Foundations of AI and Machine Learning

Alex Olson

UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING
Centre for Analytics and Artificial Intelligence Engineering

Welcome!

- My name is Alex Olson
- Senior Research Associate at CARTE
- Bachelor's in AI from the University of Edinburgh
- Master's in AI from UofT in collaboration with the School of Cities
- Published papers in collaboration with a wide array of disciplines
- Work closely with students and faculty on all types of AI

Wi-Fi Access

- Eduroam Visitor Access (EVA)
- *Permanent* username and password
- Must be re-authorized each day you visit UofT
- New code each day, same phone number

Text `52utoronto`

To `(833) 338-7626`

Key AI Technologies

- **Time Series Analysis**
  - Enables systems to predict sequential data
  - Useful for demand forecasting
- **Natural Language Processing (NLP)**
  - Helps machines understand human language
  - Applications in chatbots, sentiment analysis
- **Computer Vision**
  - Allows machines to interpret visual data
  - Used in facial recognition, autonomous vehicles

AI Maturity Model

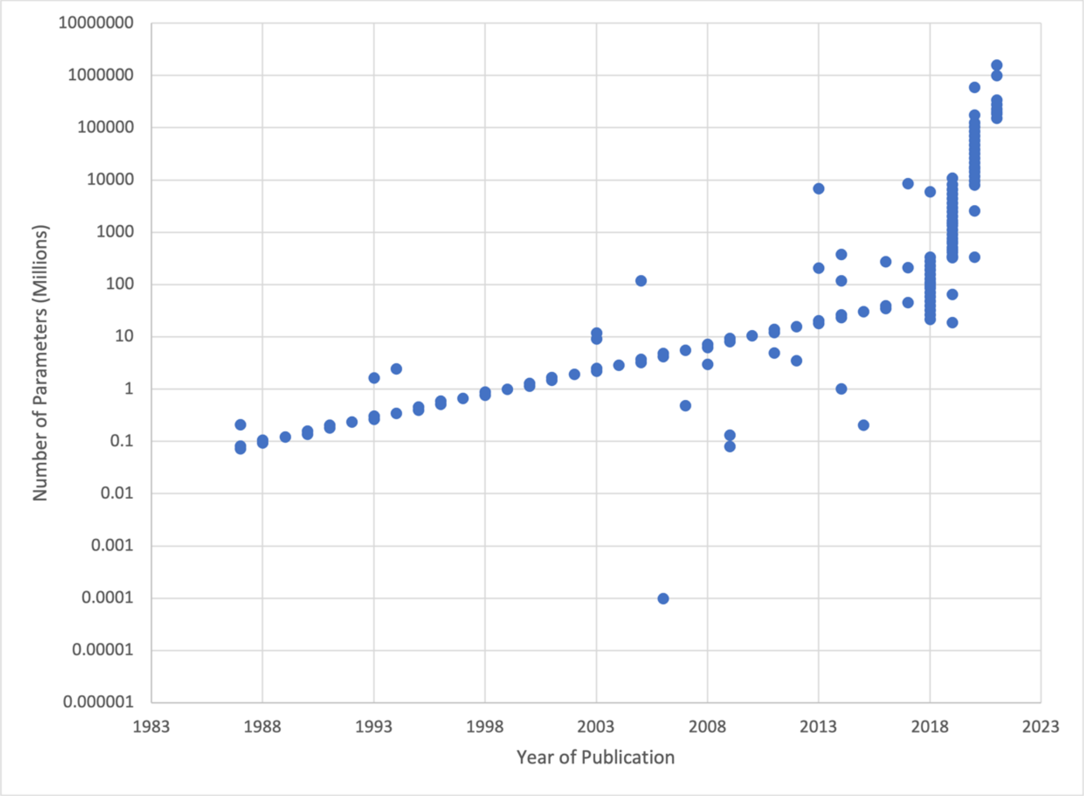
Reactive	Problem-solving focused Limited data utilization
Organized	Centralized data management Initial AI projects
Integrated	AI embedded in multiple business functions Advanced analytics capabilities
Transformative	AI at the core of business strategy Continuous innovation and adaptation

Impact on industries

- Healthcare
  - Drug discovery: Insilico Medicine found new treatments for fibrosis using AI in just 21 days
- Finance
  - Fraud detection: a global bank reduced fraudulent transactions by 50% using AI
- Manufacturing
  - Quality control: Noodle.ai collaborated with a steel mill to deploy an AI application for quality control, reducing suboptimal coil production from 50% to less than 1%

Why now?

- **Data Availability**
  - Explosion of Big Data
  - Improved Data Storage and Management
- **Computational Power**
  - Advances in GPU Technology
  - Cloud Computing Resources
- **Advanced Algorithms**
  - Breakthroughs in Machine Learning Models
  - Accessibility of Pre-trained Models



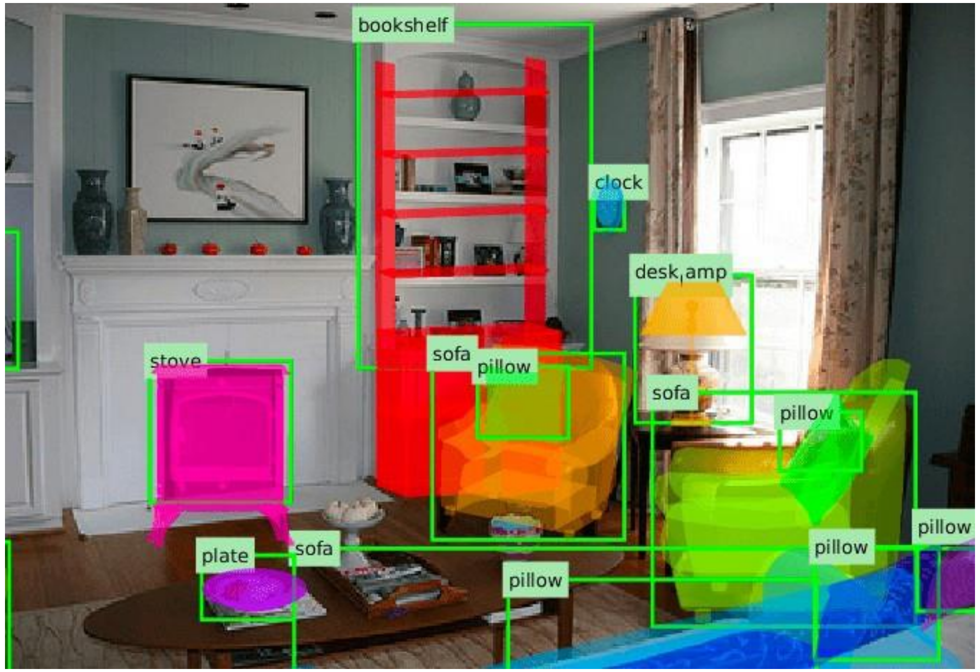
Defining Artificial Intelligence

- Many terms out there with overlapping or confused meanings
  - Artificial Intelligence
  - Machine Learning
  - Deep Learning
  - Data Science
- You will find that in AI, we like to have many terms meaning the same thing!

