# Course Logistics and Introduction

CS771: Introduction to Machine Learning

# Course Logistics

- Timing and Venue: Mon/Thur 6:00-7:15pm, L-20
- Course website: https://tinyurl.com/cs771-a24 (slides, readings, etc)
- Online discussion/QA: Piazza (https://tinyurl.com/cs771-a24-piazzasignup)
- Instructor's contact email: piyush@cse.iitk.ac.in, office: RM-502 (CSE dept)
  - Prefix email subject with CS771, else might get ignored
  - Use of Piazza is encouraged for course-related matters (also has private messaging)
  - Office hours: By appointment
- TAs: Team of 20 TAs. Their contact and office hours details shared soon
- Unofficial auditors are welcome. However, can't participate in exams/quizzes
  - Can attempt homeworks, quizzes, exams on their own. Won't be graded

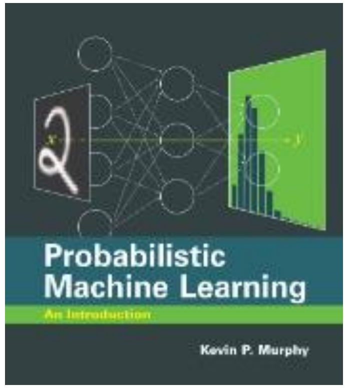
# Workload and Grading Policy

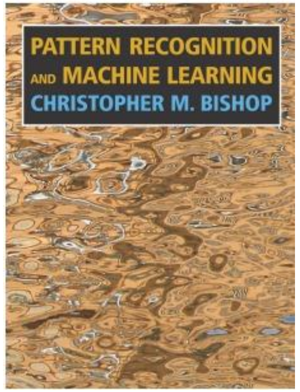
Quizzes will be closed-book. For exams, one A4 size cheat-sheet will be allowed

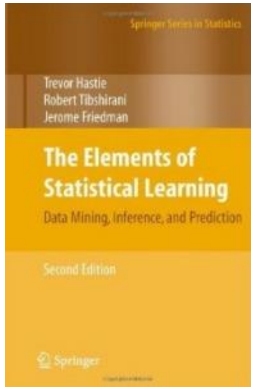
- 4 quizzes: 20%
- 2 homeworks/mini-projects: 30%
  - Writeups must be prepared in PDF using the provided LaTeX template
  - Knowledge of Python programming is assumed
  - To be done in groups of 5 students. Form your groups NOW
- Mid-sem exam: 20%
- End-sem exam: 30%
- Quiz dates (tentative): Aug 13, Sept 3, Oct 1, Oct 22 (duration: 30 mins)
  - Quiz timing and venue: will be announced closed to the quiz date
- HW/mini-project dates (tentative): Aug 19, Oct 3 (roughly 3 work-weeks given)
- Mid-sem and end-sem exam dates: As per DOAA announcements

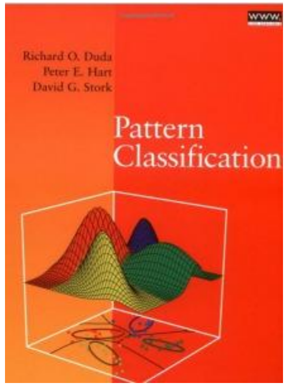
# Textbook and References

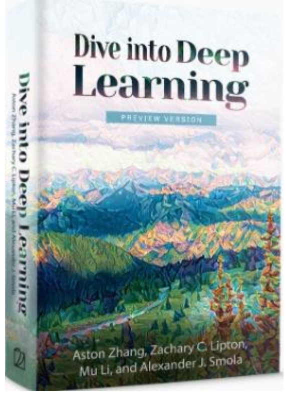
- Many excellent texts but none “required”. Some include:

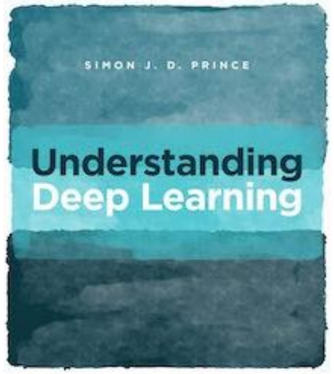



- See the course website for links and other relevant texts and references
- Different books might vary in terms of
  - Set of topics covered
  - Flavor (e.g., classical statistics, deep learning, probabilistic/Bayesian, theory)
  - Terminology and notation (beware of this especially)
- For each topic in the course, we will provide you recommended readings

# Course Goals

- Introduction to the foundations of machine learning (ML)
- Focus on developing the ability to
  - Understand the underlying principles (and maths ☺) behind ML models and algos
  - Understand how to implement and evaluate them
  - Understand/develop intuition on choosing the right ML model/algo for your problem
- (Hopefully) inspire you to work on and learn more about ML
- Not an intro to popular software frameworks and libraries, such as scikit-learn, PyTorch, Tensorflow, etc
  - However, you are encouraged to explore these as the course progresses

