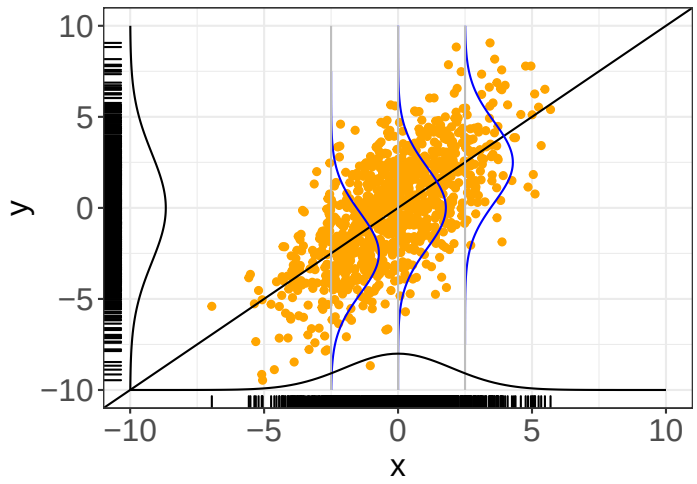
Introduction to Machine Learning

ML-Basics: Data



Learning goals

- Understand structure of tabular data in ML
- Understand difference between target and features
- Understand difference between labeled and unlabeled data
- Know concept of data-generating process

IRIS DATA SET

Introduced by the statistician Ronald Fisher and one of the most frequently used toy examples.

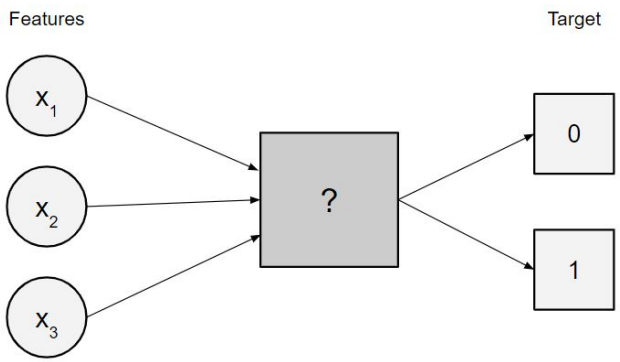
- Classify iris subspecies based on flower measurements.
- 150 iris flowers: 50 versicolor, 50 virginica, 50 setosa.
- Sepal length / width and petal length / width in [cm].

Source: `https://rpubs.com/vidhividhi/irisdataeda`

Word of warning: "iris" is a small, clean, low-dimensional data set, which is very easy to classify; this is not necessarily true in the wild.

DATA IN SUPERVISED LEARNING

- The data we deal with in supervised learning usually consists of observations on different aspects of objects:
  - **Target**: the output variable / goal of prediction
  - **Features**: measurable properties that provide a concise description of the object
- We assume some kind of relationship between the features and the target, in a sense that the value of the target variable can be explained by a combination of the features.



Features x				Target y
Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4.3	3.0	1.1	0.1	setosa
5.0	3.3	1.4	0.2	setosa
7.7	3.8	6.7	2.2	virginica
5.5	2.5	4.0	1.3	versicolor

ATTRIBUTE TYPES

- Both features and target variables may be of different data types
  - **Numerical** variables can have values in $\mathbb{R}$
  - **Integer** variables can have values in $\mathbb{Z}$
  - **Categorical** variables can have values in $\{C_1, ..., C_g\}$
  - **Binary** variables can have values in $\{0, 1\}$
- For the **target** variable, this results in different tasks of supervised learning: *regression* and *classification*.
- Most learning algorithms can only deal with numerical features, although there are some exceptions (e.g. decision trees can use integers and categoricals without problems). For other feature types, we usually have to pick or create an appropriate **encoding**, i.e., cast them to numerical values.
- If not stated otherwise, we assume numerical features.

