Harnessing the Power of HPC for Machine Learning

Tools and Techniques

Charles Peterson

👋 Welcome Everyone! 💻

This workshop provides an overview of topics and practical examples of using Machine Learning tools on HPC resources. ⭐

🔑 Key Topics:

- Python/R in HPC (Hoffman2)
- ML Package Installation
- Interactive and Batch Job Submission
- Big Data Insights

For suggestions: cpeterson@oarc.ucla.edu

📖 Access the Workshop Files

This presentation and accompanying materials are available on 🔗 UCLA OARC GitHub Repository

You can view the slides in:

- 📄 PDF format - WS_MLonHPC.pdf
- 🌐 HTML format: Workshop Slides

Each file provides detailed instructions and examples on the various topics covered in this workshop.

> **Note:** 🛠️ This presentation was built using Quarto and RStudio.

Machine Learning and HPC

Machine Learning Basics

- What is Machine Learning?
  - 🤖 **Machine Learning (ML)** is a subset of artificial intelligence (AI) focused on building systems that learn from and make decisions based on data.
- Key Concepts:
  - **Data:** The foundation of any ML model. It can be labeled (supervised learning) or unlabeled (unsupervised learning).
  - **Algorithms:** Procedures or formulas for solving a problem. Common ML algorithms include linear regression, decision trees, and neural networks.
  - **Training:** The process of teaching a machine learning model to make predictions or decisions based on data.
  - **Inference:** Applying the trained model to new data to make predictions.
- Types of Machine Learning:
  - 🔍 **Supervised Learning:** The model learns using labeled data (e.g., spam detection).
  - 🧠 **Unsupervised Learning:** The model identifies patterns in data without any labels (e.g., customer segmentation).

❓ What is HPC and Why Should I Care

- 🚀 HPC uses **MANY computers** to solve large problems faster than a normal computer.
- 🕒 If your task takes a long time to run on a laptop or a lab's server, HPC can '**speed up**' your application.
- 📚 You can store **LARGE amounts of data**, too big for your laptop.

🌐 General vs High-Performance Computing

General Purpose Computing

- 💁‍♂️ Only one person at a time.
- 🖥️ Calculations run on the machine directly.
- 🧮 Can only run 1-2 calculations at a time.

High-Performance Computing

- 👥 Multiple people can log in at one time.
- ⏱️ Calculations are ‘scheduled’ to run on a different machine.
- 💻 Can run hundreds of calculations at one time.

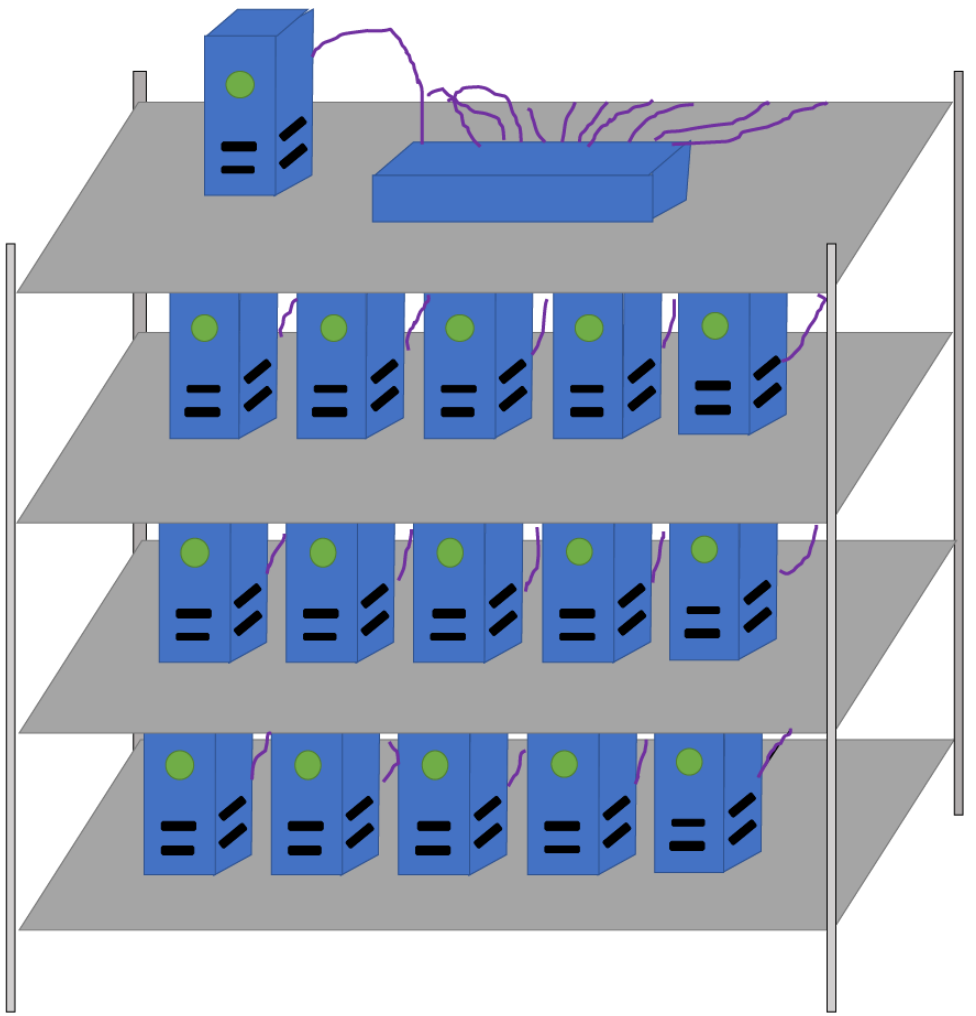
🤝 HPC provides an excellent platform for collaborations and faster results.

