10-605/805 – ML for Large Datasets
Lecture 1: Course Overview

Henry Chai

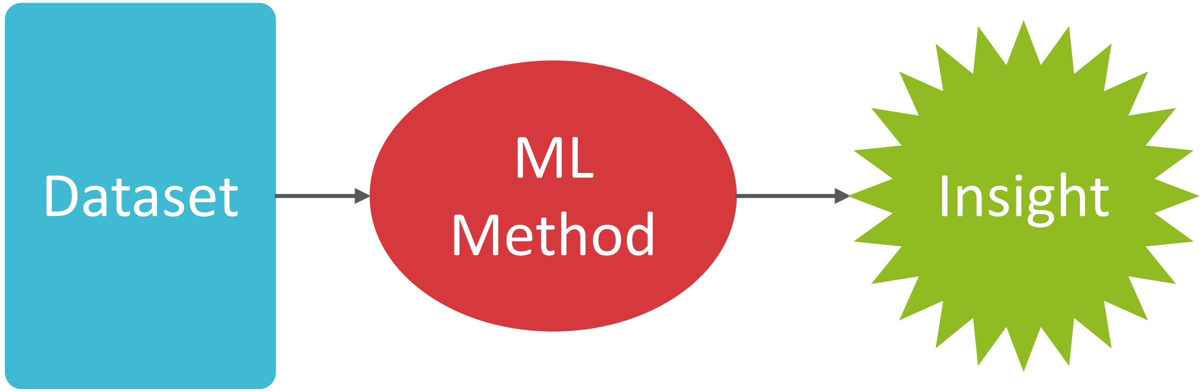
8/30/22

Machine Learning

- Premise:
  - There exists some pattern/behavior of interest
  - The pattern/behavior is difficult to describe
  - There is data
  - Use data to “learn” the pattern

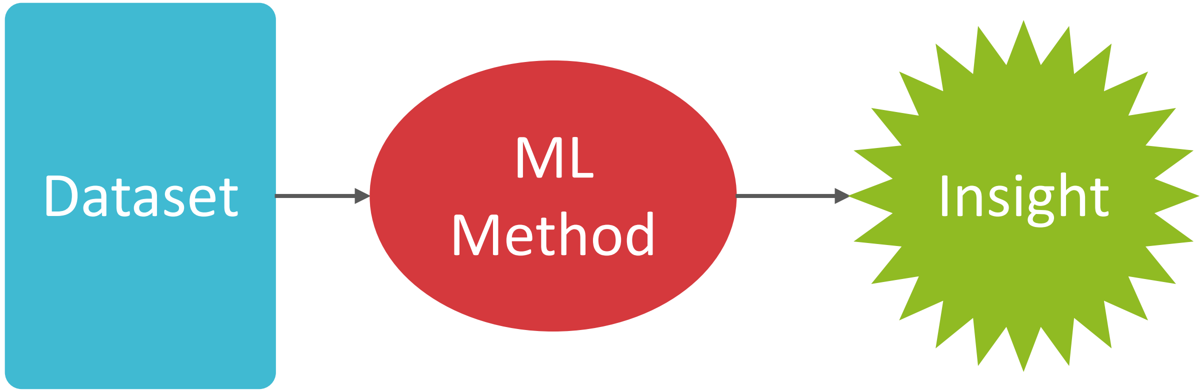
- Definition:
  - A computer program **learns** if its *performance*, *P*, at some *task*, *T*, improves with *experience*, *E*.