# Expectations from you

- Attend classes regularly (even though we have no attendance policy)

- Please make yourself acquainted with the maths required for the course
  - We will provide some refresher slides and reference material
  - Some of the maths will be introduced as and when it is needed

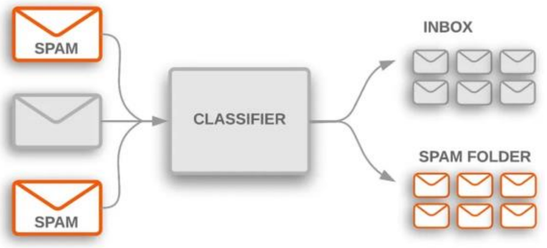
- Please ensure that you understand the maths on the slides
  - We won't do all the derivations on the slides
  - In class, our focus will be on key steps and intuition
  - If not obvious, you should try to work out the detailed steps at home (will be good practice for quizzes and exams) on your own or with classmates
  - If things are unclear, please do reach out to us (e.g., on Piazza or office hours)

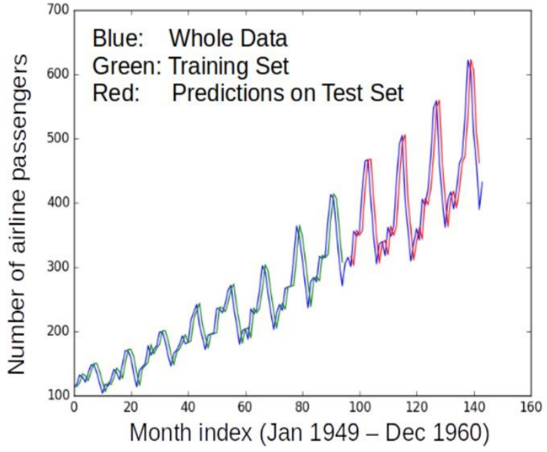
# What Is Machine Learning?

# Machine Learning (ML)

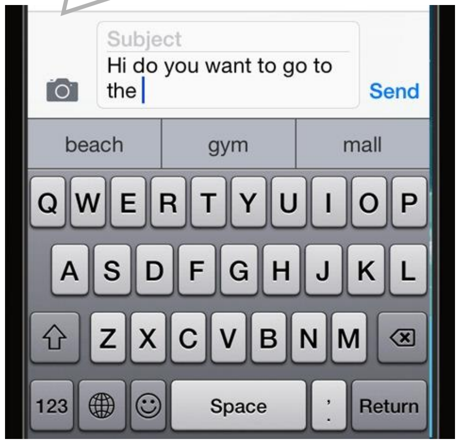
- Designing algorithms that ingest data and learn a model of the data
- The learned model can be used to
  - Detect patterns/structures/themes/trends etc. in the data
  - Make predictions about future data and make decisions

Next word prediction (key task in large-language models like ChatGPT)








- Modern ML algorithms are heavily “data-driven”
  - No need to pre-define all the rules by humans (infeasible/impossible anyway)
  - The rules are not “static”; can adapt as the ML algo ingests more and more data

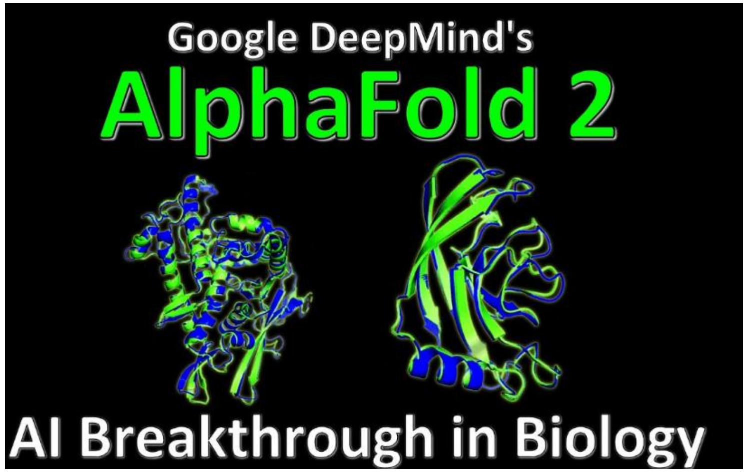
# Where Should We Use ML?