Artificial Intelligence

- Getting computers to behave intelligently:
    - Perform non-trivial tasks as well as humans do
    - Perform tasks that even humans struggle with

- Many sub-goals:
    - Perception
    - Reasoning
    - Control
    - Planning



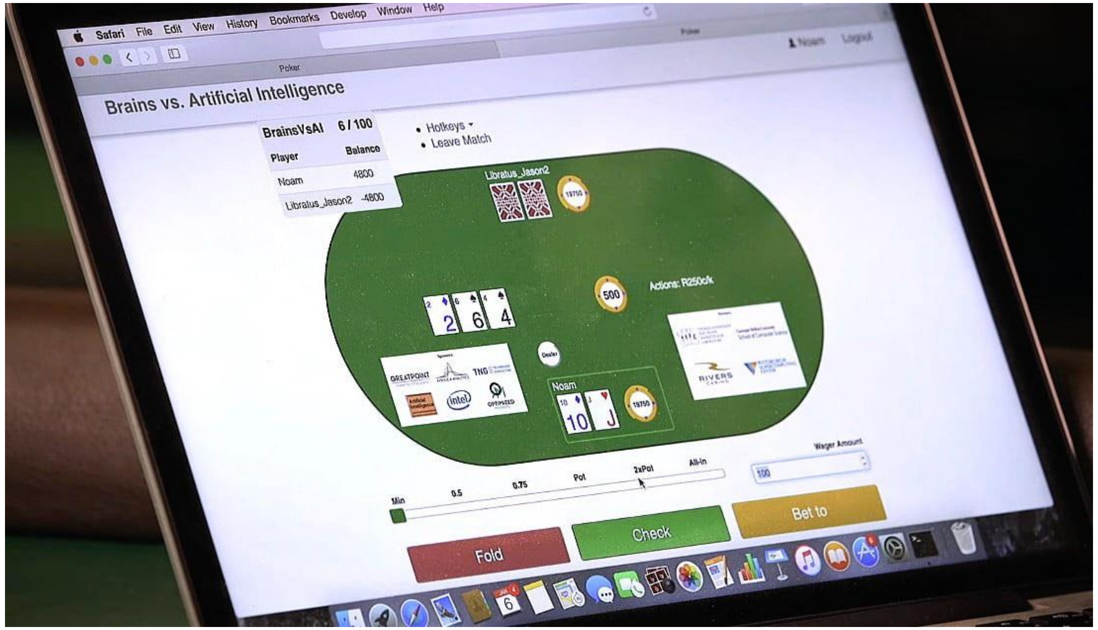
My poker face: AI wins multiplayer game for first time

Pluribus wins 12-day session of Texas hold'em against some of the world's best human players

Speech Recognition: Perception + Reasoning

Autonomous Driving: Perception + Reasoning Control + Planning

Game Playing: Reasoning + Planning

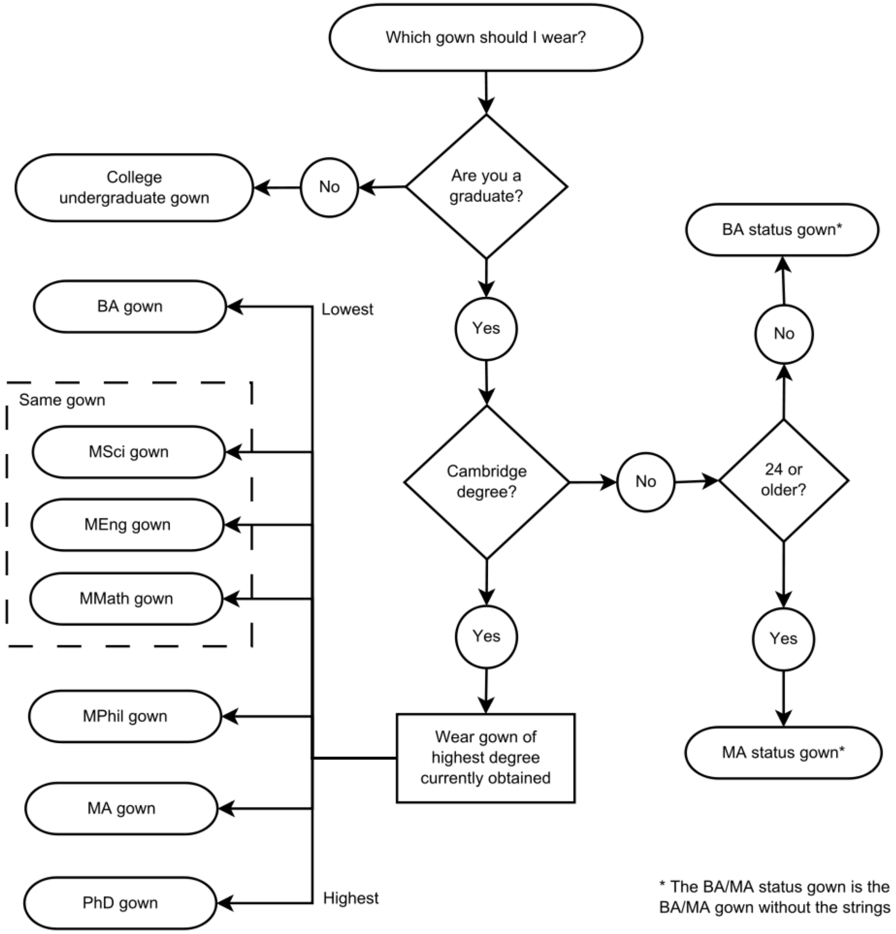


Knowledge-Based AI

Write programs that simulate how people solve the problem

Fundamental limitations:

- Will never get better than a person
- Requires deep domain knowledge
- We don't know how we do some things (e.g., riding a bicycle)



Data-Based AI = Machine Learning

Write programs that learn the task from examples

✅ No need to know how we do it as humans
✅ Performance should improve with more examples

❌ May need many examples!
❌ May not understand how the program works!

Machine Learning

- Study of algorithms that
  - Improve their performance P
  - At some task T
  - With experience E
- Well defined learning task: <P,T,E>

The Machine Learning Process

- Study of algorithms that
  - Improve their <u>performance</u> P
  - At some <u>task</u> T
  - With <u>experience</u> E
- Well defined learning task: <P,T,E>

- Experience
  - Examples of the form

    (input, correct output)
- Task
  - Mapping from input to output
- Performance
  - "Loss function" that measures error w.r.t. desired outcome

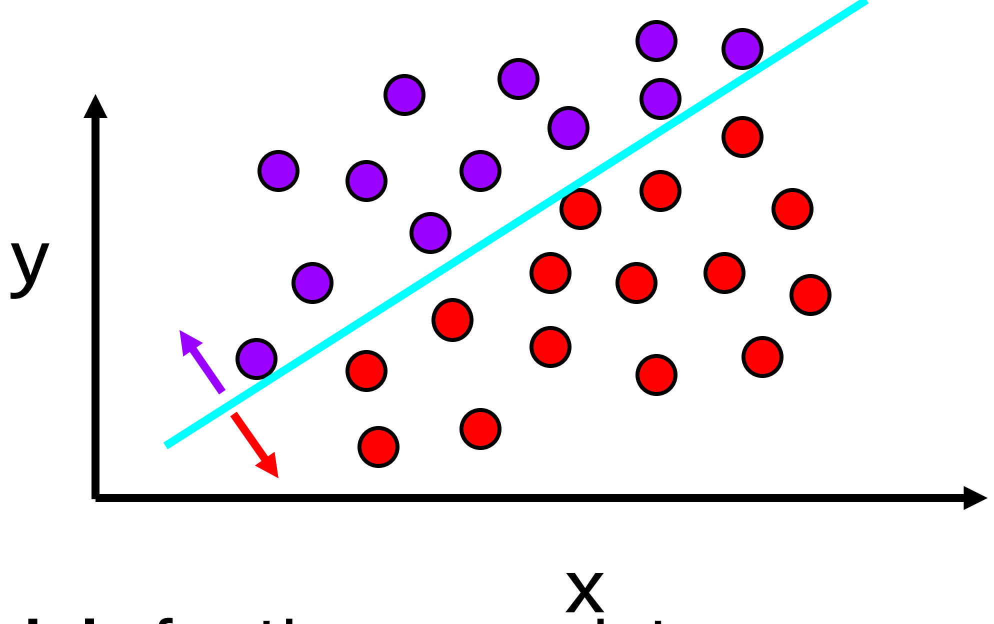
Choices in ML Problem Formulation

- Experience
  - Examples of the form (input, correct output)
- Task
  - Mapping from input to output
- Performance
  - "Loss function" that measures error w.r.t. desired outcome

Loan Applications

- What historical examples do I have? What is a correct output?
- Predict probability of default? Loan decision? Credit score?
- Do I care more about minimizing False Positives? False negatives?