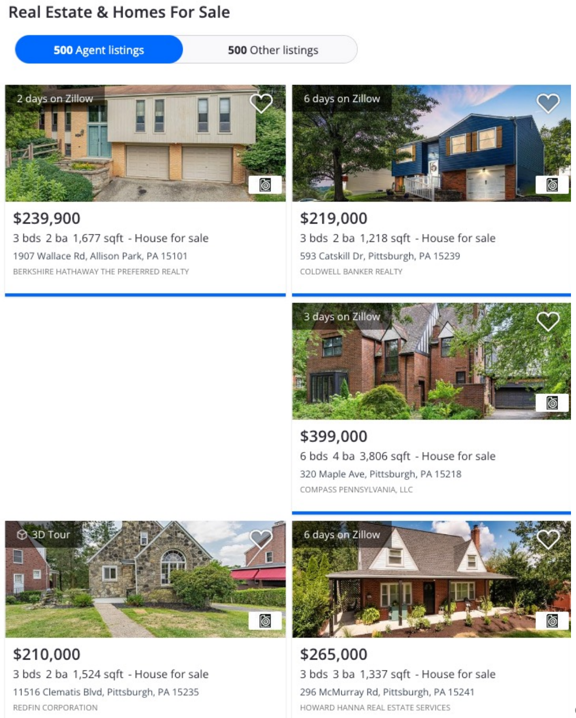
Machine Learning: Pipeline

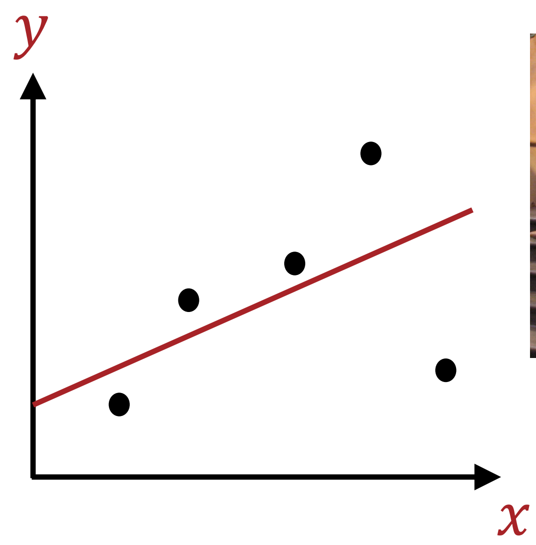


Machine Learning: Example

Regression







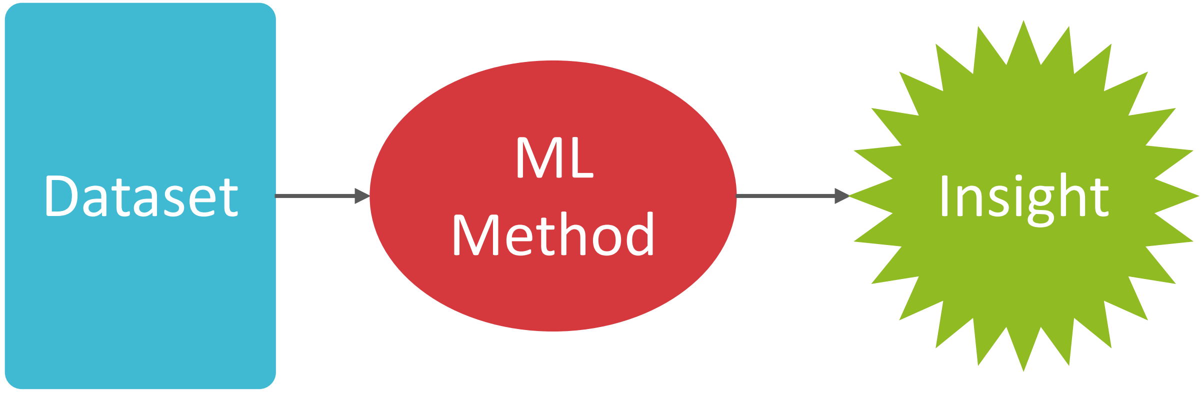
=

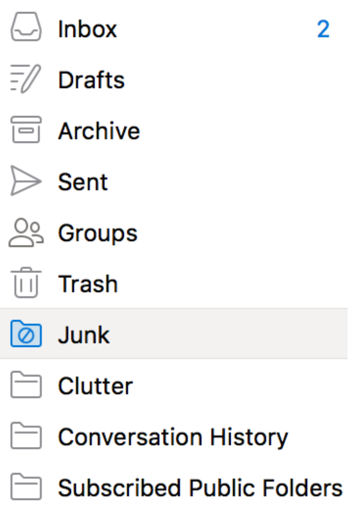
$1,000,000

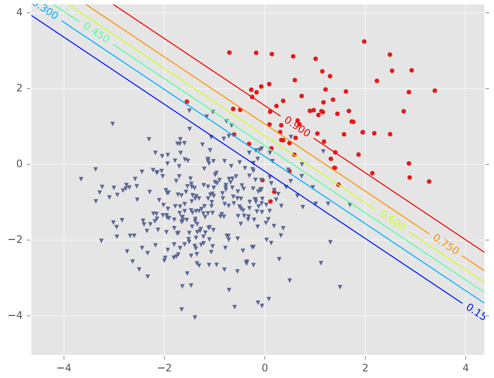
Source: https://muppet.fandom.com/wiki/Oscar's_trash_can?file=IMG_0815.jpg

