CS446 Introduction to Machine Learning (Spring 2015)
University of Illinois at Urbana-Champaign
http://courses.engr.illinois.edu/cs446

LECTURE 3: DECISION TREES

Prof. Julia Hockenmaier
juliahmr@illinois.edu

Admin

Office hours

Julia Hockenmaier: Tue/Thu, 5:00PM – 6:00PM, 3324 SC

TAs (on-campus students):

Mon, 1:00PM–3:00PM, 1312 Siebel Center (Stephen)

Tue, 5:00PM–6:00PM, 1312 Siebel Center (Ryan)

Wed, 9:30 AM–11:30 AM, 1312 Siebel Center (Ray)

If 1312 is not available, office hours will be held by 3407 Siebel Center
(at the east end of the third floor)

TAs (on-line students):

Tue, 8:00 PM – 9:00 PM (Ryan)

Textbooks

Comprehensive resource:
Samut and Webb (eds.), *Encyclopedia of Machine Learning*
Gentle introductions:
Mitchell, *Machine Learning* (a bit dated)
Flach, *Machine Learning* (more recent)
More complete introductions:
Bishop, *Pattern Recognition and Machine Learning*
Shalev-Shwartz & Ben-David, *Understanding Machine Learning*
Alpaydın, *Introduction to Machine Learning*
Murphy, *Machine Learning: a Probabilistic Perspective*
Barber, *Bayesian Reasoning and Machine Learning*
Hastie et al., *The Elements of Statistical Learning*
Duda et al., *Pattern Classification*
and many more… (see Resources page on class website)

Last lecture's key concepts

Supervised Learning:

– What is our instance space?

What features do we use to represent instances?

– What is our label space?

Classification: discrete labels

– What is our hypothesis space?

– What learning algorithm do we use?

Today's lecture

Decision trees for (binary) classification

- Non-linear classifiers

Learning decision trees (ID3 algorithm)

- Batch algorithm
- Greedy heuristic (based on information gain)
- Originally developed for discrete features

Overfitting

What are decision trees?

Will customers add sugar to their drinks?

Will customers add sugar to their drinks?

Data

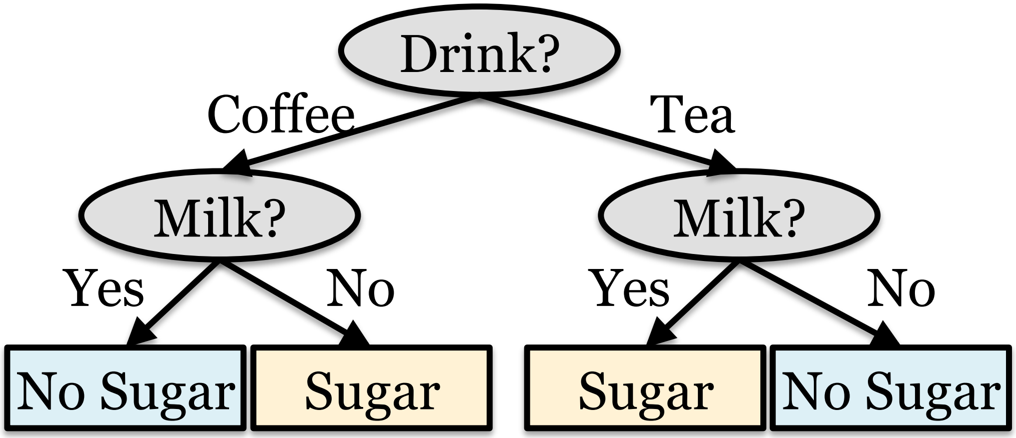
	Features		Class
	Drink?	Milk?	Sugar?
#1	Coffee	No	Yes
#2	Coffee	Yes	No
#3	Tea	Yes	Yes
#4	Tea	No	No

Will customers add sugar to their drinks?

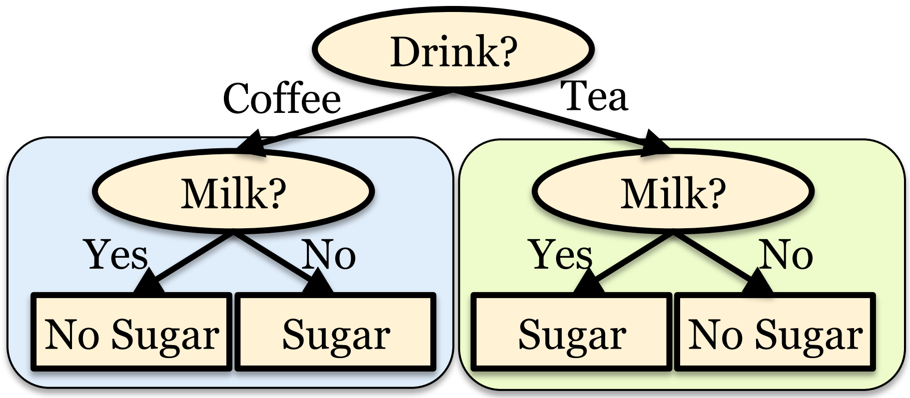
Data

	Features		Class
	Drink?	Milk?	Sugar?
#1	Coffee	No	Yes
#2	Coffee	Yes	No
#3	Tea	Yes	Yes
#4	Tea	No	No

