOR QUANTIZATION

- An interesting application of K-means is to reduce the number of colors in a figure, while preserving the overall quality
- Initial data: every pixel has three color channels (RGB) expressed with 8 bit (0…255)
- For comparison, a clustering based on randomly picked colors will be shown

The image of the Summer Palace (China) is among the sample data contained on SciKit-Learn

HANDS-ON

K-MEANS EXAMPLE: COLOR QUANTIZATION

`kmeans.ipynb`

The hands-on is based on six main steps:

1) Data preparation
2) Computing the cluster centers
3) Labelling the data
4) From scikit-learn to daal4py: command line
5) From scikit-learn to daal4py: monkey-patch in the script
6) **K-means with daal4py: kmeans-daal4py.py**

MACHINE LEARNING ALGORITHM: PCA

DIMENSIONALITY REDUCTION ALGORITHM

High-dimensional datasets are more and more common in data science

Their analysis often involves a dimensionality reduction:

- selecting a subset of features which best describes the dataset
- constructing a new set of features which provides a good description

Why to reduce data dimensionality?

- Removing noise
- Making the results easier to understand
- Making the dataset easier to be used (data handling and movement)
- Reducing computational cost of algorithms (data processing)

PRINCIPAL COMPONENT ANALYSIS (PCA)

PCA: popular method for dim. reduction

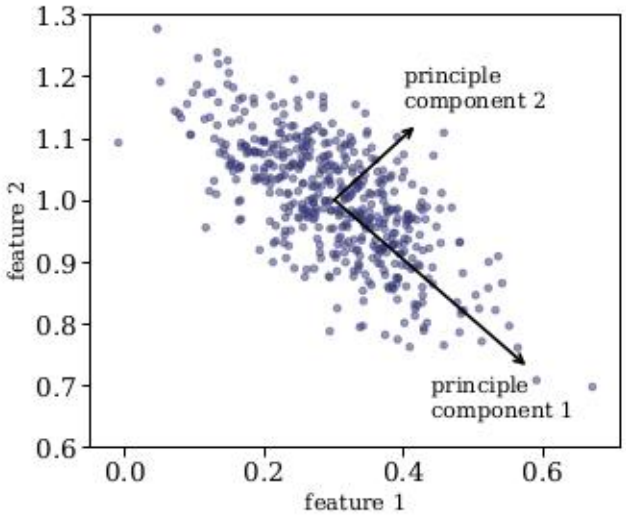
In essence, it is a coordinate transformation in the dataset

In the new coordinate system, the first coordinate (component) is chosen so to have the largest variance in the data.

The second component is orthogonal and has the second largest variance.

Number of components = number of features of the data.

Lower number of components → dimensionality reduction.



DEMO

PCA COMPARISON EXAMPLE

scikit-learn

```
infile =
"./data/pca_normalized.csv"

data = read_csv(infile)

result= PCA()

result.fit(data)
```

daal4py

```
infile =
"./data/pca_normalized.csv"

data = read_csv(infile)

algo =
d4p.pca(resultsToCompute="mean|v
ariance|eigenvalue",isDeterminis
tic=True)

result = algo.compute(data)
```

Legal Disclaimer & Optimization Notice

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 complete information visit www.intel.com/benchmarks.

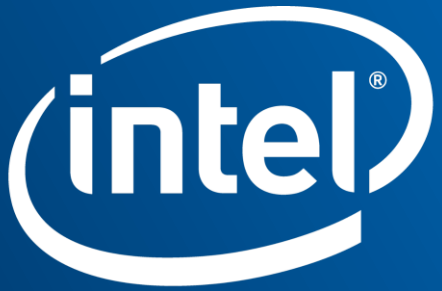
INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING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

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

BACKUP



Software