HPC Overview

Beowulf style cluster

Multiple computers

Single computing resource



⚡ The Power of HPC

Single Computer Limitations:

- 🧠 Only one CPU.
- 🚧 Large problems cannot fit.
- ⏳ Long processing times.
- 📉 Limited memory and disk space.

HPC Solutions:

- 🖥️ **More CPUs** for faster processing.
- 📈 **More memory** to handle bigger tasks.
- 💾 **More disk space** for extensive data.
- 🎮 **Access to GPUs** for advanced computations.

💪 The Power of HPC: Parallelization

Single CPU Program

- Task A ➡️ Task B ➡️ Task C
- 🕒 Total Time: 3 hours

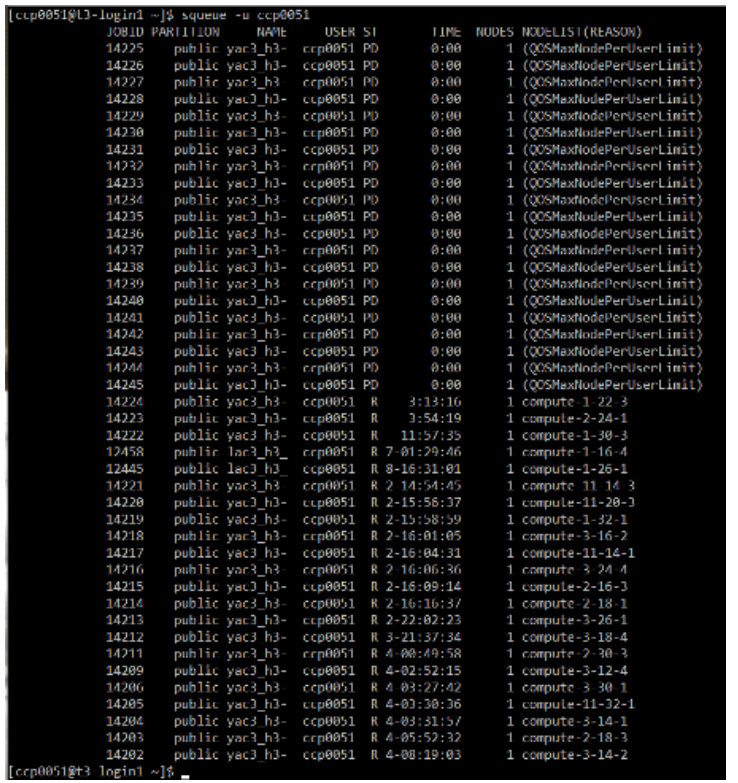
Parallel Tasks

- Task A on CPU 1 🕒 1 hour
- Task B on CPU 2 🕒 1 hour
- Task C on CPU 3 🕒 1 hour
- ⚡ Total Time: 1 hour

🤖 Most Machine Learning packages can utilize multiple CPUs and GPUs to run your models in parallel!

Multi-tasking with Machine Learning Jobs

- 🚀 With HPC resources, you can have **multiple machine learning jobs running concurrently**.
- 📈 This parallel processing greatly increases efficiency and productivity.
- 🌐 Ideal for complex computations and large-scale data analysis.