What is a “model”?



A **useful approximation** of the world

Typically, there are **many reasonable models** for the same data

Training a model = finding appropriate values for (a,b,c,…)

- An **optimization** problem
- “appropriate” = **minimizes the Loss (cost) function**
- We will focus on a common training algorithm later on

Machine Learning Decisions

- Classification vs Regression
  - Classification: predict between set categories
  - Regression: predict a value (real number)
- Supervised vs Unsupervised
  - Supervised: data with examples of what we want to predict
  - Unsupervised: data but no examples of what we want to predict

Classification: Three Elements

1. Data:
   - x: data example with d attributes
   - y: label of example (what you care about)

2. Classification model: a function $f_{(a,b,c,\ldots)}$
   - Maps from X to Y
   - (a,b,c,…) are the parameters

3. Loss function:
   - Penalizes the model's mistakes

Song	Rating
Some nights	⭐⭐⭐⭐⭐
Skyfall	⭐
Comfortably numb	⭐⭐⭐
We are young	⭐⭐⭐⭐
…	…
…	…
Chopin's 5^{th}	???

Machine Learning Decisions

- **Classification** vs Regression
  - Classification: predict between set categories
  - Regression: predict a value (real number)
- **Supervised** vs Unsupervised
  - Supervised: data with examples of what we want to predict
  - Unsupervised: data but no examples of what we want to predict

Classification Loss Function

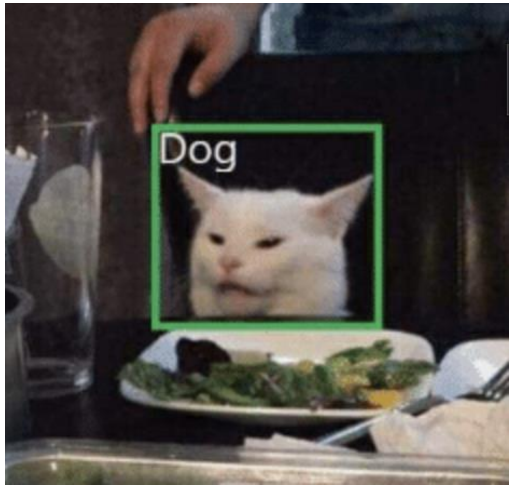
- How unhappy are you with the answer that the model gave?

- $\mathbf{L_{0\text{-}1}(y, f(x))} = 1$ if: $\mathbf{y \neq f(x)}$

  0 otherwise

- **0-1 loss** function: intuitive but hard to optimize = train

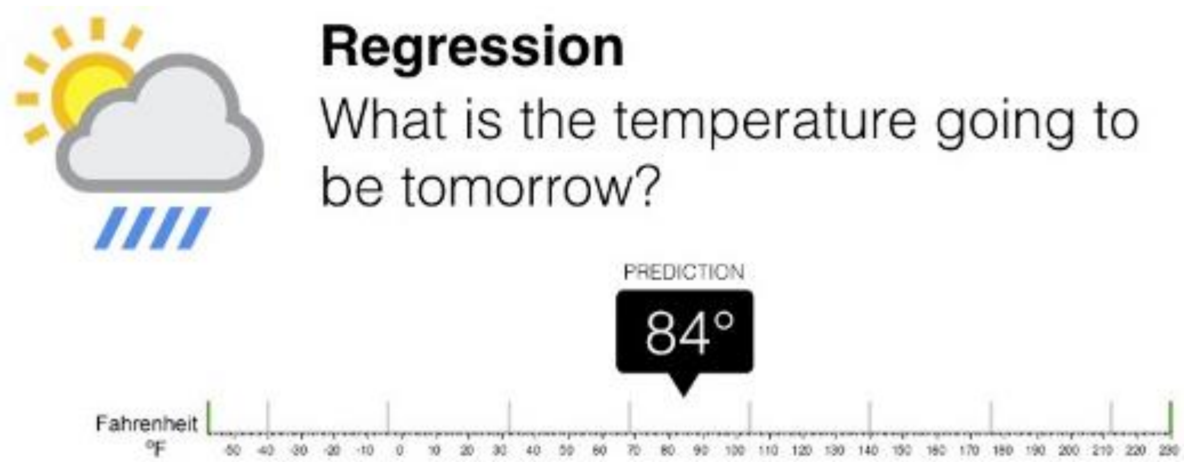
- In practice, we use **approximations** of the 0-1 loss – getting warmer or getting colder

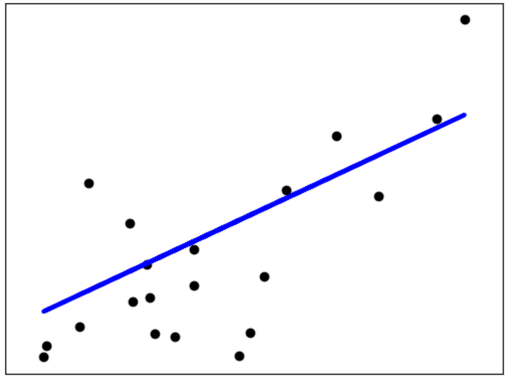


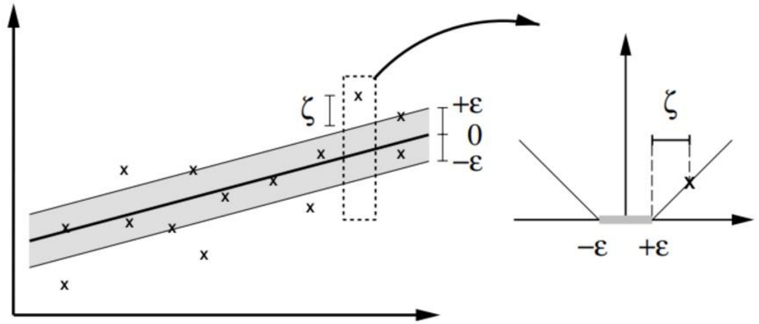
Regression

Examples:

- Stock price prediction
- Forecasting epidemics
- Weather prediction







Machine Learning Decisions

- Classification vs **Regression**
  - Classification: predict between set categories
  - Regression: predict a value (real number)
- **Supervised** vs Unsupervised
  - Supervised: data with examples of what we want to predict
  - Unsupervised: data but no examples of what we want to predict

