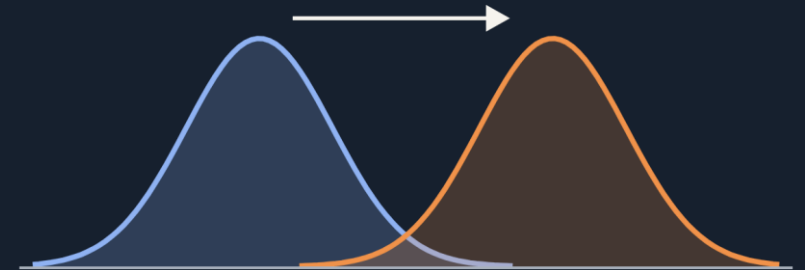


When the future doesn't look like the past

Why a robot that worked in the lab fail in your home?



ABOUT ME

Hello, I'm Julio Hurtado

Where I work

Innovation Research Associate -
Centre for Applications of Mathematical &
Computing Sciences (CAMAcs)

What I work on

Transfer knowledge from academia to the world
Multidisciplinary Projects

Why this topic

Continual Learning – Model
adaptation – Robustness



Meet the Model



The camera sees an object



The classifier decides:
 $P(y | x)$



Pick it up

Leave it alone

- **Pick it up:** toys, socks, clothes
- **Leave it alone:** cables, pet bowls, plants, rocks

94%

test accuracy in the company's lab

Trained on thousands of labelled photos of objects on lab floors, with a random train / validation / test split.



SIX MONTHS AFTER LAUNCH



"In our dark living room, it misses half the socks."

customer review, October



"It stored the Christmas lights every time we put them."

customer review, December



"Since our baby started crawling, it leaves rocks everywhere."

customer review, March

Why does a model that worked yesterday fail tomorrow?

Nothing crashed. No error message. The robot just quietly got worse.

ONE QUIET ASSUMPTION

IID Assumption:

Test data is going to be like training data

The lab is not customer home: $P_{\text{train}}(x, y) \neq P_{\text{deploy}}(x, y)$.
When the two differ, the lab test score stops telling the truth.

PART 1 · THE FAILURE

Why the 94% lied

◀ oldest data

■ train ■ test

newest data ▶

Random split

Test objects from the same homes and months as training: optimistic



Temporal split

Test on the months after launch: honest



Which one looks more like shipping a robot?

Can you think of another example?

