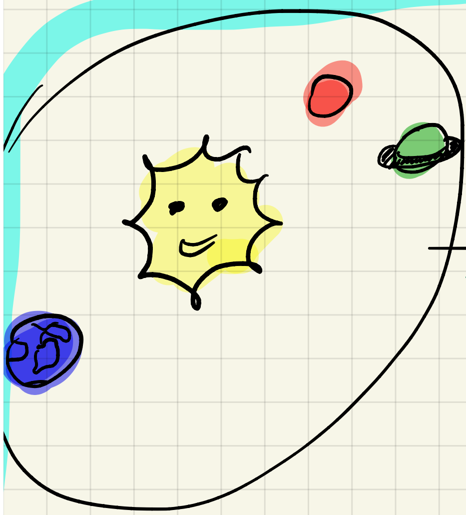
ML Setup

Supervised Learning:



Universe

Consists of everything

feature extraction → $\vec{x} \in \mathbb{R}^d$ → $y \in \mathcal{Y}$

Model/Program/Hypothesis

$h(\vec{x}) \approx y$

Input

h

What we learn

Output

feature vector

consists of the statistics that you deem relevant and obtainable to predict y.

label

The thing we would like to predict (from $\vec{x}$).

Stuff we don't know.

Stuff we do know

$\mathcal{Y} = \mathbb{R}$: Regression

$\mathcal{Y} = \{0,1\}$: Binary Classification

$\mathcal{Y} = \{1,2,\dots,k\}$: Multiclass classification

Examples:
- Predict if the Coca-Cola stock will go up tomorrow.
- Predict if an email is Spam or not.
- — " — if a photo contains a human face.
- — " — what a user said to a home assistant device (e.g. Alexa)

Goal: Learn h from available data.

Ingredients:

labeled Data: D

$D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)\} \sim P^n$

where we know $(\vec{x}, y) \sim P$ ← unknown distribution, i.i.d.!

Hypothesis Class $\mathcal{H}$:

a set of possible functions $h: \mathcal{X} \to \mathcal{Y}$

loss function ℓ:

$\ell: \mathcal{H} \to \mathbb{R}_0^+$ to tell us how bad any $h \in \mathcal{H}$ is

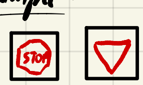
Algorithm $\mathcal{A}$:

To pick a good $h \in \mathcal{H}$ for D

Data Scientist:

To extract features, to choose pick $\mathcal{A}$

Example:



: 'stop' 'yield'

convolutional neural networks

Cross-entropy loss

Stochastic Gradient Descent

ML Stages and Concepts:

Learning / Training

Use $\mathcal{A}$ to find $h \in \mathcal{H}$ with low loss, $\mathcal{L}(h)$, on training data D.

Inference / Testing

For some testing data $\vec{x}$, not in the training data, predict the label $y = h(\vec{x})$

Train and Test data must be drawn i.i.d. from the same distribution P.

Training Data:

Data used to find $h \in \mathcal{H}$

gives rise to the training loss:

$$\boxed{\frac{1}{|D|}\sum_{(\vec{x},y)} \ell(h(\vec{x}),y)}$$

Testing Data:

Data used to evaluate h.

approximates:

Generalization loss:

$$\boxed{E_{(\vec{x},y)\sim P}\left[\ell(\vec{x},y)\right]}$$

WLLN

Typically you split your data $\approx$ 80/20 into train/test.

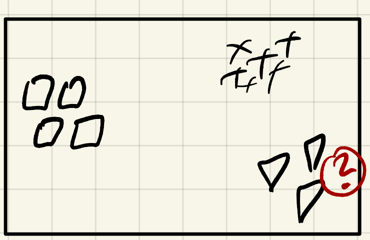
(Often people split into train/validation/test. Why?)
80% 10% 10%

Typical ways to split your data:

- uniformly at random
- by time (eg. Ja, Fe, Ma | Ap)
- by patient / instance

Standard rule: Simulate the test case.
Never predict the past from the future!

Assumptions:



No free lunch Theorem: You must make assumptions in order to learn.

$\Rightarrow$ there is no single ML algorithm that works for all settings.

Example assumptions:
- data is locally smooth
- — " — consists of natural images
- P is a mixture of Gaussians
- Features are independent given the label