Section 1: Introduction

DO NOT POLLUTE! AVOID PRINTING, OR PRINT 2-SIDED MULTIPAGE.

1.1 Introduction

Artificial intelligence (AI) can be used for a wide variety of tasks. For example:

- Predict stock prices.
- Identify who may develop a disease (diabetes, cancer, Alzheimer, etc.).
- Identify which genes trigger a certain disease.
- Understand how species of bacteria and viruses evolve, and how they react to antibiotics and drugs.
- Classify things. For example, people's faces, images, sounds, or walk patterns.
- Understand brain diseases by analyzing images.
- Predict solar flares.
- Identify fraud (credit, clicks, identity, etc.).
- Identify objects in images.
- Targeted advertisement (recommender systems).

Most of these tasks can be broadly classified in two main categories: supervised, and unsupervised learning.

1.2 Supervised Learning

The main idea is to provide a computer with enough *training examples* (a.k.a. *instances*, or *samples*, or *data points*) so that the computer *learns* to perform a task on its own and can replicate it on new data. For example, imagine we want to classify edible vs. poisinous mushrooms. Our data may look like a collection of images of known mushrooms, together with a *label* indicating whether they are edible or poisonous:

Edible 

Poisonous

The goal is to find the common patterns, so that we can determine whether a new mushroom not in our *training dataset* is edible or poisonous:

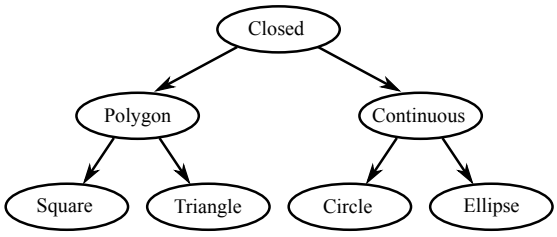
Edible or poisonous?

To this end we need some way to represent each sample. One of the most common ways is to use data arrays, often called *vectors*, where each entry/coordinate corresponds to a different *feature* (a.k.a. *attributes* or *variables*). Choosing the right features is a crucial step towards learning. In our example, we could use the following features to represent each mushroom: cap shape, cap color, stem length, bruises, and odor. Then we can represent the mushrooms in our data with the following *feature vectors*:

$$\begin{aligned}
\mathbf{x}_1 &= [bell,\ purple,\ 1",\ true,\ foul] \\
\mathbf{x}_2 &= [flat,\ yellow,\ 1.5",\ false,\ musty] \\
&\vdots \\
\mathbf{x}_8 &= [bell,\ red,\ 2",\ true,\ musty]\,.
\end{aligned}$$

Here variables may be of several *types*. Most common ones include:

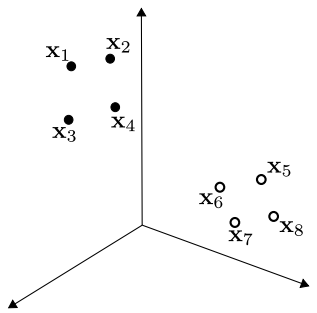
- **Nominal** (including boolean), e.g., color $\in \{red,\ green,\ blue\}$, or DNA nucleotides $\in \{Adenine,\ Cytosine,\ Guanine,\ Thymine\}$.
- **Ordinal**, e.g., size $\in \{small,\ medium,\ large\}$.
- **Numeric**, e.g., weight $\in [0, 500]$.
- **Hierarchical**, e.g.: shapes can take the following values:



Nominal variables offer no ordering among values, so it is often convenient to assign them numerical representatives. The same is true about other non-numerical values. In our mushroom example, we can map *bell* $\mapsto 1$, $flat \mapsto 2$, *purple* $\mapsto 1$, *yellow* $\mapsto 2$, *red* $\mapsto 3$, *true* $\mapsto 1$, $false \mapsto 0$, $foul \mapsto 1$, *musty* $\mapsto 2$, etc., so that now mushrooms can be represented in the following fully numerical feature vectors:

$$\begin{aligned}\mathbf{x}_1 &= [1, 1, 1, 1, 1] \\ \mathbf{x}_2 &= [2, 2, 1.5, 0, 2] \\ &\vdots \\ \mathbf{x}_8 &= [1, 3, 2, 1, 2]\,.\end{aligned}$$

This way we can think of each sample as a point in a D-dimensional *feature space*, where D is the number of features:



Feature spaces open the door to assessing similarities between samples, and finding patterns. For example, one option to determine how similar mushrooms i and j are is to compute the distance between their feature vectors $\|\mathbf{x}_\mathrm{i} - \mathbf{x}_\mathrm{j}\|$. To perform computations like these in bulk, it is often convenient to use matrix operations, so data is often equivalently stored in the form of a matrix whose rows are the feature vectors. In our example this would be:

$$\mathbf{X} \;=\; \begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_8\end{bmatrix} \;=\; \begin{bmatrix}1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 1.5 & 0 & 2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 3 & 2 & 1 & 2\end{bmatrix}.$$

In general we will use the standard notations in Table 1.1 to make it clear whether something is a scalar, vector, or matrix.