🐍 Python

Hoffman2 supports running 🐍 Python applications.

Hoffman2 supports 🐍 Python applications, and it is **HIGHLY** recommended to use Python versions built and tested by Hoffman2 staff.

🚫 Avoid using system python builds (e.g., `/usr/bin/python`). Instead, use `module load` commands to access optimized versions.

- To see all Python versions installed on Hoffman2:

```
modules_lookup -m python
```

- Load a Python module

```
module load python/3.7.3
which python3
```

- This example shown:
  - Python version 3.7.3
  - Location of python
    - `/u/local/apps/python/3.7.3/gcc-4.8.5/bin/python3`
    - (Location of the Hoffman2 installed python)

🐍 Python Packages in Machine Learning

⚙️ Scikit-learn:

- Versatile tools for machine learning, including classification, regression, and clustering.

🟠 TensorFlow:

- Google’s library for deep learning and neural networks.

💙 Keras:

- Python interface for neural networks, primarily an interface for TensorFlow.

🔥 PyTorch:

- Flexible deep learning library by Facebook’s AI Research lab.

📈 XGBoost:

- Efficient gradient boosting library, ideal for structured data.

📊 Pandas:

- Essential for data manipulation and analysis, a cornerstone in machine learning.

➕ NumPy:

- Fundamental for scientific computing, supports large arrays and matrices.

🔬 SciPy:

- For scientific and technical computing, extends NumPy’s capabilities.

🐍 Python Installation on Hoffman2

Basic Builds on Hoffman2: The Python builds on Hoffman2 include only the basic compiler/interpreter and a few essential packages.

User-Installed Packages:

- 📦 Most Machine Learning Python applications will require additional packages, installed by the user.
- 🚫 Hoffman2 staff do **not** install extra packages in the supported Python builds to avoid conflicts.

Installing Machine Learning Packages:

- ⭐ When using Python (or R), you'll need to install the ML packages yourself.
- 📚 We have a workshop covering this topic in detail:
  - 🔗 Python/R Installation Workshop

🔧 User-Installed Packages on Hoffman2

- 🚫 Users cannot install packages in the main Python build directories.
  - This is to avoid version conflicts and dependencies issues that could break Python.
- 👤 Users can install packages in their own directories:
  - `$HOME`, `$SCRATCH`, or any project directories.

Installation Methods:

- 🔧 **Using pip package manager:** Ideal for standard Python package installations.
- 🌐 **Using Python Virtual Environments:** Creates isolated environments for specific projects.
- 🐍 **Using Anaconda:** Suitable for managing complex package dependencies and environments.

📦 Using pip Package Manager

Installing scikit-learn with pip:

- To install the `scikit-learn` package via pip (PyPI) package manager:

```
module load python/3.7.3
pip3 install scikit-learn --user
```

Understanding the –user Flag:

- 🏠 The `--user` flag ensures the package installs in your $HOME directory.
- 🚫 By default, pip tries to install in the main Python build directory, where users lack write access.
- 📁 Using `--user`, packages install in $HOME/.local, avoiding permission errors.

📊 R on Hoffman2

Finding Available Versions of R:

- ⭐ Hoffman2 supports various versions of R.
- To view all available versions of R on Hoffman2:

```
modules_lookup -m R
```

Loading a Specific Version of R: Example to load R version 4.2.2 with GCC version 10.2.0:

```
module load gcc/10.2.0
module load R/4.2.2
```

Ensuring Correct Module Loads:

-🔧 Load the gcc or intel modules first, as indicated by `modules_lookup`. This step ensures that the correct versions of gcc and intel libraries are loaded for R.

📊 R Packages

⚙️ Caret:

- Framework for building machine learning models. Offers tools for data pre-processing, feature selection, and model tuning.

🌲 RandomForest:

- Implements the random forest algorithm. Known for performance in classification and regression.

🔍 e1071:

- Contains functions for SVMs, naive Bayes classifier, and more.

🧠 nnet:

- For training single-hidden-layer neural networks and multinomial log-linear models.

🌳 rpart:

- Recursive partitioning for decision tree models.

📈 xgboost:

- Efficient gradient boosting, effective for large datasets.

🔗 glmnet:

- Fitting generalized linear and Cox models via penalized likelihood.

📜 tm:

- Text mining framework, managing and mining text data.

🎨 ggplot2:

- Powerful data visualization tool based on the Grammar of Graphics.

