Machine learning: a very quick introduction

Why is learning important?

- Many AI techniques require that we know how the world works
  - Rules of puzzles
  - Rules of games
  - Logic
  - Planning
  - Probabilistic reasoning
  - Decision making
- At that point “just” need to solve/optimize
- In the real world this information is often not immediately available
- Also, some things are hard to formalize
- AI needs to be able to learn from experience
- (Sometimes learning is a good way to optimize too!)

Different kinds of learning…

- Supervised learning:
  - Someone gives us examples and the right answer (*label*) for those examples
  - We have to predict the right answer for unseen examples
- Unsupervised learning:
  - We see examples but get no feedback (no labels)
  - We need to find patterns in the data
- Semi-supervised learning:
  - Small amount of labeled data, large amount of unlabeled data
- Reinforcement learning:
  - We take actions and get rewards
  - Have to learn how to get high rewards

Example of supervised learning: classification

- We lend money to people
- We have to predict whether they will pay us back or not
- People have various (say, binary) features:
  - do we know their Address? do they have a Criminal record? high Income? Educated? Old? Unemployed?
- We see examples: (Y = paid back, N = not)

  +a, -c, +i, +e, +o, +u: Y

  -a, +c, -i, +e, -o, -u: N

  +a, -c, +i, -e, -o, -u: Y

  -a, -c, +i, +e, -o, -u: Y

  -a, +c, +i, -e, -o, -u: N

  -a, -c, +i, -e, -o, +u: Y

  +a, -c, -i, -e, +o, -u: N

  +a, +c, +i, -e, +o, -u: N
- Next person is +a, -c, +i, -e, +o, -u. Will we get paid back?

