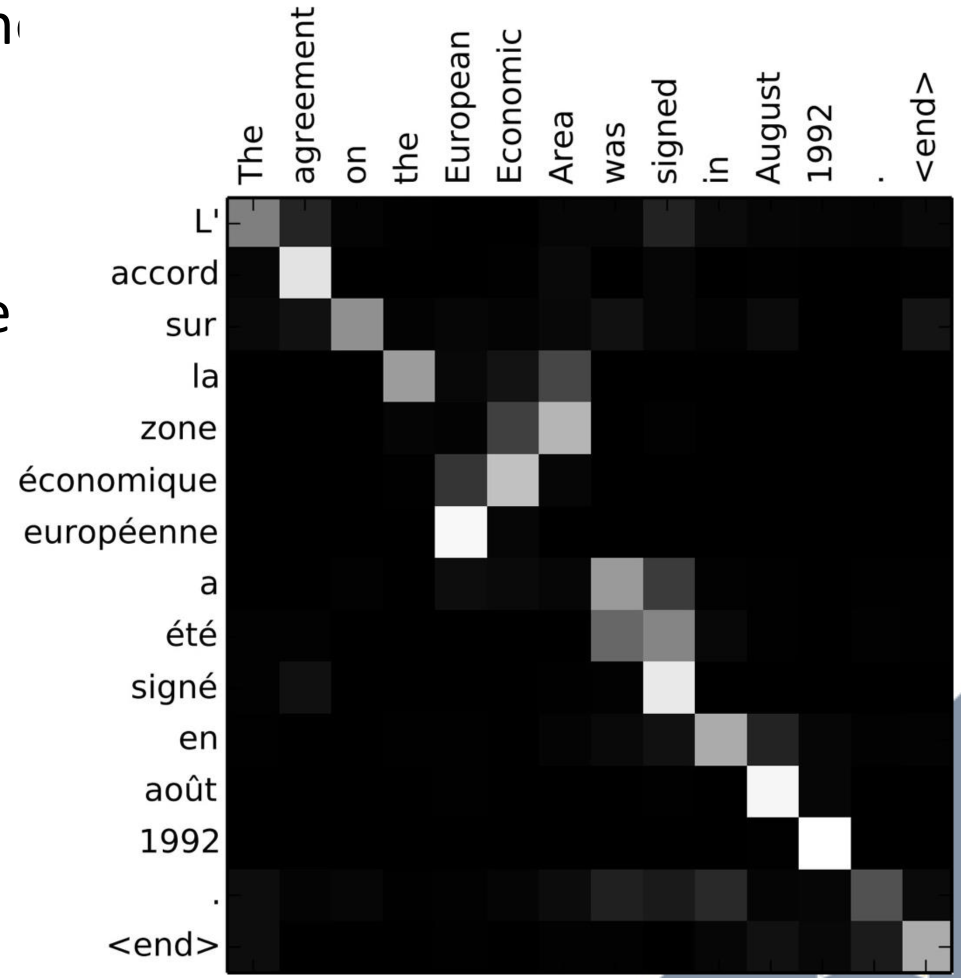
CSE 176 Introduction to Machine Learning
Lecture 14: Decision Tree