Mathematically, *supervised learning* aims to find a function $f^\star$ that produces the desired output when given a feature vector $\mathbf{x}$ as input. In our running example, $f^\star$ would be the function that correctly determines whether a mushroom is edible or poisonous. To this end we use a collection of *training pairs*. Each training pair $(\mathbf{x}_\mathrm{i}, \mathrm{y}_\mathrm{i})$ is formed by the feature vector $\mathbf{x}_\mathrm{i}$, and its corresponding label y_i, which is equal to $f^\star(\mathbf{x}_\mathrm{i})$. Typically, the primary objective is *generalization*, that is, the ability to predict the label y of previously unseen feature vectors $\mathbf{x}$.

	Examples	Regular	Bold	Lower	Capital	Roman	Script
Scalar	x, X	✓		✓	✓	✓	
Vector	$\mathbf{x}$		✓	✓		✓	
Matrix	$\mathbf{X}$		✓		✓	✓	
Random variable	x	✓		✓			✓
Random vector	$\boldsymbol{x}$		✓	✓			✓
Random matrix	$\boldsymbol{X}$		✓		✓		✓

Table 1.1: Standard notations.

1.2.1 Flavors and Jargon

Most of the time the *labels* y_i's are scalars, but they can also be multivariable. If y_i's are discrete, we say we have a *classification* task. If y_i's are continuous we say we have a *regression* task. If all training pairs are available at once, we say we are in a *batch* setting. On the other hand, if training pairs become available in a sequential manner, we say we are in an *online* setting. Because there exist infinitely many functions, we often want to restrict our search to a family of functions. This family of functions determine the type of Machine Learning algorithm we will be using. For example, *linear regression* searches over *linear* functions, *decision trees* search over *tree* functions, and *deep learning* searches over *neural network* functions. Each of these functions has parameters. For example, linear functions have a *slope* and an *offset* as parameters (see Figure 1.1 to build some intuition). The goal of *learning* is to find the parameters that yield the function that best predicts the response as a function of the features.

1.3 Unsupervised Learning

The main difference between supervised and *unsupervised learning* is that in the later we do *not* have access to the labels y_i's, and the goal is to rather discover interesting regularities/structures/patterns that characterize the data. Some common examples of these tasks include *clustering*, *anomaly detection*, and *dimensionality reduction*.

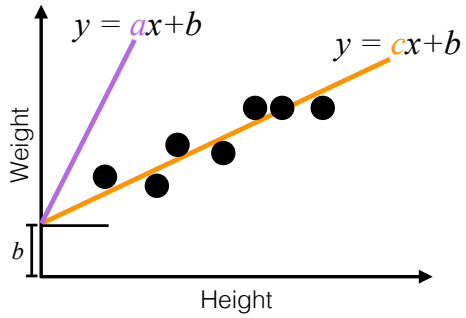
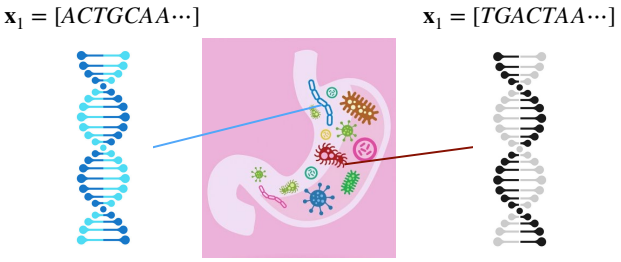


Figure 1.1: Circles represent samples (feature vectors) of the height and weight of people in a dataset. Each line has a slope and an offset parameter. Here, the slope of the violet line is a, the slope of the orange line is c, and the offset of both lines is b. Our goal is to learn the parameters that produce the line that best explains our data, so that it is more likely to make more accurate predictions. Which of the two lines would you choose?

For example, in *metagenomics* one has a collection of *reads* (small DNA segments) from several taxa (e.g., virus or bacteria) in a microbiome (e.g., human gut), but does not know a priori which reads correspond to the same specie/strain.

