Machine Learning with Python (+ HPC)

Fei Mao

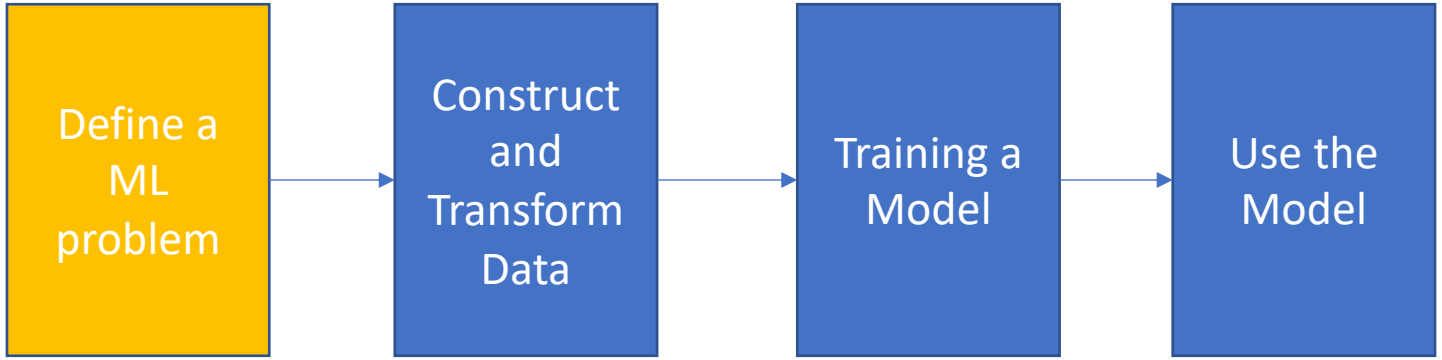
SOSCIP/SciNet

Audience Background

- Computer Science?
- Engineering?
- Natural Science?
- Social Science?
- Python Programming?

(Not) just another intro to ML course

1. Machine Learning Workflow

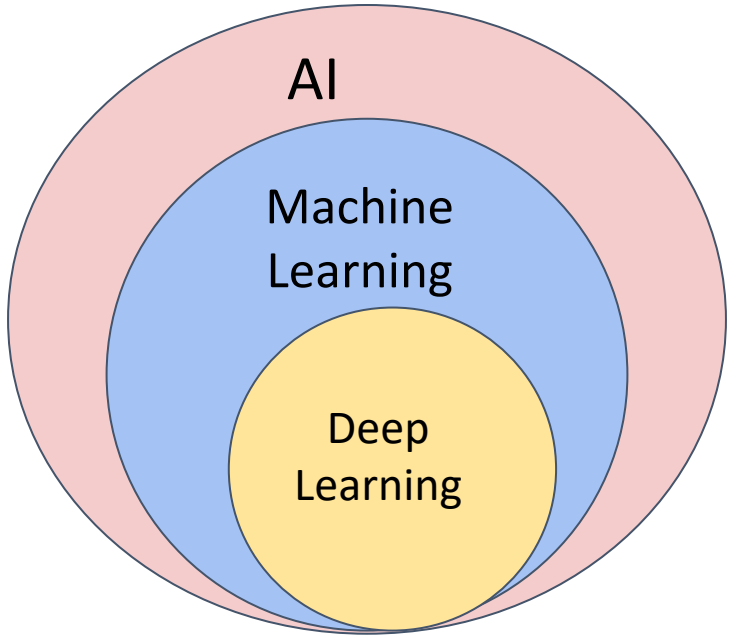


2. Machine Learning with Python

(3. Machine Learning on HPC)

Problem Framing

- What is Machine Learning?
  - In short, Training a "model" on existed data to "predict" unseen data
- What do people mean when they say: AI, Machine Learning and Deep Learning?
  - AI > ML > DL



