How to validate AI algorithms

[How do I use turing test in practise]

- We get the data
- We train our algorithms

Q. How do you know it has learned what it was supposed to learn ?

Ans. We should test it.

If good enough, the release

First factor to make sure that it is good enough is to have a target/objective function.

- error ↓
- reward ↑
- fitness ↑
- punishment ↓

Give the entire data $X = \{x_t\}$

$$\underbrace{\underbrace{\text{input/features}}_{\text{unsupervised}} \;\Big|\; \text{Output (Desired)}\ x_t^d}_{\text{supervised}}$$

Given the set of all hypotheeses H (set of of all possible solutions), find 1 Hypothesis s.t $\sum_{x_t \in X} (x_t^{*} - x_t^{d}) = \varepsilon \longrightarrow 0$

This constitutes a good fit for the mode h into X.
Supervised   hypothesis

X   h

train

↓

It doesn't converge

Relationship with output can't described with a line

Problem is big / non-linear / non-Stationary

eg. 5000 features
not how many data points
Complexity of feature space

position of the solution changes every time

Solution (hypothesis) is not capable of capturing the complexity.

X   h

train

↓

fast convergence ⟶ crappy solution
most of time

problem – linear / stationary / small
solution – too big s.t completely owns the problem

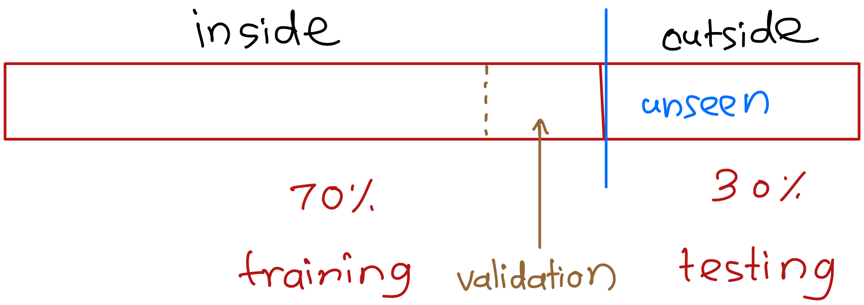
⟶ h is memorizing X

What is the ultimate sign that algorithm has really learnt ?

→ It can generalize the inherent x-y relationship to unseen data

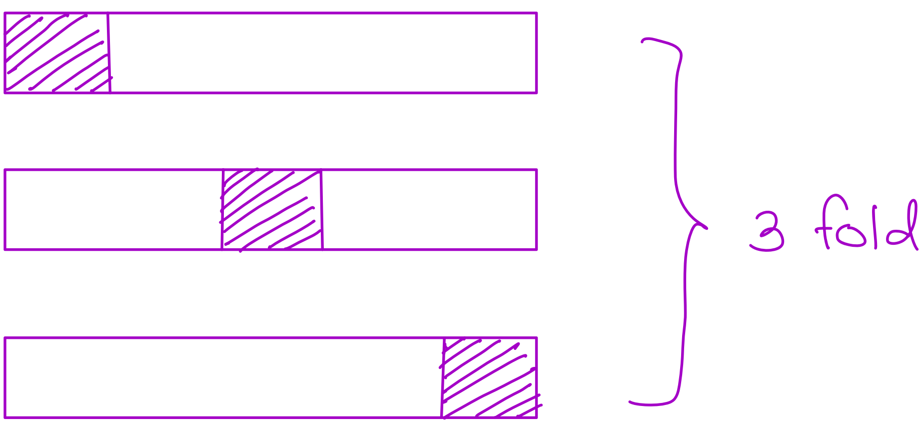
Validation = test for generalization

Idea = keep one part of the data for testing



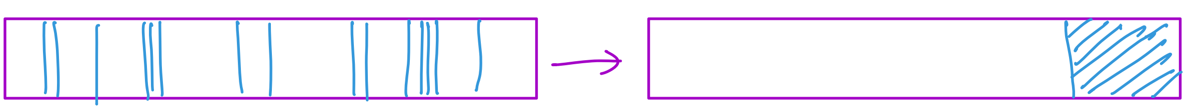
But this may not be reliable
The split may be luck, unfortunate

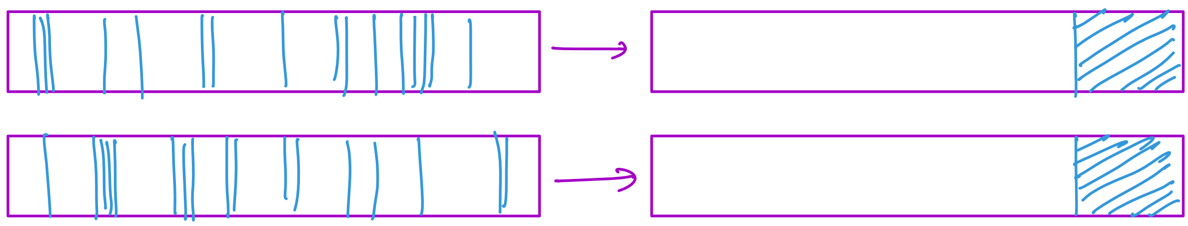
K-fold partioning



Random sampling

K-fold cross validation

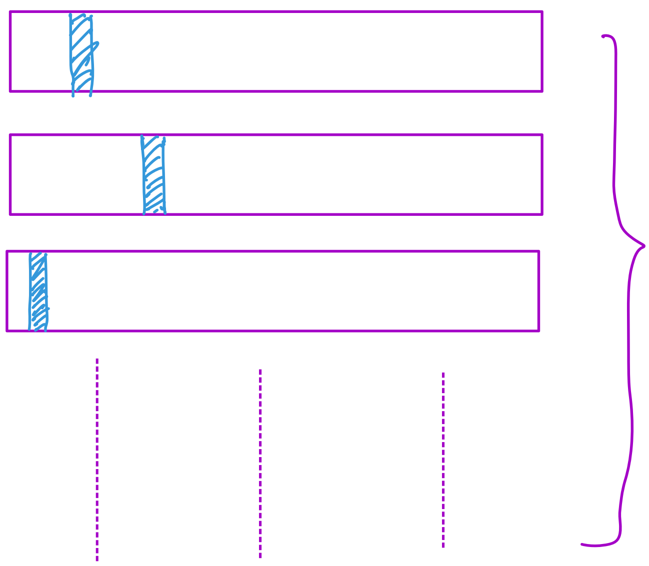




But all this would work if we had a lots of data. what if we dont ?

