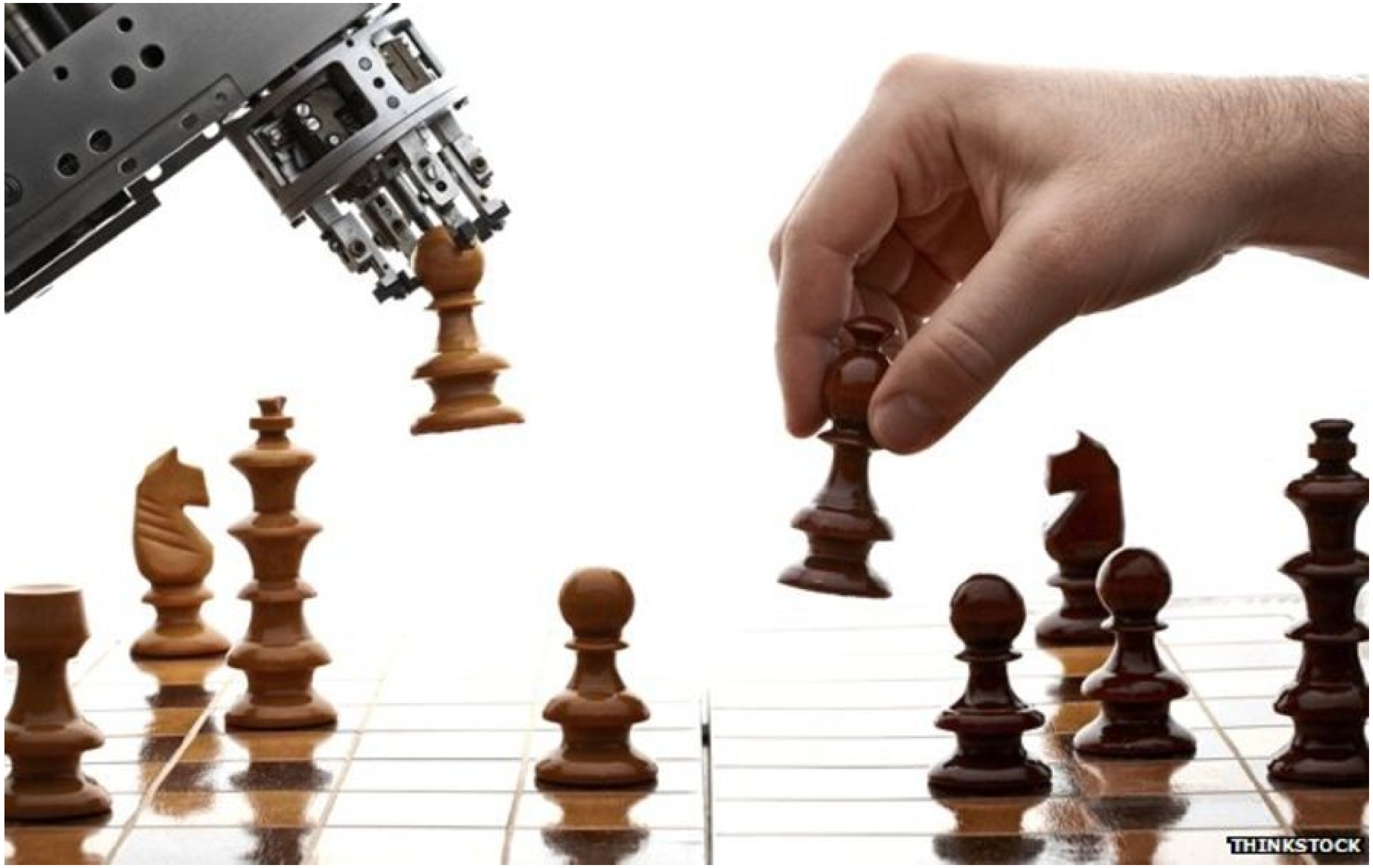


Introduction to Artificial Intelligence

COSC 4550 / COSC 5550

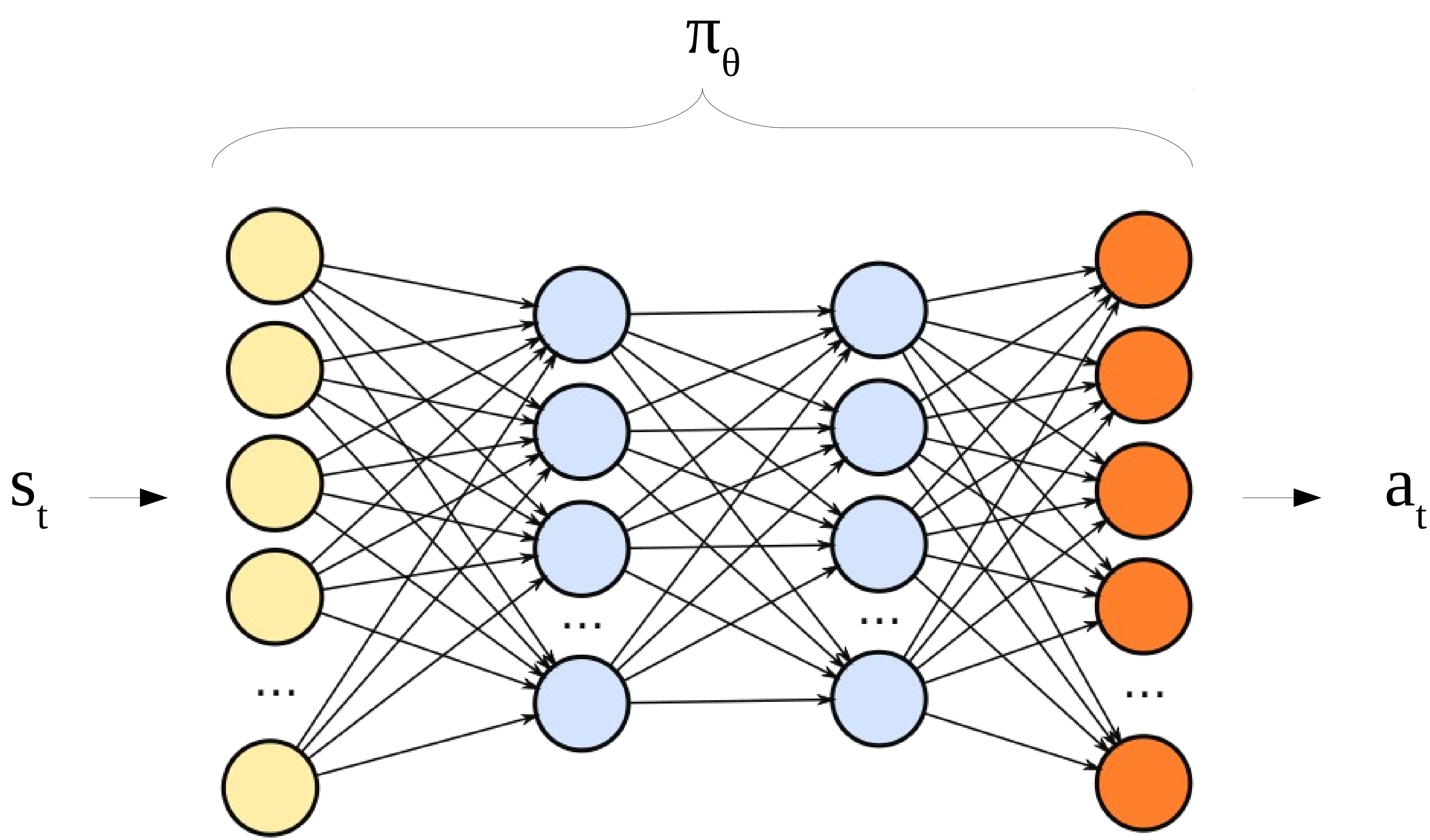
Professor Cheney

10/11/17

Q-learning is not easily extended to continuous action spaces
(since you must iterate over actions)

what other methods could we use here?

policy search


π_θ
s_t
a_t

rather than try to learn value functions,
let's just optimize the policy directly

make small changes to the policy ($\theta+\Delta$),
and see if they improve the agent's cumulative rewards

very similar in spirit to hillclimbing

most common variant is “Cross-Entropy Method”
(uses importance sampling to estimate performance)

or we can try to approximate the gradients of the reward with respect to our parameters directly from rollouts

this assumes a stochastic policy
(output is the probability to choosing some action)

thus we'll take lots of paths through the state space
(making a different decision each time we come to a state)
similar in spirit to our Monte Carlo Tree Search method
for estimating value functions at each node
(but now we're estimating value of a policy)

most popular variant in practice is “REINFORCE”

we've talked a lot about gradients,
backpropogating error signals,
learning models from the interactions of features

but up to now we've been kind of vague about these ideas

machine learning!!!

$y = f(x)$

given lots of examples of (x, y) pairs,
try and guess at what *f* is?