🔢 dplyr:

- Essential for data manipulation, providing a set of tools for dataset management.

📦 R Package Installation

Standard Installation Command:

- Use the following command to install R packages:

```
install.packages('PKG_name')
```

- 🚫 On Hoffman2 (and most other HPC resources), you cannot modify the main R global directory.
- Example Installation:

```
install.packages("dplyr")
```

- 🏠 R will suggest a new path in your $HOME directory, determined by $R_LIBS_USER.
- Each R module on Hoffman2 has a unique $R_LIBS_USER to prevent conflicts between different R versions.

🐍 Anaconda

Anaconda is a popular Python and R distribution, ideal for simplifying package management and pipelines.

Hoffman2 has Anaconda installed, allowing users to create their own conda environments.

```
1  module load anaconda3
```

> **Warning**
>
> 🚫 No Need for Other Python/R Modules:
>
> - Your Anaconda environment includes a build of Python and/or R. Loading other modules may cause conflicts.

> **Note**
>
> For more information, we had done a workshop on using Anaconda on Hoffman2 that you can review.

📦 Containers

Containers, like Apptainer and Docker, are excellent for running Machine Learning applications on Hoffman2.

Advantages of Containers:

- 🏗️ **Isolated Environments:** Comes with all necessary Machine Learning software pre-installed.
- 🚚 **Portability:** Use the same container on different computers, ensuring version control and reproducibility.

Apptainer on Hoffman2:

- 🔧 Hoffman2 uses Apptainer for running containers.
- 🔍 For more information, refer to our previous workshop:
  - 🔗 Containers Workshop

Example: Fashion MNIST

👗 Fashion MNIST

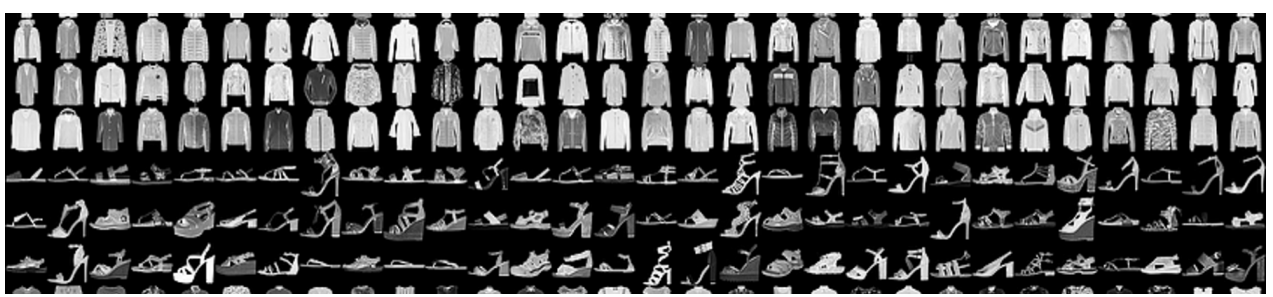
This example focuses on the “Fashion MNIST” dataset, a collection used frequently in machine learning for image recognition tasks.

Approach:

- 🌲 We will use a Random Forest algorithm to train a model for predicting fashion categories.

Dataset Overview:

- 📸 **Images:** 28x28 grayscale images of fashion products.
- 📊 **Categories:** 10, with 7,000 images per category.
- 🧱 **Total Images:** 70,000.



🔬 ML Packages for Python and R

Using Scikit-learn with Python:

- 🤖 Ideal for algorithms like classification and clustering.
- 🧩 Useful for preprocessing, model building, and evaluation.

Package Installation:

Python:

- Install Python and Scikit-learn:

```
module load python/3.9.6
pip3 install sklearn --user
```

R:

- Install R and necessary packages:

```
module load gcc/10.2.0
module load R/4.2.2
# Needed for OpenML package
module load libxml2
R -e 'install.packages(c("randomForest", "OpenML", "dplyr", "ggplot2", "caret", "farff"), repos = "https://cran.r-project.org/"
```

Python Example Run

Getting Started with Interactive Compute Node

- Start by requesting an interactive compute node:

```
qrsh -l h_data=10G
```

Cloning and Navigating to the Code Repository

- Clone the repository and navigate to the mnist-ex directory:

```
cd $SCRATCH
git clone https://github.com/ucla-oarc-hpc/WS_MLonHPC
cd WS_MLonHPC/mnist-ex
```

Lets look at the code, `minst.py`

Running the Python Script:

- Load Python module and run the mnist.py script:

```
module load python/3.9.6
python3 mnist.py
```

Parallel Processing with Python

The initial training took about 1 minute over 1 CPU core.