Decision tree



Decision trees in code




```
if Drink == Coffee
    if Milk == Yes
        Sugar := Yes
    else if Milk == No
        Sugar := No
else if Drink == Tea
    if Milk == Yes
        Sugar := No
    else if Milk == No
        Sugar := Yes
```

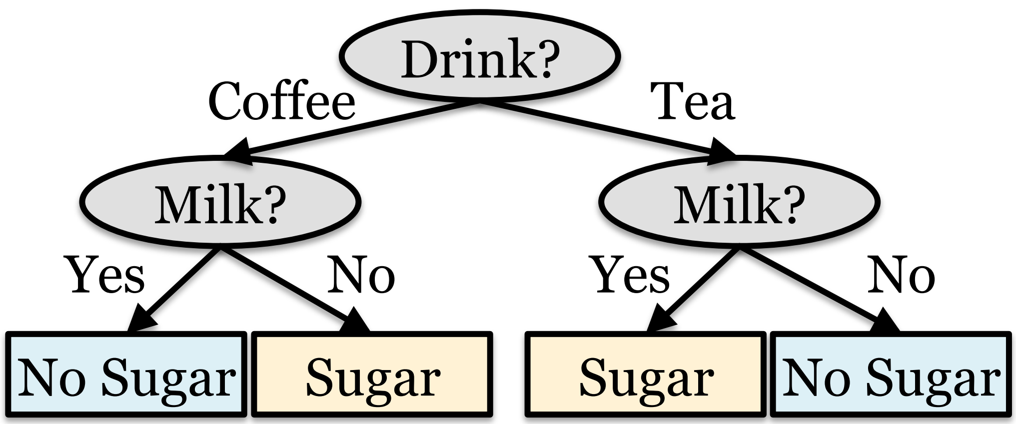
```
switch (Drink)
case Coffee:
    switch (Milk):
    case Yes:
        Sugar := Yes
    case No:
        Sugar := No
case Tea:
    switch (Milk):
    case Yes:
        Sugar := No
    case No:
        Sugar := Yes
```

Decision trees are classifiers

Non-leaf nodes test the value of one feature

- Tests: yes/no questions; switch statements
- Each child = a different value of that feature

Leaf-nodes assign a class label



How expressive are decision trees?

Hypothesis spaces for binary classification:

Each hypothesis h $\in \mathcal{H}$ assigns *true*
to one subset of the instance space $\mathcal{X}$

Decision trees do not restrict $\mathcal{H}$:

There is a decision tree for every hypothesis

Any subset of $\mathcal{X}$ can be identified via yes/no questions

Hypothesis space for our task

The target hypothesis…

Drink		Milk: Yes	Milk: No
	Coffee	No Sugar	Sugar
	Tea	Sugar	No Sugar

… is equivalent to

x_1		x_2: 0	x_2: 1
	0	y=0	y=1
	1	y=1	y=0

Hypothesis space for our task

$\mathcal{H}$

		x_2	
		0	1
x_1	0	0	0
	1	0	0

		x_2	
		0	1
x_1	0	0	0
	1	1	0

		x_2	
		0	1
x_1	0	0	0
	1	0	1

		x_2	
		0	1
x_1	0	1	0
	1	0	0

		x_2	
		0	1
x_1	0	0	1
	1	0	0

		x_2	
		0	1
x_1	0	0	0
	1	1	1

		x_2	
		0	1
x_1	0	1	0
	1	1	0

		x_2	
		0	1
x_1	0	0	1
	1	1	0

		x_2	
		0	1
x_1	0	1	0
	1	0	1

		x_2	
		0	1
x_1	0	0	1
	1	0	1

		x_2	
		0	1
x_1	0	1	1
	1	0	0

		x_2	
		0	1
x_1	0	1	0
	1	1	1

		x_2	
		0	1
x_1	0	0	1
	1	1	1

		x_2	
		0	1
x_1	0	1	1
	1	1	0

		x_2	
		0	1
x_1	0	1	1
	1	0	1

		x_2	
		0	1
x_1	0	1	1
	1	1	1

How do we learn (induce) decision trees?

How do we learn decision trees?

We want the *smallest* tree that is consistent with the training data

(i.e. that assigns the correct labels to training items)

But we can't enumerate all possible trees.

$|\mathcal{H}|$ is exponential in the number of features

We use a heuristic: greedy top-down search

This is guaranteed to find a consistent tree,
and is biased towards finding smaller trees