Handwritten digit recognition: Not too complex but still reasonably complex that an ML approach is desirable

- When the learning problem is very complex, e.g.,
  - Enumerating all rules is infeasible or too time-consuming
  - Rules might evolve with time
- In such cases, hard-coding the rules in a computer program may not work
  - Difficult to define and code all possible rules
  - Difficult to update the program if rules evolve
- ML replaces the idea of humans writing code by humans supplying data
  - The ML algorithm automatically learns the model (the rules) from the supplied data
  - The model can evolve with more and more data

# ML: Some Success Stories

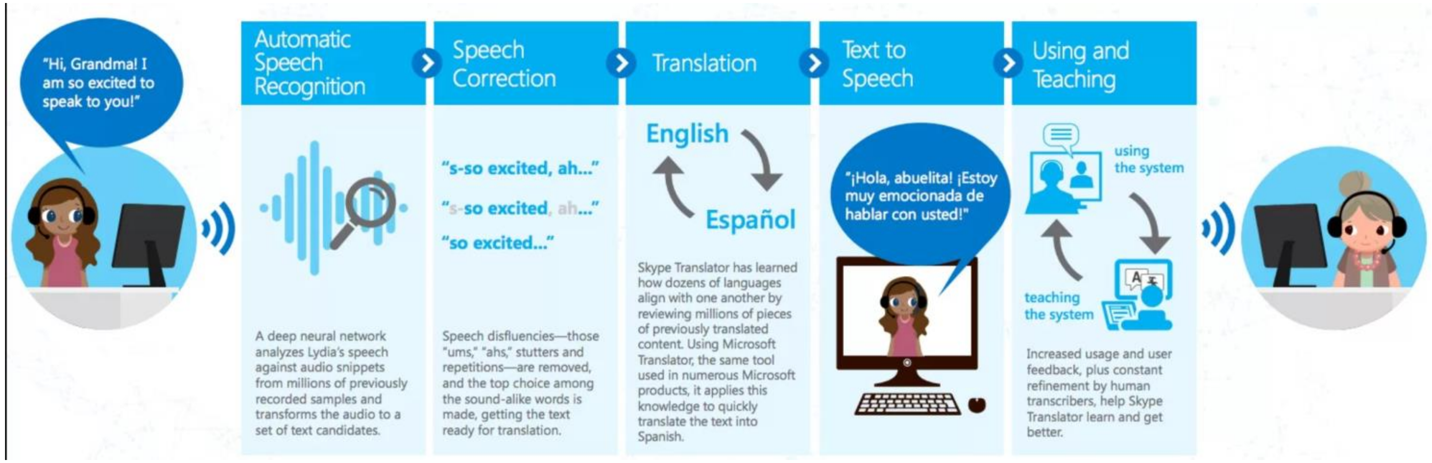
Protein Structure Prediction




Autonomous Driving


AI Generated Digital Art (Dall E 2)

