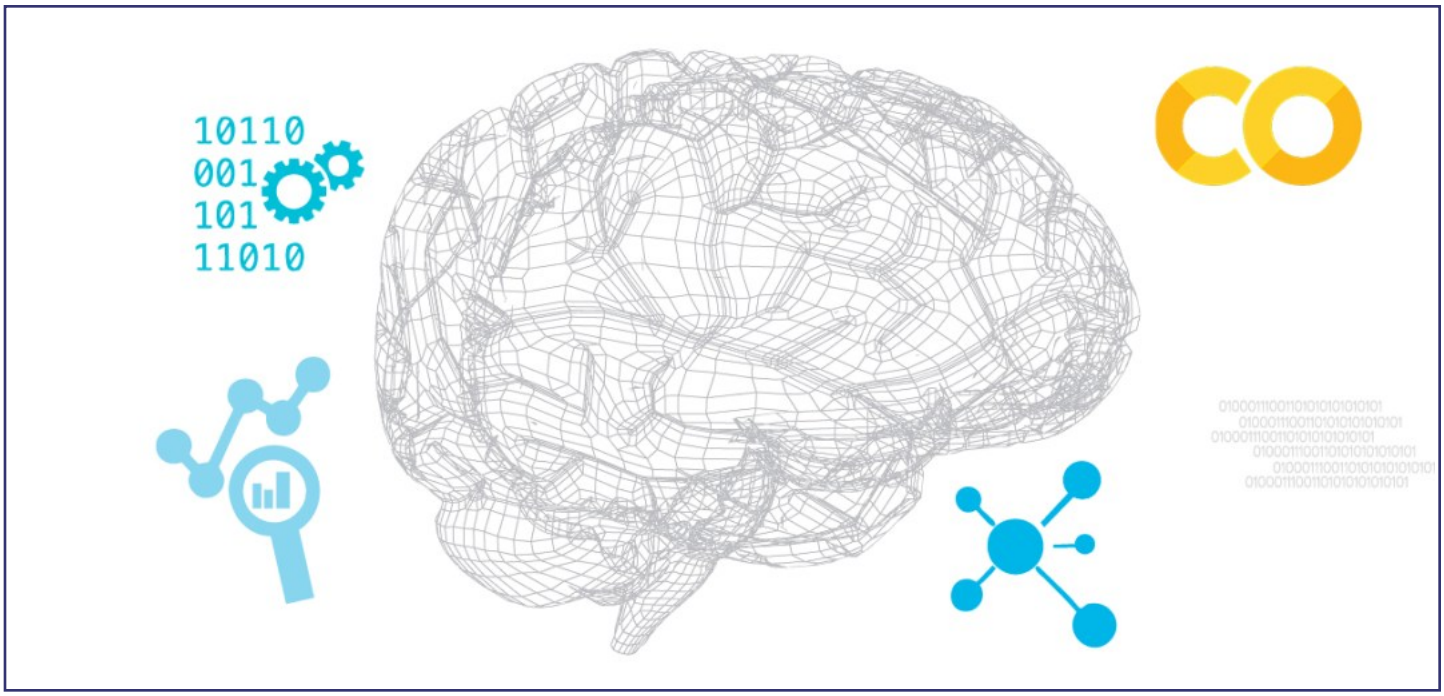
1/10 4:39:11 ***

← →

DM/ML tools



Theory => practice

In the two previous lectures, we looked at a variety of algorithms for DM and ML, eg. kNN, clustering, neural networks.

Now we'll examine how these have been/can be implemented - using tools/frameworks or APIs/languages/hardware.

Note that in 'industry' (ie. outside academia and gov't), tools/APIs... are heavily used to build RL products - so you **need** to be aware of, and be knowledgeable in, as many of them as possible.

APIs/frameworks: part 1

These are the most heavily used:

- TensorFlow ('TF')
- Spark MLlib: https://spark.apache.org/mllib/ and https://spark.apache.org/docs/2.2.0/ml-pipeline.html
- Keras: https://keras.io/ [a higher level lib, compared to TF etc]; here are all the types of Keras layers
- Torch, PyTorch: https://pytorch.org, http://torch.ch/
- scikit-learn: https://scikit-learn.org/stable/
- Caffe: https://caffe2.ai/, http://caffe.berkeleyvision.org/ [→ Caffe2 → PyTorch]
- Apache mxnet: https://mxnet.apache.org/ [multi-language APIs, GPU and cloud support...]
- CNTK: https://docs.microsoft.com/en-us/cognitive-toolkit/

TL;DR: simply learn Keras or PyTorch, and if necessary, TF.