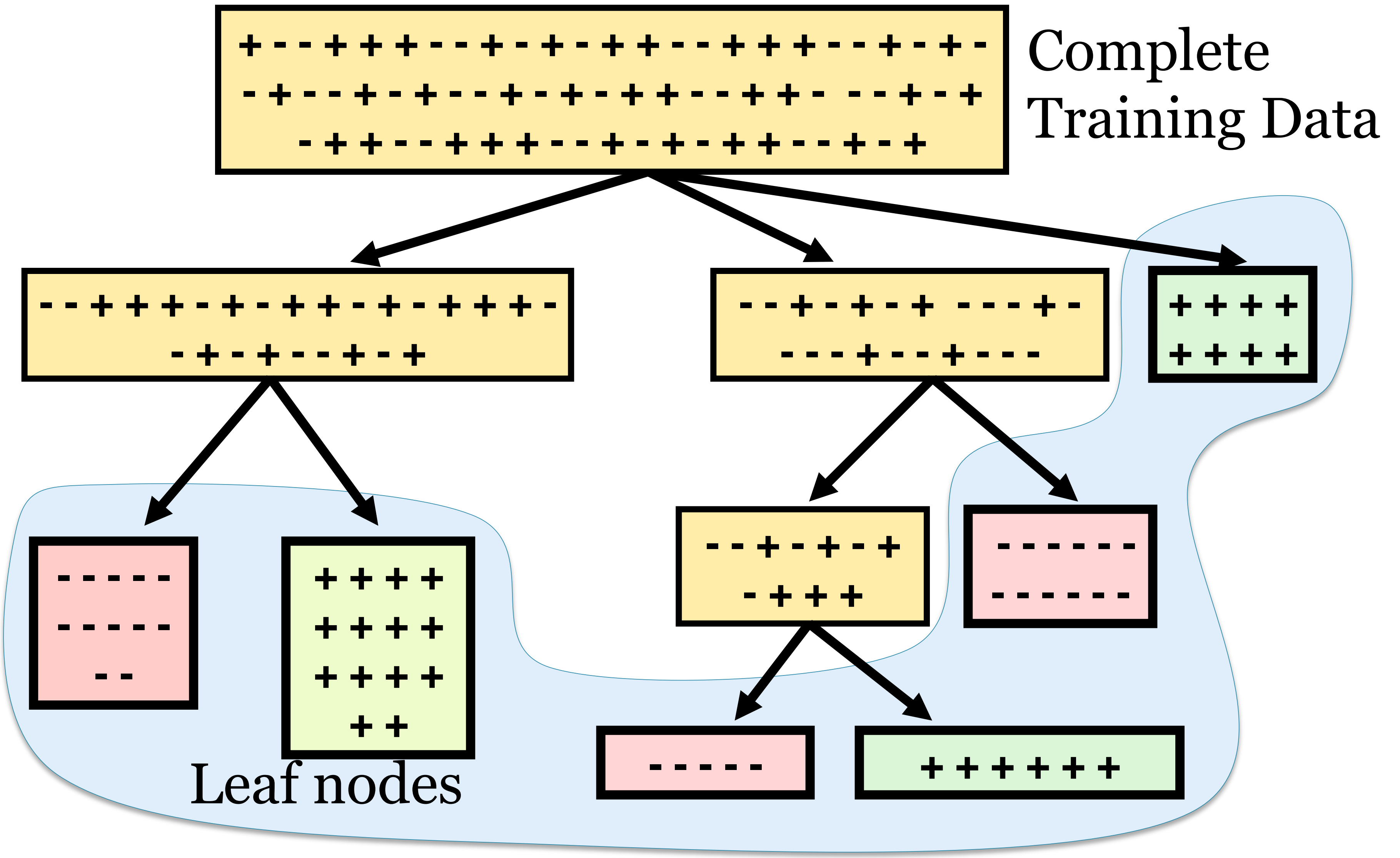
Learning decision trees

Each node is associated with a subset of the training examples

– The root has all items in the training data

– Add new levels to the tree until each leaf has only items with the same class label

Learning decision trees



How do we split a node **N**?

The node **N** is associated with a subset S of the training examples.

- If all items in S have the same class label, **N** is a leaf node
- Else, split on the values $V_F = \{v_1, ..., v_K\}$ of the most informative feature F :

  For each $v_k \in V_F$: add a new child $\mathbf{C}_k$ to **N**.

  $\mathbf{C}_k$ is associated with S_k, the subset of items in S where the feature F takes the value v_k

Which feature to split on?

We add children to a parent node in order to be more certain about which class label to assign to the examples at the child nodes.

Reducing uncertainty = reducing entropy

We want to reduce the entropy of the label distribution P(Y)

Entropy (binary case)

The class label Y is a binary random variable:

– Y takes on value 1 with probability p

$$P(Y=1) = p$$

– Y takes on value 0 with probability $1-p$

$$P(Y=0) = 1-p$$

The entropy of Y, $H(Y)$, is defined as

$$H(Y) = -p \log_2 p - (1-p) \log_2(1-p)$$

Entropy (general discrete case)

The class label Y is a discrete random variable:

– It can take on K different values

– It takes on value k with probability p_{k}

$$\forall k \in \{1 \ldots K\}: \; P(Y = k) = p_{\mathrm{k}}$$

The entropy of Y, $H(Y)$, is defined as:

$$H(Y) = -\sum_{i=1}^{K} p_k \log_2 p_k$$

Example

$$H(Y) = -\sum_{i=1}^{K} p_k \log_2 p_k$$

P(Y=a) = 0.5
P(Y=b) = 0.25
P(Y=c) = 0.25

$$H(Y) = -0.5\log_2(0.5) - 0.25\log_2(0.25) - 0.25\log_2(0.25)$$
$$= -0.5\,(-1) \quad - 0.25(-2) \quad - 0.25(-2)$$
$$= \quad 0.5 \quad + 0.5 \quad + 0.5 = 1.5$$