Classification…

- We want some hypothesis h that predicts whether we will be paid back

  +a, -c, +i, +e, +o, +u: Y

  -a, +c, -i, +e, -o, -u: N

  +a, -c, +i, -e, -o, -u: Y

  -a, -c, +i, +e, -o, -u: Y

  -a, +c, +i, -e, -o, -u: N

  -a, -c, +i, -e, -o, +u: Y

  +a, -c, -i, -e, +o, -u: N

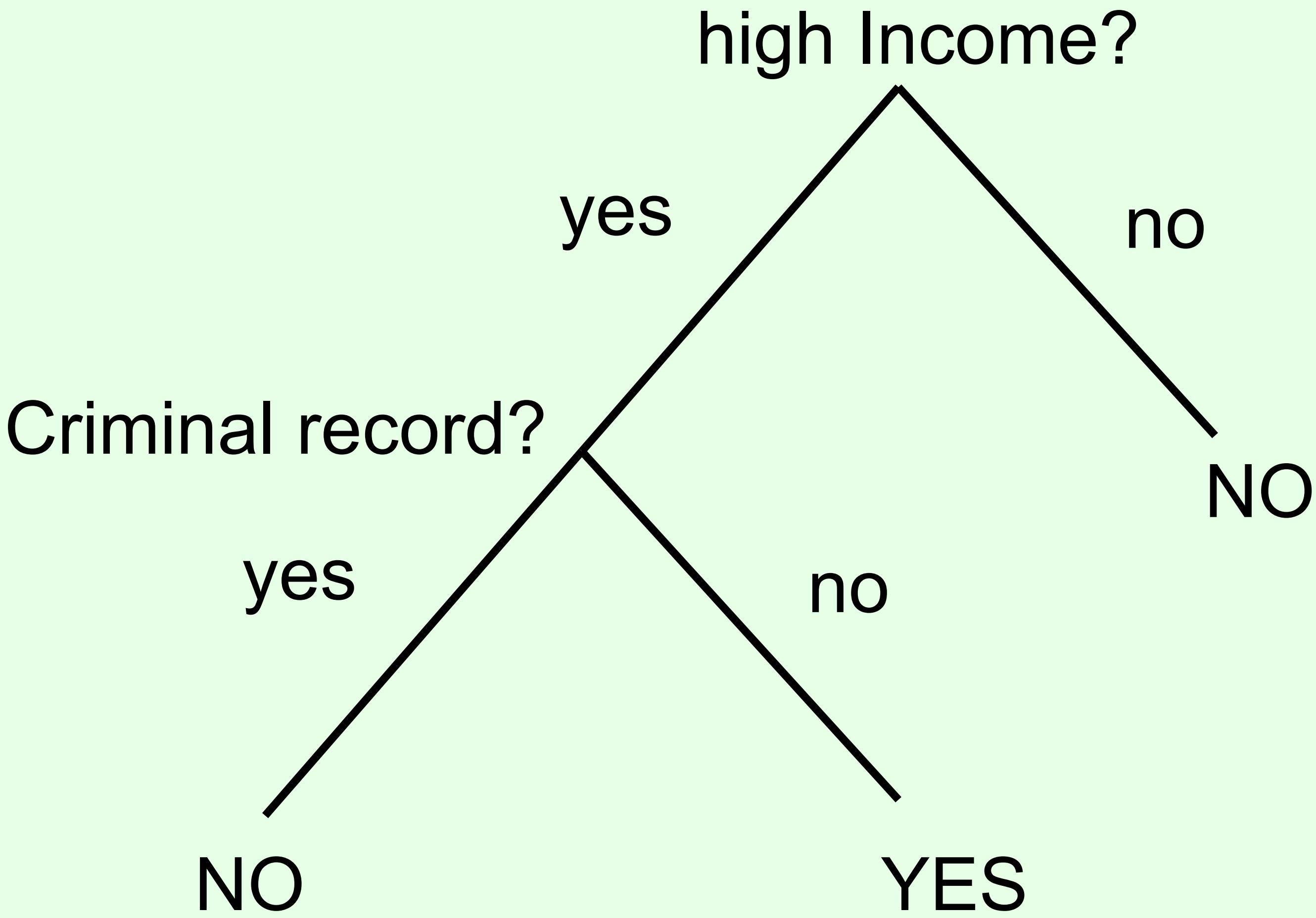
  +a, +c, +i, -e, +o, -u: N

- Lots of possible hypotheses: will be paid back if…
  - Income is high *(wrong on 2 occasions in training data)*
  - Income is high and no Criminal record *(always right in training data)*
  - (Address is known AND ((NOT Old) OR Unemployed)) OR ((NOT Address is known) AND (NOT Criminal Record)) *(always right in training data)*
- Which one seems best? Anything better?

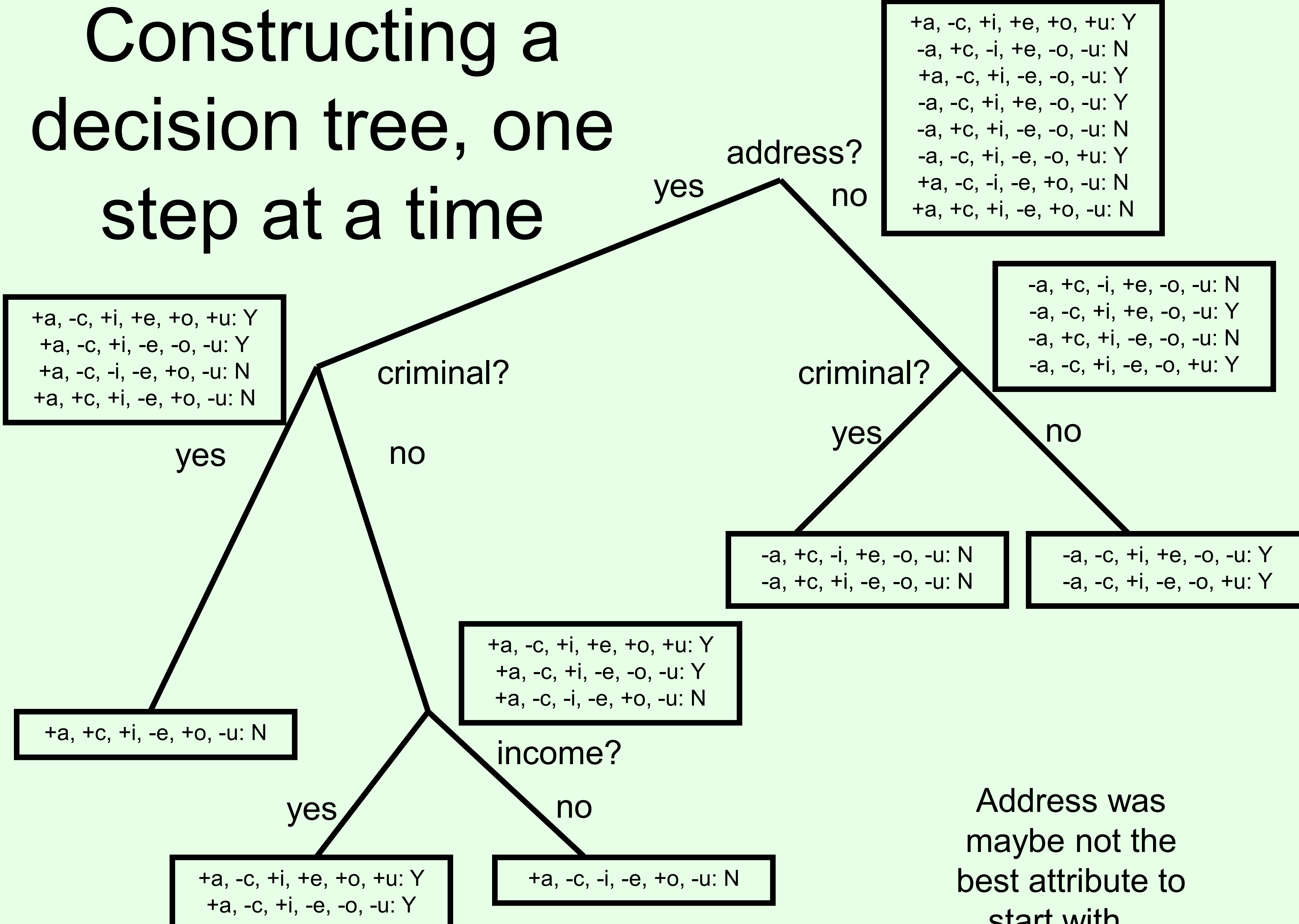
Occam's Razor

- Occam's razor: *simpler hypotheses tend to generalize to future data better*
- Intuition: given limited training data,
  - it is likely that there is some complicated hypothesis that is not actually good but that happens to perform well on the training data
  - it is less likely that there is a simple hypothesis that is not actually good but that happens to perform well on the training data
    - There are fewer simple hypotheses
- Computational learning theory studies this in much more depth