Machine Learning: Example

Classification







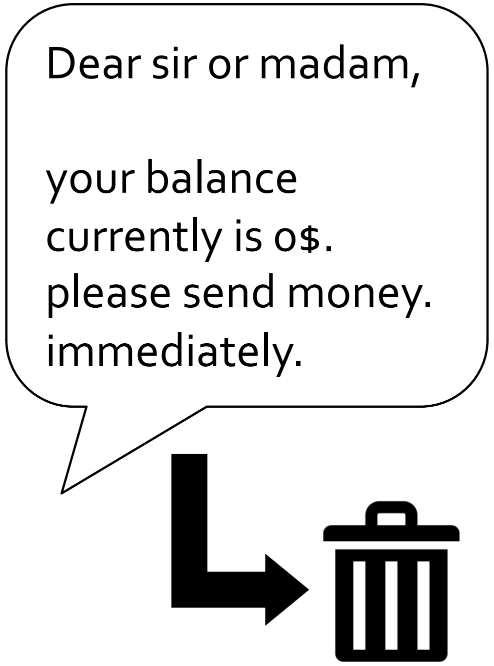
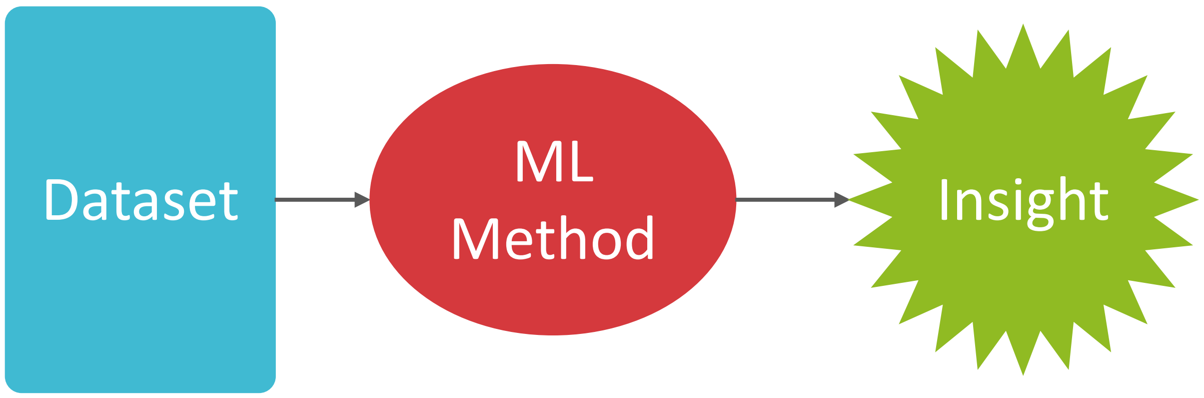