Linear Regression

Data: $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i)\}$

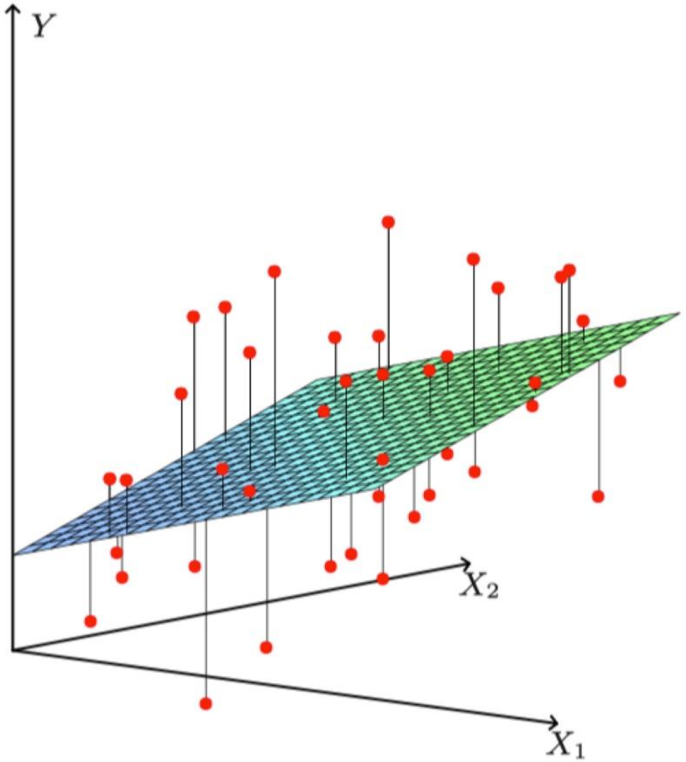
- x_i: data example with d attributes
- y_i: target of example (what you care about)

Model:

$$f(x; \boldsymbol{\beta}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$$

Loss function: Residual Sum of Squares

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i - f(x_i; \boldsymbol{\beta})\right)^2$$



Ridge Regression

- Linear Regression uses all features; model may be complicated
- **Ridge Regression** penalizes large parameter values

Model:

$$f(x;\boldsymbol{\beta}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$$

Loss function: Residual Sum of Squares + penalty term

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i - f(x_i;\boldsymbol{\beta})\right)^2 + \boldsymbol{\lambda} \sum_{j=0}^{d} \beta_j^2$$

Lasso Regression

- **As in Ridge Regression, Lasso** penalizes large parameters
- Penalizes **absolute** instead of squared coefficient values
- **Zeroes out** more coefficients **BUT** optimization is more involved

Model:

$$f(x;\boldsymbol{\beta}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$$

Loss function: Residual Sum of Squares + penalty term

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i - f(x_i;\boldsymbol{\beta})\right)^2 + \lambda \sum_{j=0}^{d} |\beta_j|$$

Example: Prostate Cancer

Stamey et al. (1989)

- x: cancer volume, prostate weight, age, …
- y: amount of prostate-specific antigen

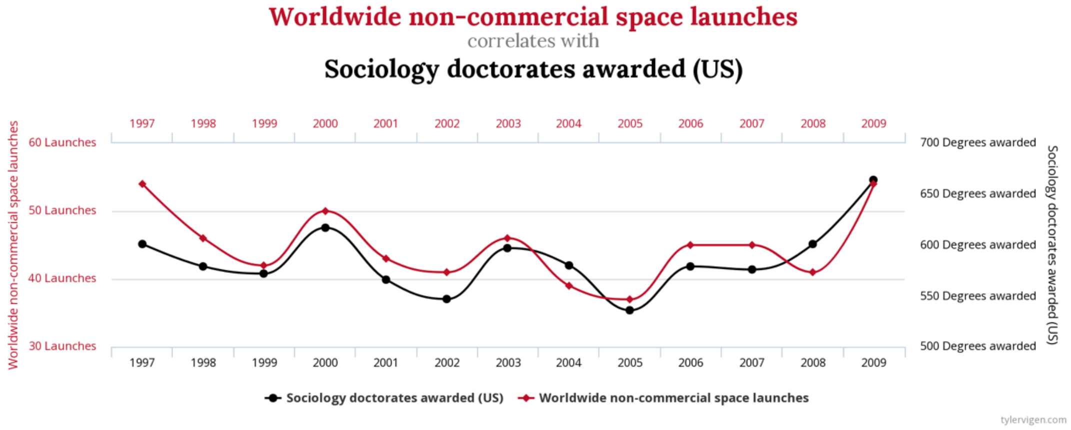
Term	LS	Best Subset	Ridge	Lasso
`Intercept`	2.465	2.477	2.452	2.468
`lcavol`	0.680	0.740	0.420	0.533
`lweight`	0.263	0.316	0.238	0.169
`age`	−0.141		−0.046	
`lbph`	0.210		0.162	0.002
`svi`	0.305		0.227	0.094
`lcp`	−0.288		0.000	
`gleason`	−0.021		0.040	
`pgg45`	0.267		0.133	
Test Error	0.521	0.492	0.492	0.479
Std Error	0.179	0.143	0.165	0.164

Correlation vs. Causation

- Correlation measures the strength and direction of a relationship between two variables
- Causation refers to a cause-and-effect relationship, where one variable directly influences the other
- It's crucial to remember that a strong correlation doesn't necessarily imply causation

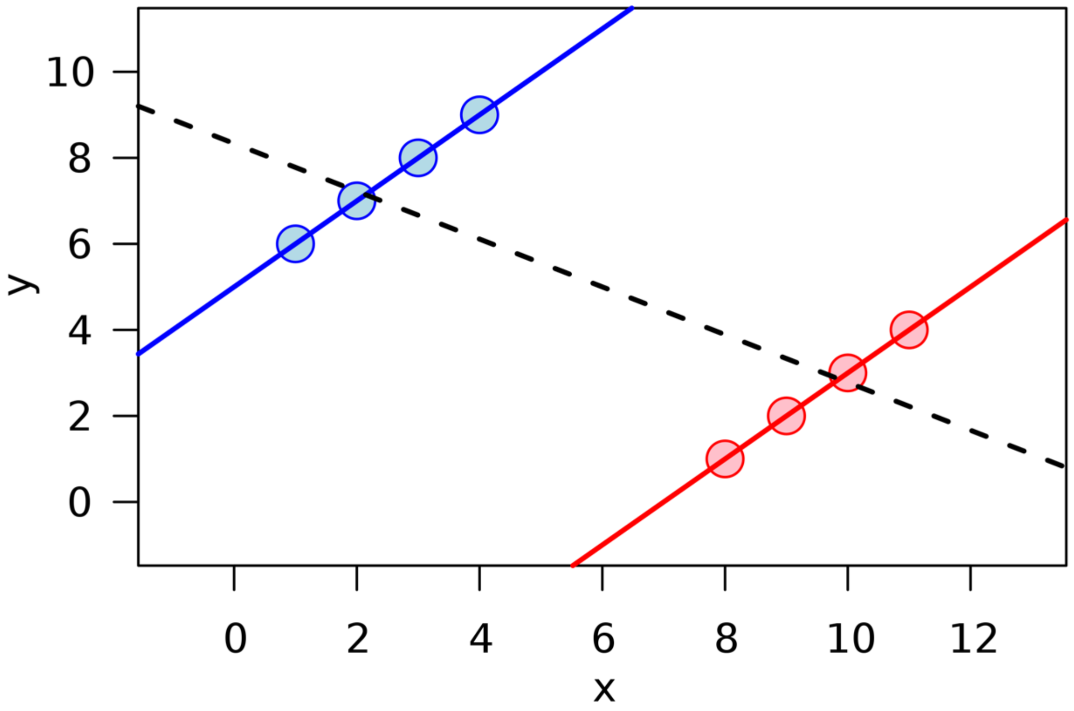
Correlation vs. Causation

- To avoid confusion between correlation and causation:
    - Consider possible confounding variables or third factors
    - Look for evidence of a causal mechanism
    - Test the relationship using controlled experiments or statistical methods



Simpson's Paradox

- A trend or relationship between two factors seems to exist when you look at separate groups but disappears or even reverses when you combine the groups together.
- To avoid Simpson's Paradox:
  - Investigate data at different levels of aggregation
  - Consider the influence of confounding variables
  - Use caution when combining data from different sources or groups