Real-time Speech Translation



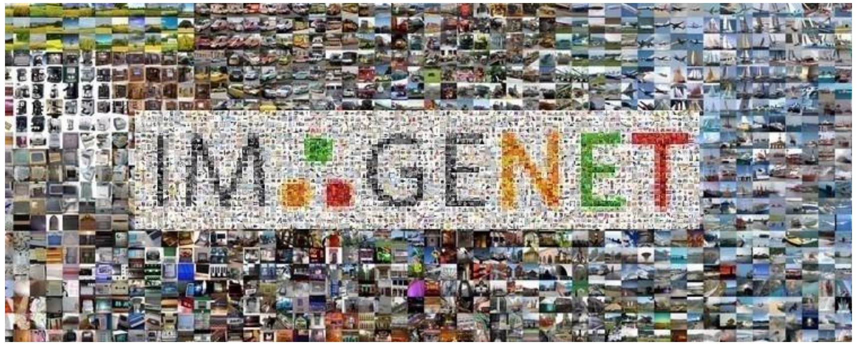

Conversational Systems


Digital art pic credit: David Schnurr

# Key Enablers for Modern ML



- Availability of large amounts of data to train ML models



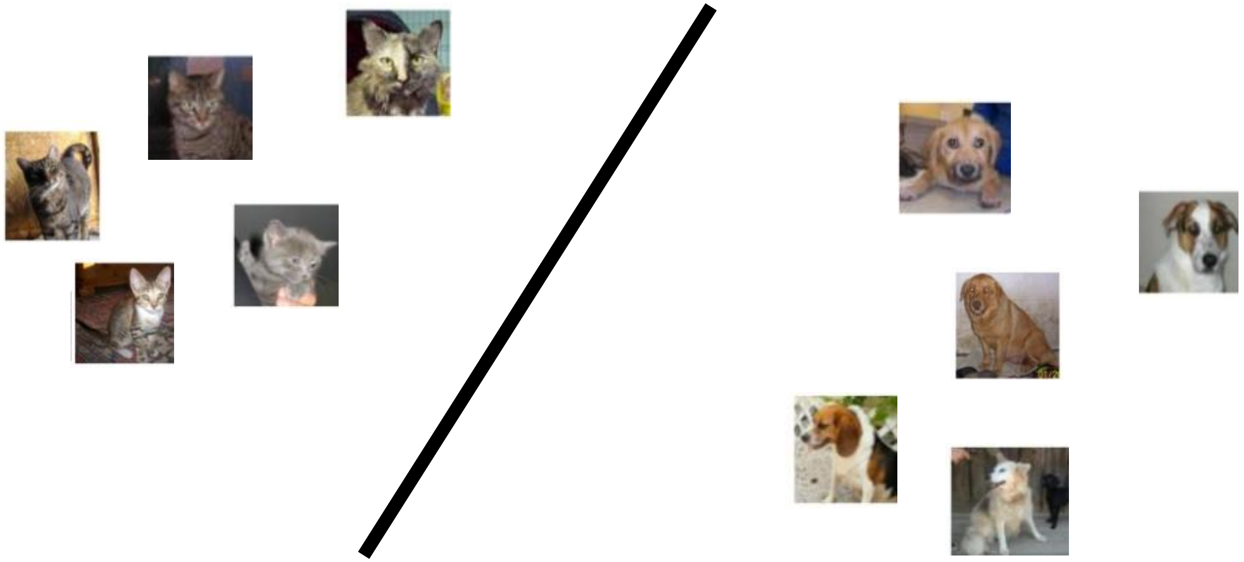


- Increased computing power (e.g., GPUs)

# ML: A Simple Illustration

- ML enables intelligent systems to be data-driven rather than rule-driven
- How: By supplying training data and building statistical models of data
- Pictorial illustration of an ML model for binary classification:

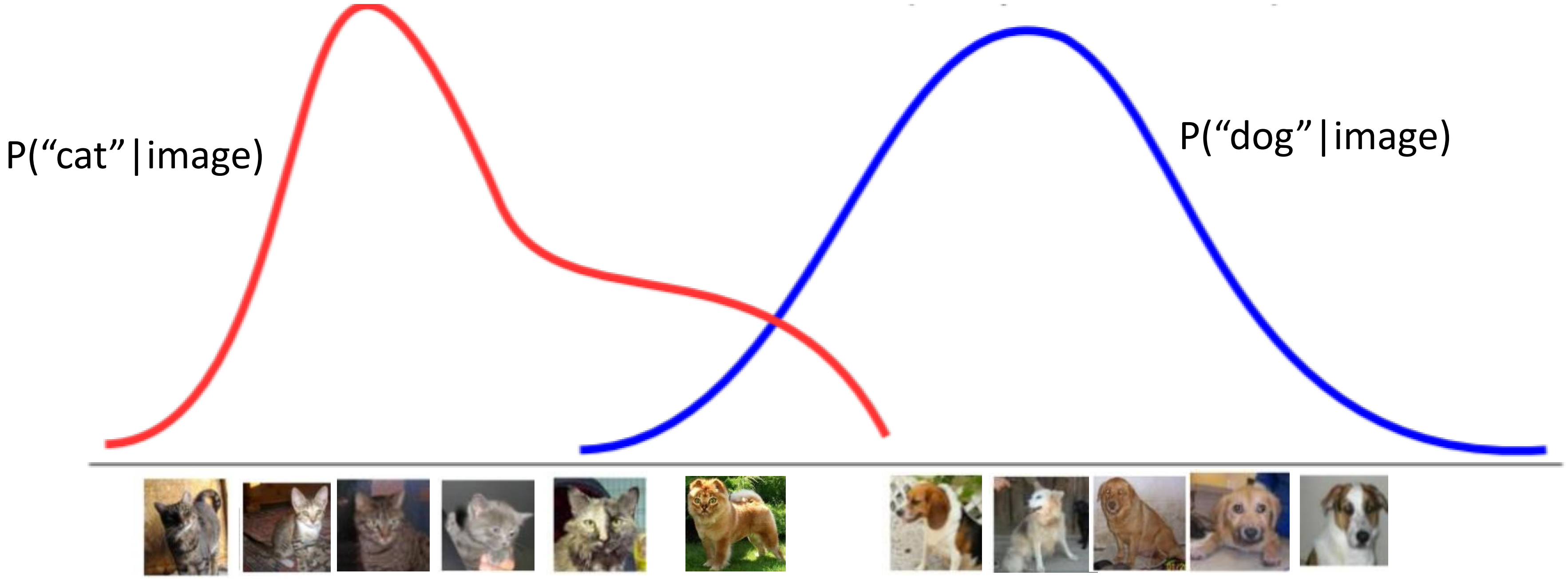
A Linear Classifier (the statistical model)



# ML: A Simple Illustration

- ML enables intelligent systems to be data-driven rather than rule-driven
- How: By supplying training data and building statistical models of data
- Pictorial illustration of an ML model for binary classification:

A Probabilistic Classifier (the statistical model)