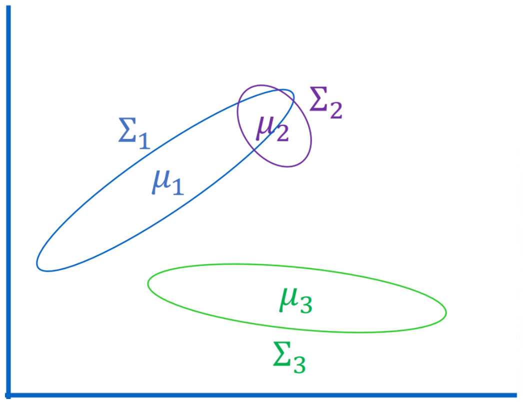
Figure courtesy of Matt Gormley

Machine Learning: Example

Clustering

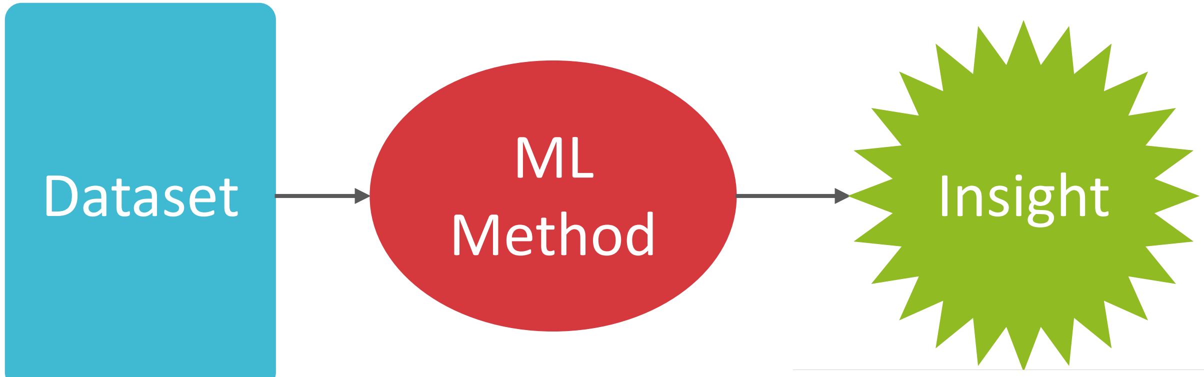


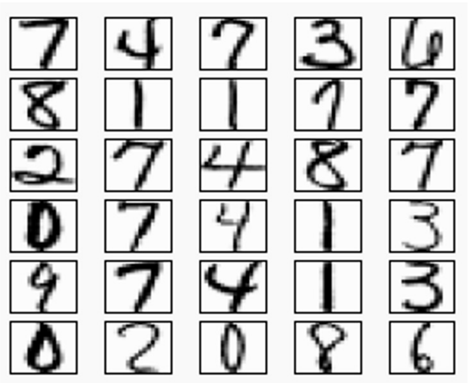
CustomerID	Purchases		
1			
2			
⋮			

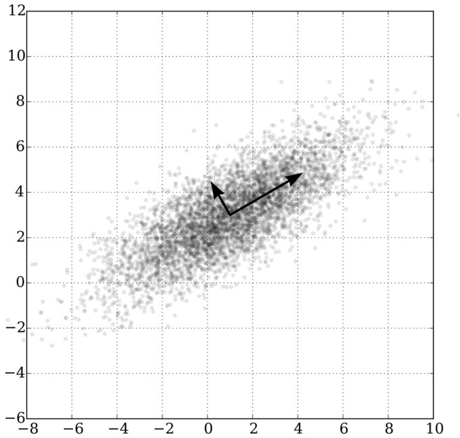


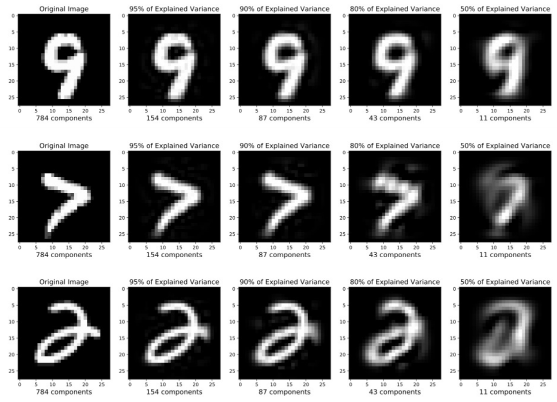
Machine Learning: Example

Dimensionality Reduction









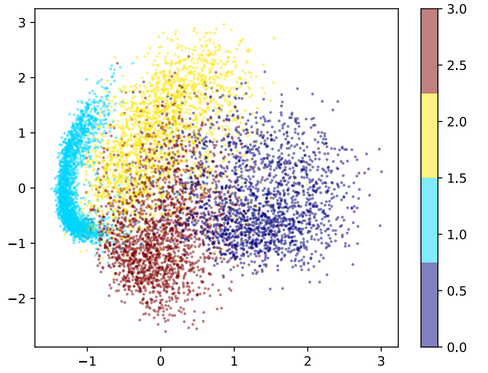


Figure courtesy of Matt Gormley

Machine Learning: Terminology

- Datasets will (usually) consist of
  - Observations – individual entries used in learning or evaluating a learned model
  - Features – attributes used to represent an observation during learning
  - Labels – values or categories associated with an observation

Machine Learning: Pipeline

- Running Example: Sentiment analysis of course evaluations
- training dataset

CHAI	Easy course, taught well
CHAI	homework takes way too long
CHAI	See above.
CHAI	Great
CHAI	Too much work
CHAI	This course had a lot of problems but none of them were Henrys fault.

Machine Learning: Pipeline

- Running Example: Sentiment analysis of course evaluations
- Raw training dataset
  - Data preprocessing

Features		Labels
CHAI	Easy course, taught well	🙂
CHAI	homework takes way too long	🙁
~~CHAI~~	~~See above.~~	
CHAI	Great	🙂
CHAI	Too much work	🙁
CHAI	This course had a lot of problems but none of them were Henrys fault.	🙃

Machine Learning: Pipeline

- Running Example: Sentiment analysis of course evaluations
- Raw training dataset
  - Data preprocessing

Features	Labels
easy course taught well	$+1$
homework takes way too long	-1
great	$+1$
too much work	-1
this course had lot problems but none them were henry fault	0

Machine Learning: Pipeline

- Running Example: Sentiment analysis of course evaluations
- Training dataset
  - Feature engineering - transform observations into a form appropriate for the machine learning method
  - Example: bag of words model

easy course taught well

homework takes way too long

great

too much work

this course had lot problems but none them were henry fault

Vocabulary
but
course
easy
fault
great
henry
homework
long
⋮

Machine Learning: Pipeline

- Running Example: Sentiment analysis of course evaluations
- Training dataset
  - Feature engineering - transform observations into a form appropriate for the machine learning method
  - Example: bag of words model