$f(1) = 2$
$f(4) = 8$
$f(3) = 6$
$f(5) = 10$
$f(2) = 4$

$f(x) = 2*x$

$$y = f(x)$$

given lots of examples of (x, y) pairs,
try and guess at what *f* is?

$$f(1) = 2$$
$$f(2) = 4$$
$$f(3) = 6$$
$$f(4) = 8$$
$$f(5) = 10$$

$$f(x) = 2*x$$

$$y = f(x)$$

given lots of examples of (x, y) pairs,
try and guess at what *f* is?

$$f(1) = 2$$
$$f(2) = 3$$
$$f(3) = 6$$
$$f(4) = 11$$
$$f(5) = 18$$

$$f(x) = x^2 - 2x + 3$$

$$y = f(x)$$

given lots of examples of (x, y) pairs,
try and guess at what *f* is?

f([T, F, T, F]) = T
f([F, F, F, T]) = F
f([T, T, F, F]) = T
f([F, F, T, T]) = F
f([T, F, T, F]) = T

$$f([x_1, x_2, x_3, x_4]) = (x_1 \wedge x_2) \cup (x_3 \wedge \sim x_4)$$

$$y = f(x)$$

$$state_{t+1} = transitionModel(\ state_t\ )$$

$$reward_t = valueFunction(\ state_t\ )$$

person? = faceDetector(cameraSensor)

stockValue = valuationFunction(companyMetrics)

word? = speechRecognitionFunction(sounds)

graduateCollege? = predictorFunction(highSchoolGrades)

$y = f(x)$

“supervised learning”
(where we are given examples of x and y pairs)
(correct answer *y* is the supervision)

“regression”
(predict a real-valued number)

“classification”
(choose which class will be true)

decision trees

supervised learning method for
when solutions are composed of “if” statements

wearWinterHatToday? = f(weather)

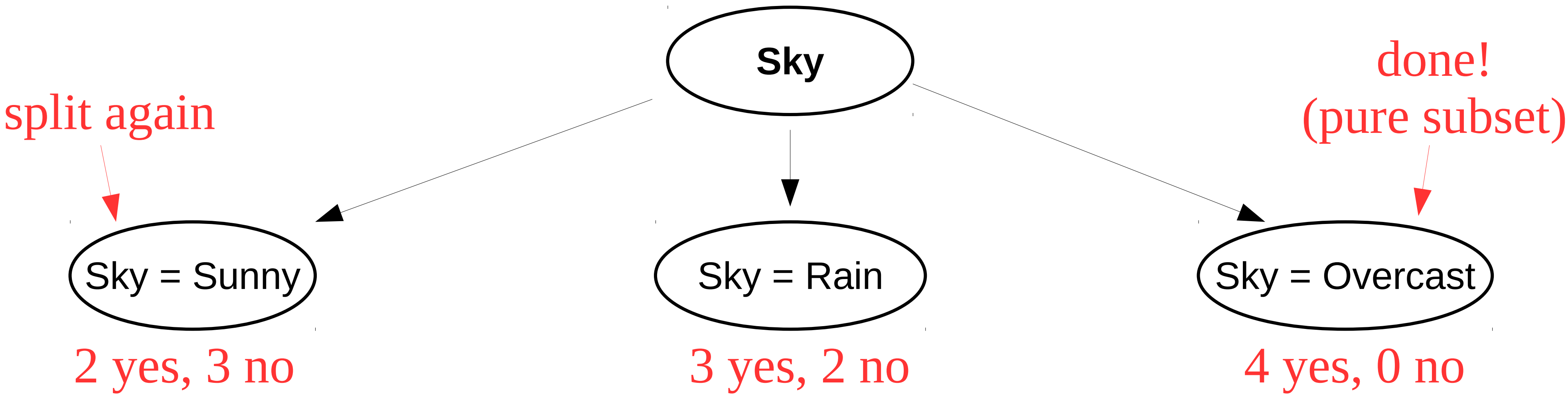
f = “if it's cold, or if it's raining and if it's windy,
but not if my hair is still wet”

on a given day, predict if John will play tennis or not?

Day	Sky	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No
D15	Rain	High	Weak	???

← 9 yes, 5 no

1) choose an attribute, and draw the tree with the subtable for each value of that attribute:

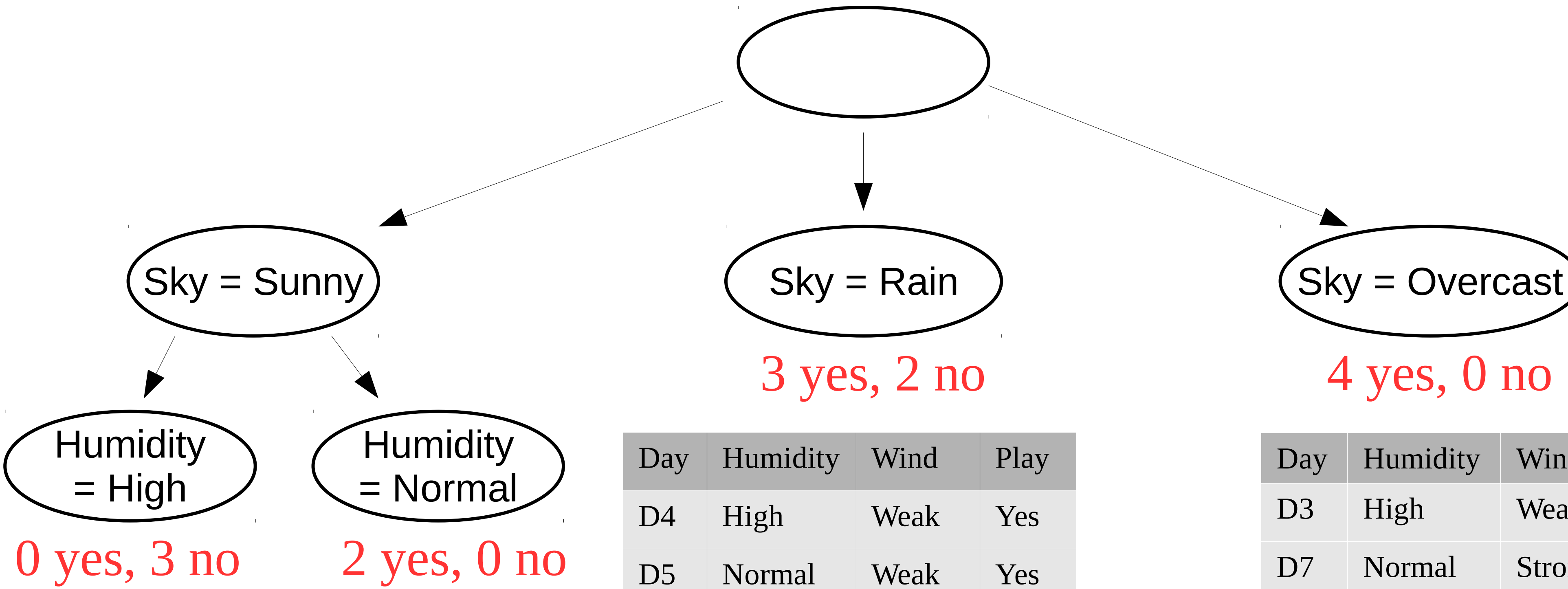


Day	Humidity	Wind	Play
D3	High	Weak	Yes
D7	Normal	Strong	Yes
D12	High	Strong	Yes
D13	Normal	Weak	Yes
D11	Normal	Strong	Yes

Day	Humidity	Wind	Play
D4	High	Weak	Yes
D5	Normal	Weak	Yes
D6	Normal	Strong	No
D10	Normal	Weak	Yes
D14	High	Strong	No

Day	Humidity	Wind	Play
D3	High	Weak	Yes
D7	Normal	Strong	Yes
D12	High	Strong	Yes
D13	Normal	Weak	Yes

1) choose an attribute, and draw the tree with the subtable for each value of that attribute:



Sky = Sunny, Humidity = High (0 yes, 3 no)

Day	Wind	Play
D1	Weak	No
D2	Strong	No
D8	Weak	No

Sky = Sunny, Humidity = Normal (2 yes, 0 no)

Day	Wind	Play
D9	Weak	Yes
D11	Strong	Yes

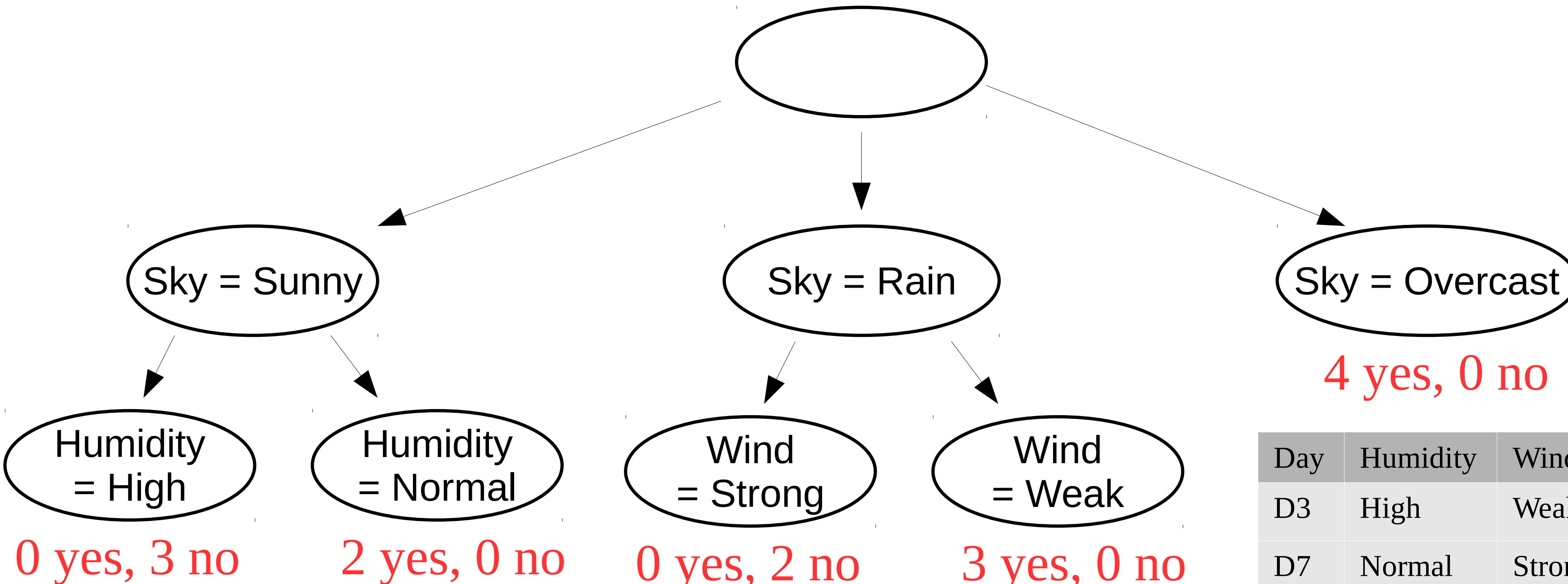
Sky = Rain (3 yes, 2 no)