Example

$$H(Y) = -\sum_{i=1}^{K} p_k \log_2 p_k$$

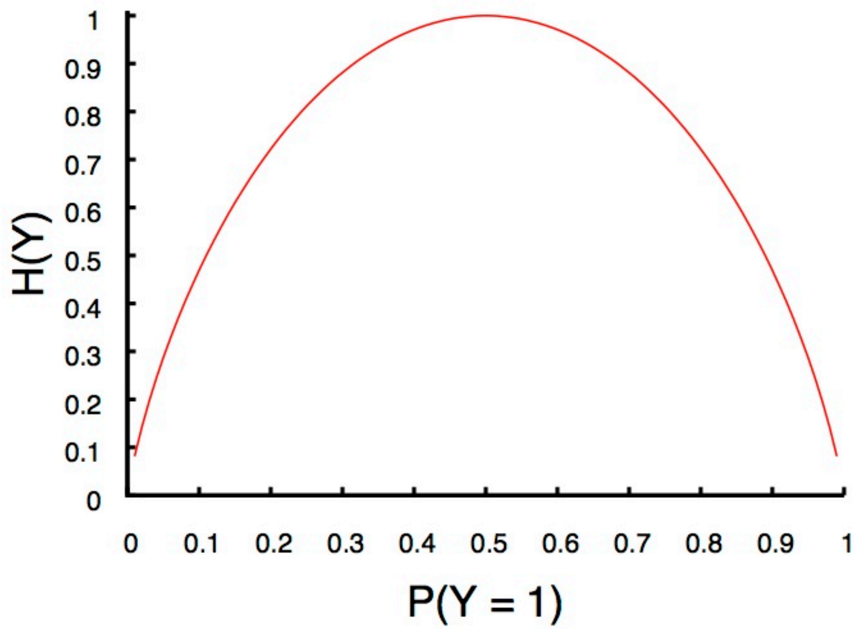
$P(Y\text{=a}) = 0.5$
$P(Y\text{=b}) = 0.25$
$P(Y\text{=c}) = 0.25$

$H(Y) = 1.5$

Entropy of Y = the average number of bits required to specify Y

Bit encoding for Y: a = 1 b = 01 c = 00

Entropy (binary case)



Entropy as a measure of uncertainty:

H(Y) is maximized when $p = 0.5$

(uniform distribution)

H(Y) is minimized when $p = 0$ or $p = 1$

Sample entropy (binary case)

Entropy of a sample (data set) S $= \{(\mathbf{x}, y)\}$
with $N = N^+ + N^-$ items

Use the sample to estimate P(Y):

$p = N^+/N$ N^+ = number of positive items (Y = 1)

$n = N^-/N$ N^- = number of negative items (Y = 0)

This gives $H(S) = -p \log_2 p - n \log_2 n$

H(S) measures the *impurity* of S

Using entropy to guide decision tree learning

At each step, we want to split a node to reduce the label entropy

- H(Y) = entropy of (distribution of) class labels P(Y)
- For decision tree learning, we only care about H(Y); We don't care about H($\mathbf{X}$), the entropy of the features $\mathbf{X}$
- Define H(S) = label entropy H(Y) of the sample S

Entropy reduction = Information gain

Information Gain = $H(S_{\text{before split}}) - H(S_{\text{after split}})$

Using entropy to guide decision tree learning

- The parent node S has entropy $H(S)$ and size $|S|$
- Splitting S on feature $\mathbf{X}_i$ with values 1,…,k yields k children S_1, …, S_k with entropy $H(S_k)$ & size $|S_k|$
- After splitting S on $\mathbf{X}_i$ the expected entropy is