Decision trees

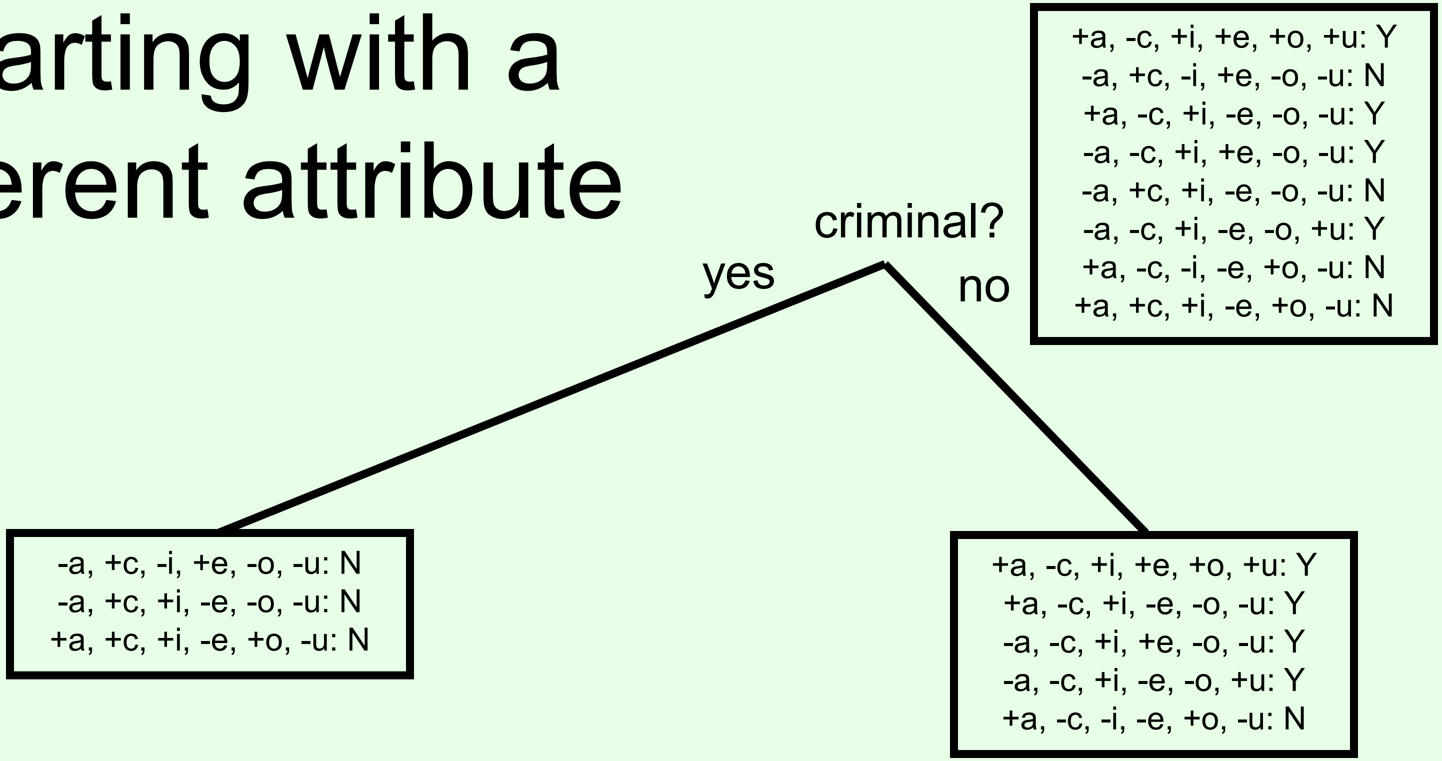


Constructing a decision tree, one step at a time



Address was maybe not the best attribute to start with…

Starting with a different attribute



- Seems like a much better starting point than address
  - Each node almost completely uniform
  - Almost completely predicts whether we will be paid back

Heuristics for which attribute to split on

+a, -c, +i, +e, +o, +u: Y
-a, +c, -i, +e, -o, -u: N
+a, -c, +i, -e, -o, -u: Y
-a, -c, +i, +e, -o, -u: Y
-a, +c, +i, -e, -o, -u: N
-a, -c, +i, -e, -o, +u: Y
+a, -c, -i, -e, +o, -u: N
+a, +c, +i, -e, +o, -u: N

criminal?

yes

-a, +c, -i, +e, -o, -u: N
-a, +c, +i, -e, -o, -u: N
+a, +c, +i, -e, +o, -u: N

no

+a, -c, +i, +e, +o, +u: Y
+a, -c, +i, -e, -o, -u: Y
-a, -c, +i, +e, -o, -u: Y
-a, -c, +i, -e, -o, +u: Y
+a, -c, -i, -e, +o, -u: N

- Can you think of a good heuristic?
- Can you think of a perfect heuristic?

An example

+a, +b, +c: Y
+a, +b, -c: Y
+a, -b, +c: N
+a, -b, -c: N
-a, +b, +c: N
-a, +b, -c: N
-a, -b, +c: Y
-a, -b, +c: Y

- What feature would you use if you can use only one?
- What if two?