Day	Humidity	Wind	Play
D4	High	Weak	Yes
D5	Normal	Weak	Yes
D6	Normal	Strong	No
D10	Normal	Weak	Yes
D14	High	Strong	No

Sky = Overcast (4 yes, 0 no)

Day	Humidity	Wind	Play
D3	High	Weak	Yes
D7	Normal	Strong	Yes
D12	High	Strong	Yes
D13	Normal	Weak	Yes

1) choose an attribute, and draw the tree with the subtable for each value of that attribute:



Day	Wind	Play
D1	Weak	No
D2	Strong	No
D8	Weak	No

Day	Wind	Play
D9	Weak	Yes
D11	Strong	Yes

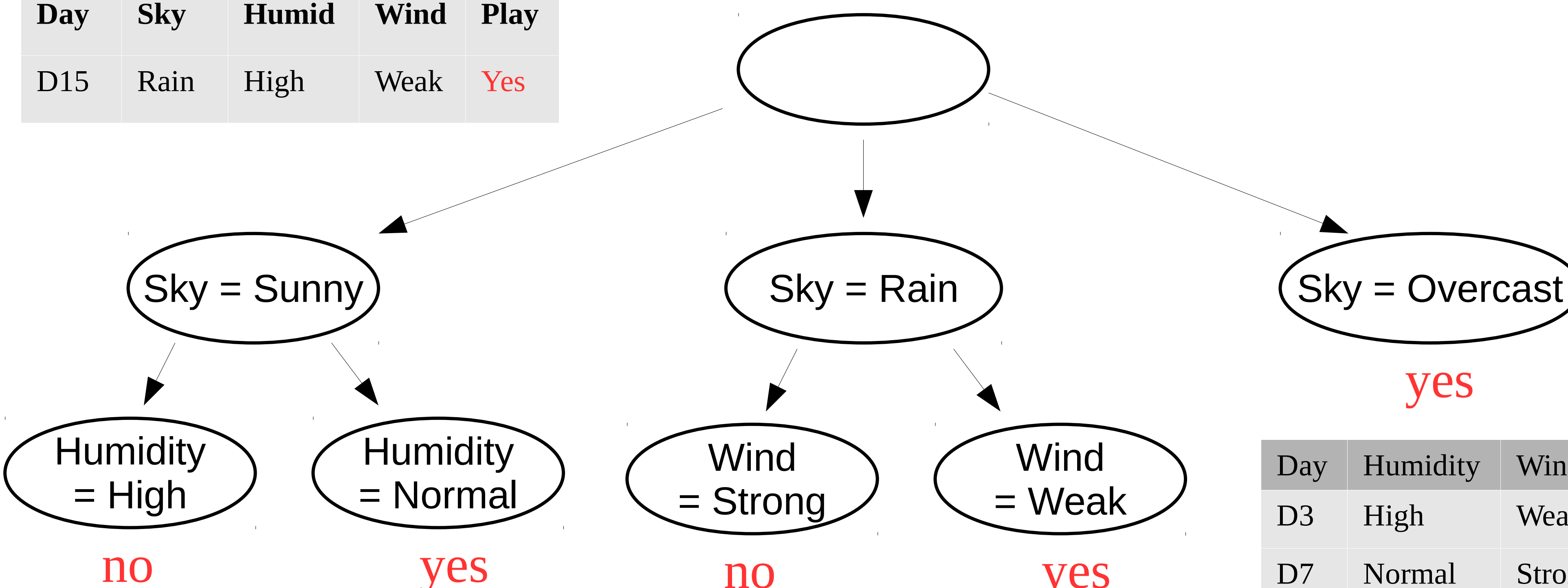
Day	Humid	Play
D6	Normal	No
D14	High	No

Day	Humid	Play
D4	High	Yes
D5	Normal	Yes
D10	Normal	Yes

Day	Humidity	Wind	Play
D3	High	Weak	Yes
D7	Normal	Strong	Yes
D12	High	Strong	Yes
D13	Normal	Weak	Yes

"John will play tennis if it's overcast, or if it's raining but not windy, or if it's sunny but not humid"

Day	Sky	Humid	Wind	Play
D15	Rain	High	Weak	Yes