Simpson's Paradox

- In 1973, UC Berkeley found that men applying were more likely to be admitted than women

	All	Men	Women
Applicants Admitted	41%	44%	35%

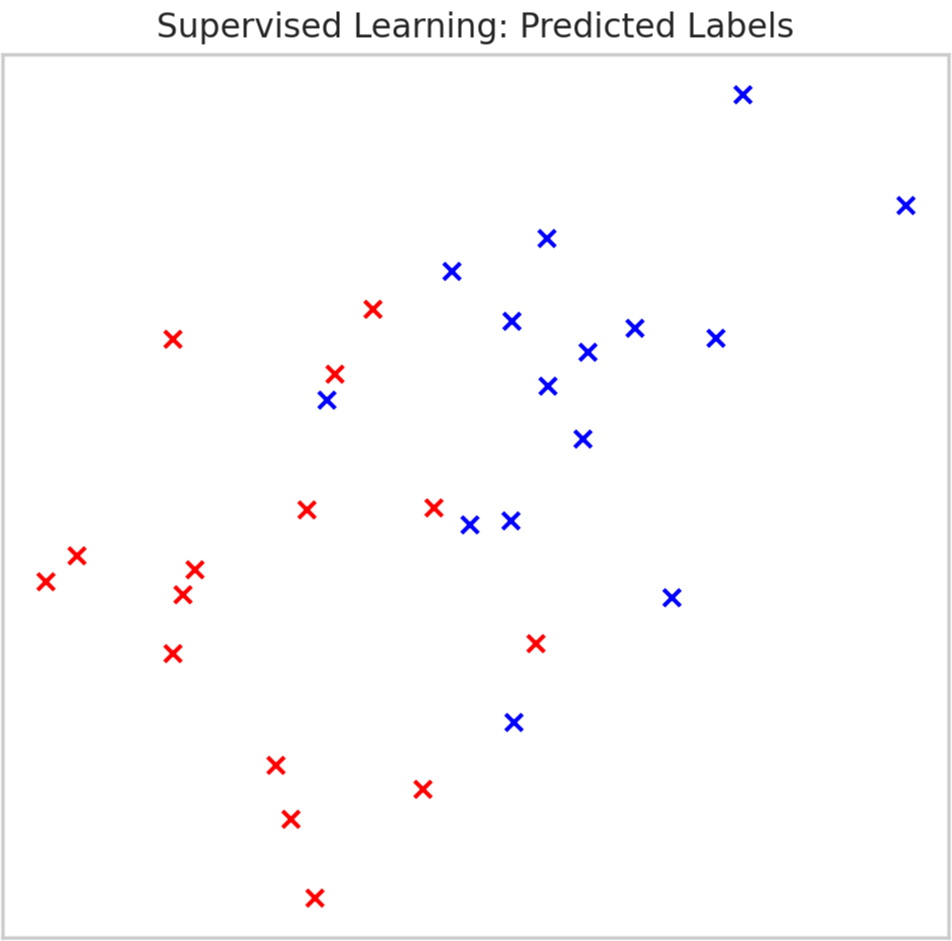
Simpson's Paradox

- In 1973, UC Berkeley found that men applying were more likely to be admitted than women
- But when analyzed at a department level, they found only a small subset of departments with a lot of applicants were biased
- Solving the problem required a targeted approach, not a general one

Department	All	Men	Women
A	64%	62%	82%
B	63%	63%	68%
C	35%	37%	34%
D	34%	33%	35%
E	25%	28%	24%
F	6%	6%	7%
Applicants Admitted	**39%**	**45%**	**30%**

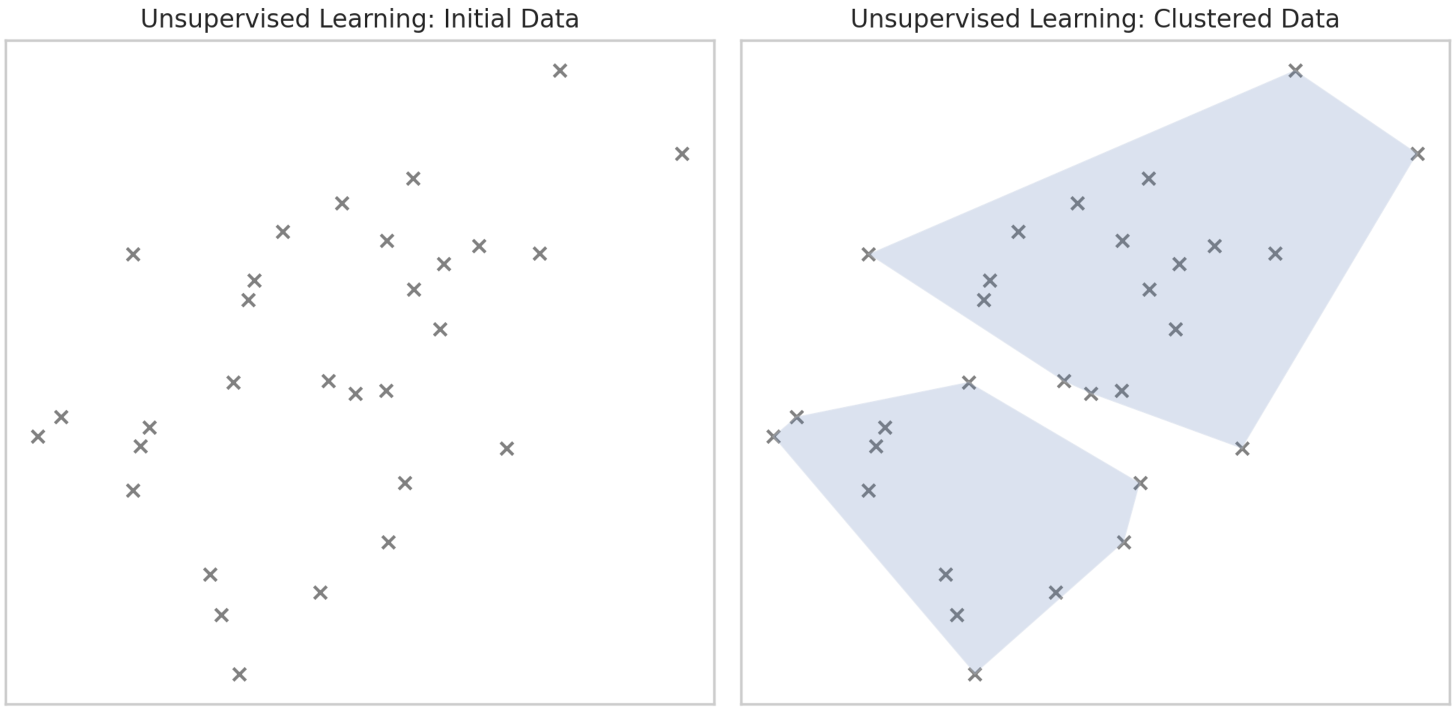
Supervised vs Unsupervised

- So far we have looked at two types of <u>supervised</u> learning
- In both our classification and regression examples, we have examples where we "know the answer"
- With supervised learning, we have a strong definition of the model's performance



Supervised vs Unsupervised

- In <u>unsupervised</u> learning, we don't know what the answer is – collecting this data may be costly, or impossible
- Unsupervised approaches attempt to uncover patterns in the data without relying on a pre-defined label