In ML, inputs and labels come together

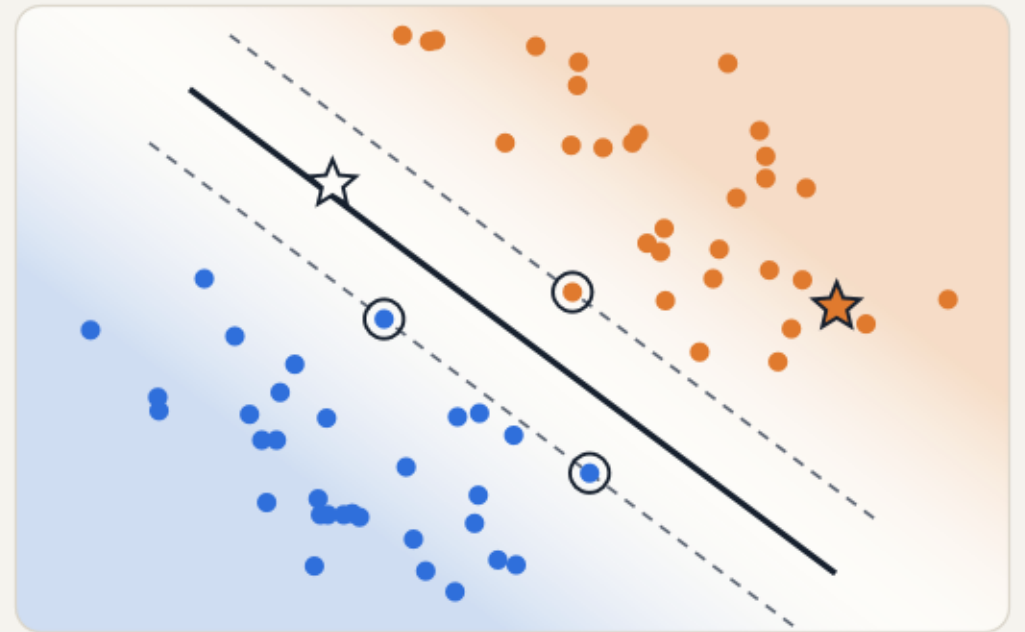
$$P(x, y) = P(y \mid x) \cdot P(x)$$

$P(x)$: what the inputs look like

Which emails arrive in the inbox.
Which patients visit the hospital.

$P(y \mid x)$: the rule

Given this email, is it spam? Given this scan, is there a tumour?
This is what the model tries to learn.



Three things can change

$$P(x, y) = P(y \mid x) \cdot P(x)$$

Covariate shift

$P(x)$ changes

What the robot sees:
Darker homes, new floors.

Label shift

$P(y)$ changes

How often each class appears:
Christmas lights are everywhere at
Christmas.

Concept drift

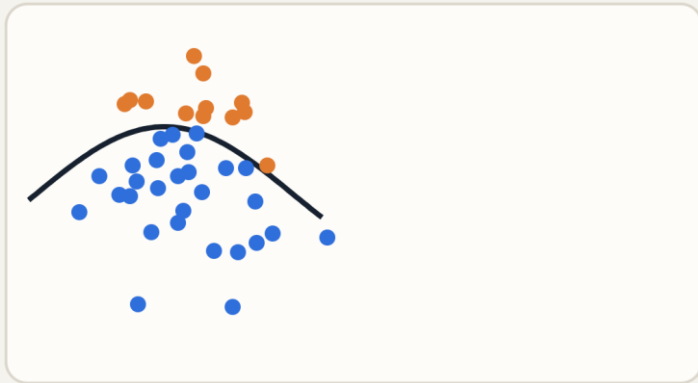
$P(y \mid x)$ changes

The right action itself:
The baby crawling change what is
needed to be picked

The robot's classifier (an SVM, a Random Forest, ...) is its **estimate of $P(y \mid x)$** : given what it sees, what should it do?

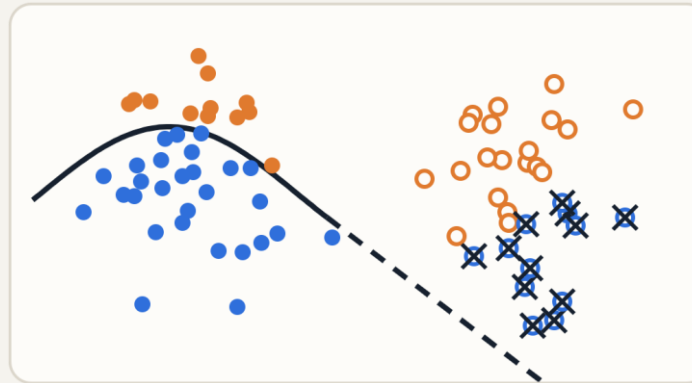
Covariate shift: new homes look different

1 · What the model learned



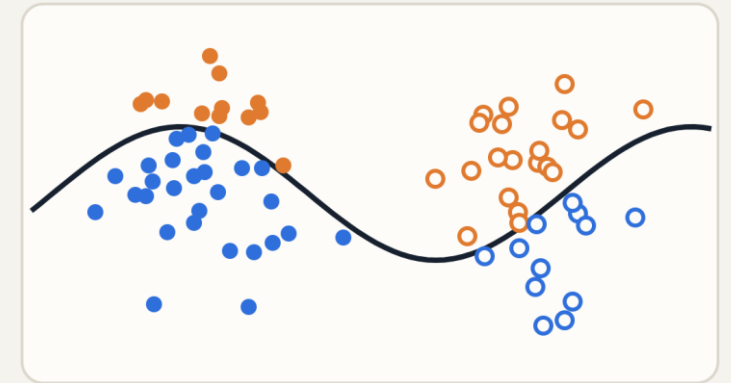
Filled: training data.
The boundary is learned only where there is data.