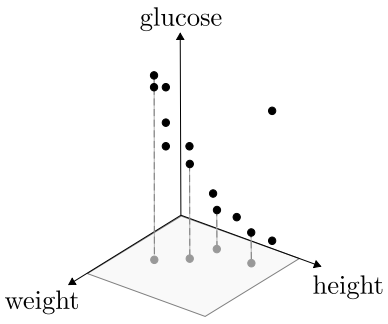


In other words, we have no labels. The goal is to cluster reads according to their taxa, with applications in soil health, human health, plant health, sustainability, climate change, and more.

For another example consider *electronic health records*, with datasets like the following:

Sample	**Height**	**Weight**	**Age**	**Sex**	**Glucose**	$\cdots$	**Cholesterol**
1	5'10"	150	30	M	145	$\cdots$	110
2	5'7"	200	50	F	195	$\cdots$	140
3	6'1"	180	25	M	150	$\cdots$	100
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
N	5'5"	145	40	F	205	$\cdots$	160

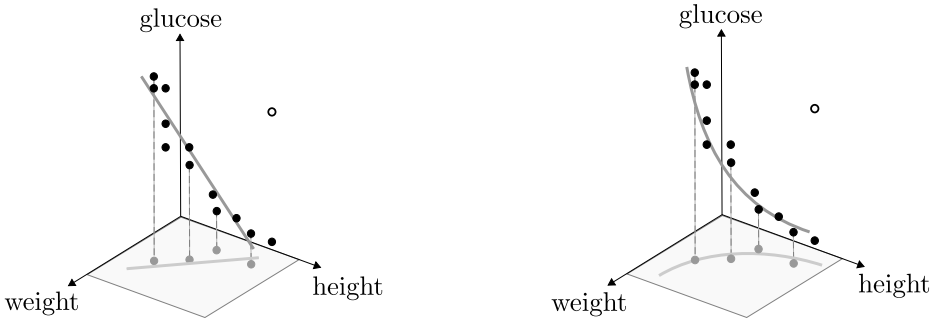
As before, each individual is represented by its feature vector. For example, $\mathbf{x}_2 = [5'7", 200, 50, F, 195, \cdots, 140]$. Geometrically, each feature vector can be thought of as a point in D-dimensional space (sadly I only know how to draw stuff in 3 dimensions, but you get the idea):



Datasets like these pose several unsupervised tasks, for example:

(a) **Anomaly Detection.** Can we identify outliers?

(b) **Dimensionality Reduction.** Can we identify a pattern, (e.g., linear or polinomial) that summarizes the dataset in just a few variables?

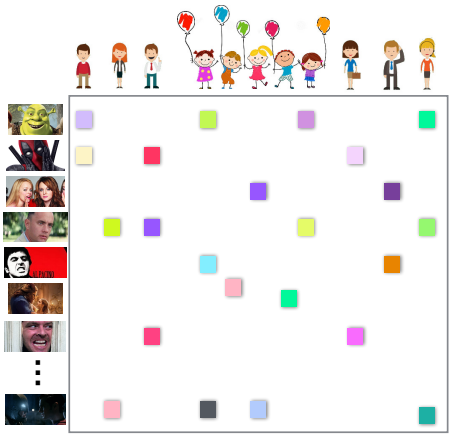


1.4 Other Motivating Applications

Example 1.1 (Recommender systems)**.** Amazon, Netflix, Pandora, Spotify, Pinterest, Yelp, Apple, etc., keep information of their users, such as age, sex, income level, and very importantly, ratings of their products. The information of the i^{th} user can be arranged as a vector:

$$\mathbf{x}_i = \Big[\text{age},\ \text{sex},\ \text{income},\ \text{rating of item 1},\ \text{rating of item 2},\ \cdots,\ \text{rating of item D} - 3\Big].$$

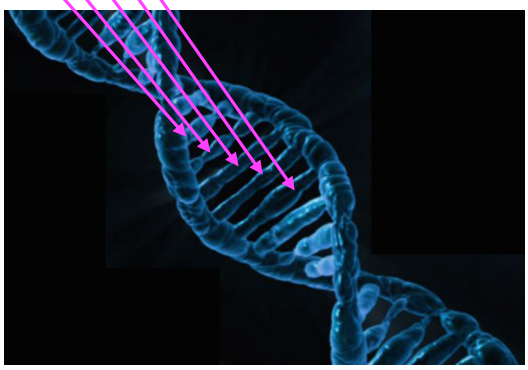
In this sort of problem we want to discover which variables (e.g., sex, age, income, etc.) can predict which items (e.g., movies, shoes, songs, etc.) users will like, in order to make good recommendations. This can be done by finding structures (e.g., *lines* or *curves*) in high-dimensions that explain the data. This structure (e.g., low-rankness or union of subspaces) can be later use to complete the matrix containing the preferences of all users, which is often known as *matrix completion*.