Different approach: nearest neighbor(s)

- Next person is -a, +c, -i, +e, -o, +u. Will we get paid back?
- Nearest neighbor: simply look at most similar example in the training data, see what happened there

  +a, -c, +i, +e, +o, +u: Y *(distance 4)*

  -a, +c, -i, +e, -o, -u: N *(distance 1)*

  +a, -c, +i, -e, -o, -u: Y *(distance 5)*

  -a, -c, +i, +e, -o, -u: Y *(distance 3)*

  -a, +c, +i, -e, -o, -u: N *(distance 3)*

  -a, -c, +i, -e, -o, +u: Y *(distance 3)*

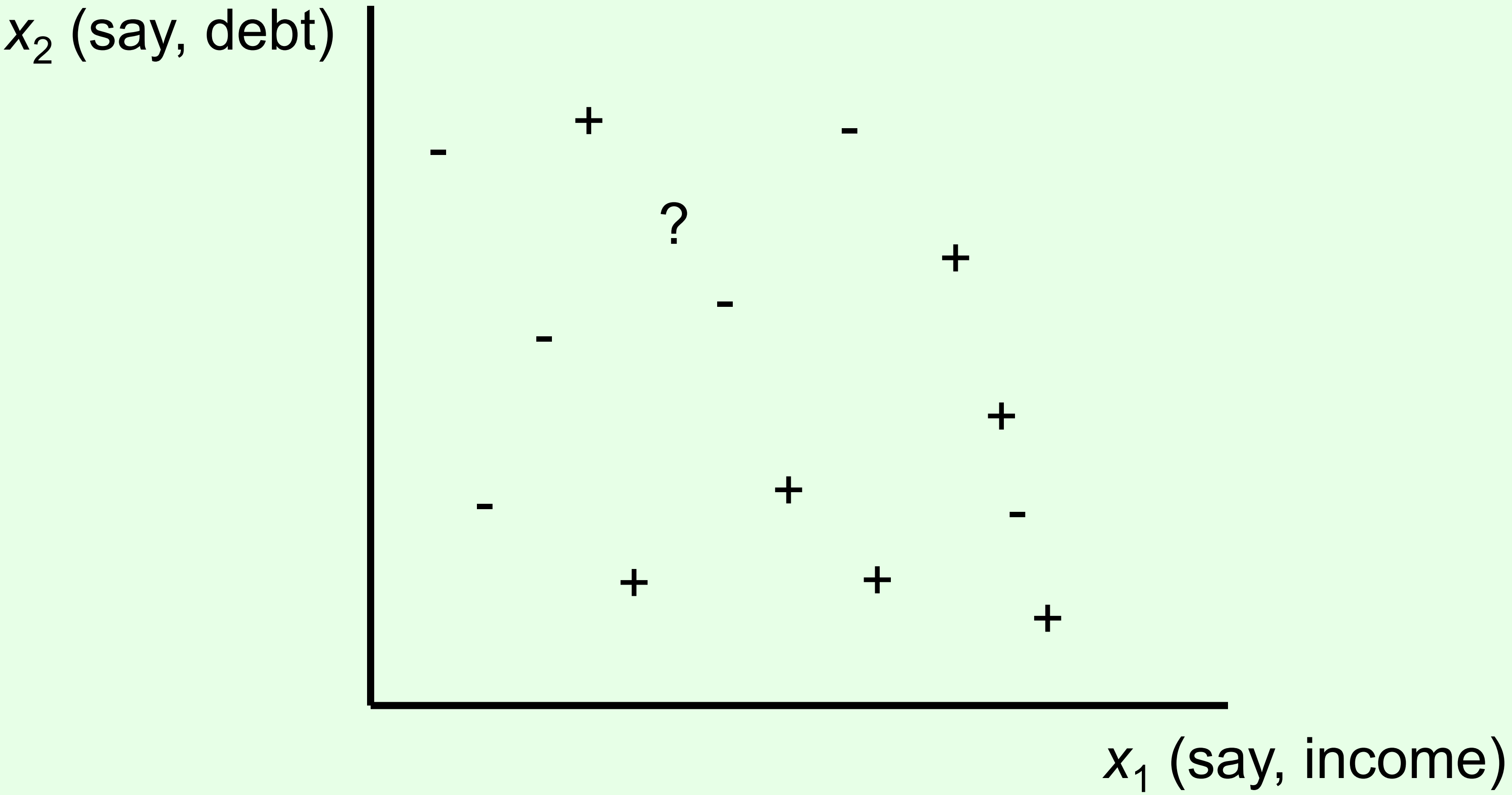
  +a, -c, -i, -e, +o, -u: N *(distance 5)*

  +a, +c, +i, -e, +o, -u: N *(distance 5)*
- Nearest neighbor is second, so predict N
- k nearest neighbors: look at k nearest neighbors, take a vote
  – E.g., 5 nearest neighbors have 3 Ys, 2Ns, so predict Y