2 · New data arrives



Hollow: new data.
Dashed: the model's guess. X = wrong prediction.

3 · The truth



The true rule never changed.
The model just never saw this part.

$P(x)$ changes

$P(y | x)$ stays the same

EXAMPLE

Lab homes are bright. Real living rooms are darker, with wooden floors.
The same socks and cables look different to the camera.

Label shift: Christmas arrives

Training data



Deployment data



■ pick up ■ leave alone

$P(y)$ changes

$P(x | y)$ stays the same

Each object still looks the same.

But the robot learned “Christmas lights are rare”...
... and now they aren't.

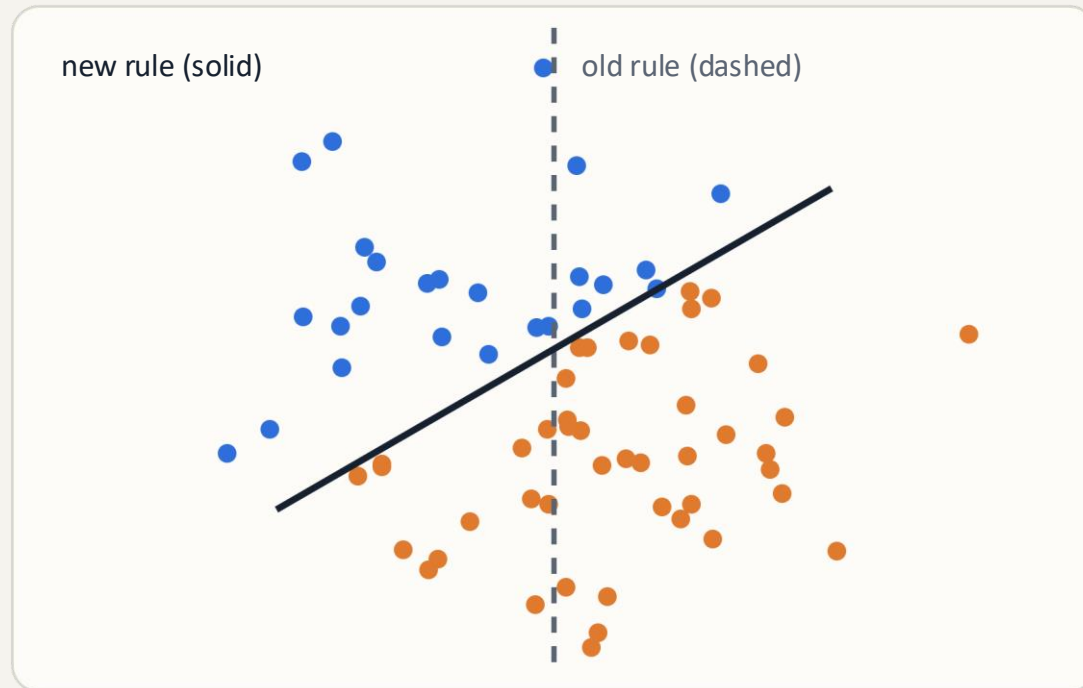
EXAMPLE

Fairy lights everywhere.

Each looks like any other cable, but there are far more of them.

Colours make them look like toys

Concept drift: the baby starts crawling



$P(y | x)$ changes

Same objects, new right action.

Data from before the baby cannot fix this

EXAMPLE

A rocks on the floor used to be “leave it alone”. Now that the baby crawls, it must be picked up.

Drift also differs in how fast it happens



Sudden

The family moves house.



Incremental

The camera lens slowly gets dusty.