If Amazon recommends you an item you will like, you are more likely to buy it. You can see why all these companies have a great interest in this problem, and they are paying *a lot* of money to people who work on this.

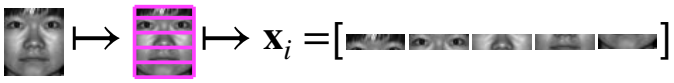
Example 1.2 (Genomics)**.** The genome of each individual can be stored as a vector containing its corresponding sequence of nucleotides

$$[\cdots A\ T\ G\ C\ T\ \cdots] \mapsto \mathbf{x}_i = [\cdots\ 1\ 4\ 3\ 2\ 4\ \cdots]$$

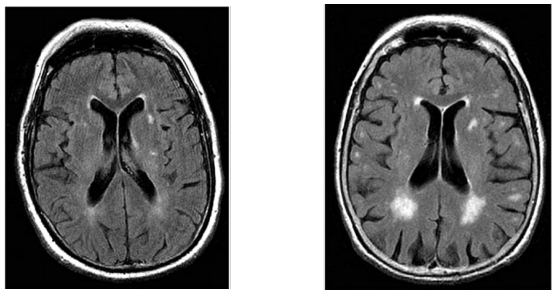


In this sort of problem we want to analyze these data vectors to determine which genes are correlated to which diseases (or features, like height or weight).

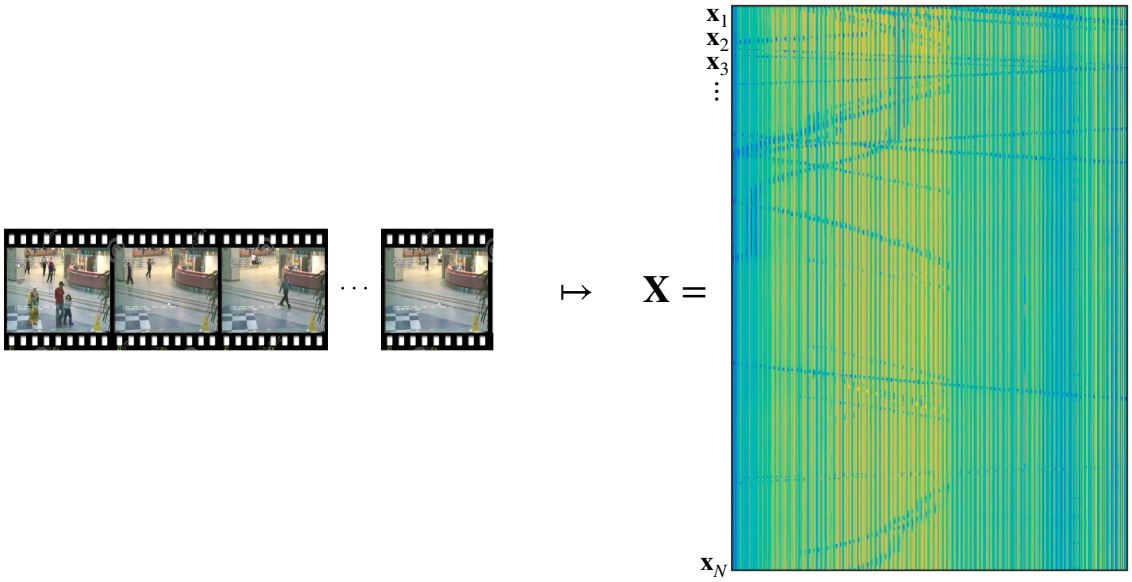
Example 1.3 (Image processing)**.** A $m \times n$ grayscale image can be stored in a data matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ whose $(i,j)^{th}$ entry contains the gray intensity of pixel (i,j). Furthermore, $\mathbf{X}$ can be *vectorized*, i.e., we can concatenate its rows to form a vector $\mathbf{x}$, with $D = mn$.



We want to analyze these vectors to interpret the image, identify the objects contained in it, classify faces, and medical diagnosis. For instance, can you tell which of these magnetic resonance images (MRIs) corresponds to an individual with Alzheimer?