# ML: The Exam Analogy

- It's the performance on the D-day which matters
- In an exam, our success is measured based on how well we did on the questions in the test (not on the questions we practiced on)
- Likewise, in ML, success of the learned model is measured based on how well it predicts/fits the future test data (not the training data)

In Machine Learning, generalization performance
on the test data matters
(we should not "overfit" on training data)

# A Loose Taxonomy of ML

Learning using labeled data

Learning using unlabeled data

"Labeled" means, during training, for each input, the corresponding output is available (i.e., the machine learner is explicitly told that a cat image is of a cat)

Some examples of supervised learning problems

- Classification
- Regression
- Ranking

Supervised Learning

Machine Learning

Unsupervised Learning

Some examples of unsupervised learning problems

- Clustering
- Dimensionality Reduction
- Unsupervised Probability Density Estimation
- Generative Models (e.g., ChatGPT)

Reinforcement Learning

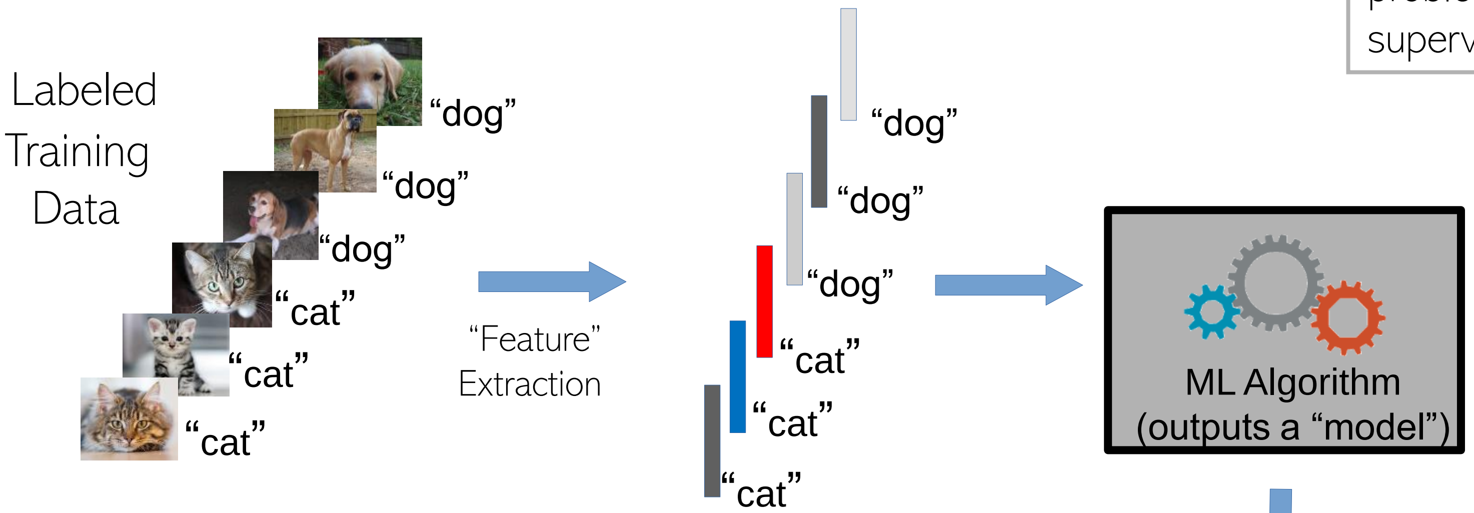
RL doesn't use "labeled" or "unlabeled" data in the traditional sense! In RL, an agent learns via its interactions with an environment

Many other specialized flavors of ML also exist, some of which include

- Semi-supervised Learning
- Self-supervised Learning (very popular these days)
- Active Learning
- Transfer Learning
- Multitask Learning
- Zero-Shot/Few-Shot Learning
- Continual learning

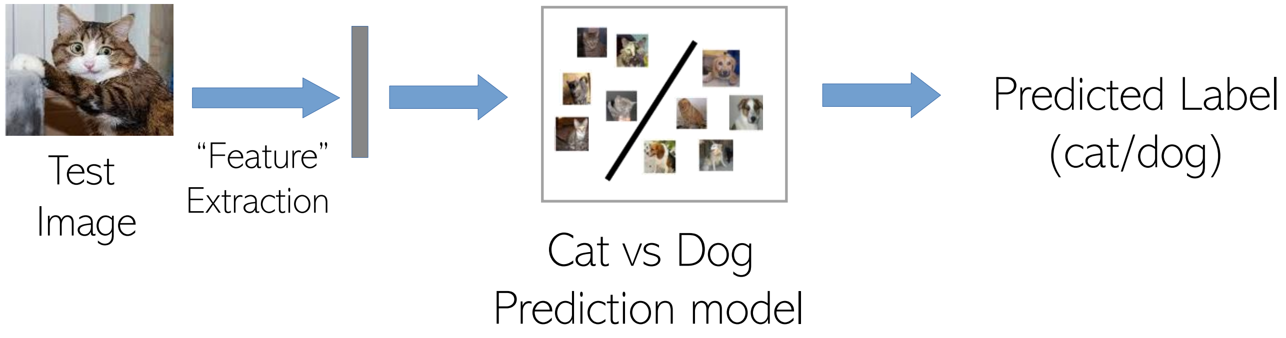
# A Typical Supervised Learning Workflow