Gradual

Interaction with the baby



Recurring

Christmas comes back every year.

Vertical axis: which situation is active. The pattern decides the fix: recurring drift punishes a robot that forgets.

Before we run it: predict!

1

Build a world we control

Two actions.

2

Train in the lab

Random Forest and SVM.
As in your labs.

3

Change the world

New homes, Christmas, the baby.

4

You vote first

Will accuracy stay, drop a little, or collapse?

Vote for each: accuracy stays the same · drops a little · collapses?

New homes

The rule never changed.

Christmas

Only the class mix changed.

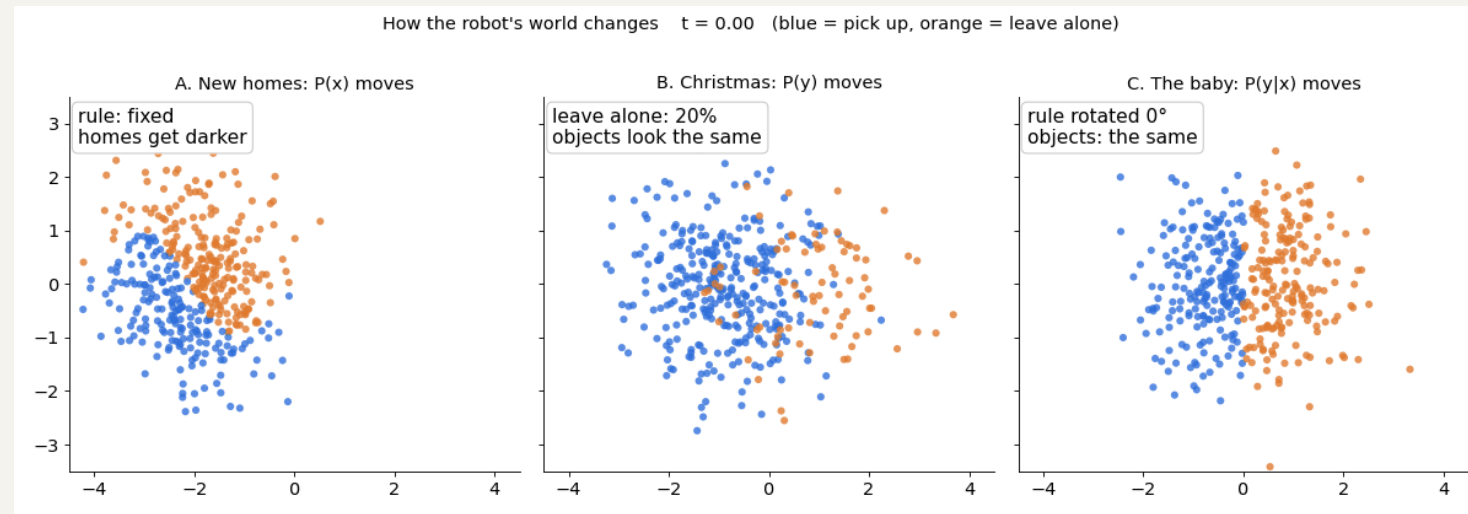
The baby

The rule itself changed.