this course had lot problems but none them were henry fault

Vocabulary	
but	1
course	1
easy	0
fault	1
great	0
henry	1
homework	0
long	0
⋮	⋮

Machine Learning: Pipeline

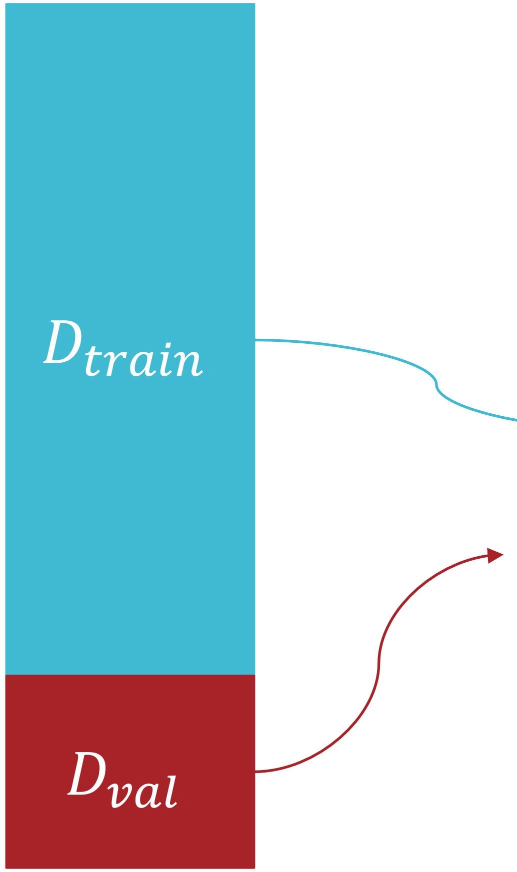
- Running Example: Sentiment analysis of course evaluations
- Model training
  - Just throw a ~~narwhal~~ neural network at it?

Machine Learning: Pipeline

- Running Example: Sentiment analysis of course evaluations
- Model training
  - Hyperparameter tuning – most machine learning/optimization methods will have values/design choices that need to be specified/made in order to run
  - Example: neural networks trained using mini-batch gradient descent
    - architecture
    - batch size
    - learning rate/step size
    - termination criteria
    - etc...

Machine Learning: Pipeline

- Running Example: Sentiment analysis of course evaluations
- Model training



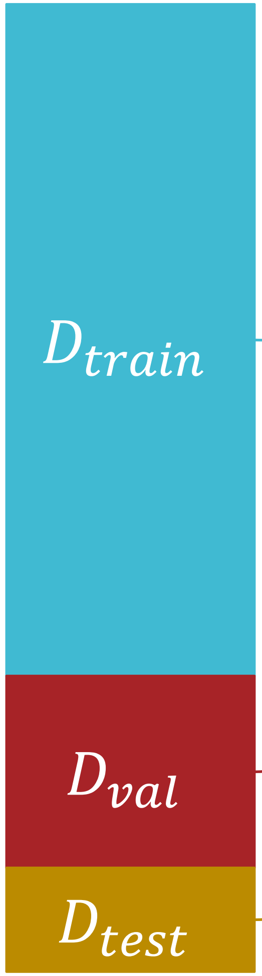
- Suppose we want to compare multiple hyperparameter settings $\theta_1, \ldots, \theta_K$
- For $k = 1, 2, \ldots, K$
  - Train a model on D_{train} using θ_k
- Evaluate each model on D_{val} and find the best hyperparameter setting, θ_{k^*}

Machine Learning: Pipeline

- Running Example: Sentiment analysis of course evaluations
- Model evaluation
  - How do you know if you've learned a good model?
  - If a model is trained by minimizing the training error, then the training error at termination is (typically) overly optimistic about the model's performance
    - The model has been *overfit* to training data
  - Likewise, the validation error is also (typically) optimistic about the model's performance
    - Usually less so than the training error
  - Idea: use a held-out *test* dataset to assess our model's ability to generalize to unseen observations