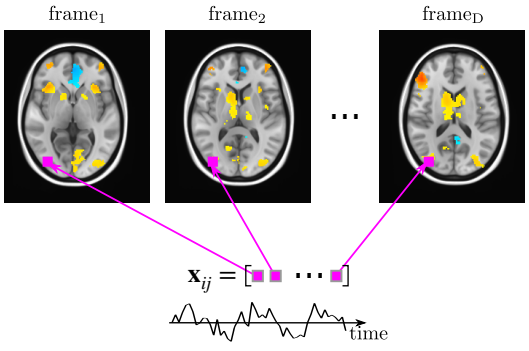


Example 1.4 (Computer vision)**.** The images $\mathbf{X}_1, \ldots, \mathbf{X}_N \in \mathbb{R}^{m \times n}$ that form a video can be vectorized to obtain vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^D$.



Similar to image processing, we want to analyze these vectors to interpret the video. For example, be able to distinguish background from foreground, track objects, etc. This has applications in surveillance, defense, robotics, etc.

Example 1.5 (Neural activity)**.** *Functional magnetic resonance imaging* (fMRI) generates a series of MRI images over time. Because oxygenated and deoxygenated hemoglobin have slightly different magnetic characteristics, variations in the MRI intensity indicate areas of the brain with increased blood flow and hence neural activity. The central task in fMRI is to reliably detect neural activity at different spatial locations (pixels) in the brain. The measurements over time at the $(i, j)^{th}$ pixel can be stored in a data vector $\mathbf{x}_{ij} \in \mathbb{R}^D$.

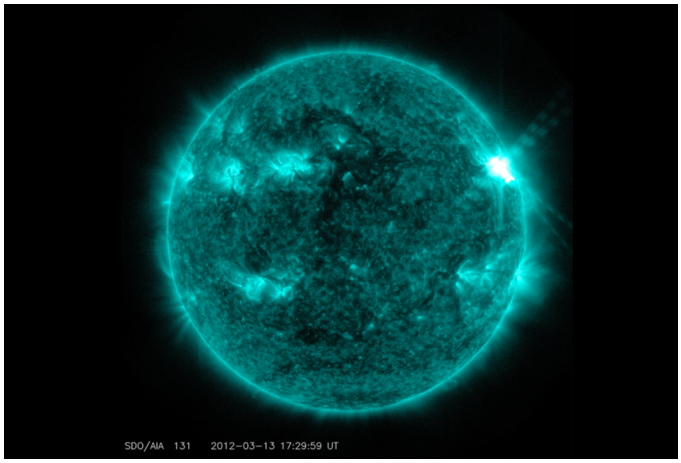


The idea is to analyze these vectors to determine the active pixels. This can help better understand how the brain works.

Example 1.6 (Sun flares)**.** The Sun, like all active stars, is constantly producing huge electromagnetic *flares*. Every now and then, these flares hit the Earth. Last time this happened was in 1859, and all that happened was that you could see the northern lights all the way down to Mexico — not a bad secondary effect! However, back in 1859 we didn't have a massive power grid, satellites, wireless communications, GPS, airplanes, space stations, etc. If a flare hits the Earth now, all these systems would be crippled, and repairing them could take *years* and would cost *trillions* of dollars to the U.S. alone! To make things worse, it turns out that these flares are not rare at all! It is estimated that the chance that a flare hits the earth in the next decade is about 12%.

Of course, we cannot stop these flares any more than we can stop an earthquake. If it hits us, it hits us. However, like with an earthquake, we can act ahead. If we know that one flare is coming, we can turn everything off, let it pass, and then turn everything back on, like nothing happened. Hence the NASA and other institutions are investing a great deal of time, effort and money to develop techniques that enable us to *predict* that a flare is coming.

So essentially, we want to device a sort of flares *radar* or *detector*. This radar would receive, for example, an image $\mathbf{X}$ of the sun (or equivalently, a vectorized image $\mathbf{x} \in \mathbb{R}^{\mathrm{D}}$), and would have to decide whether a flare is coming or not.



Example 1.7 (Fraud detection)**.** Credit cards are a classical example of fraud detection. The main idea is to look at usage patterns, and identify *outliers* (unusual activity).

For a more interesting example, consider *click fraud*. Companies pay popular websites to advertise their products. How much they pay depends on the popularity of each website, measured in number of clicks. Hence, companies often cheat, using *bots* that click their websites (to have a higher click count, and charge more for advertising). How would you detect whether a click is genuine or fraudulent?