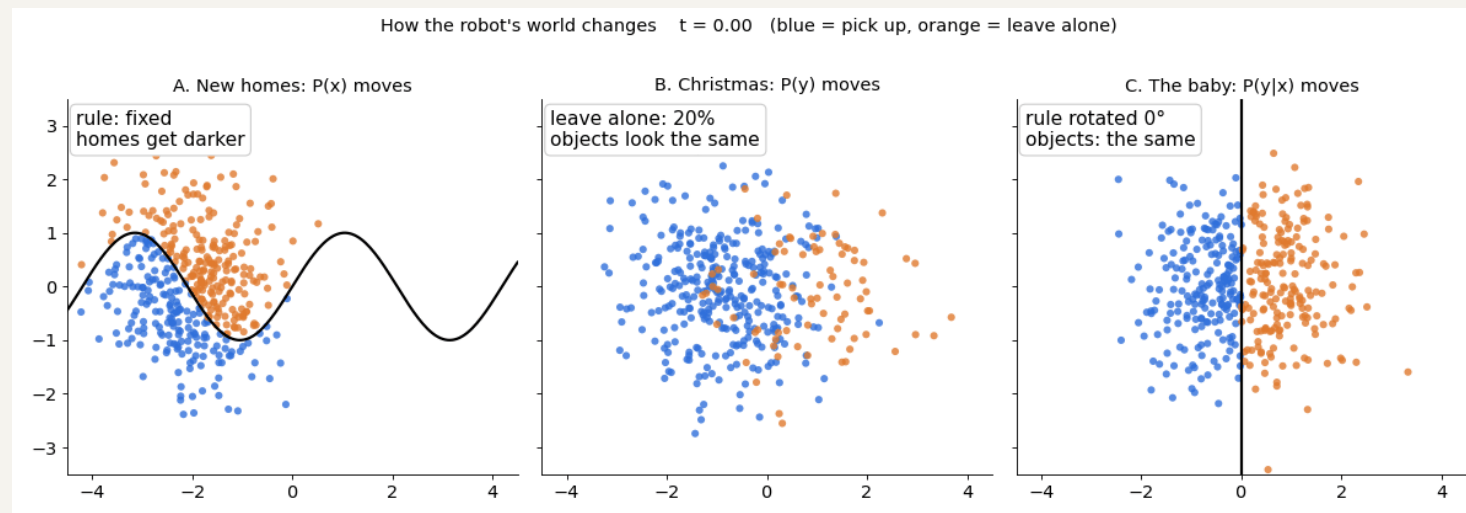
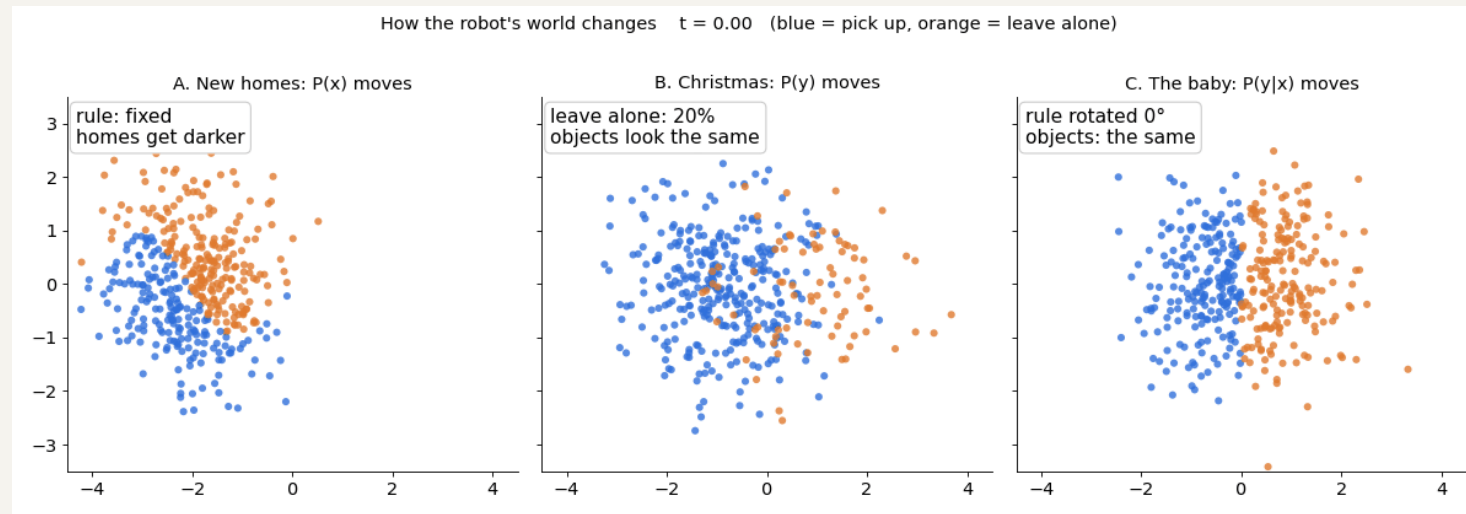
`</> distribution_shift_demo.ipynb`

PART 3 · DEMO

How the robot's world changes



How the robot's world changes



Did you predict it?