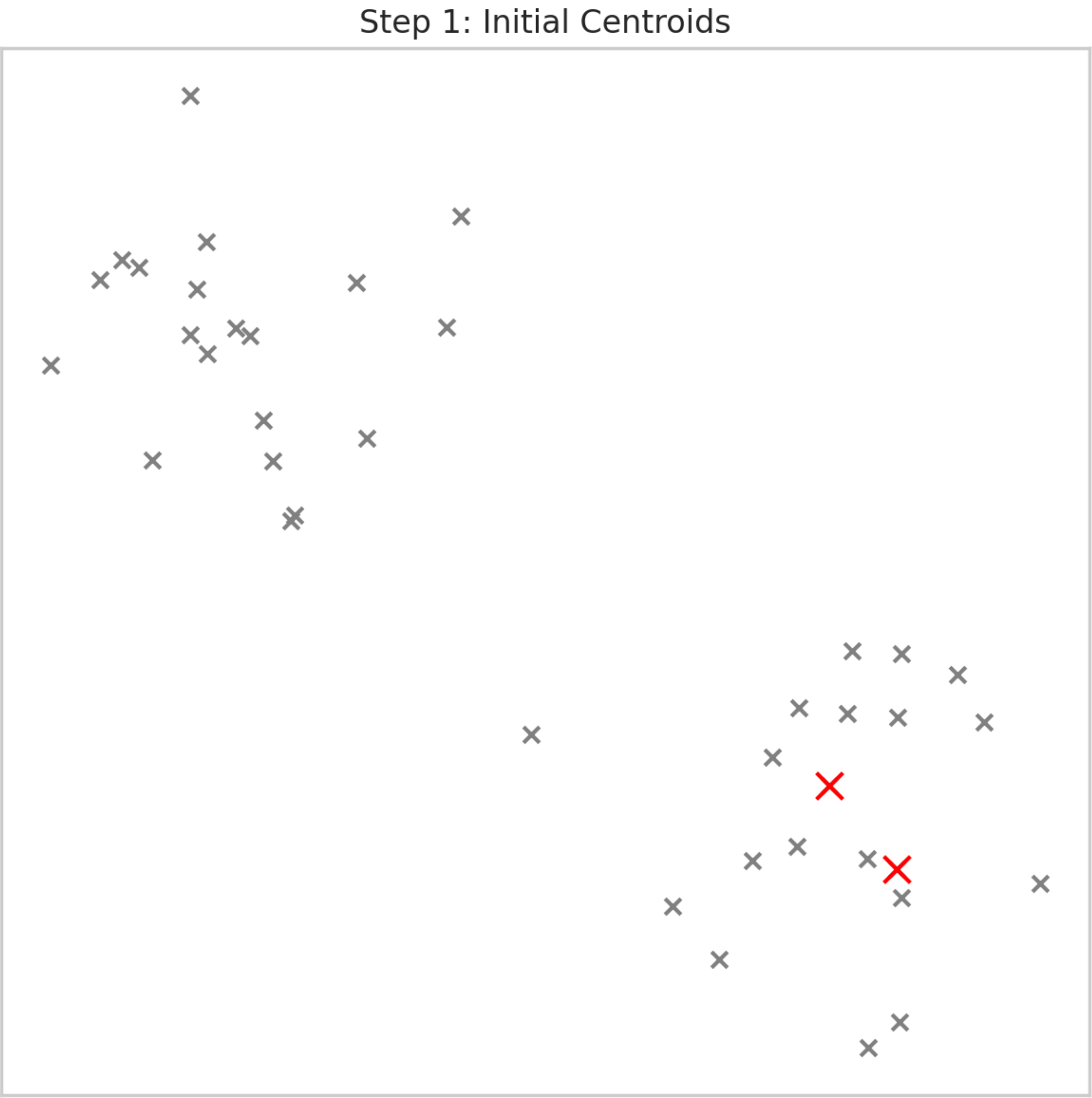


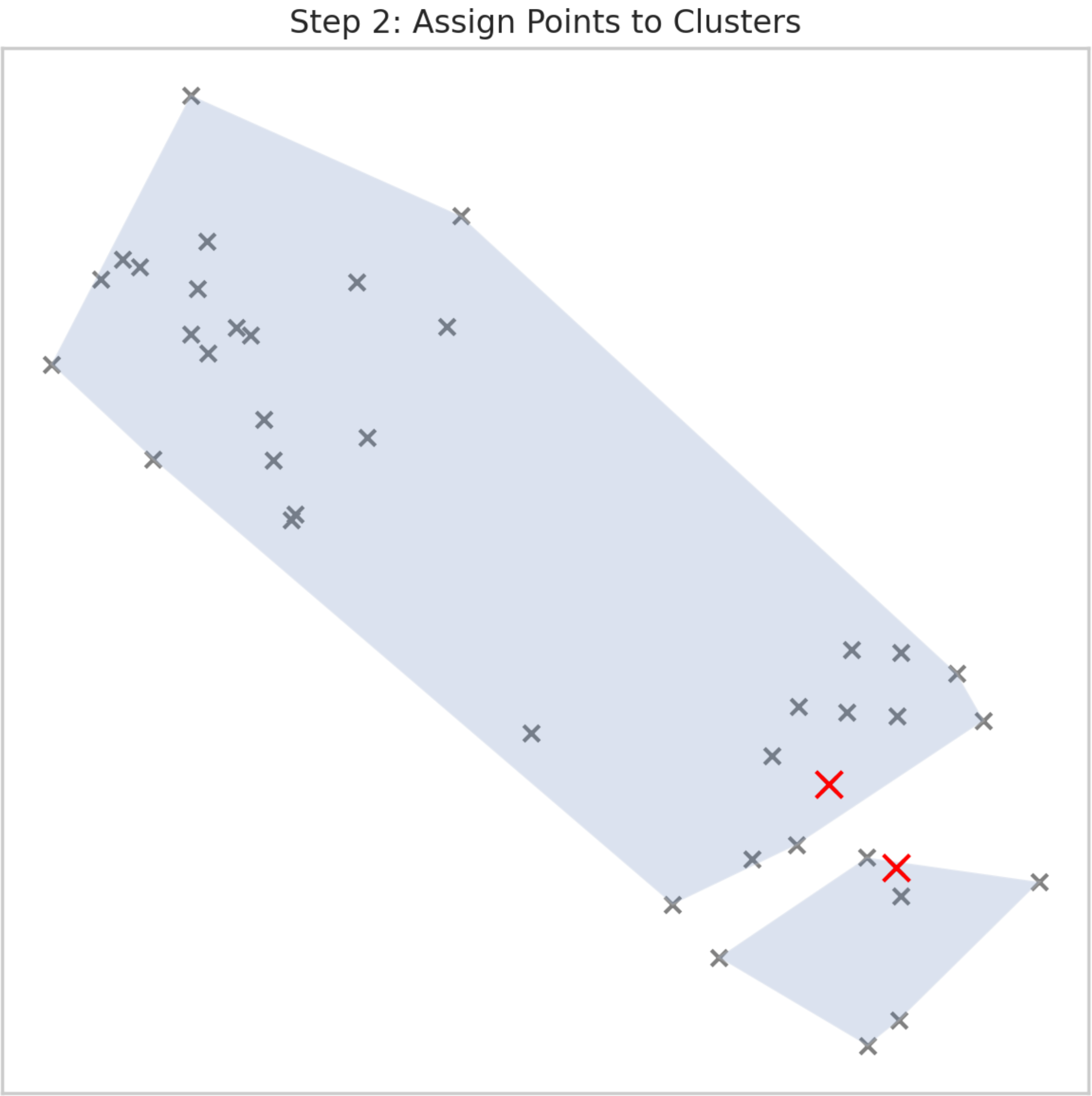
Unsupervised Learning: K-Means Clustering

- Clustering approaches seek to uncover groups within data
- Starting with randomly set groups, we measure the similarity of each point to the possible groups, and re-assign
- This process continues until no points change group