Some materials from Stuart Russell and Miguel Carreira-Perpiñán

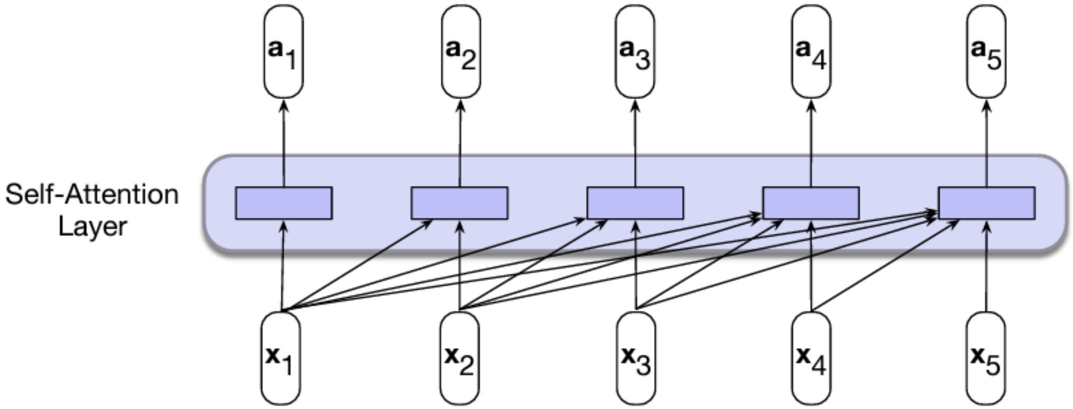
Recap: Attention between words

- ❑Example: English to French translation
- ❑Input: "The agreement on the European Economic Area was signed in August 1992."
- ❑Output: "L'accord sur la zone économique européenne a été signé en août 1992."



Recap: Casual or backward-looking self-attention

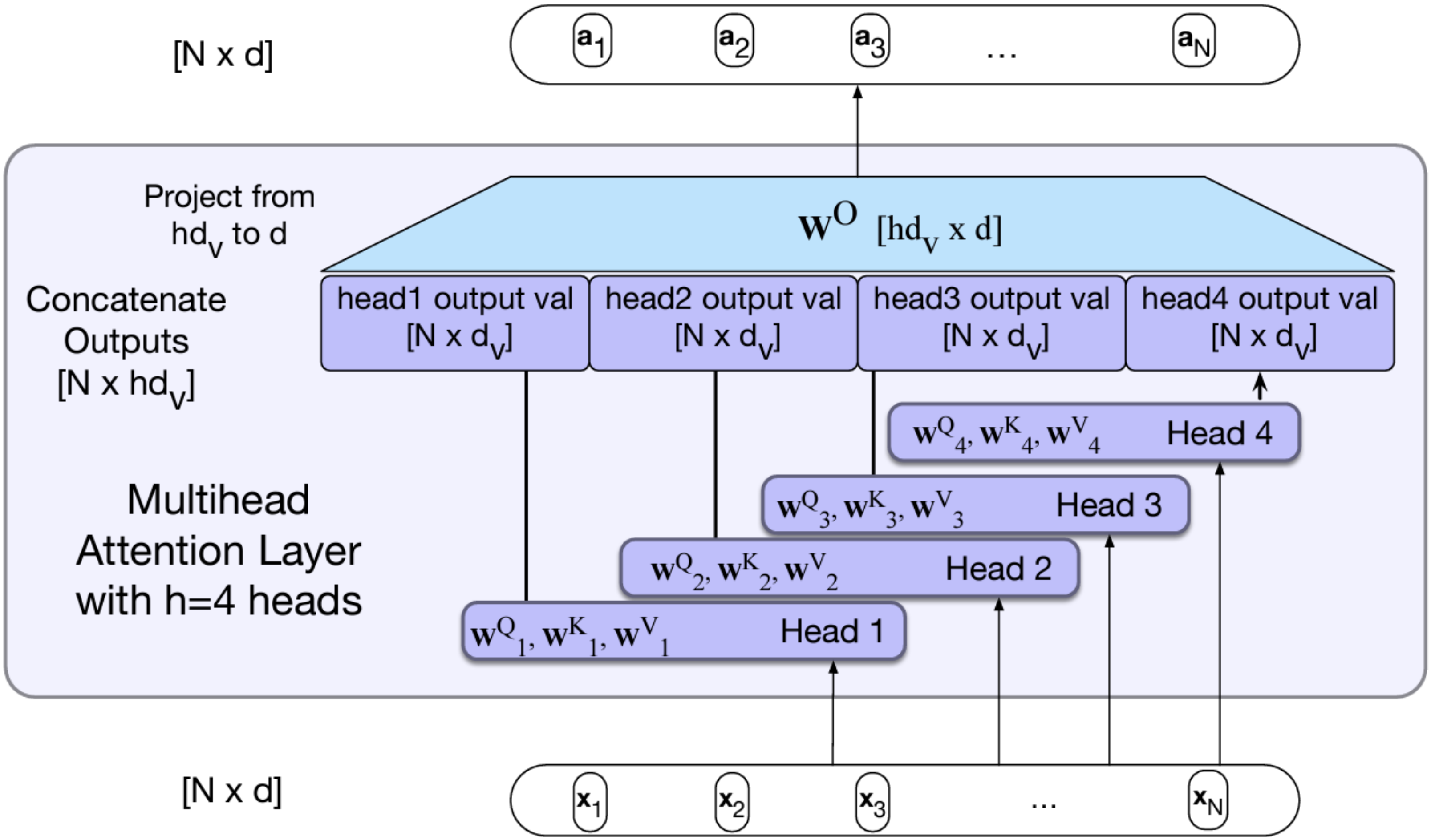
- ❑Attends to all the inputs up to, and including, the current one