Note: This example is for the problem of binary classification, a supervised learning problem




Is feature extraction done "manually" as a pre-processing step before the ML algo starts working? Can't we "automate" this part? Can't we "learn" good features directly from raw inputs?

Feature extraction converts raw inputs to a numeric representation that the ML algo can understand and work with. More on feature extraction later.



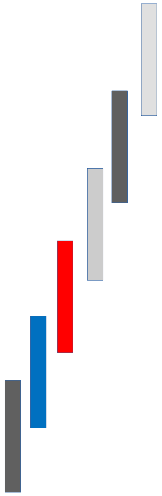

Indeed. Deep Learning algos do precisely that! (feature + model learning). More on Deep Learning later

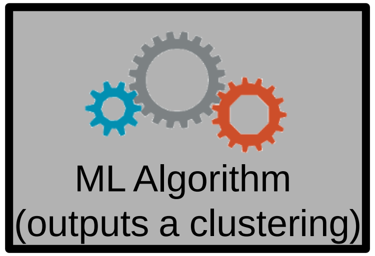
# A Typical Unsupervised Learning Workflow

Note: This example is for the problem of data clustering, an unsupervised learning problem

Unlabeled Data

"Feature" Extraction






Yes. In this example, given a new "test" cat/dog image, we can assign it to the cluster with closer centroid

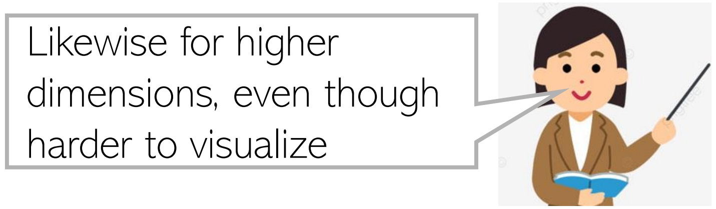
Does unsupervised learning also have a test phase? That is, can we also predict the cluster of a new test input?

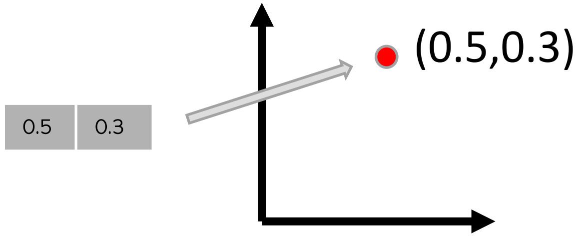
# Notation and Convention

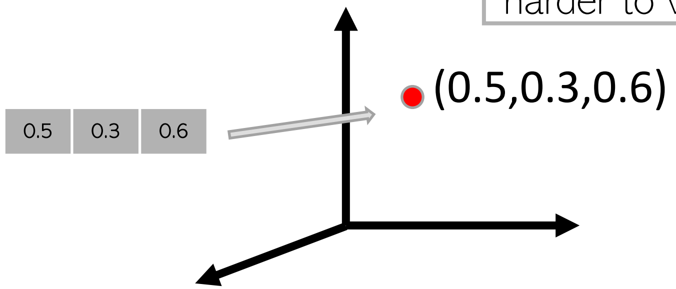
- In ML, inputs are usually represented by vectors
- A vector consists of an array of scalar values

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0.5 | 0.3 | 0.6 | 0.1 | 0.2 | 0.5 | 0.9 | 0.2 | 0.1 | 0.5 |

- Geometrically, a vector is just a point in a vector space, e.g.,
  - A length 2 vector is a point in 2-dim vector space
  - A length 3 vector is a point in 3-dim vector space




- Unless specified otherwise
  - Small letters in bold font will denote vectors, e.g., $\mathbf{x}, \mathbf{a}, \mathbf{b}$ etc.
  - Small letters in normal font to denote scalars, e.g. $x, a, b$, etc
  - Capital letters in bold font will denote matrices (2-dim arrays), e.g., $\mathbf{X}, \mathbf{A}, \mathbf{B}$, etc

# Notation and Convention

- A single vector will be assumed to be of the form $\mathbf{x} = [x_1, x_2, \dots, x_D]$