Another approach: perceptrons

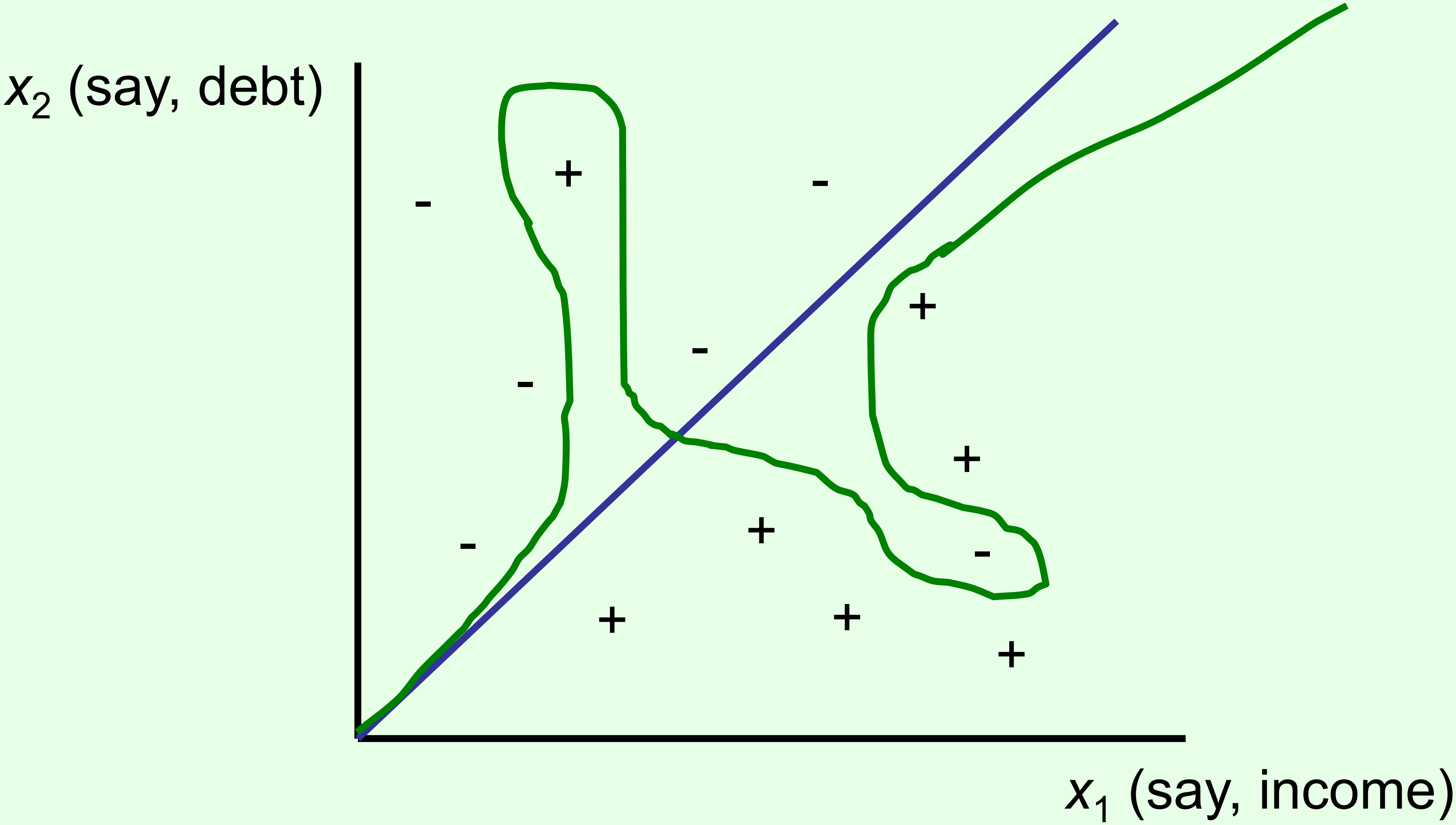
- Place a weight on every attribute, indicating how important that attribute is (and in which direction it affects things)
- E.g., $w_a = 1$, $w_c = -5$, $w_i = 4$, $w_e = 1$, $w_o = 0$, $w_u = -1$
  - +a, -c, +i, +e, +o, +u: Y *(score 1+4+1+0-1 = 5)*
  - -a, +c, -i, +e, -o, -u: N *(score -5+1=-4)*
  - +a, -c, +i, -e, -o, -u: Y *(score 1+4=5)*
  - -a, -c, +i, +e, -o, -u: Y *(score 4+1=5)*
  - -a, +c, +i, -e, -o, -u: N *(score -5+4=-1)*
  - -a, -c, +i, -e, -o, +u: Y *(score 4-1=3)*
  - +a, -c, -i, -e, +o, -u: N *(score 1+0=1)*
  - +a, +c, +i, -e, +o, -u: N *(score 1-5+4+0=0)*
- Need to set some threshold above which we predict to be paid back (say, 2)
- May care about combinations of things (nonlinearity) – generalization: neural networks