Speeding Up with Parallel Processing:

- Request 10 cores for parallel processing:

```
qrsh -l h_data=10G -pe shared 10
```

Note: Use the shared parallel environment as sci-kit learn doesn't support multi-node parallelism.

Code Adjustment for Parallelism:

- Lets look at the code, `minst-par.py`
- The main change is n_jobs option in the Classifier

```
clf = RandomForestClassifier(random_state=42, n_jobs=10)
```

Run the code!

```
module load python/3.9.6
python3 mnist-par.py
```

📋 Batch Submission on Hoffman2

Submitting Non-Interactive Jobs:

- For tasks that don't require interactive sessions, you can submit jobs to be processed in the background.

Command to Submit a Job:

- Use the `qsub` command to submit your job script to the queue:

```
1  qsub mnist-py.job
```

Advantages:

- 🚀 Efficient for longer or resource-intensive tasks.
- 🕒 Allows you to free up your session while the job runs in the background.

Running R

Executing Code with a Single CPU:

- Start with requesting an interactive compute node:

```
qrsh -l h_data=10G
module load gcc/10.2.0
module load R/4.2.2
Rscript mnist.R
```

Running Code with Parallel Processing (10 CPUs):

- Request multiple cores for parallel execution:

```
qrsh -l h_data=10G -pe shared 10
module load gcc/10.2.0
module load R/4.2.2
Rscript mnist-par.R
```

Submitting as a Batch Job:

- For non-interactive execution, submit the job script:

```
qsub mnist-R.job
```

Example: DNA Sequence

DNA Sequence classification

🧬 DNA Sequence Classification with PyTorch

- 🧬 **Objective:** Create a model to classify DNA sequences into ‘gene’ or ‘non-gene’ regions.
- **Gene Regions:** Segments of DNA containing codes for protein production.
- **Dataset Creation:** Generate random DNA sequences labeled as ‘gene’ or ‘non-gene’.
- 🤖 **Model Development:** Use PyTorch to build a model predicting the presence of ‘gene’ regions.
- 🚀 **Leveraging GPUs:** Utilize the parallel processing power of GPUs for efficient training.

🐍 Creating a Conda Environment

Setting Up for GPU-Enabled PyTorch:

- Begin by loading the Anaconda module:

```
module load anaconda3
```

- Create a new Conda environment named biotest with Python, scikit-learn, and scipy:

```
conda create -n biotest python=3.11 scikit-lear
```

- Activate the newly created environment:

```
conda activate biotest
```

- Install PyTorch with GPU support using pip:

```
pip3 install torch
```

🧬 Running PyTorch on Hoffman2

Code Location and Versions:

- The code for this task is located in the `dna-ex` directory.
- There are two versions:
  - 💻 `dna-cpu.py` for the CPU version.
  - 🎮 `dna-gpu.py` for the GPU version.

Executing the Examples:

- Request a node with GPU resources:

```
qrsh -l h_data=10G,gpu,A100
```

- Run CPU version

```
conda activate biotest
python3 dna-cpu.py
```

- Run GPU version

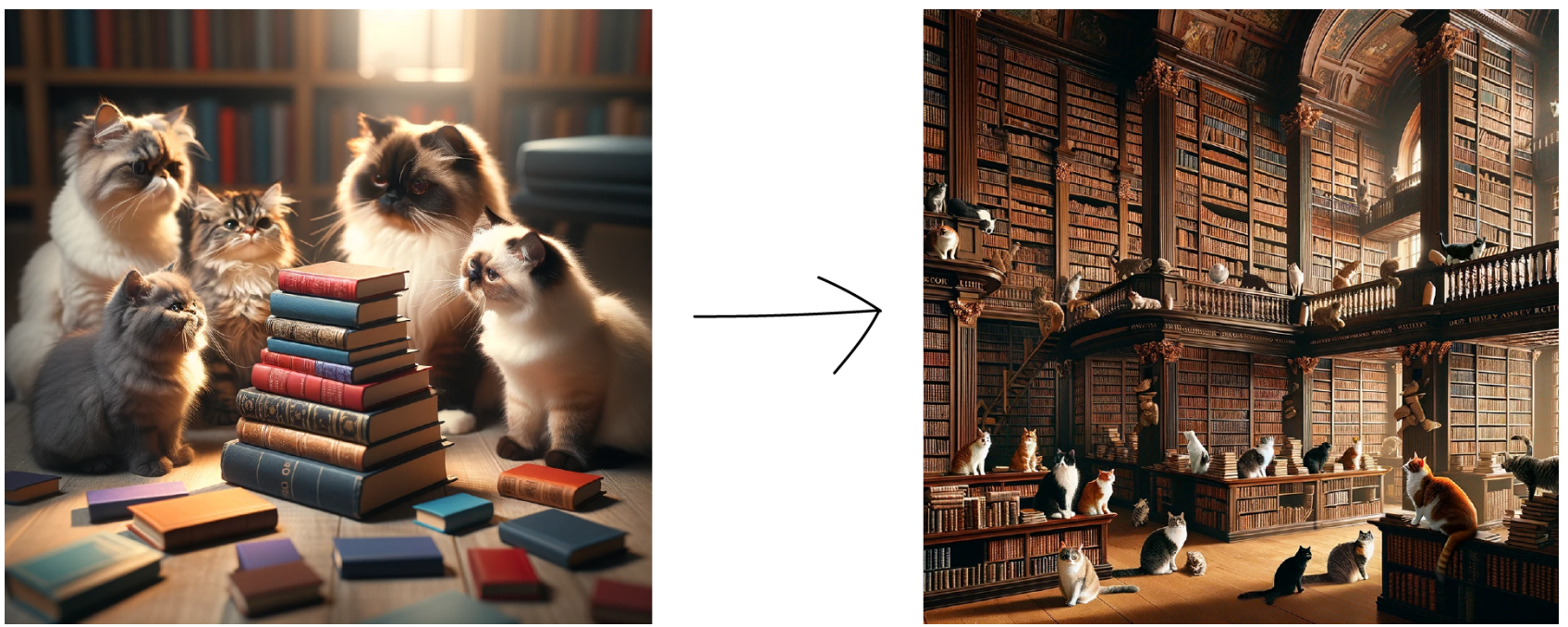
```
python3 dna-gpu.py
```

Understanding Big Data

Big Data

The term **Big Data** refers to datasets and data science tasks that become too large and complex for traditional techniques.

- For comprehensive information, check our workshop on Big Data on HPC



🛠️ Big Data Tools

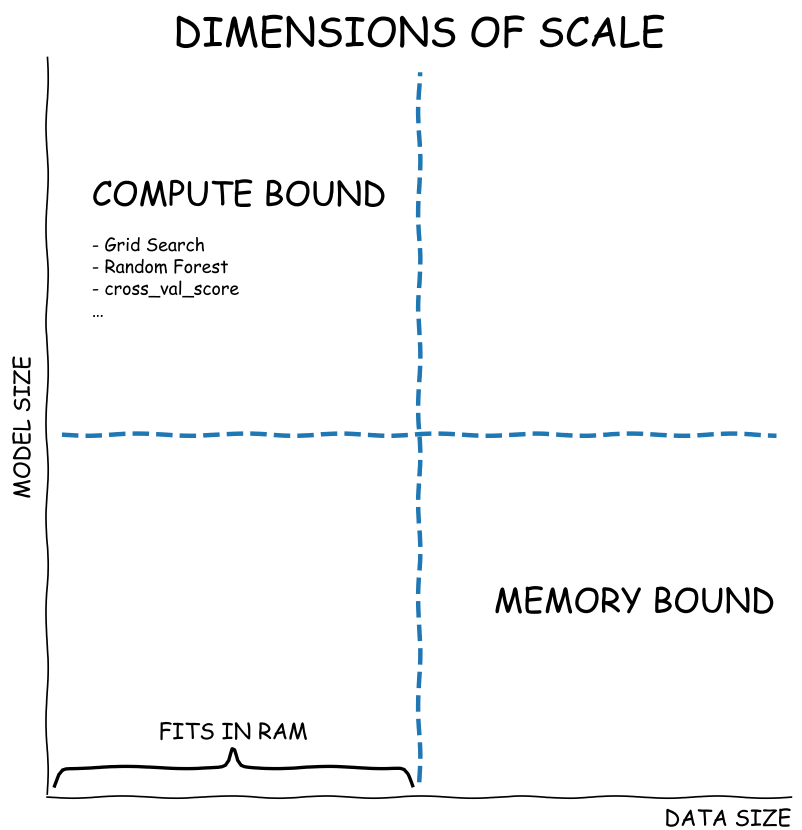
Explore various frameworks, APIs, and libraries for handling Big Data