Day	Wind	Play
D1	Weak	No
D2	Strong	No
D8	Weak	No

Day	Wind	Play
D9	Weak	Yes
D11	Strong	Yes

Day	Humid	Play
D6	Normal	No
D14	High	No

Day	Humid	Play
D4	High	Yes
D5	Normal	Yes
D10	Normal	Yes

Day	Humidity	Wind	Play
D3	High	Weak	Yes
D7	Normal	Strong	Yes
D12	High	Strong	Yes
D13	Normal	Weak	Yes

but won't the tree (and the resulting logic) look different depending on what order we expand the nodes?

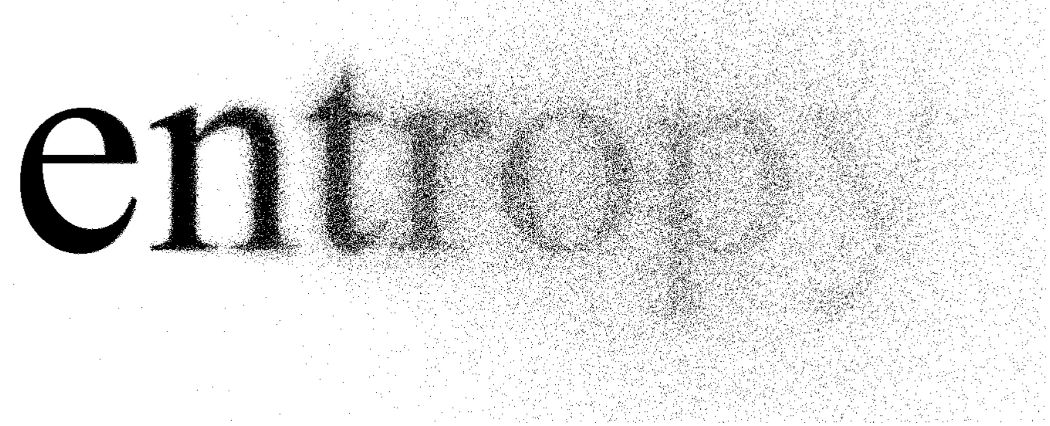
how did we know which node to expand first?

expand the attribute that would
provide the most information!
(reduce the most uncertainty)

the information theoretic metric
for uncertainty is entropy (H)!

$$H(V) = -\Sigma_{i=1:n} P(v_i)*\log_2(P(v_i))$$

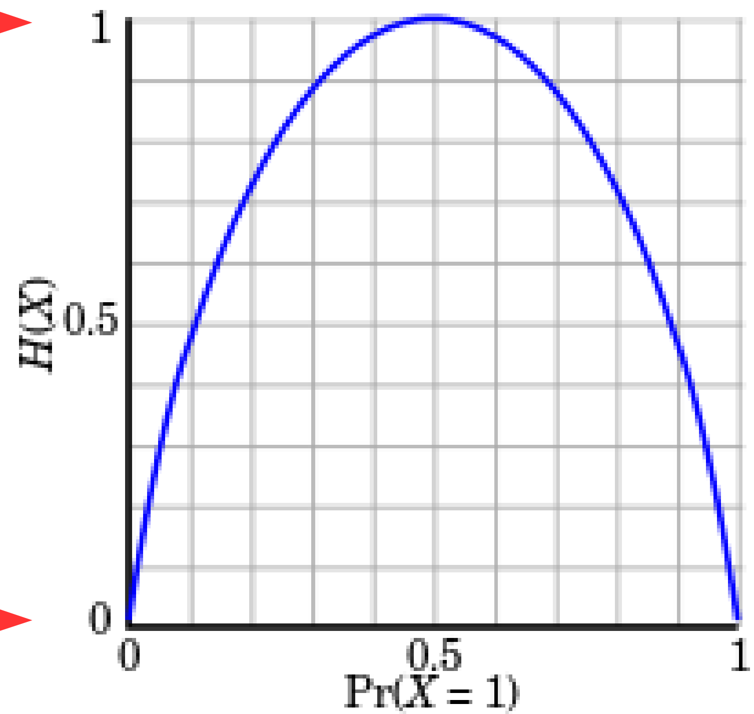
e.g. $(n \in \{T,F\})$: $H = - P(v_T)*\log_2(P(v_T)) - P(v_F)*\log_2(P(v_F))$



