

CPS 170: Artificial Intelligence

<http://www.cs.duke.edu/courses/spring09/cps170/>

Machine learning

Instructor: Vincent Conitzer

Why is learning important?

- So far we have assumed **we know how the world works**
 - Rules of queens puzzle
 - Rules of chess
 - Knowledge base of logical facts
 - Actions' preconditions and effects
 - Probabilities in Bayesian networks, MDPs, POMDPs, ...
 - Rewards in MDPs
- At that point “just” need to solve/optimize
- In the real world this information is often not immediately available
- AI needs to be able to **learn from experience**

Different kinds of learning...

- **Supervised learning:**
 - Someone gives us examples and the right answer for those examples
 - We have to predict the right answer for unseen examples
- **Unsupervised learning:**
 - We see examples but get no feedback
 - We need to find patterns in the 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?

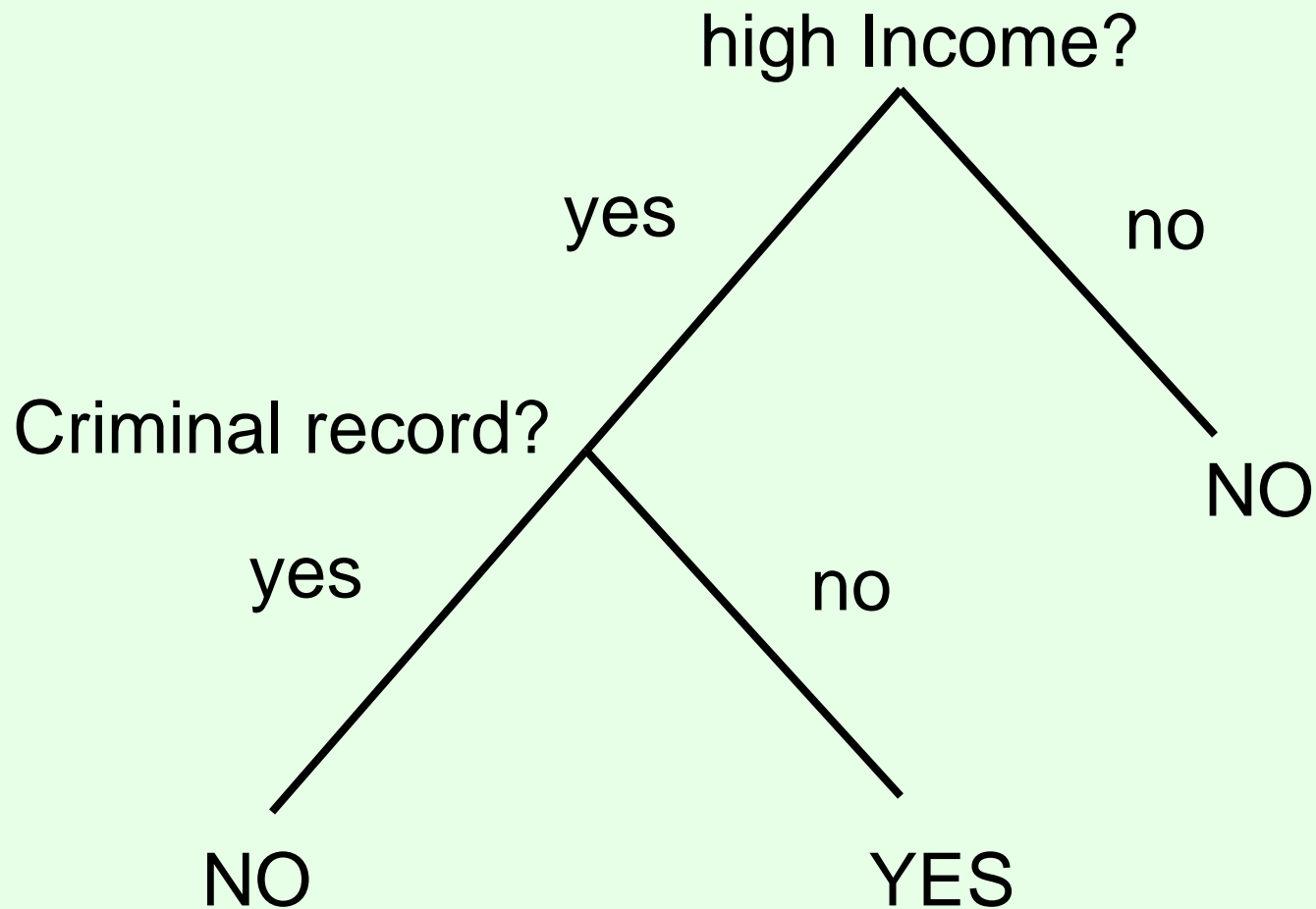
Classification...