APIs/frameworks: part 2

Upcoming/lesser-used/'internal'/specific:

- here is FBLearner Flow - Facebook's version of TensorFlow :)
- Apache Mahout - a collection of ML algorithms, in Java/Scala
- .NET ML: https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet
- fastai [on top of PyTorch]: https://github.com/fastai/fastai
- OpenVINO: https://software.intel.com/en-us/openvino-toolkit and https://www.youtube.com/watch?v=rUwayTZKnmA&t=1s [a tutorial]
- Turi: an alternative to Apple's CreateML: https://github.com/apple/turicreate
- LibSVM: https://www.csie.ntu.edu.tw/~cjlin/libsvm/
- LightGBM: https://github.com/Microsoft/LightGBM
- XGBoost: https://xgboost.ai/ [and, look at Tianqi's slides and talk]
- CatBoost: https://tech.yandex.com/catboost/
- Google - SEED: https://ai.googleblog.com/2020/03/massively-scaling-reinforcement.html
- Uber's 'Fiber', for distributed ML training: https://venturebeat.com/2020/03/26/uber-details-fiber-a-framework-for-distributed-ai-model-training/
- LOTS of smaller efforts: https://github.com/EthicalML/awesome-production-machine-learning

Here is an article about deep learning tools.

Cloud

The virtually unlimited computing power and storage that a cloud offers, make it an ideal platform for data-heavy and computation-heavy applications such as ML.

Amazon: https://aws.amazon.com/machine-learning/ Their latest offerings make it possible to 'plug in' data analysis anywhere.

Google: https://cloud.google.com/products/ai/ [in addition, Colab is an awesome resource!]

Microsoft: https://azure.microsoft.com/en-us/services/machine-learning-studio/ [and AutoML] [aside: alternatives to brute-force 'auto ML' include 'Neural Architecture Search' [incl. this], pruning, and better network design (eg using ODEs - see this).

IBM Cloud, Watson: https://www.ibm.com/cloud/ai [eg. look at https://www.ibm.com/cloud/watson-language-translator]

Others:

- h2o: https://www.h2o.ai/products/h2o/ [supports R, Python, Java, Scala, JSON, native Flow GUI [similar to Jupyter], REST...]
- BigML: https://bigml.com/features#platform
- FloydHub: https://www.floydhub.com/
- Paperspace: https://ml-showcase.paperspace.com/
- Algorithmia, eg. https://info.algorithmia.com/ and https://demos.algorithmia.com/

With so much available out of the box, it's time for citizen data scientists?

Pretrained ML models

A pre-trained model includes an architecture, and weights obtained by training the architecture on specific data (eg. flowers, typical objects in a room, etc) - ready to be deployed.

Eg. this is simple object detection in the browser! You can even run this detector on a command line.

TinyMOT: https://venturebeat.com/2020/04/08/researchers-open-source-state-of-the-art-object-tracking-ai

Apple's CreateML is useful for creating a pre-trained model, which can then be deployed (eg. as an iPad app) using the companion CoreML product. NNEF and ONNX are other formats, for NN interchange.

Pre-trained models in language processing, include Transformer-based BERT and GPT-2. Try this demo (of GPT etc). There is GPT-3 currently available, GPT-4 in the works, Wu Dao 2.0, MT-NLG...

There are also, combined (bimodal) models, based on language+image data.

Tools

Several end-to-end applications exist, for DM/ML. Here popular ones.

Weka is a Java-based collection of machine learning algorithms.

RapidMiner uses a dataflow ("blocks wiring") approach for building ML pipelines.

KNIME is another dataflow-based application.

TIBCO's 'Data Science' software is a similar (to WEKA etc) platform. Statistica [similar to Mathematica] is a flexible, powerful analytics software [with an old-fashioned UI].

bonsai is a newer platform.

To do ML at scale, a job scheduler such as from cnvrg.io can help.