- Unless specified otherwise, vectors will be assumed to be column vectors
  - So we will assume $\mathbf{x} = [x_1, x_2, \dots, x_D]$ to be a column vector of size $D \times 1$
  - Assuming each element to be real-valued scalar, $\mathbf{x} \in \mathbb{R}^{D \times 1}$ or $\mathbf{x} \in \mathbb{R}^D$ ($\mathbb{R}$: space of reals)

- If $\mathbf{x} = [x_1, x_2, \dots, x_D]$ is a feature vector representing, say an image, then
  - $D$ denotes the dimensionality of this feature vector (number of features)
  - $x_i$ (a scalar) denotes the value of $i^{th}$ feature in the image

- For denoting multiple vectors, we will use a subscript with each vector, e.g.,
  - N images denoted by N feature vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_\mathrm{N}$, or compactly as $\{\mathbf{x}_n\}_{n=1}^N$
  - The vector $\mathbf{x}_n$ denotes the $n^{th}$ image
  - $x_{ni}$ (a scalar) denotes the $i^{th}$ feature ($i = 1,2, \dots, D$) of the $n^{th}$ image

# Notation and Convention

- Sup. learning requires training data as $N$ input-output pairs $\{(\mathbf{x_n}, y_n)\}_{n=1}^{N}$

- Unsupervised learning requires training data as $N$ inputs $\{\mathbf{x_n}\}_{n=1}^{N}$

RL and other flavors of ML problems also use similar notation

- Each input $\mathbf{x_n}$ is (usually) a vector containing the values of the features or attributes or covariates that encode properties of the object it represents, e.g.,
  - For a $7 \times 7$ image: $\mathbf{x_n}$ can be a $49 \times 1$ vector of pixel intensities

Size or length of the input $\mathbf{x_n}$ is commonly known as data/input dimensionality or feature dimensionality

- (In sup. learning) Each $y_n$ is the output or response or label associated with input $\mathbf{x_n}$ (and its value is known for the training inputs)
  - Output can be a scalar, a vector of numbers, or even a structured object (more on this later)

# Some Basic Operations on Vectors

- Addition/subtraction of two vectors gives another vector of the same size

- The mean $\boldsymbol{\mu}$ (average or centroid) of $N$ vectors $\{\mathbf{x}_n\}_{n=1}^{N}$

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$ (of the same size as each $\mathbf{x}_n$)

- The inner/dot product of two vectors $\boldsymbol{a} \in \mathbb{R}^D$ and $\boldsymbol{b} \in \mathbb{R}^D$

Assuming both $\boldsymbol{a}$ and $\boldsymbol{b}$ have unit Euclidean norm

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^\top \boldsymbol{b} = \sum_{i=1}^{D} a_i b_i$$ (a real-valued number denoting how "similar" $\boldsymbol{a}$ and $\boldsymbol{b}$ are)

- For a vector $\boldsymbol{a} \in \mathbb{R}^D$, its Euclidean norm is defined via its inner product with itself

$$||\boldsymbol{a}||_2 = \sqrt{\boldsymbol{a}^\top \boldsymbol{a}} = \sqrt{\sum_{i=1}^{d} a_i^2}$$

- Also the Euclidean distance of $\boldsymbol{a}$ from origin
- Note: Euclidean norm is also called L2 norm

# Computing Distances

- Euclidean (L2 norm) distance between two vectors $\boldsymbol{a} \in \mathbb{R}^D$ and $\boldsymbol{b} \in \mathbb{R}^D$

$$d_2(\boldsymbol{a}, \boldsymbol{b}) = ||\boldsymbol{a} - \boldsymbol{b}||_2 = \sqrt{\sum_{i=1}^{D} (a_i - b_i)^2}$$

Another expression in terms of inner products of individual vectors

$$= \sqrt{(\boldsymbol{a} - \boldsymbol{b})^\top (\boldsymbol{a} - \boldsymbol{b})} = \sqrt{\boldsymbol{a}^\top \boldsymbol{a} + \boldsymbol{b}^\top \boldsymbol{b} - 2\boldsymbol{a}^\top \boldsymbol{b}}$$

- Other types of distances can be defined too, such as L1 norm

$$d_1(\boldsymbol{a}, \boldsymbol{b}) = ||\boldsymbol{a} - \boldsymbol{b}||_1 = \sum_{i=1}^{D} |a_i - b_i|$$

- Even more general type of distances can be defined

$\mathbf{W}$ is a DxD diagonal matrix with weights $\boldsymbol{w}_i$ on its diagonals. Weights may be known or even learned from data (in ML problems)

$$d_w(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{\sum_{i=1}^{D} w_i (a_i - b_i)^2} = \sqrt{(\boldsymbol{a} - \boldsymbol{b})^\top \mathbf{W} (\boldsymbol{a} - \boldsymbol{b})}$$