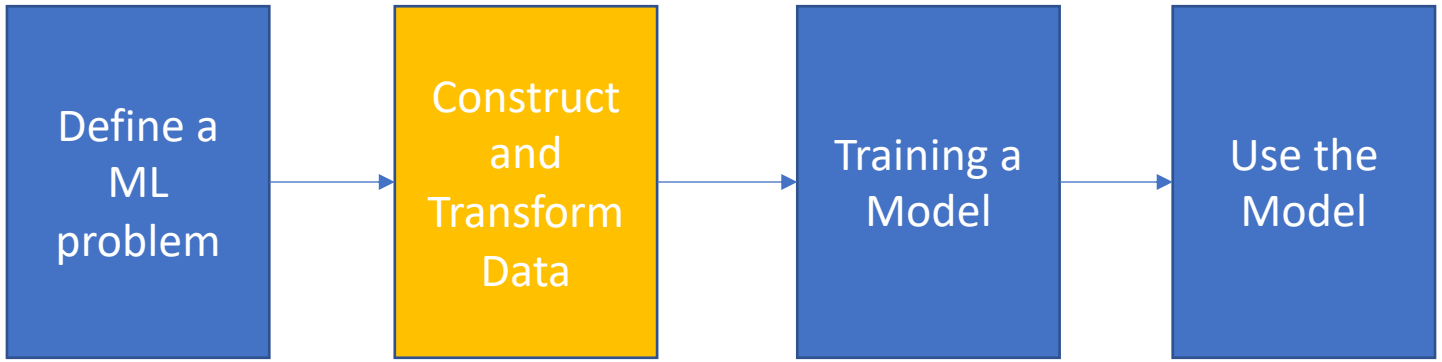
Types of ML Problems

- Classification
  - Pick one of N labels
- Regression
  - Predict numerical values
- Clustering
  - Group similar examples
- Others: Structured output, Recommendation, Ranking, etc.

Types of approaches

- Supervised Learning
  - model is provided with labeled training data
- Unsupervised Learning
  - to identify meaningful patterns in the unlabeled data
- Reinforcement Learning
  - no examples with labels, agent learn under the rule

Data Preparation and Feature Engineering



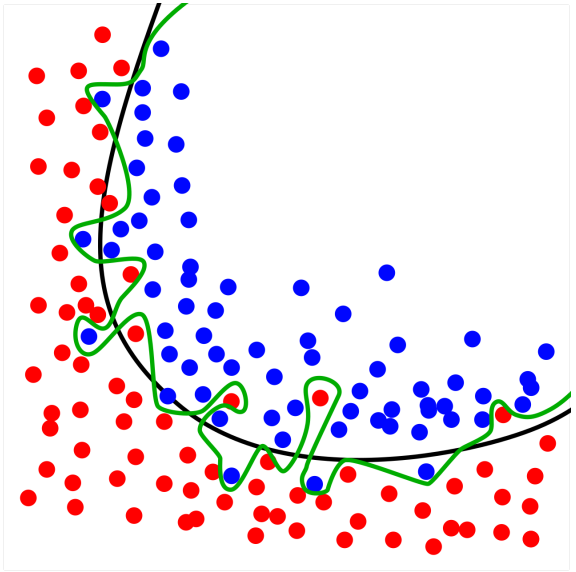
Data

- Any type: characters, images, videos, … , combinations
- Examples:
  - one example is a particular instance of data
- Features:
  - measurable variables to represent the data
  - e.g. properties of a example, image pixels, extracted higher level features
- Labels:
  - the "thing" we're predicting
  - for supervised machine learning

Collecting Data

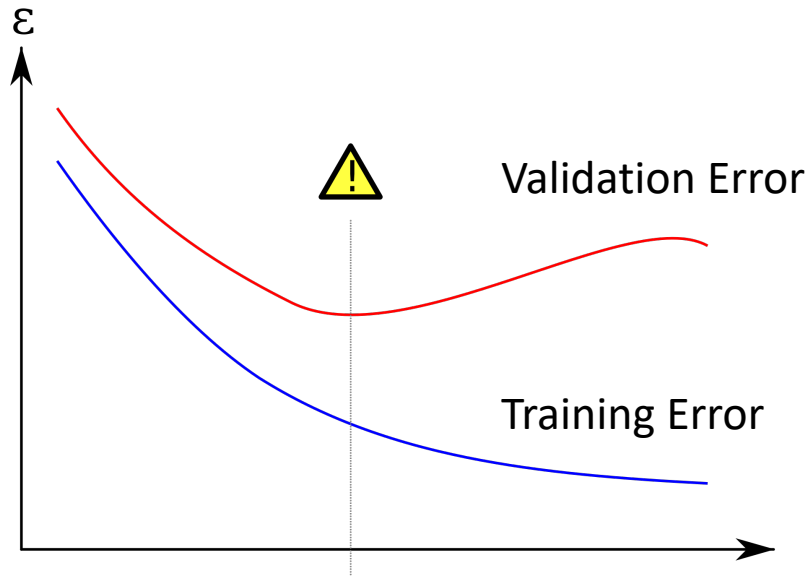
- Size of a Data Set
  - How big is enough?
    - Iris flower data set: 150
    - Heart Disease Data Set: 303
    - MNIST handwritten digit database: 60,000
    - ImageNet: 1,200,000
    - Google Gmail SmartReply: 238,000,000
    - Google Translate: trillions
- Quality of a Data Set:
    - Label errors
    - Features noisy/errors
    - Missing values
    - Duplicate examples

Splitting Data



- Training Data
- Testing Data
  - For the final evaluation
- Validation Data
  - Measuring the model during training and discovering overfitting
- Randomly pickup is not always good
- For small dataset, use cross-validation
  - *k*-fold cross-validation

**Don't use Test/Val data to train the model



Feature engineering

- Data Transformation
  - Mandatory transformations
    - Converting non-numeric features into numeric
    - Resizing inputs to a fixed size
    - …
  - Optional quality transformations
    - Normalized numeric features (most models perform better afterwards)
    - Tokenization or lower-casing of text features
    - …