Trained once in the lab: accuracy as the robot's world changes



New homes

The rule never changed: the robot had never seen this region.

Christmas

Objects look the same, but the robot's idea of "rare" is now wrong.

The baby

Same objects, new rule: accuracy slides toward a coin flip.

What the robot learned no longer fits



Four scientific questions in deployment

1 Has the world changed?

Detect: watch the robot's mistakes, compare today's camera images with the lab's.

2 What changed?

Diagnose: what the robot sees, how often things appear, or the right action?

3 What should we update?

Respond: retrain on recent data, learn online, or learn continually.

4 How fast should we adapt?

Tune: fast enough to follow the change, slow enough to stay stable.

First, notice that it happened



Monitor performance

Track the robot's mistakes over time:
missed socks, dragged cables.
Alert when they rise.



Compare input distributions

No labels needed.
Compare condition with the lab's,
e.g. brightness.



Drift detectors

Compare rules.
Allow adaptation

Comparing images catches new homes. A new rule (the baby) only shows up through mistakes, once you have labels.

The diagnosis decides the fix

What changed	How you notice	What to update
New homes (covariate)	Camera images look different from the lab's.	Collect and label data in the new homes; keep the old homes too.
Christmas (label)	Same-looking objects, but the class mix shifts.	Re-estimate how common each class is; adjust the decision threshold.
The baby (concept)	Only through mistakes: the images look the same.	Get new labels; drop or down-weight data from before the baby.

Two families of fixes



Retrain from scratch

Throw the model away and train a new one on recent data.

The model is rebuilt.

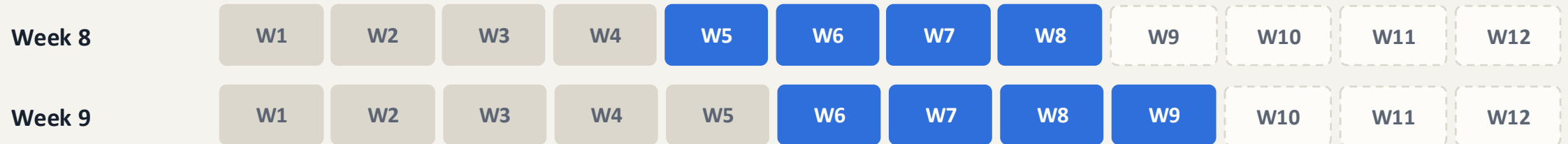


Continual / online learning

Keep the model and update it a little with new data.

The model evolves.

Retrain on a moving window



■ used for training ■ dropped: too old

Small window

Adapts quickly, but has little data and is noisy.

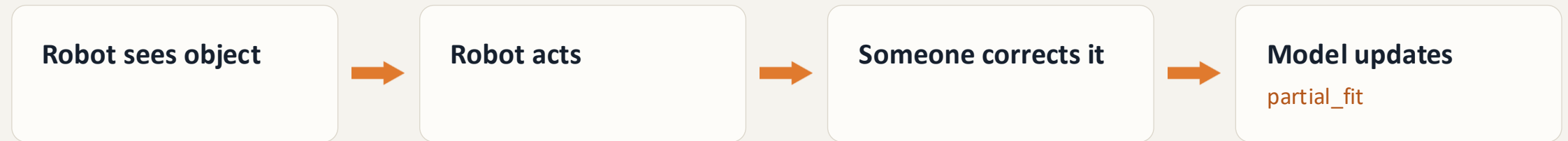
Large window

Stable, but slow to adapt and keeps outdated data.