$$\sum_k \frac{|S_k|}{|S|} H(S_k)$$

Using entropy to guide decision tree learning

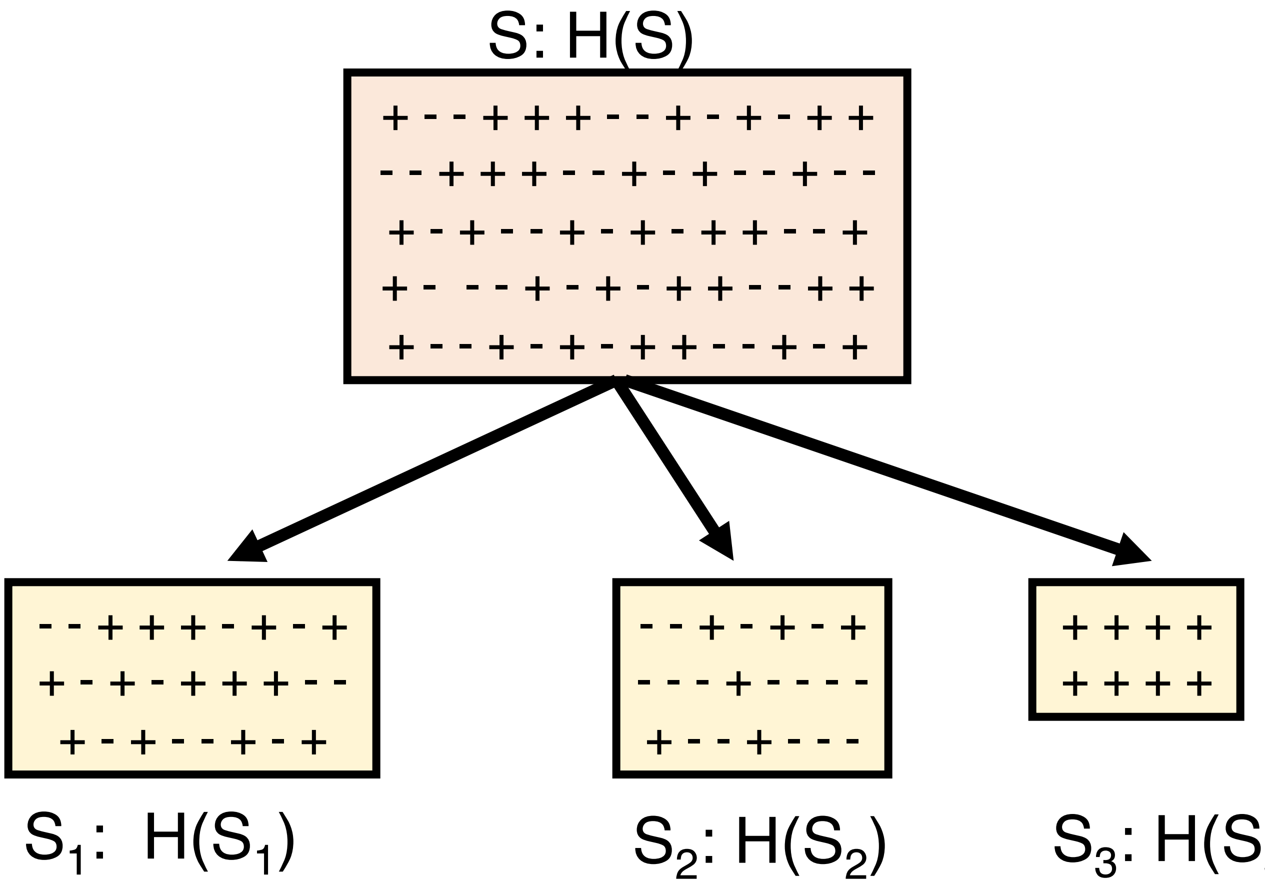
- The parent S has entropy $H(S)$ and size |S|
- Splitting S on feature $\mathbf{X}_i$ with values 1,…,k yields k children S_1, …, S_k with entropy $H(S_k)$ & size $|S_k|$
- After splitting S on $\mathbf{X}_i$ the expected entropy is

$$\sum_k \frac{|S_k|}{|S|} H(S_k)$$

- When we split S on $\mathbf{X}_i$, the information gain is:

$$\boxed{Gain(S, X_i) = H(S) - \sum_k \frac{|S_k|}{|S|} H(S_k)}$$

Information Gain



$$Gain(S, X_i) = H(S) - \left(\frac{|S_1|}{|S|} H(S_1) + \frac{|S_2|}{|S|} H(S_2) + \frac{|S_3|}{|S|} H(S_3) \right)$$

Will I play tennis today?

Features

- Outlook: {Sun, Overcast, Rain}
- Temperature: {Hot, Mild, Cool}
- Humidity: {High, Normal, Low}
- Wind: {Strong, Weak}

Labels

- Binary classification task: $Y = \{+, -\}$

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny), O(vercast), R(ainy)

Temperature: H(ot), M(edium), C(ool)

Humidity: H(igh), N(ormal), L(ow)

Wind: S(trong), W(eak)

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Current entropy:

$p = 9/14$

$n = 5/14$

$H(\mathrm{Y}) =$

$-(9/14)\log_2(9/14)$

$-(5/14)\log_2(5/14)$

≈ 0.94

Information Gain: Outlook

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook = sunny:

$p = 2/5 \quad n = 3/5 \quad \mathbf{H_S = 0.971}$

Outlook = overcast:

$p = 4/4 \quad n = 0 \quad \mathbf{H_o = 0}$

Outlook = rainy:

$p = 3/5 \quad n = 2/5 \quad \mathbf{H_R = 0.971}$

Expected entropy:

$(5/14)\times 0.971 + (4/14)\times 0$
$+ (5/14)\times 0.971 = \mathbf{0.694}$

Information gain:

$0.940 - 0.694 = \mathbf{0.246}$

Which feature to split on?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Information gain:

Outlook: 0.246
Humidity: 0.151
Wind: 0.048
Temperature: 0.029

→ Split on Outlook

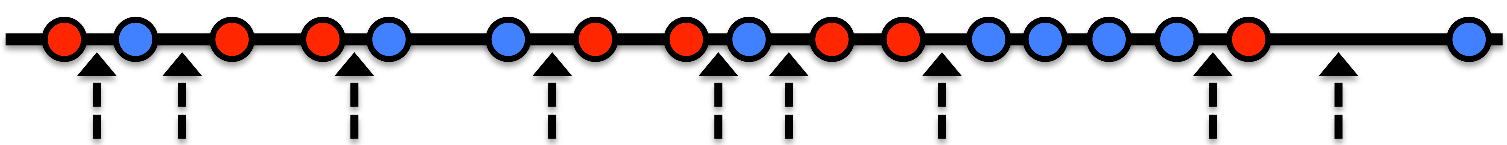
Continuous-valued features

If a feature X_i has continuous (real) values, we need to find a threshold T to split X_i on:

– Left child: $X_i \leq T$

– Right child: $X_i > T$

Possible thresholds occur between items with different class labels:



induceDecisionTree(S)

1. Does S uniquely define a class?

    if all $s \in S$ have the same label y: **return** S;

2. Find the feature with the most information gain:

    $i = \text{argmax}_{\,i}\ Gain(S, X_i)$

3. Add children to S:

    for k **in** Values(X_i):

    $S_k = \{s \in S \mid x_i = k\}$

    addChild(S, S_k)

    induceDecisionTree(S_k)

    return S;

Caveat:
No item in S has value $X_i = k$

- **Training:**
  $|S_k| = 0$, so the k-th value of X_i contributes 0 to *Gain*(S, X_i)

- **Testing:**
  If a test item that reaches S has $X_i = k$:
  Assign the most common class label (in S)

Caveat: Value of feature X_i is missing for **s**

NB: This means the value of X_i is unknown for **s**, not 'false'

Compute the probability of each value at S:

$$P(X_i = k) = |S_k| / |S|$$

Two possibilities:

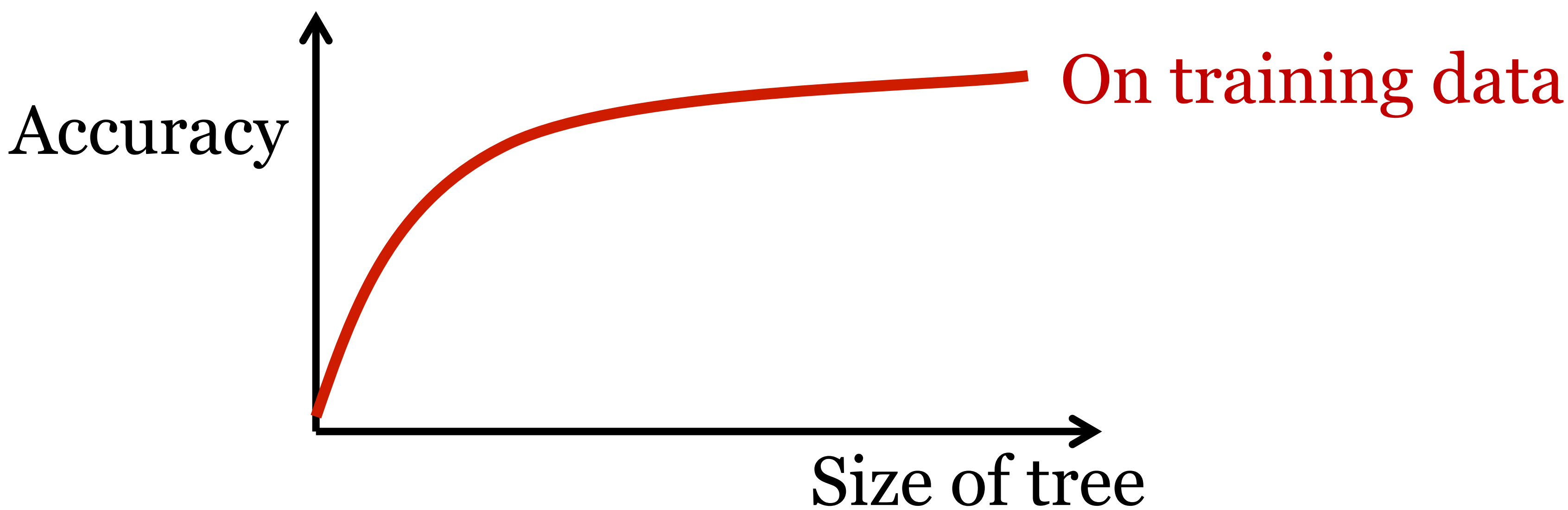
– Assign the most likely value of X_i to **s**:

$$\text{argmax}_k\, P(X_i = k)$$

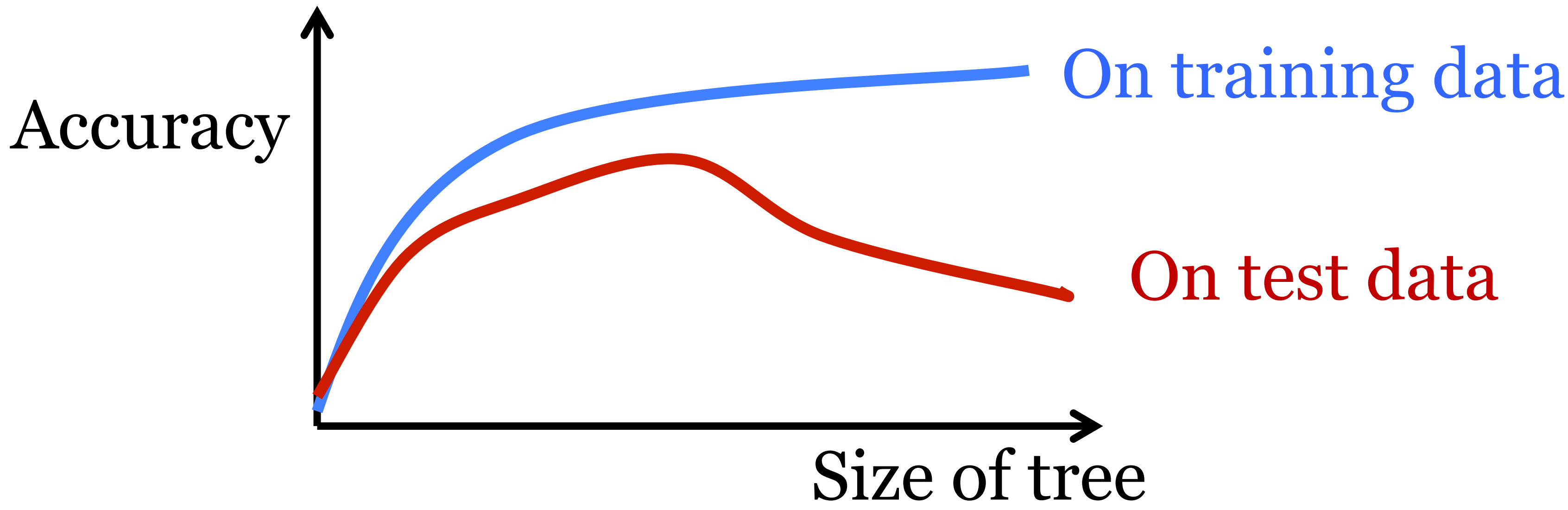
– Assign fractional counts $P(X_i = k)$ for each value of X_i to **s**

Learning curve

The accuracy on the training data will increase as we add more levels to the tree