Features don't need to be binary



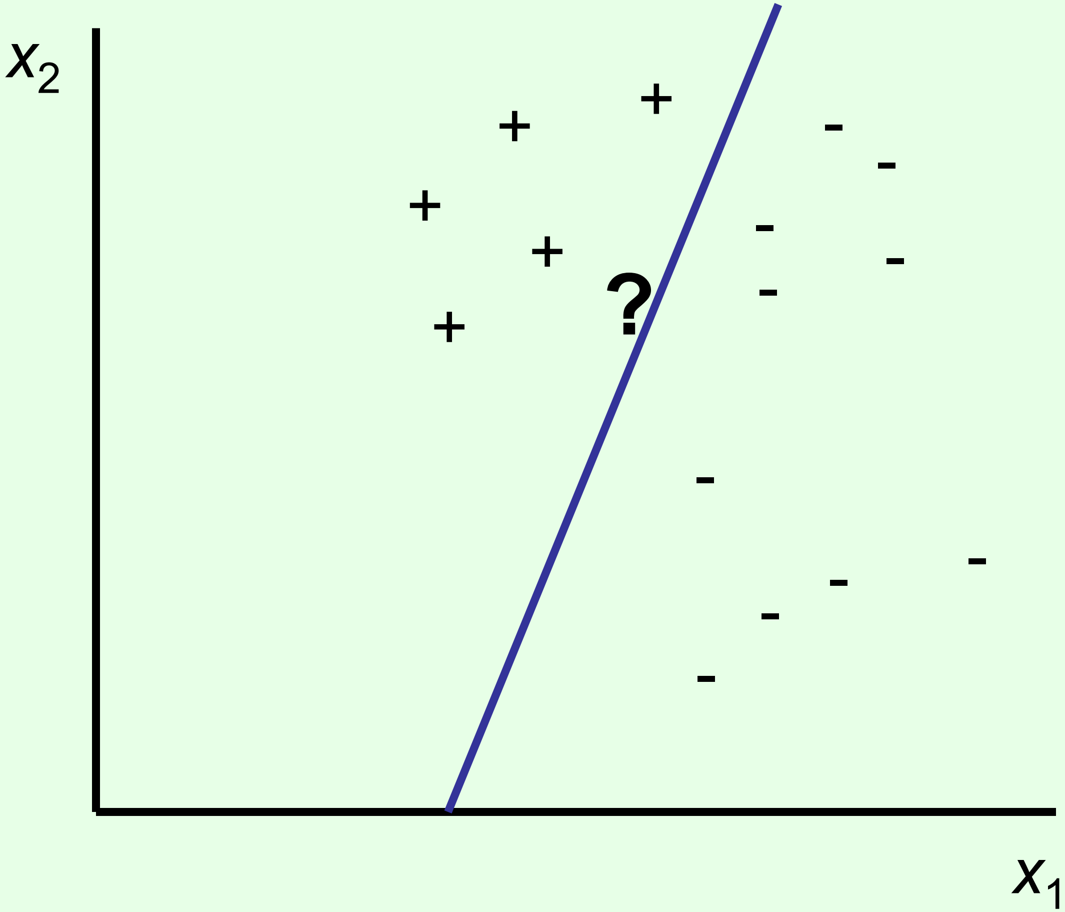
- Do you think the ? will pay back?

Which classifier is better?



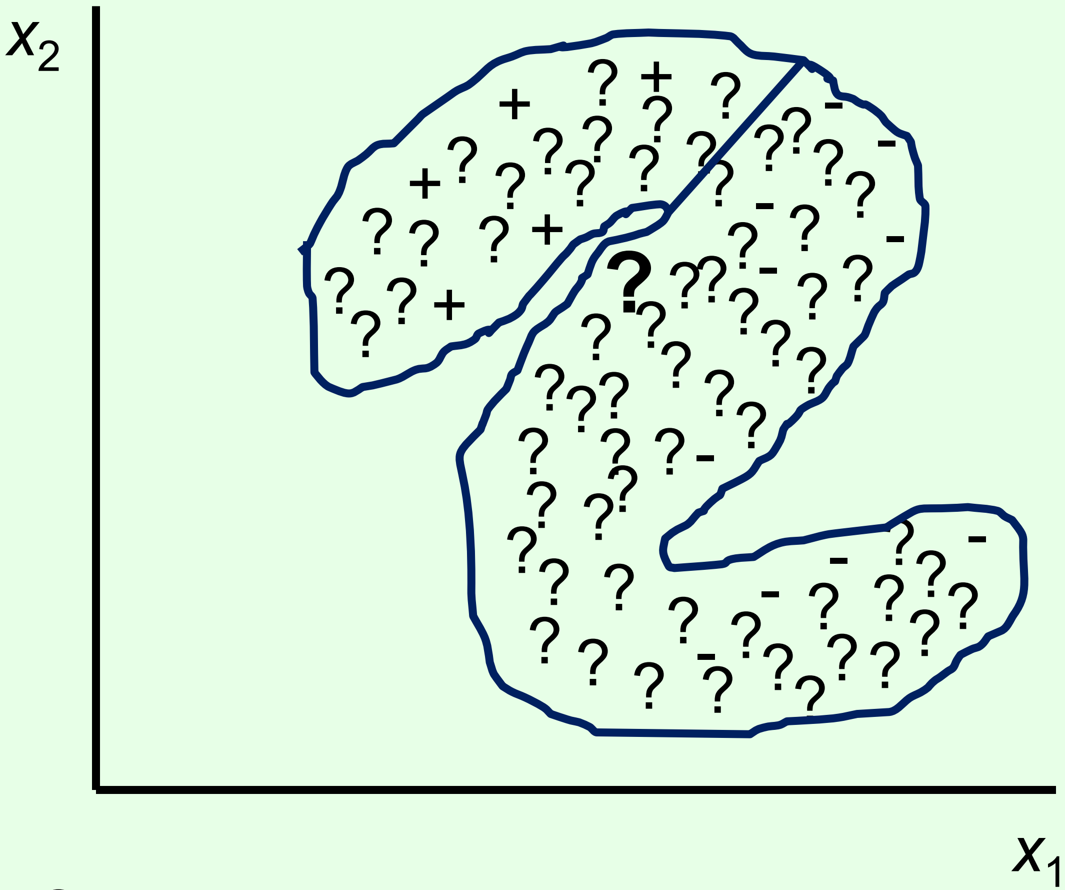
- Both blue and green classify points on the "southeast" side of them as positive

Another example (say, novel scientific dataset)



- Do you think the big bold **?** is positive?