Works with any model

Including your Random Forests and SVMs.

Online learning: update as you go



Online learning is about how data arrives: one example at a time, so the robot always follows the latest world.

Cheap and fast

A small update instead of a full retrain.

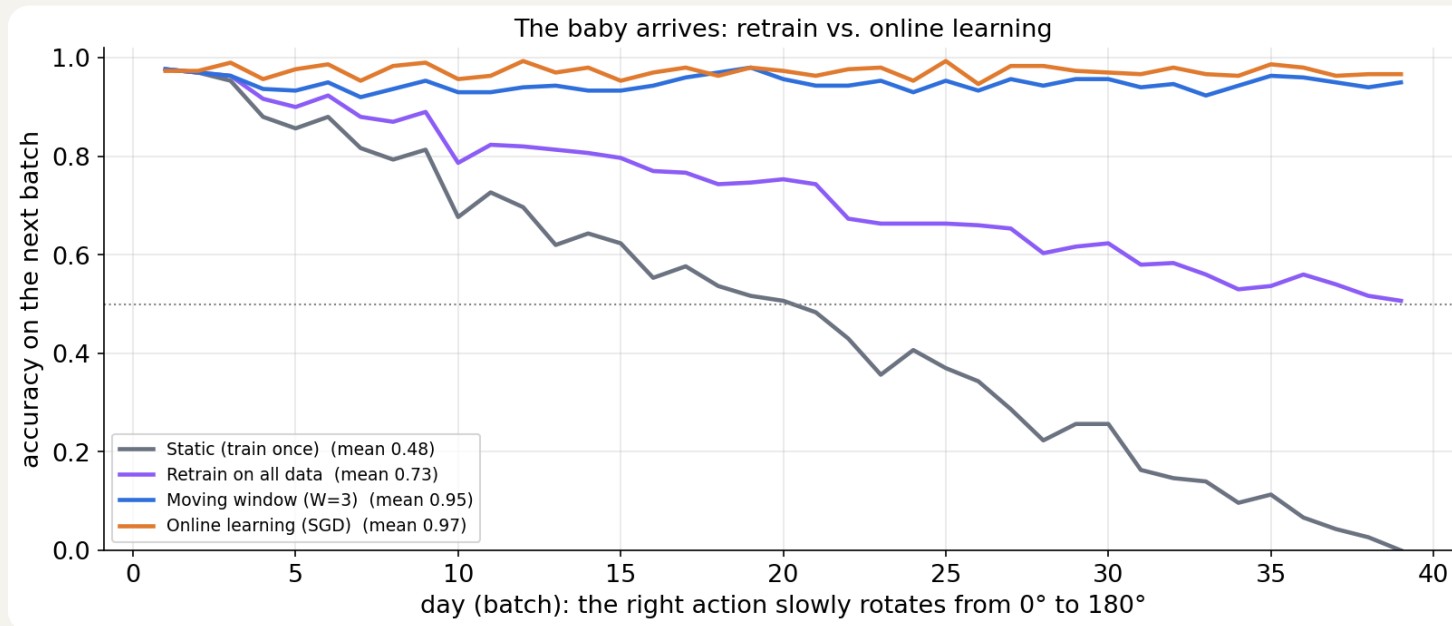
Needs the right model

Only certain models support `partial_fit`.

Streaming libraries

River offers online trees and forests

The baby: adapting beats remembering



Static

Falls to almost 0%: the rule has flipped.

Retrain on all data

Old data teaches the wrong rule.

Window and online

Keep up: here, forgetting is good.

Test-then-train: each batch is predicted first, then used for learning.

Online learning $\neq$ continual learning



Online learning

About how data arrives: one example at a time, the model is updated immediately.

Goal: follow the latest world.