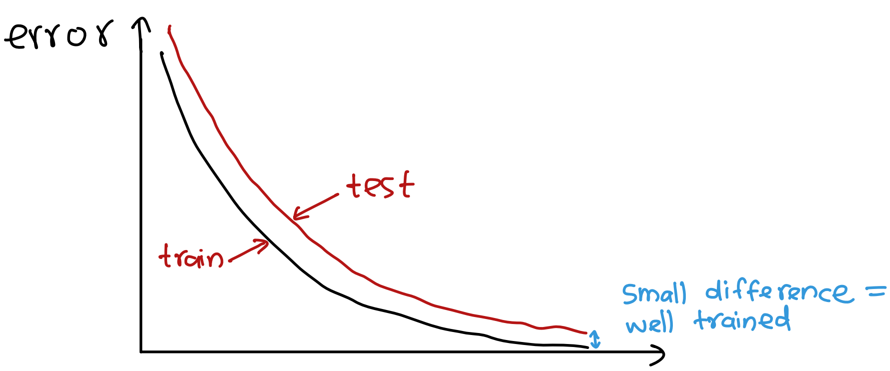
n data samples $\langle$ 1 for testing / n-1 for training

$|X| = n \longrightarrow$ n-fold cross validation

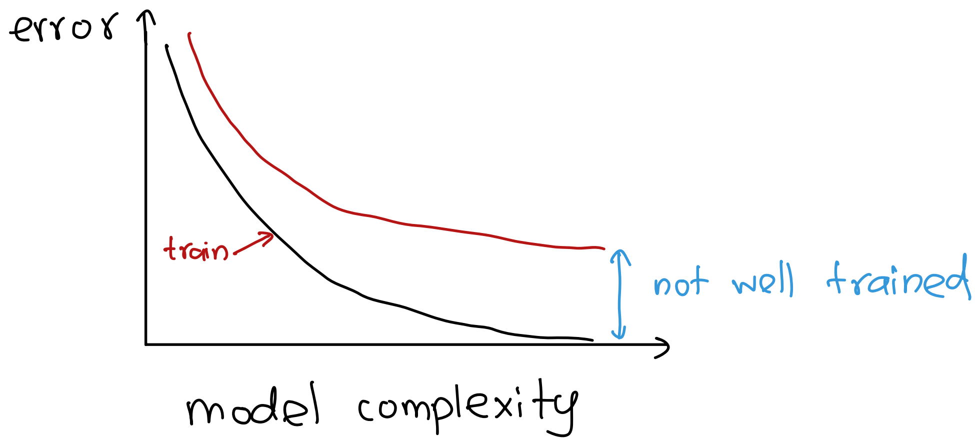


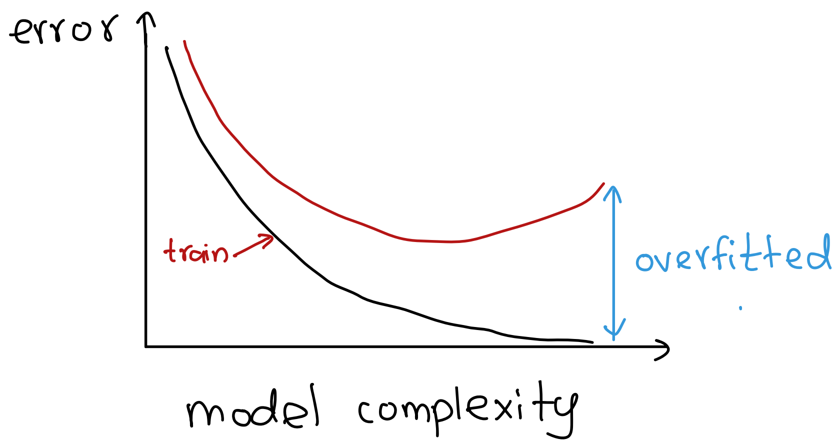
n times $\longrightarrow$ very expensive

* suitable for small data



model complexity





model complexity $= |P|$

P is set of parameters of $h \in H$ ← universe of discourse

Occam's Razor (old)

→ keep it simple = Regularization

$$\text{minimize} \left[\sum_{x_t \in X} (x_t^* - x_t^d) + |P| \right]$$

Augmented error function

$$E' = E_{total} + \lambda \cdot (\text{model complexity})$$

penalize complex solutions with large variance

λ too large ⟶ small solutions ⟶ increase bias

So hence we use cross validation to optimize λ

1. Basian approach is used if we have prior knowledge.

how many times that model is a good model

Bayes Rule :

$$P(\text{model} \mid \text{data}) = \frac{P(\text{data} \mid \text{model}) * P(\text{model})}{P(\text{data})}$$

frequency of data

2. Structural Risk minimization

3. Minimum description length

small solution ⟵ good solution ⟶ big solution

bias ⟵ good solution ⟶ high variance