SynapseML is a new ML library from Microsoft.

There are a variety of DATAFLOW ('connect the boxes') tools! This category is likely to become HUGE:

- Perceptilabs: https://www.perceptilabs.com/
- Lobe: https://insights.dice.com/2018/05/07/lobe-deep-learning-platform/
- https://www.producthunt.com/posts/datature
- smartpredict: https://smartpredict.ai/
- StackML: https://stackml.com/ [RIP]
- Baseet: https://baseet.ai/ [RIP]

Languages

These languages are popular, for building ML applications (the APIs we saw earlier, are good examples):

- Python
- R
- Julia [Python 'replacement'?!]
- Wolfram
- JavaScript - this is a good list of JS-based libraries [look at ConvnetJS for nice demos]
- Scala - a functional+OO language - here is a roundup of libraries [these are in addition to Spark's MLlib Scala API]
- Java - another robust language for building ML libs [we already saw WEKA] and apps
- Jupyter [an environment, not a language] (eg. here is a collection of ML notebooks - as an exercise, run them all in Colab!) [also, here are notebooks for 'everything'!]
- ...

Hardware

Because (supervised) ML is computationally intensive, and detection/inference needs to happen in real-time almost always, it makes sense to accelerate the calculations using hardware. Following are examples.

Google TPU: TF is in hardware! Google uses a specialized chip called a 'TPU', and documents TPUs' improved performance compared to GPUs. Here is a pop-sci writeup, and a Google blog post on it.

Amazon Inferentia: a chip, for accelerating inference (detection): https://aws.amazon.com/machine-learning/inferentia/

NVIDIA DGX-1: an 'ML supercomputer': https://www.nvidia.com/en-us/data-center/dgx-1/ [here is another writeup]

Intel's Movidius (VPU): https://www.movidius.com/ - on-device computer vision

In addition to chips and machines, there are also boards and devices:

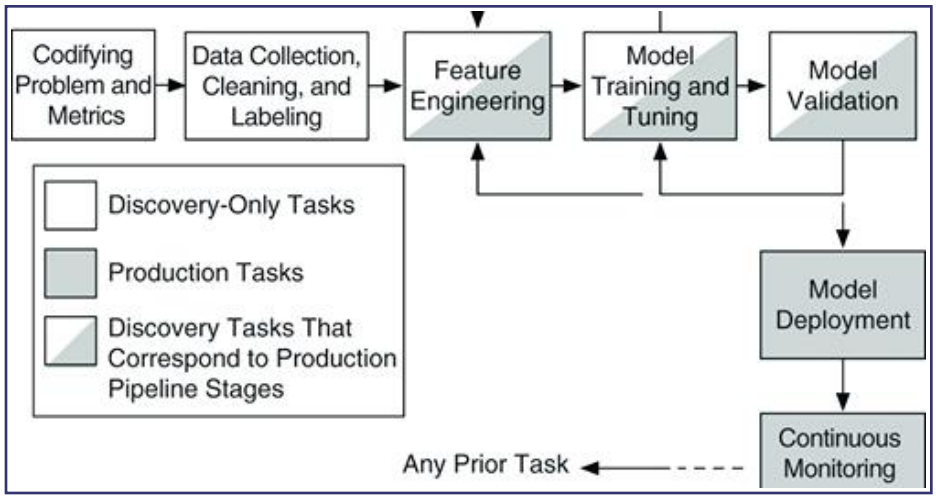
- Pixy2: https://pixycam.com/ - camera + ML in a single board
- Coral: https://coral.withgoogle.com/
- Jetson Nano: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/
- Movidius NCS: https://software.intel.com/en-us/movidius-ncs
- ...