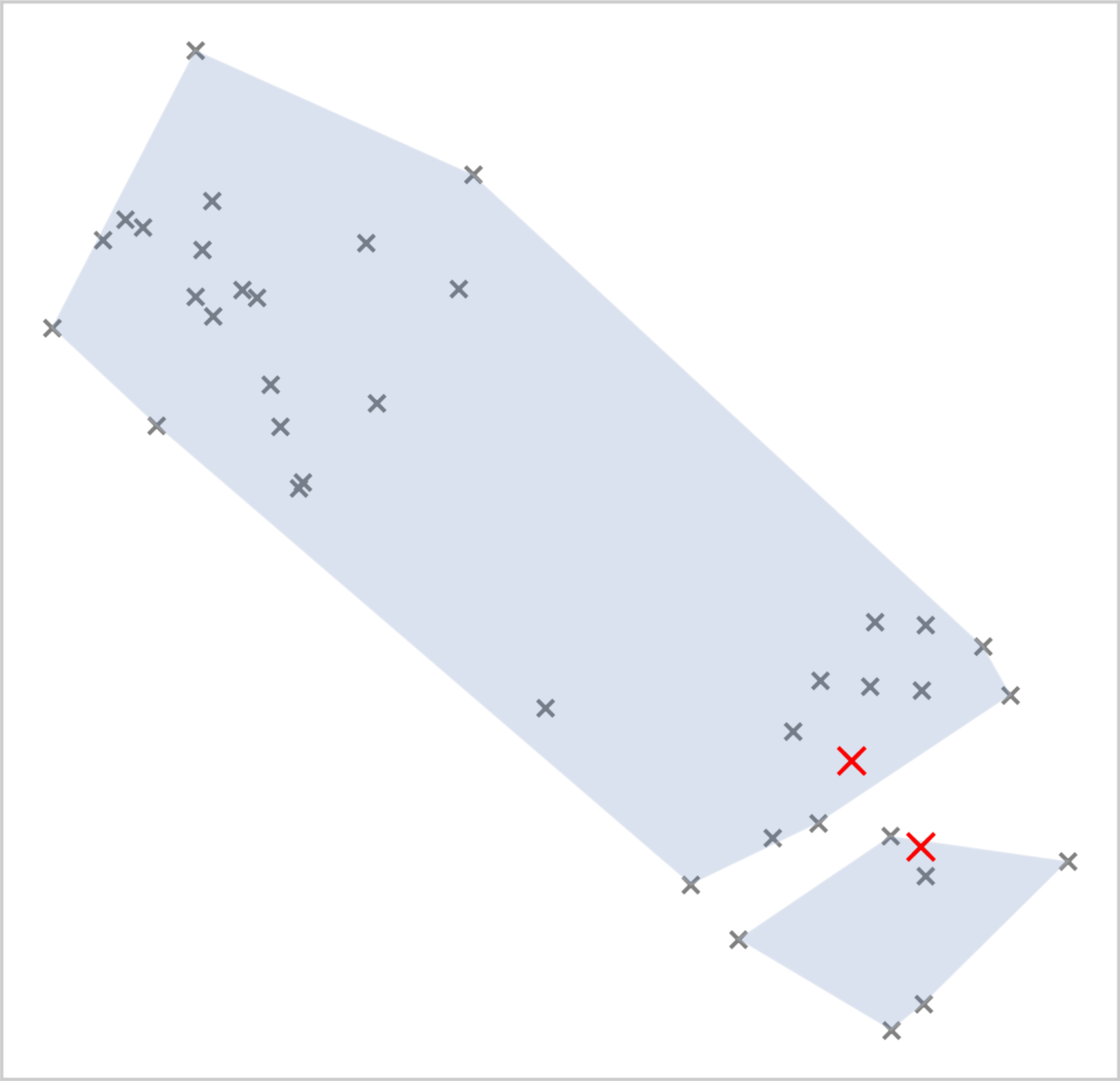
Machine Learning Decisions

- **Classification** vs Regression
  - Classification: predict between set categories
  - Regression: predict a value (real number)
- Supervised vs Unsupervised
  - Supervised: data with examples of what we want to predict
  - Unsupervised: data but no examples of what we want to predict