🚧 Challenges with LOTS of Data

Dealing with extensive **DATA** presents unique challenges 😰:

- 🧠 **Insufficient RAM:** Struggling to accommodate large datasets.
- ⏳ **Time-Consuming Processing:**
  - Difficulty in managing large datasets with traditional techniques.
  - Prolonged computation times.
- 🤖 **Complex Machine Learning Models:**
  - Training advanced models requires significant computational power for accuracy.
- 🤖 **Solution: High-Performance Computing (HPC)**
  - HPC resources supercharge solving Big Data challenges with superior computing power 💪
  - Many Big Data tools are designed to run efficiently across multiple compute nodes in HPC systems.

🚧 Big Data Challenges



- **Scaling Data Size** 📈
  - Datasets can become so large that they can't fit into RAM 😱
- **Scaling Model/Task Size** 🤖
  - Machine Learning or other tasks become so complex that a single CPU core is not adequate 🐌

Example: Million Song Dataset

Million Song Example

Using Spark's MLlib for Music Data Analysis:

- This example utilizes Spark's Machine Learning library (MLlib).
- We will analyze data from the Million Song Subset.

Dataset Characteristics:

- 🎵 The subset contains approximately 500,000 songs.
- 📊 Features include:
  - Year of the song.
  - 90 features related to the timbre average and covariance.

🔧 Installing Spark and PySpark

Creating and Activating the Conda Environment:

- Load Anaconda and create a new environment named `mypyspark`:
- Installing Spark

```
module load anaconda3
conda create -n mypyspark openjdk pyspark python \
                          pyspark=3.3.0 py4j jupyterlab findspark \
                          h5py pytables pandas matplotlib \
                          -c conda-forge -c anaconda -y
conda activate mypyspark
pip install ipykernel
ipython kernel install --user --name=mypyspark
```

Environment Features:

- 📚 This Conda environment, mypyspark, is configured with Jupyter.
- 🚀 It includes both Spark and PySpark, ready for big data processing tasks.

PySpark: Basic Operations 📋

Let’s practice basic PySpark functions with examples.

- Download the workshop content from the GitHub repository
- We’ll work with a Jupyter Notebook: Spark_basics.ipynb
- Jupyter Notebook: `MSD.ipynb` from `MSD_ex`

```
cd $SCRATCH
git clone https://github.com/ucla-oarc-hpc/WS_MLOnHPC
cd WS_MLonHPC/MSD-ex
```

Downloading the Dataset:

- Retrieve the dataset to your workspace:

```
cd $SCRATCH/WS_MLonHPC/MSD-ex
wget https://archive.ics.uci.edu/ml/machine-learning-databases/00203/YearPredictionMSD.txt.zip
unzip YearPredictionMSD.txt.zip
```

PySpark: Basic operations: Starting the notebook

We will use the `h2jupynb` script to start Jupyter on Hoffman2

You will run this on your LOCAL computer.

```
wget https://raw.githubusercontent.com/rdauria/jupyter-notebook/main/h2jupynb
chmod +x h2jupynb

#Replace 'joebruin' with you user name for Hoffman2
#You may need to enter your Hoffman2 password twice

python3 ./h2jupynb -u joebruin -t 5 -m 10 -e 2 -s 1 -a intel-gold\\* \
                   -x yes -d /SCRATCH/PATH/WS_MLonHPC/MSD-ex
```

> **Note**
>
> The `-d` option in the `python3 ./h2jupynb` will need to have the `$SCRATCH/WS_MLonHPC` full PATH directory

This will start a Jupyter session on Hoffman2 with ONE entire intel-gold compute node (36 cores)

More information on the `h2jupynb` can be found on the Hoffman2 website

AutoML with H2O.ai

Introduction to AutoML

AutoML, or Automated Machine Learning, is an innovative approach to automating the process of applying machine learning to real-world problems.

Key Benefits

- **Efficiency:** Streamlines the model development process.
- **Accessibility:** Makes ML more accessible to non-experts.
- **Optimization:** Automatically selects the best models and parameters.

Components of AutoML

- **Data Preprocessing:** Automatic handling of missing values, encoding, and normalization.
- **Feature Engineering:** Automated feature selection and creation.
- **Model Selection:** Choosing the best model from a range of algorithms.
- **Hyperparameter Tuning:** Optimizing parameters for peak performance.
- **Model Validation:** Ensuring robustness through cross-validation.