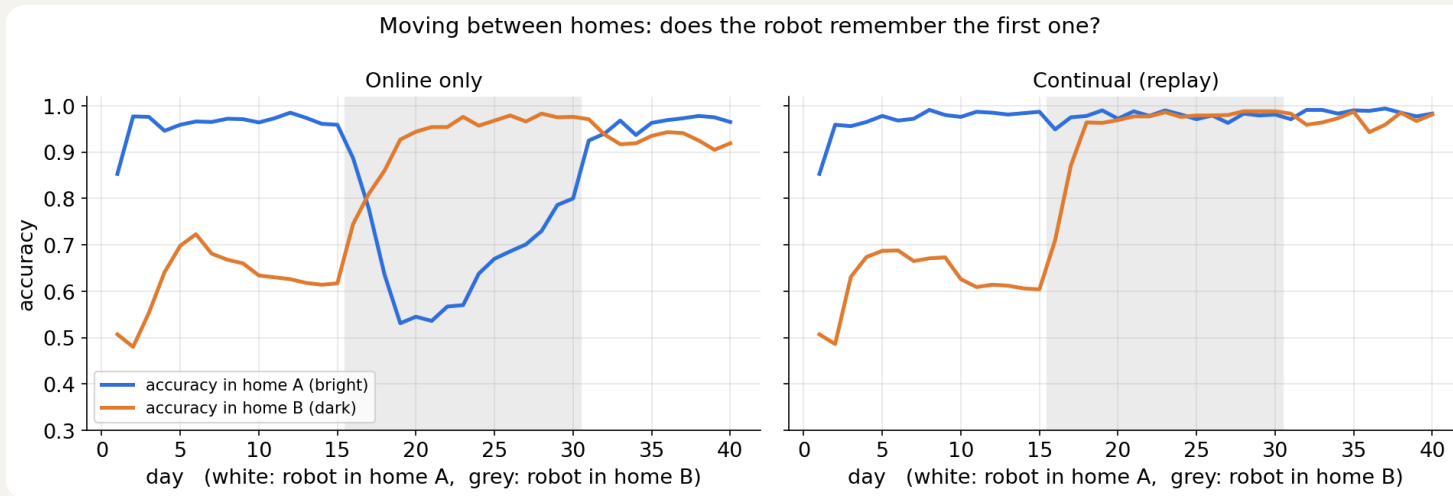
Continual learning

About what the model keeps: learn each new situation without losing the old ones.

Goal: remember worlds that come back.

The robot moves from home A to home B, then goes back to home A for Christmas. Does it still know home A?

The catch: catastrophic forgetting



Living in home B, the online robot forgot home A.

Fine if home A is gone for good. A problem when it comes back.

Continual learning balances plasticity (learn new things) and stability (keep old ones). Here, replaying a small memory of old examples keeps both homes.

Retrain or learn continually?

	Retrain (moving window)	Continual / online
Cost per update	Full retraining: high	Small update: low
Speed of adapting	At the next retrain	With every new batch
Old knowledge	Dropped outside the window	Fades: risk of forgetting
Which models	Any: RF, SVM, ...	Incremental: SGD, online trees
Main knob	Window size, retrain frequency	Learning rate

In practice, many systems combine both: small frequent updates plus a full retrain now and then.

KEY TAKEAWAYS

1 The lab is not your home.

Models assume $P_{\text{train}} = P_{\text{deploy}}$; the real world rarely agrees.

2 Ask what changed.

What the robot sees, how often things appear, or the right action?

3 Test like you deploy.

Split by a variable that makes sense for the problem, and keep monitoring after launch.

4 Choose the response that fits the change.

Retrain, update online, or learn continually, and mind the forgetting.

ONE LAST THING

Training a model is an event. Maintaining a model is a process.

The 94% in the lab was the beginning of the job, not the end.

THANK YOU

Questions?

Julio Hurtado · julio.hurtado@warwick.ac.uk

Demo notebook: [distribution_shift_demo.ipynb](#)