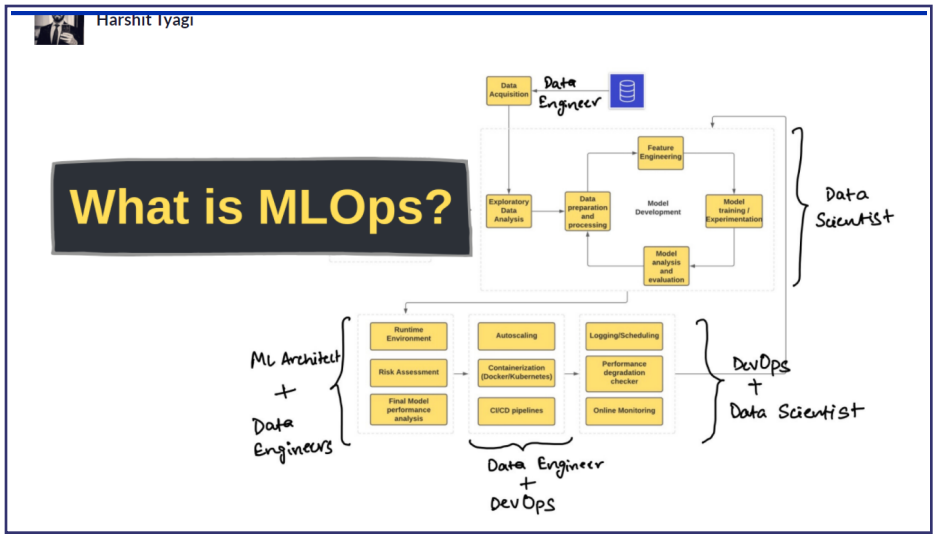
Overall, there's an explosion/resurgence in 'chip design', for accelerating AI training, inference. In April '21, NVIDIA announced its new A30 and A10 GPUs, at the annual [GTC] conference.

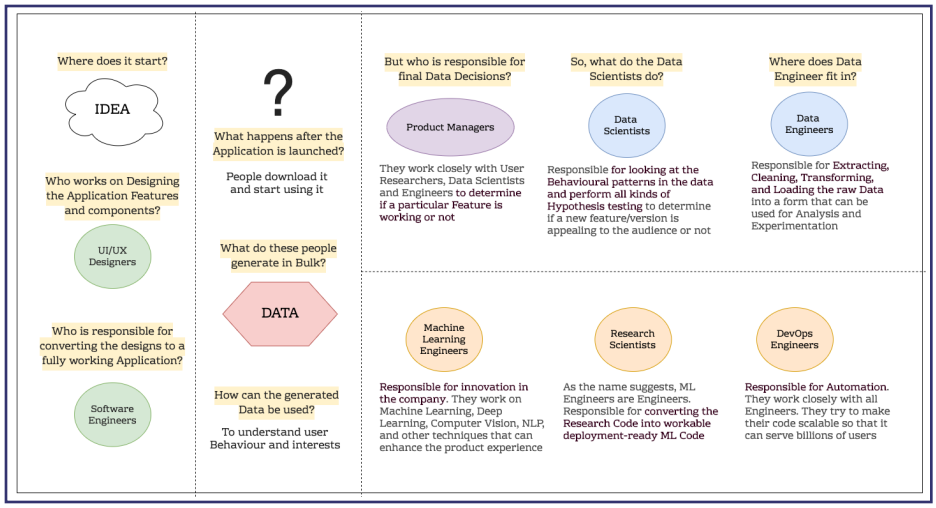
Summary

We looked at a plethora of ways to 'do' ML. Pick a few, and master them - they complement your coursework-based (theoretical) knowledge, and, make you marketable to employers! Aside: LOTS of salaries etc., revealed here :)

Also, FYI - in industry (G-MAFIA/FAANG/MAMMA MIA, BAT, more!), ML is part of a bigger 'production pipeline':







So, how do you prepare for the Data Roles?

- **SQL ! SQL ! SQL !** Practice a lot of SQL
- Be a savvy **Python** Developer
- **Think like a Product Manager**. Take up your favorite Application, and think of KPI (Key Performance Indicators). Determine the criteria for Decision Making
- **Teamwork and Collaboration** are essential skills needed for any Data Role. Be a good Communicator. Whether it be an interview or a Team Meeting, make sure to speak your mind
- Learn different **Visualization Techniques** and present your findings in the best way possible. Make it impressive
- Study **Datawarehousing concepts** for Data Engineering Roles
- Have a **basic understanding** of Data pipelines, MapReduce Concepts, Graph Models, Data Analytics platforms, Database Concepts, Kubernetes, Containers, and various open-source Apache Products. (Depth Knowledge is not needed – But, basic information will help you understand the bigger picture)