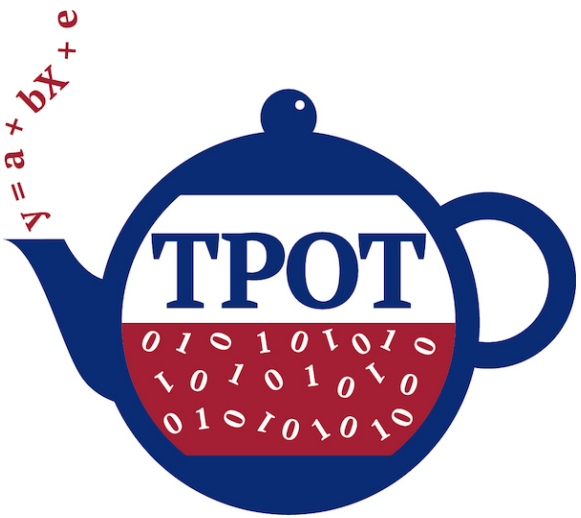
AutoML Tools

HPC resouces can be use to echance AutoML since it can be very computationally demanding

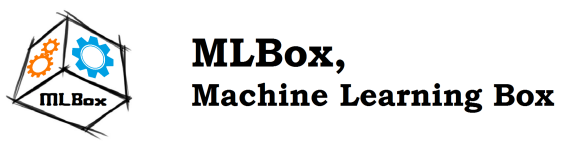
H2O.ai AutoML: An open-source platform that automates the process of training and tuning a large selection of candidate models within H2O, a popular machine learning framework.

Auto-sklearn: An automated machine learning toolkit based on the scikit-learn library, focusing on automating the machine learning pipeline, including preprocessing, feature selection, and model selection.

TPOT (Tree-based Pipeline Optimization Tool): An open-source Python tool that uses genetic algorithms to optimize machine learning pipelines.



MLBox: A powerful Automated Machine Learning python library that provides robust preprocessing, feature selection, and model tuning capabilities.



Auto-Keras: Auto-Keras is a AutoML program built on the Keras platform.

💧 Using H2O.ai for AutoML

Setting Up H2O.ai for Automated Machine Learning:

- Start by loading Anaconda and creating a new environment named `h2oai`:

```
module load anaconda3
conda create -n h2oai python matplotlib -c conc
```

- Activate the newly created environment:

```
conda activate h2oai
```

- Install essential packages including H2O:

```
pip install requests tabulate future h2o
```

- Install IPython kernel and configure it for the h2oai environment:

```
pip install ipykernel
ipython kernel install --user --name=h2oai
```

🚀 Your environment is now set up with H2O.ai, ready for AutoML tasks.

💧 H2O AutoML Example

Exploring AutoML with H2O:

- We will work through an AutoML example from H2o-tutorials.
- The focus is on the Combined Cycle Power Plant dataset.
- ⭐ **Objective:** Predict the energy output of a Power Plant using temperature, pressure, humidity, and exhaust vacuum values.
- This example, we will use the Python API, but H2O.ai has a R API as well

Accessing the Notebook:

- The Jupyter notebook for this example is in the `automl-ex` directory.
- To start Jupyter, execute the following command, adjusting the path as necessary:

```
python3 ./h2jupynb -u joebruin -t 5 -m 50 -e 2 -s 1 -a intel-gold\\* \
                   -x yes -d /SCRATCH/PATH/WS_MLonHPC/automl-ex
```

Wrap-up

🌟 Workshop Highlights

- High-Performance Computing (HPC) and Machine Learning:
  - 🚀 Introduction to HPC and its benefits for Machine Learning.
  - 🐍 Utilizing Python and R on HPC for advanced data processing.
- Key Tools and Frameworks:
  - 📦 Installation and usage of vital Python packages like Scikit-learn, PyTorch.
  - 📊 R package installation and management in HPC environment.
  - 🐍 Setting up Anaconda environments for Python and machine learning libraries.
- Big Data and Its Challenges:
  - 💥 Understanding Big Data, its challenges, and tools to handle large datasets.
  - 🛠️ Introduction to various Big Data frameworks and libraries.
- Conclusion:
  - This workshop offered a comprehensive overview of leveraging HPC

👏 Thanks for Joining! ❤️

Questions? Comments?

- cpeterson@oarc.ucla.edu
- Look at for more Hoffman2 workshops