ENCODING FOR CATEGORICAL FEATURES

- We typically choose one of the following encoding schemes for a variable x with g mutually exclusive categories:
  - **One-hot encoding**: g dummy variables, each assuming values from {0, 1}, are used to represent x with only one at a time $\neq 0$.
    E.g., $x \in \{\text{black, white}\} \mapsto x_{\text{black}} \in \{0, 1\}, x_{\text{white}} \in \{0, 1\}$
  - **Dummy encoding**: Like one-hot encoding but more parsimonious, using just $g - 1$ dummies and thus cutting the redundancy inherent to the one-hot representation (which some algorithms that cannot deal with non-singular input matrices, such as linear regression learners, demand).
    E.g., $x \in \{\text{black, white}\} \mapsto x_{\text{black}} \in \{0, 1\}$
- We effectively create a vector $o(x) = [\mathbb{I}(x = k)]_{k=1,2,\ldots,\tilde{g}} \in \{0, 1\}^{\tilde{g}}$.
- For features with a natural order in their categories we might resort to different encodings that reflect this ordinality (e.g., a sequence of integer values – note that equidistant values signify a linear ordering here.)

ENCODING FOR CATEGORICAL FEATURES

- We effectively expand the representation of a variable x with c mutually exclusive categories from one to $\tilde{c}$ columns in the design matrix, each element being a **binary dummy variable** assuming values from {0, 1}.
- Every dummy is treated as a separate variable.
- x is thus represented by a length-$\tilde{c}$ vector with at most one 1-element and zeroes otherwise: $\boldsymbol{o}(x) = [\mathbb{I}(x = j)]_{j=1,2,\ldots,\tilde{c}} \in \{0, 1\}^{\tilde{c}}$.
- Two popular ways to do this are
  - **One-hot encoding**: $\tilde{c} = c$ dummies, so *exactly one* element is 1 ("hot"). E.g., $x \in \{a, b, c\} \mapsto \boldsymbol{o}(x) = (x_a, x_b, x_c)$, with $x_a = x_b = 0, x_c = 1$ and $\boldsymbol{o}(x) = (0, 0, 1)$ for $x = c$.
  - **Dummy encoding**: $\tilde{c} = c - 1$ dummies, so *at most one* element is 1, cutting the redundancy of one-hot encoding (necessary for learners that require non-singular input matrices, such as in linear regression). E.g., $x \in \{a, b, c\} \mapsto \boldsymbol{o}(x) = (x_a, x_b)$ for reference category c, with $x_a = x_b = 0$ and $\boldsymbol{o}(x) = (0, 0)$ for $x = c$.
- For features with a natural **order** in their categories we might resort to different encodings that reflect this ordinality (e.g., a sequence of integer values – note that equidistant values signify a linear ordering here.)

OBSERVATION LABELS

- We call the entries of the target column **labels**.
- We distinguish two basic forms our data may come in:
  - For **labeled** data we have already observed the target
  - For **unlabeled** data the target labels are unknown

	Features x				Target y
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
labeled data	4.3	3.0	1.1	0.1	setosa
	5.0	3.3	1.4	0.2	setosa
	7.7	3.8	6.7	2.2	virginica
	5.5	2.5	4.0	1.3	versicolor
unlabeled data	5.9	3.0	5.1	1.8	?
	4.4	3.2	1.3	0.2	?

NOTATION FOR DATA

In formal notation, the data sets we are given are of the following form:

$$\mathcal{D} = \left(\left(\mathbf{x}^{(1)}, y^{(1)} \right), \ldots, \left(\mathbf{x}^{(n)}, y^{(n)} \right) \right) \in (\mathcal{X} \times \mathcal{Y})^n.$$

We call

- $\mathcal{X}$ the input space with $p = \dim(\mathcal{X})$ (for now: $\mathcal{X} \subset \mathbb{R}^p$),
- $\mathcal{Y}$ the output / target space,
- the tuple $\left(\mathbf{x}^{(i)}, y^{(i)} \right) \in \mathcal{X} \times \mathcal{Y}$ the i-th observation,
- $\mathbf{x}_j = \left(x_j^{(1)}, \ldots, x_j^{(n)} \right)^T$ the j-th feature vector.

We denote

- $(\mathcal{X} \times \mathcal{Y})^n$, i.e., the set of all data sets of size n, as $\mathbb{D}_n$,
- $\bigcup_{n \in \mathbb{N}} (\mathcal{X} \times \mathcal{Y})^n$, i.e., the set of all finite data sets, as $\mathbb{D}$.

So we have observed n objects, described by p features.