Another example (say, novel scientific dataset)

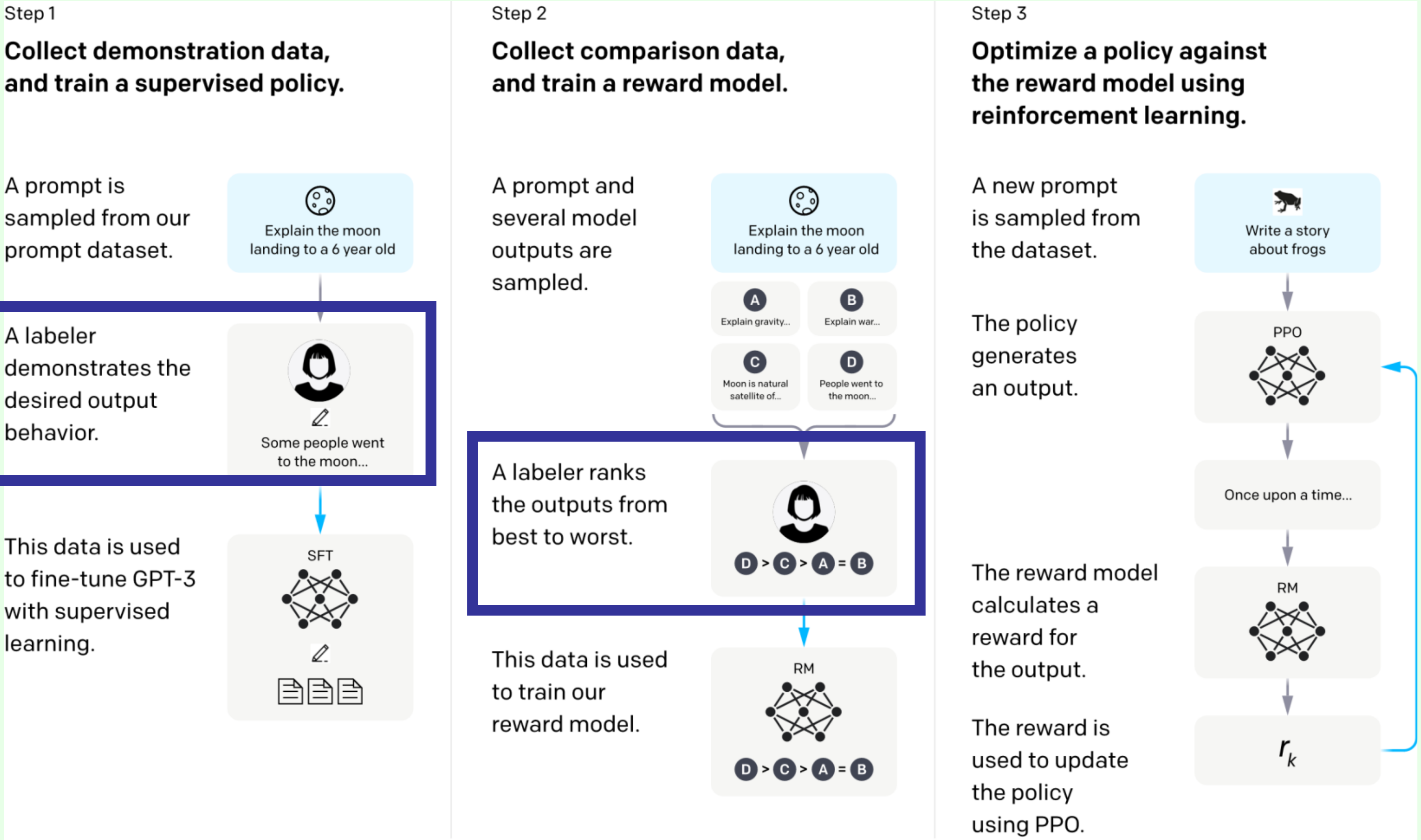


- Regular ?s are unlabeled
- Do you think the big bold **?** is positive?

Semi-supervised learning

- Sometimes there's a lot of unlabeled data that's easily available…
  - E.g., images
- … while labeled data is harder to come by
  - E.g., have to pay someone to label those images

- Compare: recent shift to large **pretrained** / **foundation** models that are then **fine-tuned** for specific purposes



"To train InstructGPT models, our core technique is [reinforcement learning from human feedback (RLHF)](), a method we helped pioneer in our earlier alignment research. This technique uses human preferences as a reward signal to fine-tune our models, which is important as the safety and alignment problems we are aiming to solve are complex and subjective, and aren't fully captured by simple automatic metrics."

https://openai.com/research/instruction-following

Self-supervised learning

- Wouldn't it be nice to do supervised learning without paying for labels?
- Common approach: remove something from the data, then try to predict the missing part
  - A word
  - Part of an image
  - The colors of an image
  - …
- Allows training on very large datasets

Reinforcement learning

- There are three routes you can take to work: A, B, C
- The times you took A, it took: 10, 60, 30 minutes
- The times you took B, it took: 32, 31, 34 minutes
- The time you took C, it took 50 minutes
- What should you do next?
- Exploration vs. exploitation tradeoff
  - Exploration: try to explore underexplored options
  - Exploitation: stick with options that look best now
- Reinforcement learning usually studied in MDPs
  - Take action, observe reward and new state