- 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?

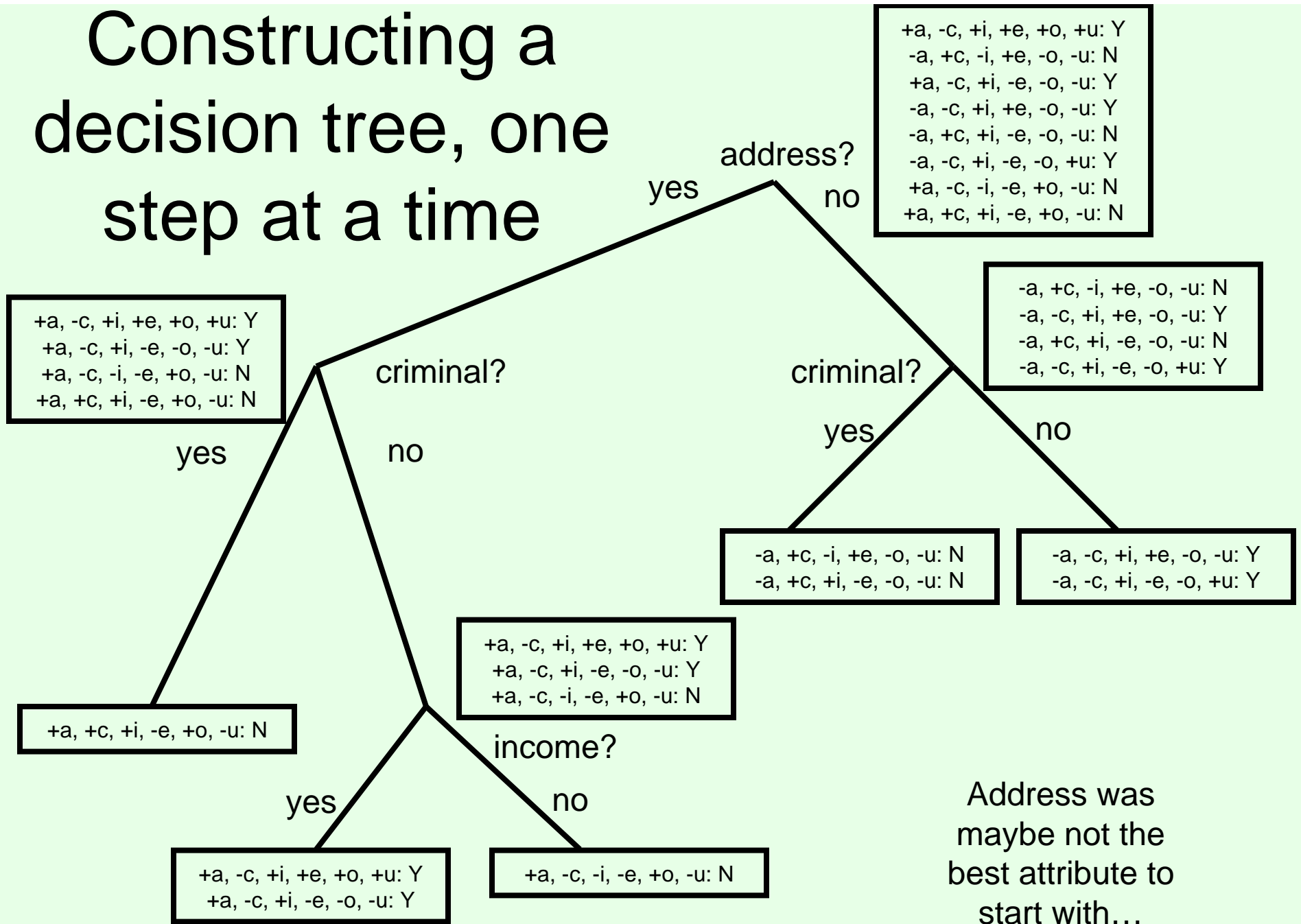
Occam's Razor

- 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

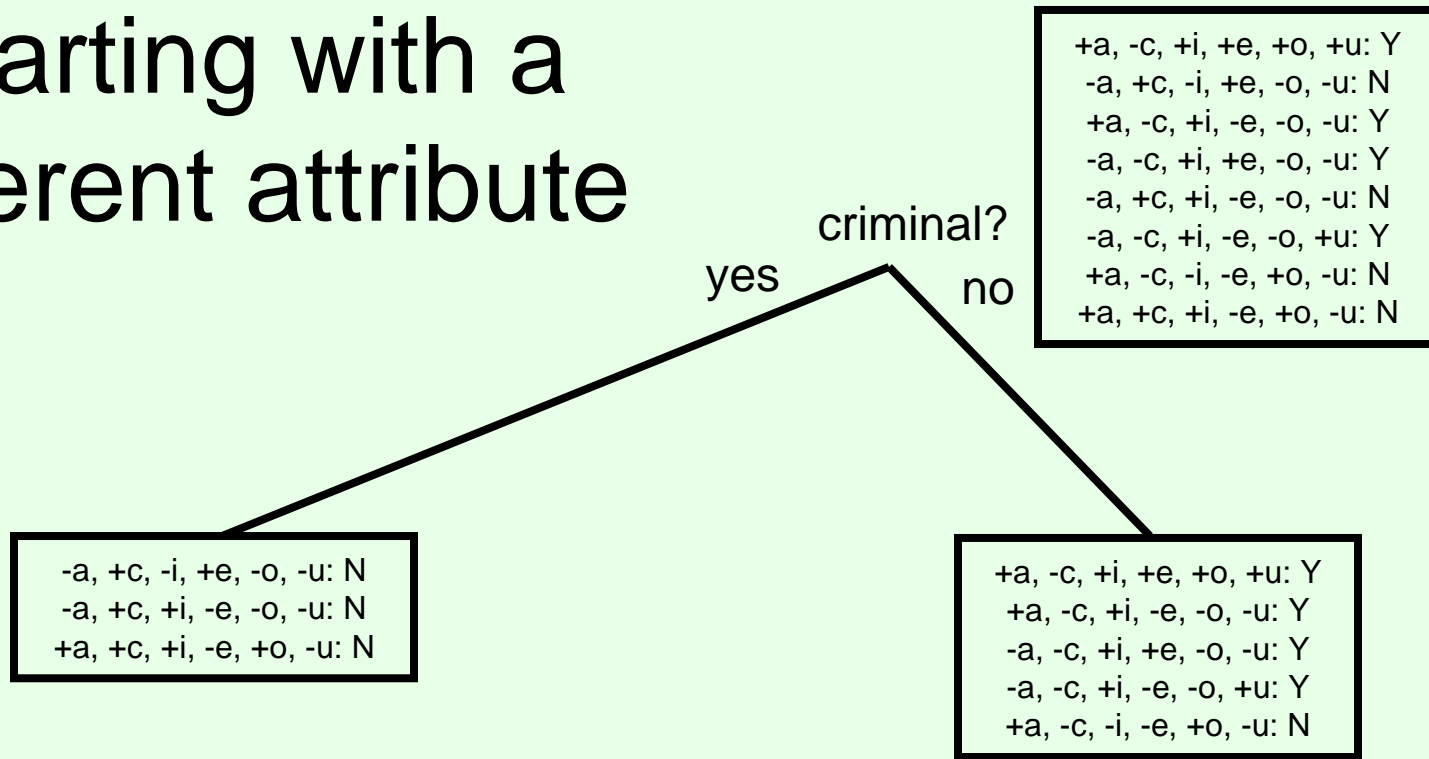
Decision trees



Constructing a decision tree, one step at a time



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

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**

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
- $P(A \text{ is fast} \mid \text{observation}) = \frac{P(\text{observation} \mid A \text{ is fast}) \cdot P(A \text{ is fast})}{P(\text{observation})} = \frac{.5 \cdot .6}{.5 \cdot .6 + .25 \cdot .4} = \frac{.3}{.3 + .1} = .75$
- Convenient approach for decision theory, game theory

Learning in game theory

- Like 2/3 of average game
- Very tricky because other agents learn at the same time
- From one agent's perspective, the environment is changing
 - Taking the average of past observations may not be good idea