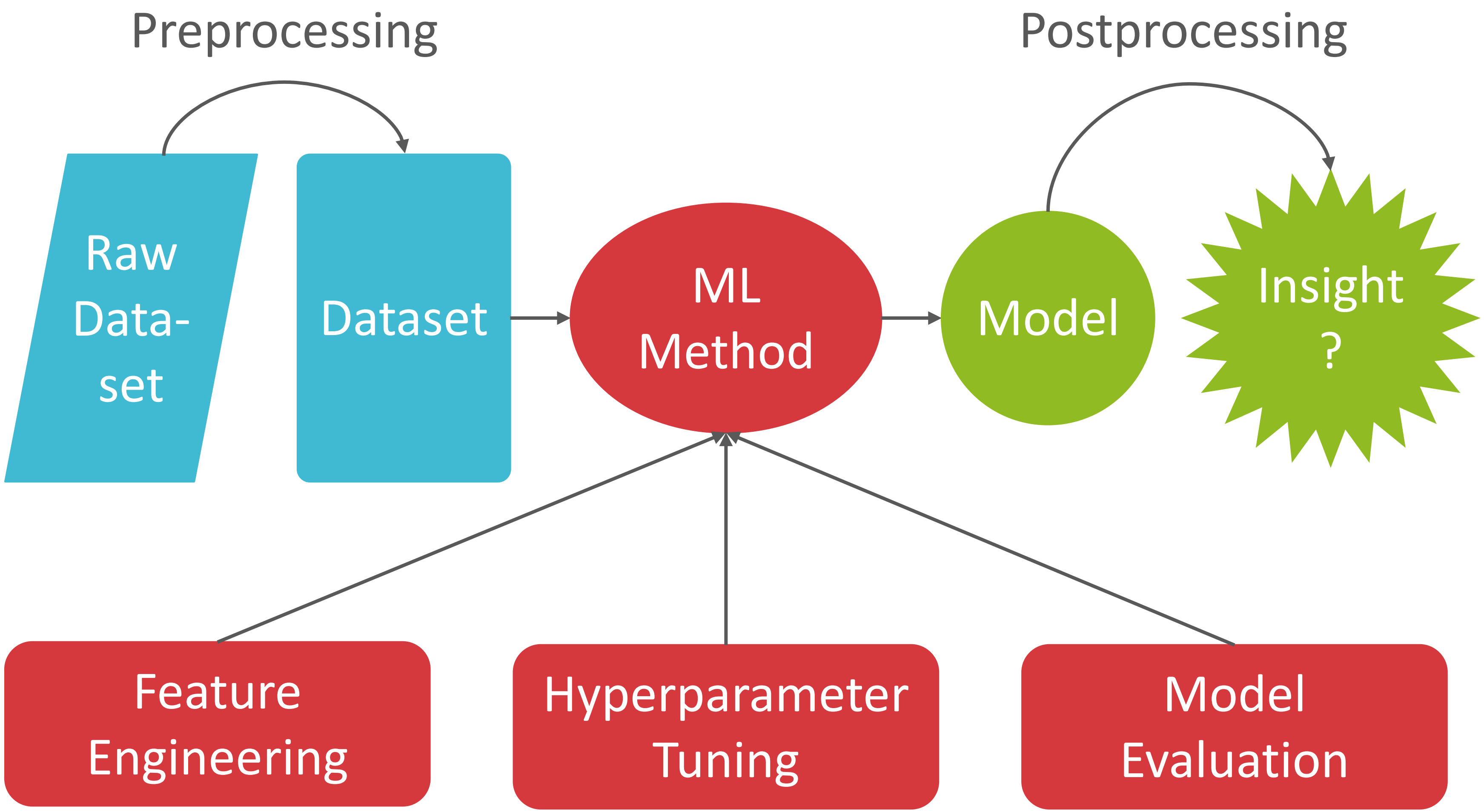
Machine Learning: Pipeline

- Running Example: Sentiment analysis of course evaluations
- Model evaluation



- Suppose we want to compare multiple hyperparameter settings $\theta_1, \ldots, \theta_K$
- For $k = 1, 2, \ldots, K$
  - Train a model on D_{train} using θ_k
- Evaluate each model on D_{val} and find the best hyperparameter setting, θ_{k^*}
- Compute the error of a model trained with θ_{k^*} on D_{test}

Machine Learning: Pipeline Revisited



Machine Learning: Challenges

- Contemporary issues in modern machine learning:
    - Privacy
    - Fairness
    - Interpretability
    - Big data

Machine Learning: Challenges

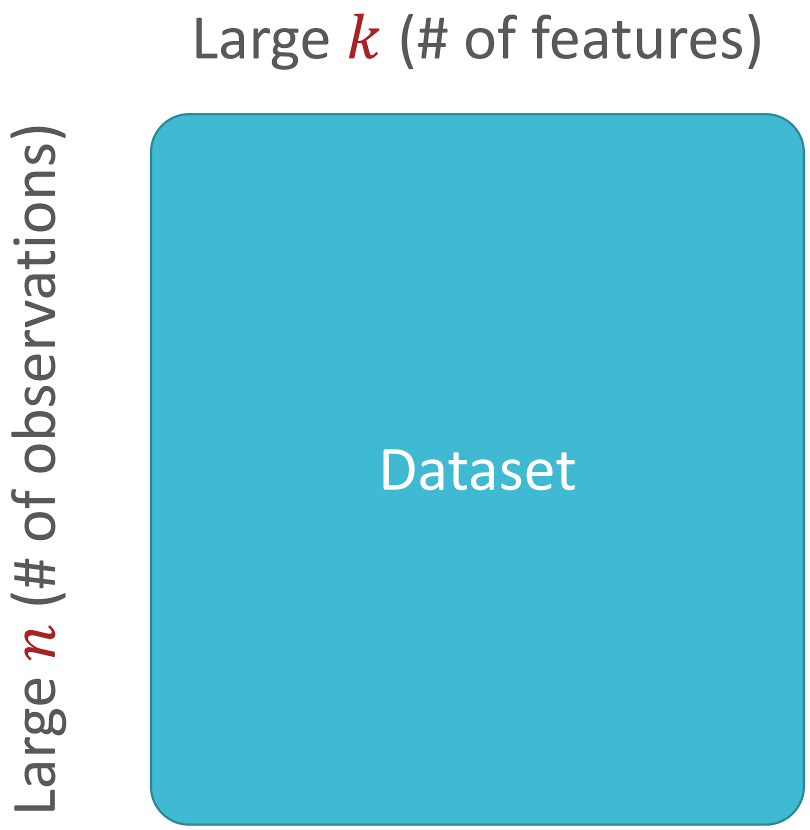
- Contemporary issues in modern machine learning:
  - Privacy
  - Fairness
  - Interpretability
  - **Big data**