Bayesian approach to learning

- Assume we have a prior distribution over the long-term behavior of A
  - With probability .6, A is a "fast route" which:
    - With prob. .25, takes 20 minutes
    - With prob. .5, takes 30 minutes
    - With prob. .25, takes 40 minutes
  - With probability .4, A is a "slow route" which:
    - With prob. .25, takes 30 minutes
    - With prob. .5, takes 40 minutes
    - With prob. .25, takes 50 minutes
- We travel on A once and see it takes 30 minutes
- P(A is fast | observation) = P(observation | A is fast)*P(A is fast) / P(observation) = .5*.6/(.5*.6+.25*.4) = .3/(.3+.1) = .75
- Convenient approach for decision theory, game theory

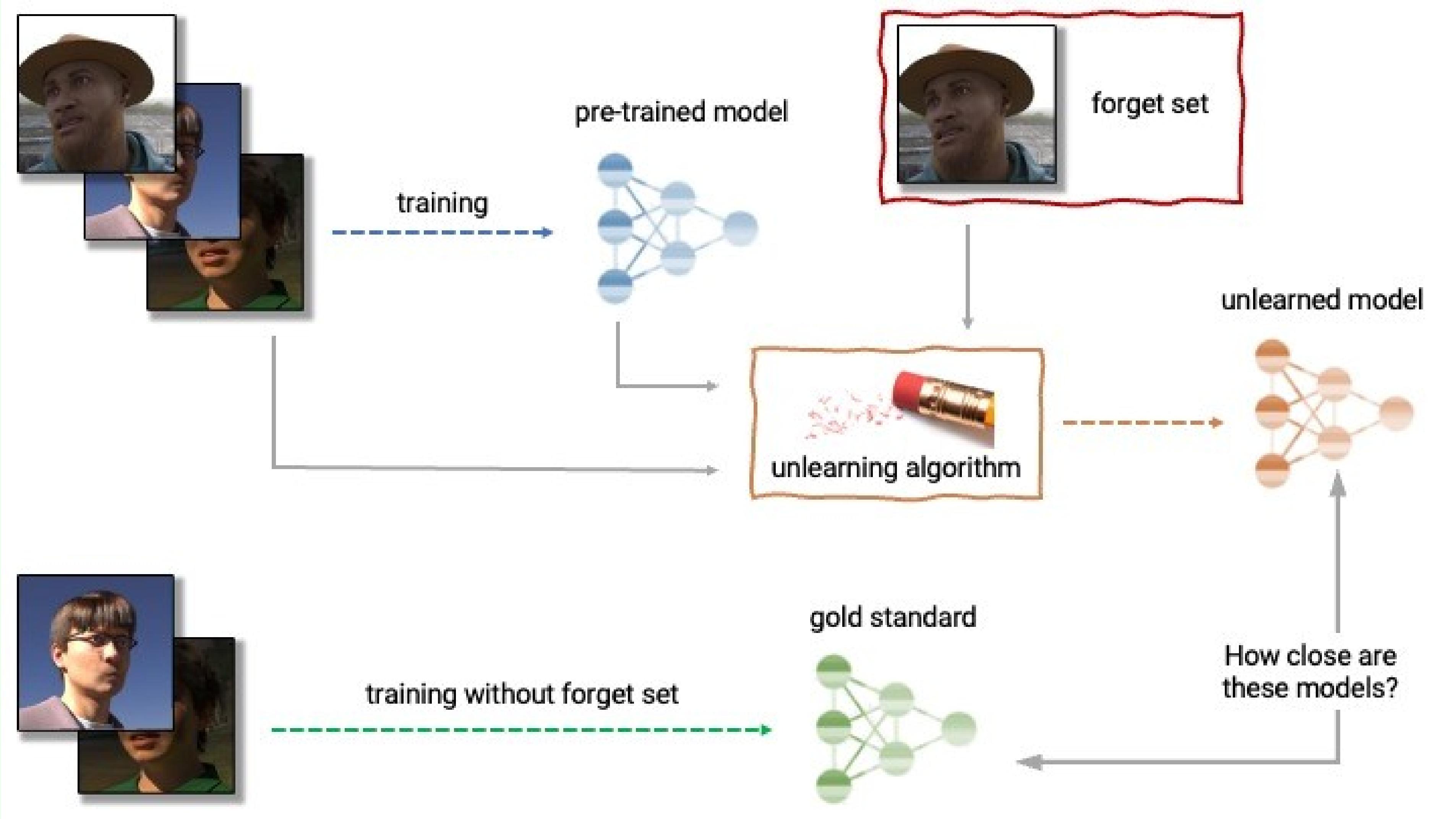
Key assumption / issue: is the real world like the training data?

- Real world may not look like training data (out of distribution, OOD)
- Distribution changes over time
  - Use of “benefits” as a signal about the economy
- Tool applied in a different setting
  - E.g., medicine: different patient population, images taken in different way, …
- People respond strategically to tool
  - E.g., resume screening
  - Learning in strategic settings / game theory

Some concerns about LLMs

- Overconfidence / hallucination / BS
  - It does not know what it does not know…
  - … or at least doesn't indicate this
- Stealing / leaking / lack of attribution
- Cybersecurity / bot armies / flood of communication / other malicious uses
- Loss of signal in text being written (cf. deepfakes)
  - College essays
  - Job applications
  - …
- Environmental cost / cheap outsourcing of human labor / …
- Inheriting human biases / uneven training data across languages and cultures
- Harmful speech / manipulating and deceiving humans
- Humans overinterpreting responses / getting directed into real-world action
- A new general / difficult-to-direct intelligence
- …

Unlearning (!)



https://unlearning-challenge.github.io/