# Computing Similarities

- Can also define similarity between two vectors $\boldsymbol{a} \in \mathbb{R}^D$ and $\boldsymbol{b} \in \mathbb{R}^D$
- Basically, opposite of distance
- For defining similarity, can use any function that gives
  - High value when $\boldsymbol{a}$ and $\boldsymbol{b}$ are close/similar
  - Small value when $\boldsymbol{a}$ and $\boldsymbol{b}$ are far/dissimilar
- Some examples
  - Dot product or cosine similarity
  - Kernel/similarity functions, such as the RBF ("radial basis function" or "Gaussian") kernel

Kernel functions like this provide a "nonlinear" similarity function unlike the standard dot product which is a linear similarity function (more on this later)

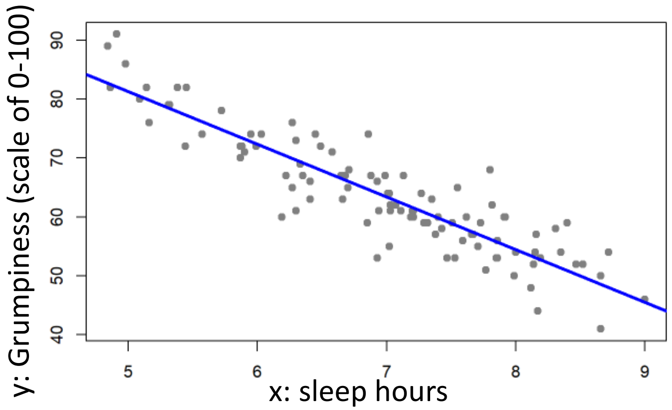
$\gamma$ is called the bandwidth hyperparameter of this kernel function

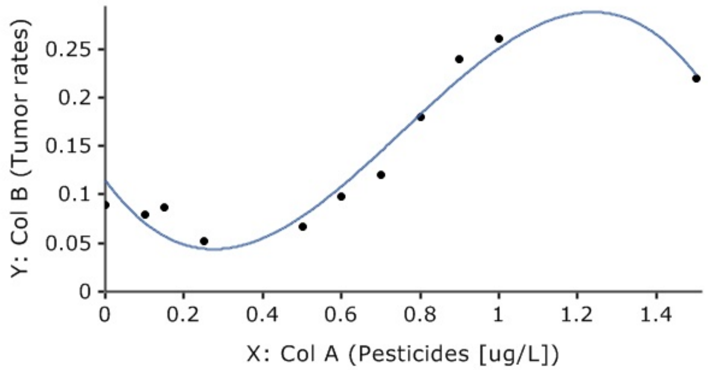
$$k(\boldsymbol{a}, \boldsymbol{b}) = \exp(-\gamma||\boldsymbol{a} - \boldsymbol{b}||^2)$$

# Remember: ML Problems can be viewed Geometrically

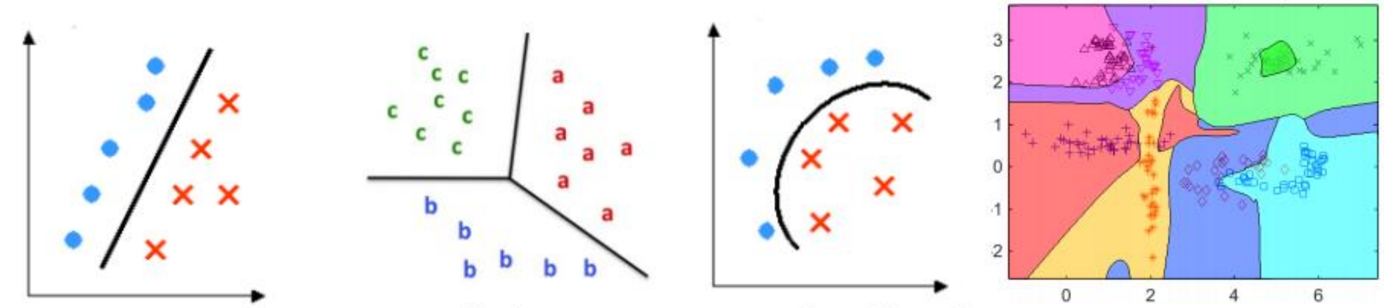
- Inputs can often be represented as points or vectors in some vector space
- Doing ML on such data can thus be seen from a geometric view. For example,

**Regression:** A supervised learning problem. Goal is to model the relationship between input (x) and real-valued output (y). This is akin to a line or curve fitting problem




**Classification:** A supervised learning problem. Goal is to learn a to predict which of the two or more classes an input belongs to. Akin to learning linear/nonlinear separator for the inputs



Two-Class (binary) Linear Classification | Multi-Class Linear Classification | Two-Class (binary) Nonlinear Classification | Multi-Class Nonlinear Classification

Pic from: https://learningstatisticswithr.com/book/regression.html, https://maxstat.de/