DATA-GENERATING PROCESS

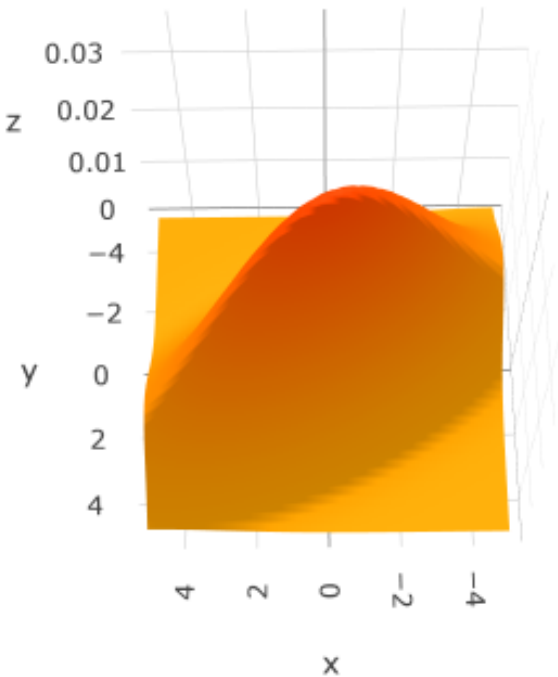
- We assume the observed data $\mathcal{D}$ to be generated by a process that can be characterized by some probability distribution

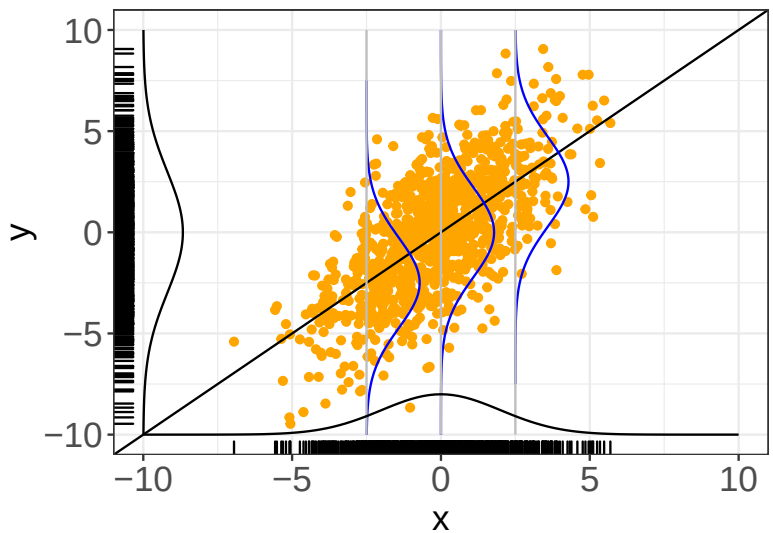
$$\mathbb{P}_{xy},$$

  defined on $\mathcal{X} \times \mathcal{Y}$.
- We denote the random variables following this distribution by lowercase **x** and y.
- It is important to understand that the true distribution is essentially **unknown** to us. In a certain sense, learning (part of) its structure is what ML is all about.

DATA-GENERATING PROCESS

- We assume data to be drawn *i.i.d.* from the joint probability density function (pdf) / probability mass function (pmf) $p(\mathbf{x}, y)$.
  - i.i.d. stands for **i**ndependent and **i**dentically **d**istributed.
  - This means: We assume that all samples are drawn from the same distribution and are mutually independent – the i-th realization does not depend on the other $n - 1$ ones.
  - This is a strong yet crucial assumption that is precondition to most theory in (basic) ML.





DATA-GENERATING PROCESS

Remarks:

- With a slight abuse of notation we write random variables, e.g., $\mathbf{x}$ and y, in lowercase, as normal variables or function arguments. The context will make clear what is meant.
- Often, distributions are characterized by a parameter vector $\boldsymbol{\theta} \in \Theta$. We then write $p(\mathbf{x}, y \mid \boldsymbol{\theta})$.
- This lecture mostly takes a frequentist perspective. Distribution parameters $\boldsymbol{\theta}$ appear behind the | for improved legibility, not to imply that we condition on them in a probabilistic Bayesian sense. So, strictly speaking, $p(\mathbf{x}|\boldsymbol{\theta})$ should usually be understood to mean $p_{\boldsymbol{\theta}}(\mathbf{x})$ or $p(\mathbf{x}, \boldsymbol{\theta})$ or $p(\mathbf{x}; \boldsymbol{\theta})$. On the other hand, this notation makes it very easy to switch to a Bayesian view.