$$H(X) = P(X)*\log_2(P(X))$$

bad (high uncertainty) →

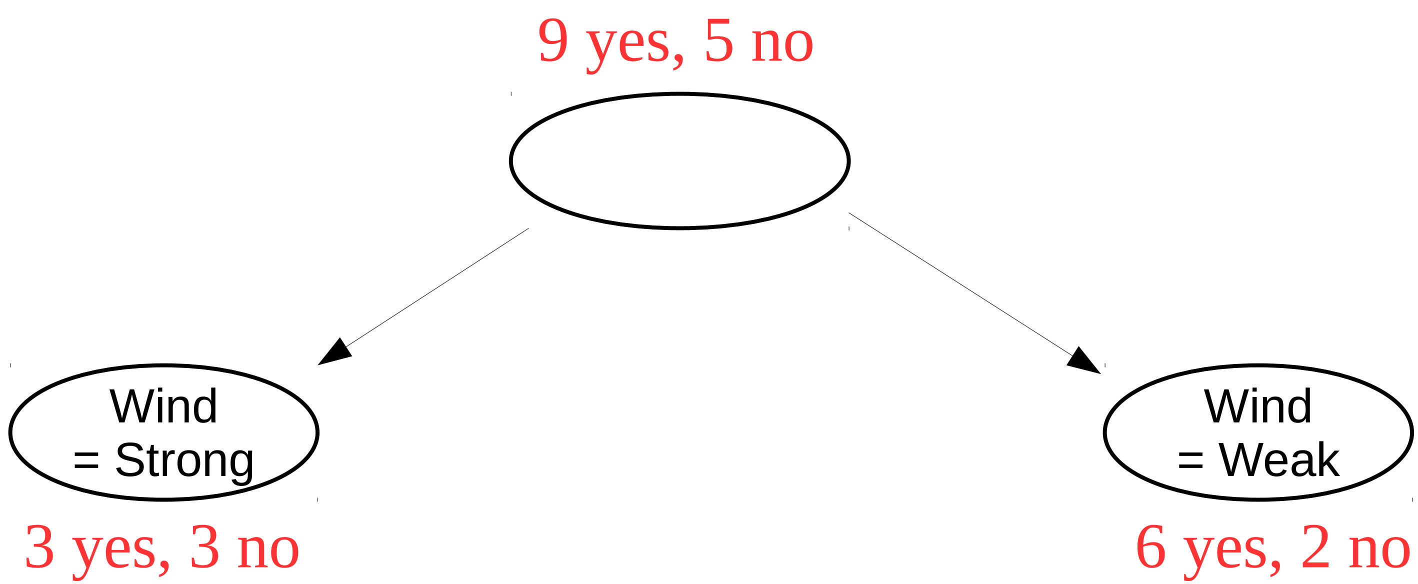
good (low uncertainty) →



on a given day, predict if John will play tennis or not?