Machine Learning with Large Datasets

- Premise:
  - There exists some pattern/behavior of interest
  - The pattern/behavior is difficult to describe
  - There is data (sometimes a lot of it!)
  - More data *usually* helps
  - Use data efficiently/intelligently to "learn" the pattern
- Definition:
  - A computer program **learns** if its *performance*, *P*, at some *task*, *T*, improves with *experience*, *E*.

Large Datasets

- Datasets can be big in two ways



Large Datasets: Example

Large Datasets: Example

- Image processing
  - Large n: potentially massive number of observations (e.g., pictures on the internet)
  - Use-cases: object recognition, annotation generation
- Medical data
  - Large k: potentially massive feature set (e.g., genome sequence, electronic medical records, etc…)
  - Use-cases: personalized medicine, diagnosis prediction
- Business analytics
  - Large n (e.g., all customers & all products) and k (e.g., customer data, product specifications, transaction records, etc…)
  - Use-cases: product recommendations, customer segmentation

Tons of Features

- High-dimensional datasets present numerous issues:
  - Curse of dimensionality
  - Overfitting
  - Computational issues
- Strategies:
  - Learn low-dimensional representations
  - Perform feature selection to eliminate “low-yield” features

Tons of Observations

- Typically, we consider exponential time complexity (e.g., $O(2^n)$) bad and polynomial complexity (e.g., $O(n^3)$) good
- However, if n is massive, then even $O(n)$ can be problematic!
- Strategies:
  - Speed up processing e.g., stochastic gradient descent vs. gradient descent
  - Make approximations/subsample the dataset
  - Exploit parallelism

Tons of Observations

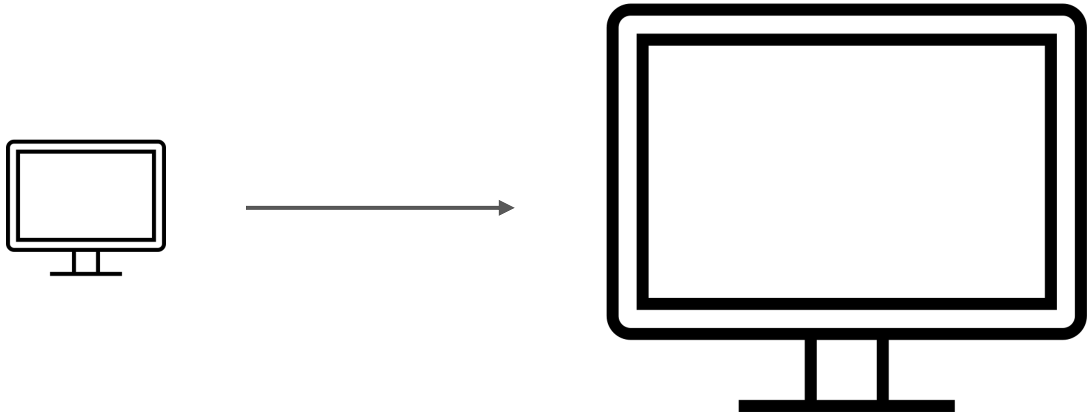
- Typically, we consider exponential time complexity (e.g., $O(2^n)$) bad and polynomial complexity (e.g., $O(n^3)$) good
- However, if n is massive, then even $O(n)$ can be problematic!
- Strategies:
  - Speed up processing e.g., stochastic gradient descent vs. gradient descent
  - Make approximations/subsample the dataset
  - **Exploit parallelism**

Parallel Computing

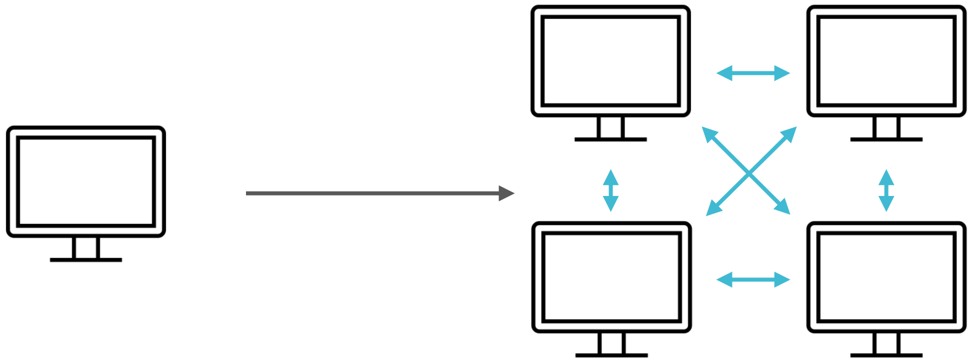
- Multi-core processing – scale up one big machine
- Distributed processing – scale out many machines