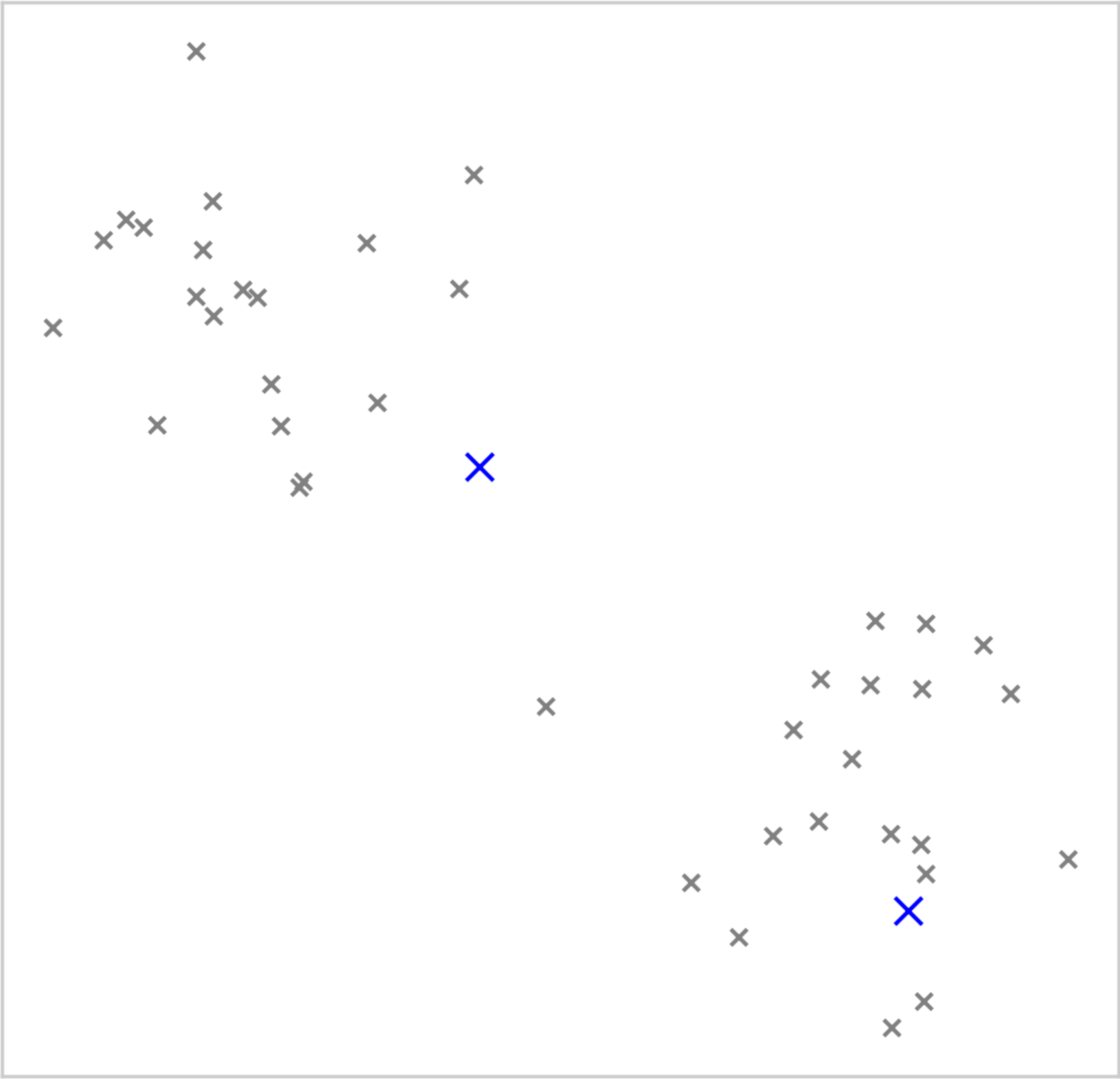

Step 1: Initial Centroids

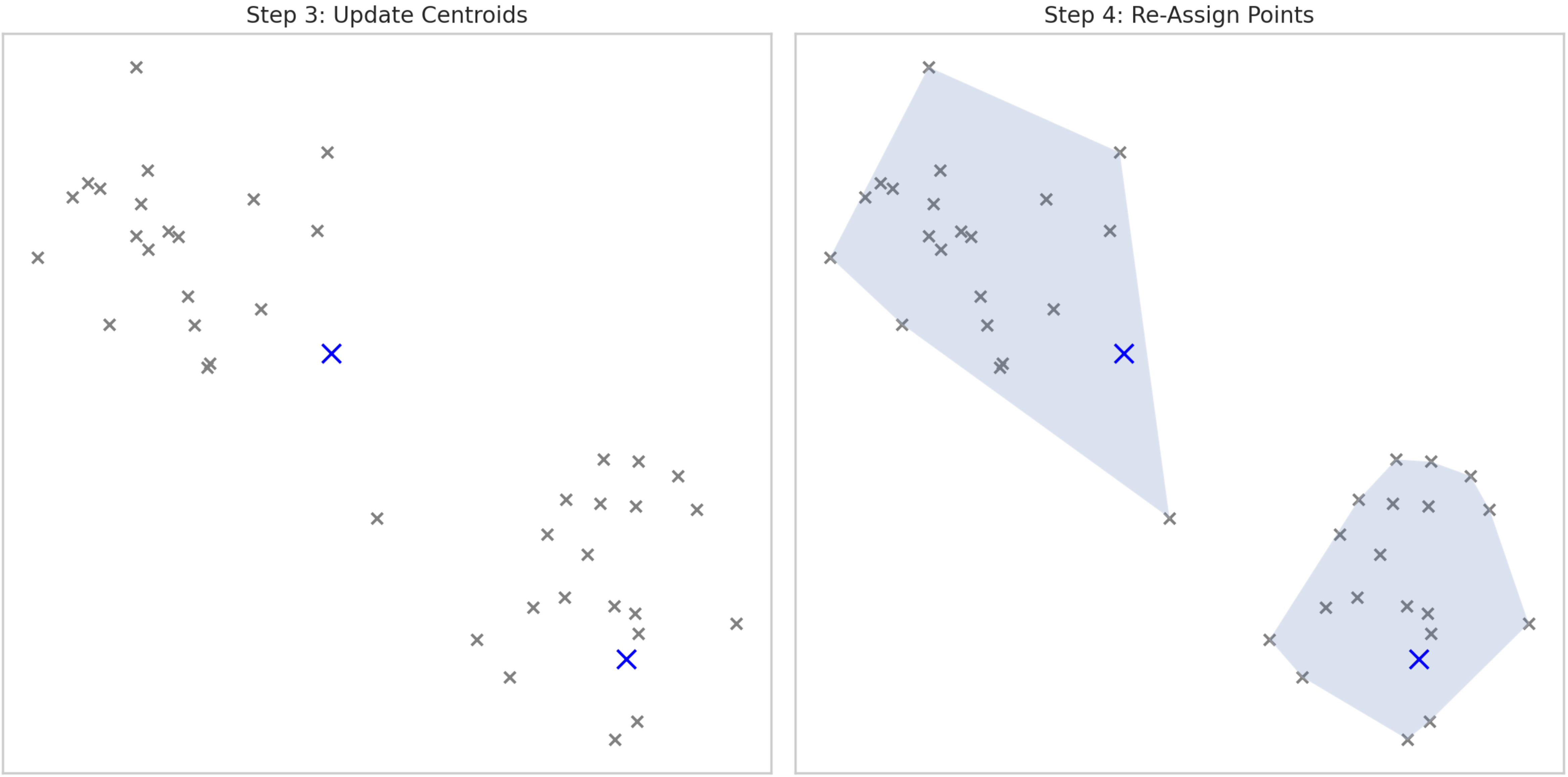

Step 2: Assign Points to Clusters

Step 2: Assign Points to Clusters



Step 3: Update Centroids




Step 3: Update Centroids
Step 4: Re-Assign Points

Machine Learning Decisions

- Classification vs Regression
  - Classification: predict between set categories
  - Regression: predict a value (real number)
- Supervised vs Unsupervised
  - Supervised: data with examples of what we want to predict
  - Unsupervised: data but no examples of what we want to predict
- Deep Learning?
  - Methods that use neural networks (this afternoon!)

Programming Languages for AI

- Python
  - Dominates AI development due to extensive libraries like TensorFlow, PyTorch, and scikit-learn.
  - Easy syntax, strong community, rich ecosystem for AI research and production.
  - Slower execution compared to lower-level languages, can struggle with very large-scale, performance-critical systems.

Programming Languages for AI

- R
  - Specialized for statistics and data visualization, with packages like caret and ggplot2.
  - Most useful for data preprocessing, statistical modeling, and some ML tasks.
  - Less suited for general-purpose AI development or production-grade systems.

Programming Languages for AI

- C++
  - High performance for real-time applications like gaming or embedded AI. Libraries like dlib and OpenCV excel in computer vision.
  - Core of many Python-based AI tools (e.g., TensorFlow's backend).
  - Steeper learning curve, verbose syntax, slower development time.
- Java
  - Scalability and enterprise use, with frameworks like Weka and Deeplearning4j.
  - Can interact with Python tools via APIs or frameworks like Apache Spark.
  - Verbose and less favored for rapid prototyping.
- Julia
  - High-performance numerical computing, increasingly adopted for AI and optimization.
  - Growing interoperability with Python (e.g., PyCall).
  - Smaller ecosystem and less community support than Python.

Machine Learning Libraries PyTorch

- PyTorch
  - Intuitive and flexible, with dynamic computation graphs allowing for easier debugging and experimentation.
  - Strong adoption in research, supported by an active community.
  - Less mature deployment tools compared to TensorFlow (though this gap is narrowing).
  - Can be slower in some production scenarios without optimization.

Machine Learning Libraries

- TensorFlow
  - Comprehensive ecosystem with tools for training (TensorFlow), deployment (TensorFlow Serving, TensorFlow Lite), and explainability (What-If Tool).
  - TensorFlow.js and TensorFlow Lite make it suitable for web and mobile development.
  - Strong community and corporate support (Google).
  - Integration with Keras offers a high-level API for beginners.
  - Steeper learning curve compared to PyTorch.
  - Debugging can be less straightforward due to static computation graphs (though this has improved with TensorFlow 2.x).

Machine Learning Libraries

- Scikit-learn
  - Easy-to-use interface for classical machine learning tasks like regression, classification, and clustering.
  - Excellent for preprocessing and feature engineering (e.g., PCA, scalers).
  - Strong documentation and wide adoption in education and small-scale projects.
- XGBoost
  - Extremely efficient and scalable for tabular data tasks.
  - Known for achieving high accuracy with minimal tuning.
  - Distributed training support for large datasets.
- HuggingFace Transformers
  - Simplifies the use of pre-trained transformers for NLP, vision, and multimodal tasks.
  - Strong community and regularly updated with state-of-the-art models.
  - Easy fine-tuning and deployment of large language models (LLMs).

Next: Lab 1

https://github.com/CARTE-Toronto/mitsubishi-workshop