Day	Sky	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

← 9 yes, 5 no

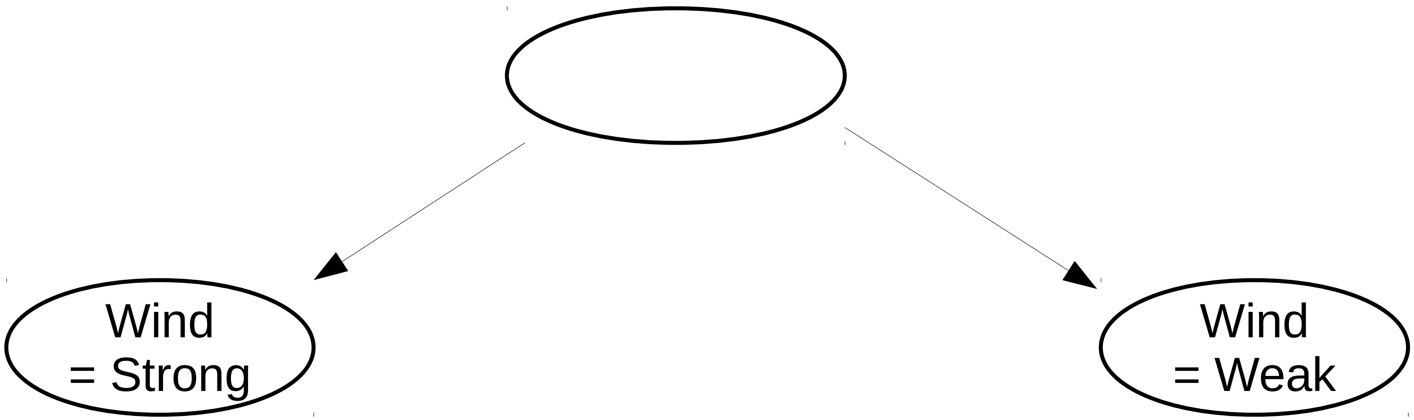


Day	Sky	Humidity	Play
D2	Sunny	High	No
D6	Rain	Normal	No
D7	Overcast	Normal	Yes
D11	Sunny	Normal	Yes
D12	Overcast	High	Yes
D14	Rain	High	No

Day	Sky	Humidity	Play
D1	Sunny	High	No
D3	Overcast	High	Yes
D4	Rain	High	Yes
D5	Rain	Normal	Yes
D8	Sunny	High	No
D9	Sunny	Normal	Yes
D10	Rain	Normal	Yes
D13	Overcast	Normal	Yes

$$H(beforeSplit) = -(9/14)*\log_2(9/14) - (5/14)*\log_2(5/14)$$
$$= 0.94$$

9 yes, 5 no



3 yes, 3 no

6 yes, 2 no

$$H(strong) = -(3/6)*\log_2(3/6) - (3/6)*\log_2(3/6)$$
$$= 1.0$$

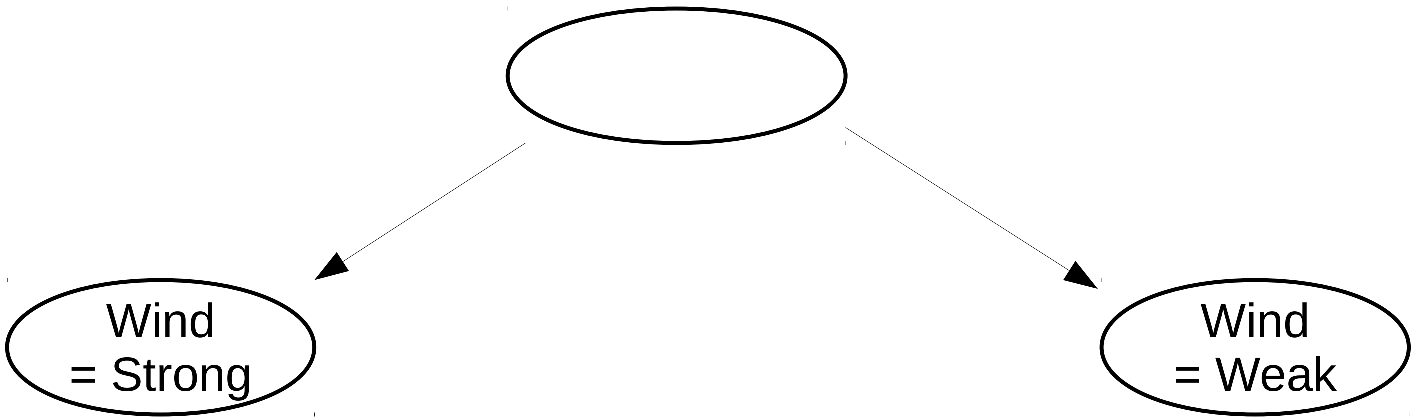
$$H(weak) = -(6/8)*\log_2(6/8) - (2/8)*\log_2(2/8)$$
$$= 0.81$$

did this split help or hurt (and by how much)?

“information gain” is the entropy change in each subset after the split, weighted by the size of the subset

$$H(\text{beforeSplit}) = -(9/14)*\log_2(9/14) - (5/14)*\log_2(5/14)$$
$$= 0.94$$

9 yes, 5 no



3 yes, 3 no

6 yes, 2 no

$$H(\text{strong}) = -(3/6)*\log_2(3/6) - (3/6)*\log_2(3/6)$$
$$= 1.0$$

$$H(\text{weak}) = -(6/8)*\log_2(6/8) - (2/8)*\log_2(2/8)$$
$$= 0.81$$

$$\text{information gain} = H(\text{beforeSplit}) - 6/14 * H(\text{strong}) - 8/14 * H(\text{weak})$$

$$\text{information gain} = 0.94 - 6/14 * 1.0 - 8/14 * 0.81 = \mathbf{0.049\ bits}$$

now we have a way to recursively split
datasets along different attributes

and we have a way to choose what order to split them

super simple! (yet effective)
frequently used in practice