Parallel Computing

- Multi-core processing – scale up one big machine
  - Data can fit on one machine
  - Usually requires high-end, specialized hardware
  - Simpler algorithms that don't necessarily scale well



Parallel Computing



- Distributed processing – scale out many machines
  - Data stored across multiple machines
  - Scales to massive problems on standard hardware
  - Added complexity of network communication

Apache Spark

- Open-source engine for parallel computing/large-scale data processing
- Lots of convenient features for machine learning specifically
  - Fast iterative procedures
  - Efficient communication primitives
  - Interactive IPython-style notebooks (Databricks)

Course Overview

- Data preprocessing
  - Cleaning
  - Summarizing/visualizing
  - Dimensionality reduction
- Model training
  - Distributed machine learning
  - Large-scale optimization
  - Scalable deep learning
  - Efficient data structures
  - Hyperparameter tuning
- Inference
  - Hardware for ML
  - Low-latency inference (Compression, Pruning, Distillation)
- Infrastructure/Frameworks
  - Apache Spark
  - TensorFlow
  - AWS/Google Cloud/Azure
- Advanced Topics
  - Federated Learning
  - Neural architecture search
  - Machine learning in practice

“Front” Matter

- HW1 released 8/30 (today!), due 9/14 at 11:59 PM
    - All HWs consist of two parts: written and programming
    - **For HW1 only, the programming part is optional (but strongly encouraged)**
    - The written part is nominally about PCA but can be solved using pre-requisite knowledge (linear algebra)
- Recitations on Friday, 11:50 – 1:10 (**different from lecture**) in GHC 4401 (**same as lecture**)
    - Recitation 1 on 9/2: Introduction to PySpark/Databricks
    - Recitation 2 on 9/9: Review of linear algebra

Course Logistics

- Course website: https://10605.github.io/

Course Components

The requirements of this course consist of participating in lectures, homework assignments, a mini-project and two exams. The grading breakdown is the following:

- 25% Exam 1
- 25% Exam 2
- 36% Homework (5 Assignments HW1 4%, other homework 8%)
- 14% Mini-Project

Course Logistics

- Course website: https://10605.github.io/

Exams

You are required to attend all in person exams. The exams will be given during class. Please plan your travel accordingly as we will not be able accommodate individual travel needs (e.g. by offering the exam early).

If you have an unavoidable conflict with an exam (e.g. an exam in another course), notify us by filling out the exam conflict form which will be released on Piazza a few weeks before the exam.

- **Exam 1: 10/11**
- **Exam 2: 12/8**

Course Logistics

- Course website: https://10605.github.io/

Late Homework Policy

You receive 4 total grace days for use on any homework assignment. We will automatically keep a tally of these grace days for you; they will be applied greedily. No assignment will be accepted more than 2 days after the deadline without written permission from Daniel, or the Professors. You may not use more than 2 grace days on any single assignment.

All homework submissions are electronic. As such, lateness will be determined by the latest timestamp of any part of your submission. For example, suppose the homework requires submissions to both Gradescope Written and Programming– if you submit your Written on time but your Programming 1 minute late, your entire homework will be penalized for the full 24-hour period.

Course Logistics

- Course website: https://10605.github.io/
- Mini-project:
  - Complete in groups of 2-3
  - Two pre-specified options:
    - Groups with only 10-605 students will pick one to complete
    - Groups with any 10-805 students must complete both
    - No late days may be used on project deliverables
  - More details about project options and deliverables will be announced later in the semester

Course Technologies

- Piazza for Q&A / announcements
- Gradescope for assignment submissions
- Canvas for hosting recordings and gradebook
- Google calendar for lecture, recitation and OH schedule

Course Staff


Akshath Jain
OH: TBA


Kunal Dhawan
OH: TBA


Nikhil Gupta
OH: TBA


Ramya Ramanathan
OH: TBA


Ruben John Mampilli
OH: TBA


Mehak Malik
OH: TBA


Utsav Dutta
OH: TBA

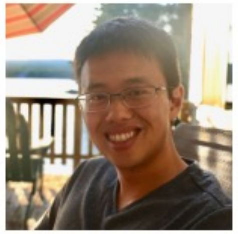

Preksha Patel
OH: TBA


Cristian Challu
OH: TBA

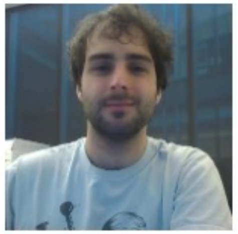

Rahul Dharani
OH: TBA

TAs


Ameet Talwalkar


Henry Chai

Instructors


Daniel Bird

EA