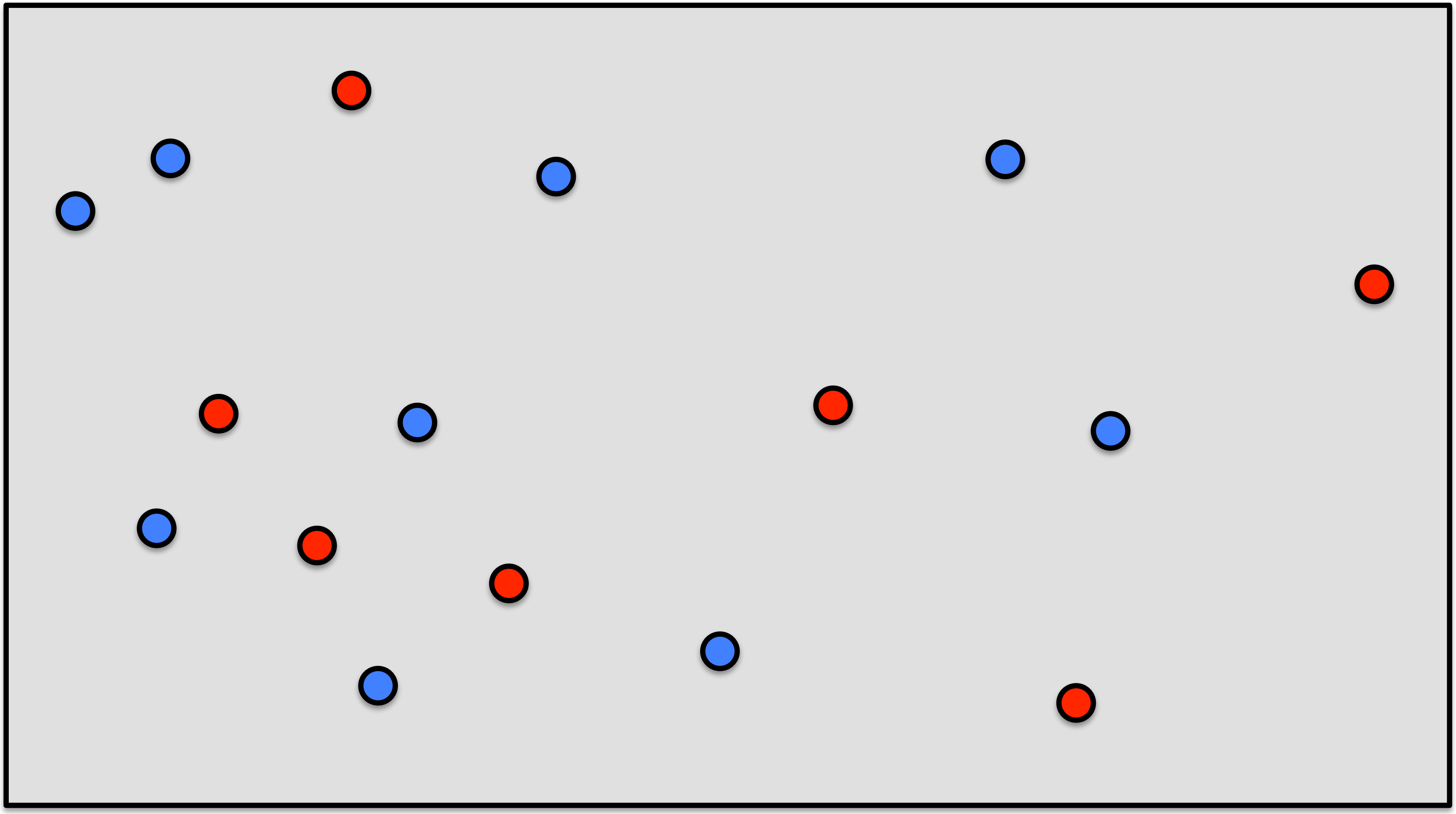


Overfitting

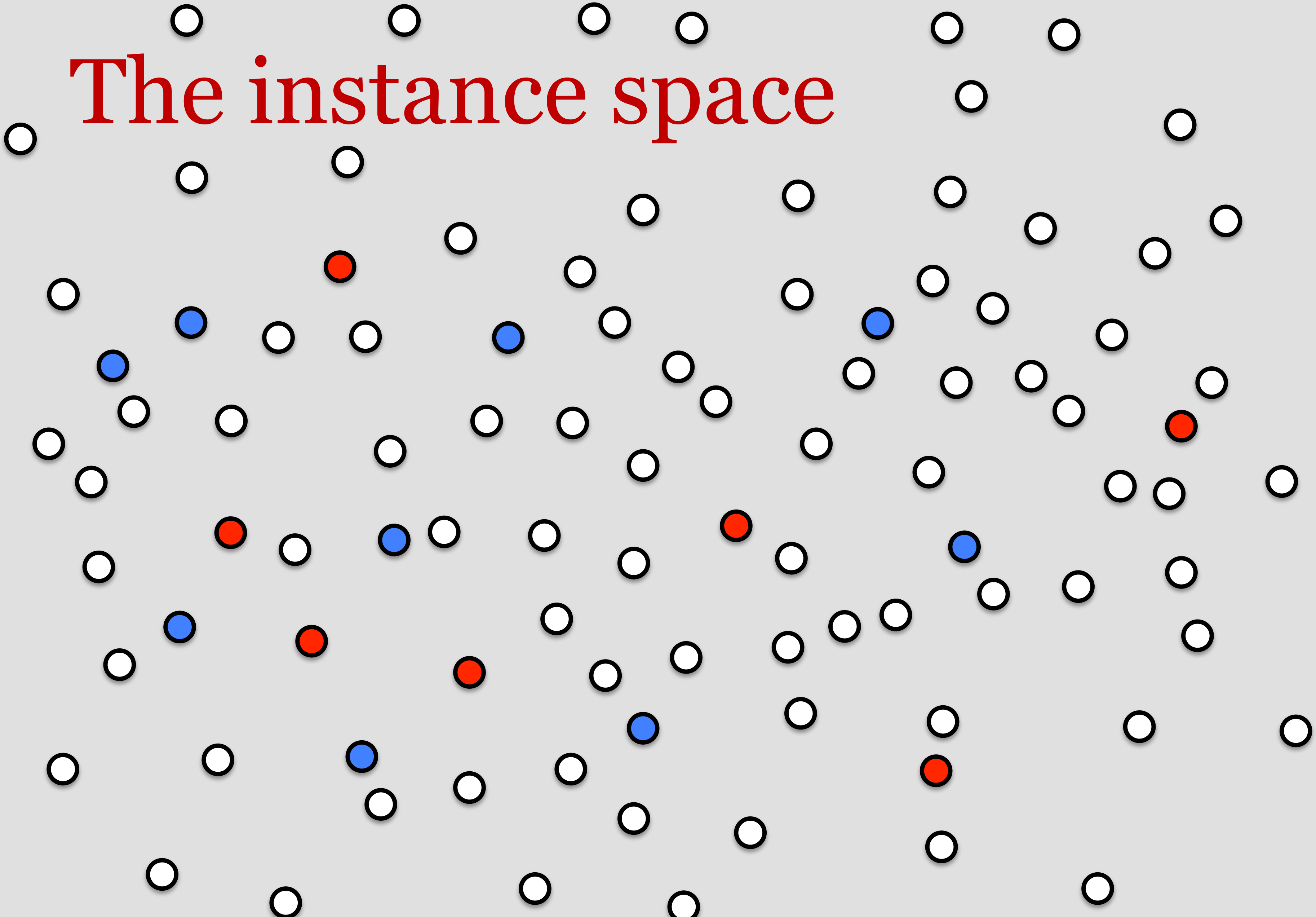


A classifier overfits the training data
when its accuracy on the training data goes up
but its accuracy on unseen data goes down

Our training data



The instance space



Reasons for overfitting

Too much **variance** in the training data

- Training data is not a representative sample of the instance space
- We split on features that are actually irrelevant

Too much **noise** in the training data

- Noise = some feature values or class labels are incorrect
- We learn to predict the noise

Reducing overfitting

Various heuristics are commonly used:

- Limit the depth of the tree
- Require a minimum number of examples per node used to select a split
- Learn a complete tree and prune, using validation (held-out) data

Pruning a decision tree

Pruning = Remove leaves and assign majority label of the parent to all items

Prune the children of S if:

– all children are leaves, and

– the accuracy on the validation set does not decrease if we assign the most frequent class label to all items at S.

Today's key concepts

Decision trees for (binary) classification

Non-linear classifiers

Learning decision trees (ID3 algorithm)

Greedy heuristic (based on information gain)
Originally developed for discrete features

Overfitting

What is it? How do we deal with it?