Transforming Numeric Data

- Normalizing
  - Why normalization?
    - transform features to be on a similar scale
    - improves the performance and training stability
  - Linear Scaling:
    - usually scale to 0 – 1
    - good choice data is approximately uniformly distributed
    - A good example is age, bad example is income
  - Feature Clipping:
    - data set contains extreme outliers
    - caps all feature values above (or below) a certain value to fixed value
  - Log Scaling
    - computes the log of your values to compress a wide range to a narrow range
    - help to improve linear model performance
  - Z-Score
    - mean = 0 and std = 1

Transforming Categorical Data

- Mapping categorical values
  - e.g. map street names to numbers
  - constraints:
    - learning a single weight that applies to all streets
    - street names may take multiple values
- One-hot encoding/multi-hot encoding
  - A vector: set corresponding elements to 1, set all other elements to 0

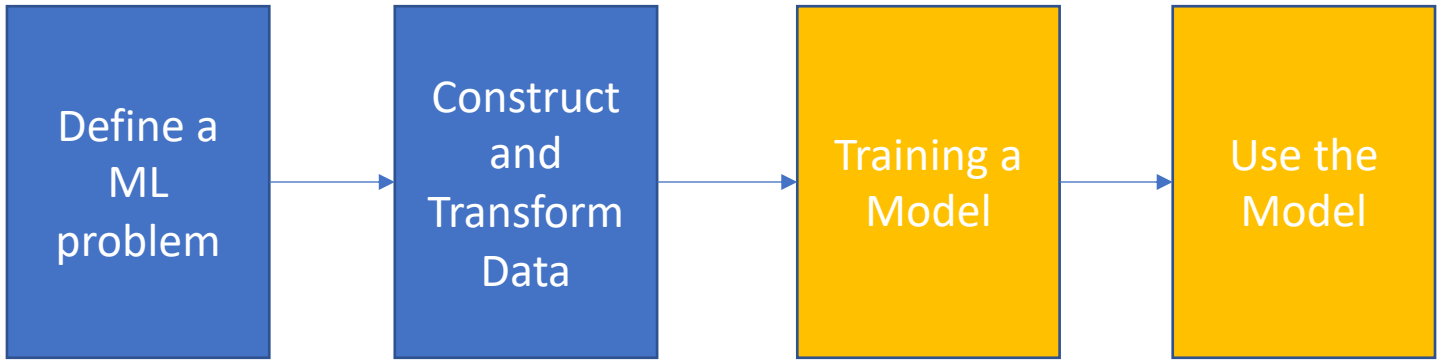
Python Practice 1

- Login to "teach" cluster and cp the materials:
  - ssh –Y teach.scinet.utoronto.ca
  - cp -r /scinet/course/ss2019/2/5_machinelearning $SCRATCH/
  - cd $SCRATCH/5_machinelearning
- Use Anaconda3
  - module load anaconda3
  - Default environment includes: numpy/scipy, scikit-learn, pandas, seaborn, etc
- Install additional packages:
  - pip install --user <package_name>
  - e.g. pip install --user xgboost

Python Practice 1

- Run python script:
  - srun -n 1 python iris.py
- Interactive ipython:
  - debugjob
  - ipython
- If network connection is lost, please kill all previous interactive jobs before asking for a new one:
  - scancel -u <user_name>

Training and Inferencing models



Training and Inferencing models

- Training the model:
  - How to determine a good model?
  - Loss function (cost function):
    - **loss** is a number indicating how bad the model's prediction was on a single example
    - popular loss function:
      - Mean square error (MSE)
        - $\frac{1}{N}\sum\left(y - prediction(x)\right)^2$
      - Cross-entropy loss (log loss)
        - $-\sum_{c=1}^{M} y\ log(P)$
  - Training is to find a model that minimizes loss
- Inferencing:
  - select single or multi models

Training and Inferencing models

- Common Machine Learning Algorithms:
  - kNN
  - Dimensionality Reduction
    - PCA, t-SNE, etc
  - SVM
  - NN
  - Decision Tree, Random Forest, Gradient Boosting
  - k-means clustering
  - …
  - …

Python Practice 2

- Full examples for Iris dataset: iris.py
  - srun -n 1 python iris.py
- Heart Disease Dataset:
  - https://www.kaggle.com/ronitf/heart-disease-uci
  - go to kaggle.com and search "heart" to check dataset details
  - train any model w/ or w/o feature engineering

Machine Learning on HPC

- Pick up the right library:
  - sklearn in Anaconda vs. sklearn in Intel Python
  - TensorFlow w/ MKL-DNN vs. TensorFlow w/ Eigen
- Pick up the right hardware:
  - CPU vs. GPU
- Distributed Training:
  - Model Parallelism vs. Data Parallelism
- I/O:
  - Remote Parallel Filesystem vs. Local SSD