Recap: Self-attention

$$
\begin{aligned}
\mathbf{q}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{Q}}; \mathbf{k}_i &= \mathbf{x}_i \mathbf{W}^{\mathbf{K}}; \mathbf{v}_i = \mathbf{x}_i \mathbf{W}^{\mathbf{V}} \\
\mathrm{score}(\mathbf{x}_i, \mathbf{x}_j) &= \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}} \\
\alpha_{ij} &= \mathrm{softmax}(\mathrm{score}(\mathbf{x}_i, \mathbf{x}_j)) \;\; \forall j \leq i \\
\mathbf{a}_i &= \sum_{j \leq i} \alpha_{ij} \mathbf{v}_j
\end{aligned}
$$

Recap: Multi-head attention

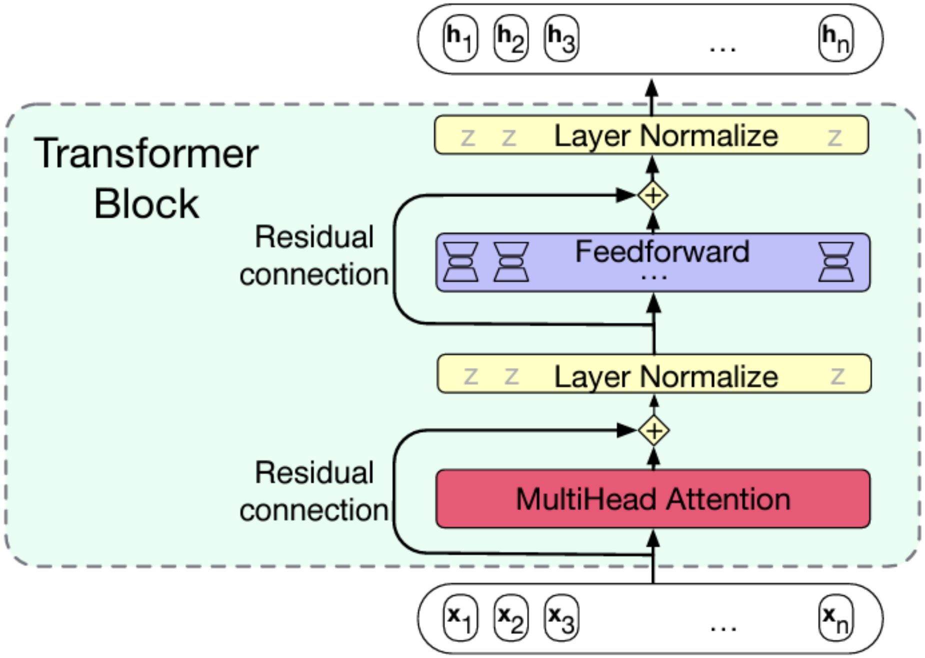


Self attention v.s. Cross attention

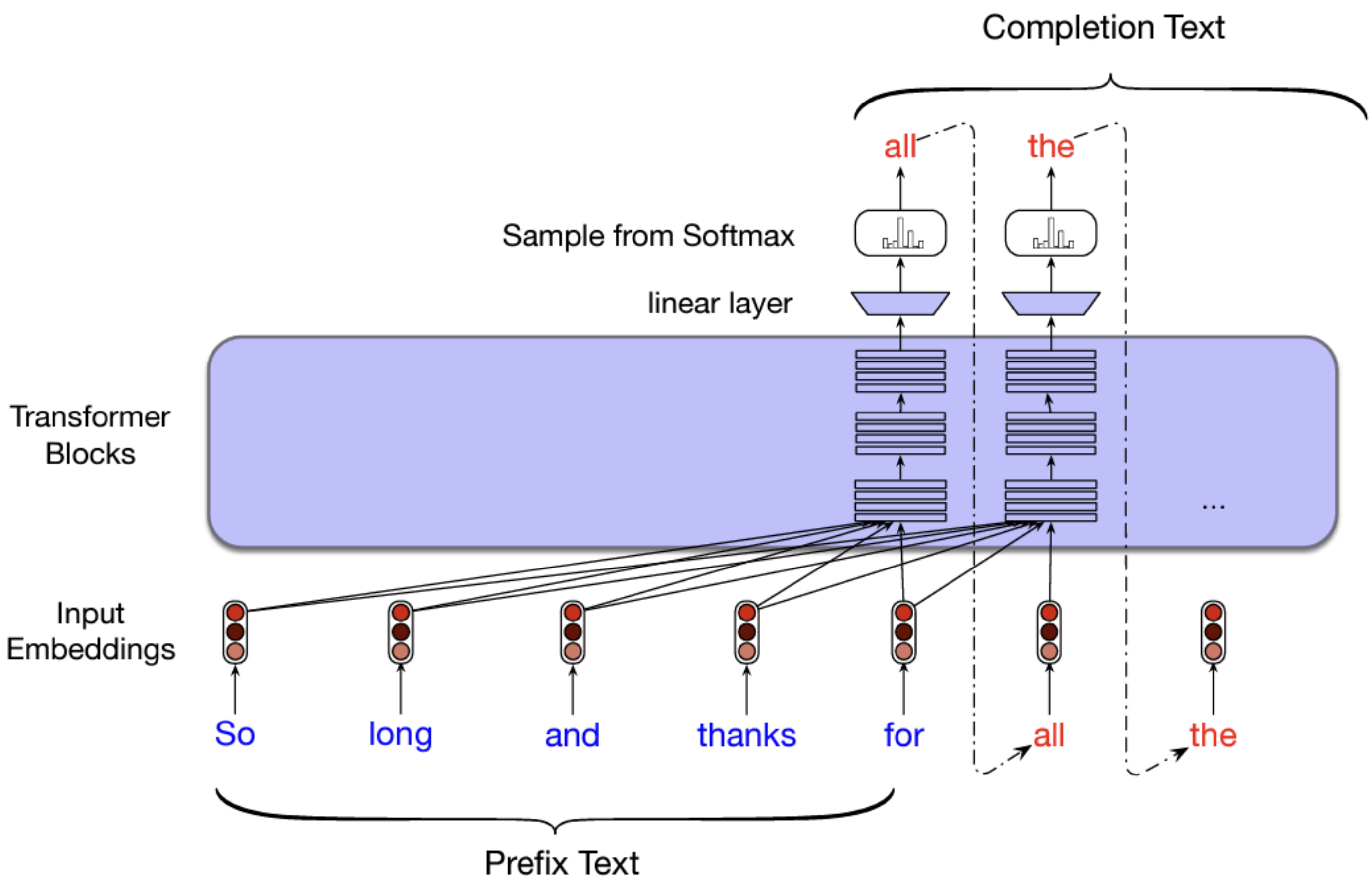
- Self Attention
  - Key, Value, and Query from the same set of tokens
- Cross Attention
  - Key, and Value from one set of tokens
  - Query from another set of tokens
  - E.g. words in one language pay attention to words in another.

From Attention to Transformer Block

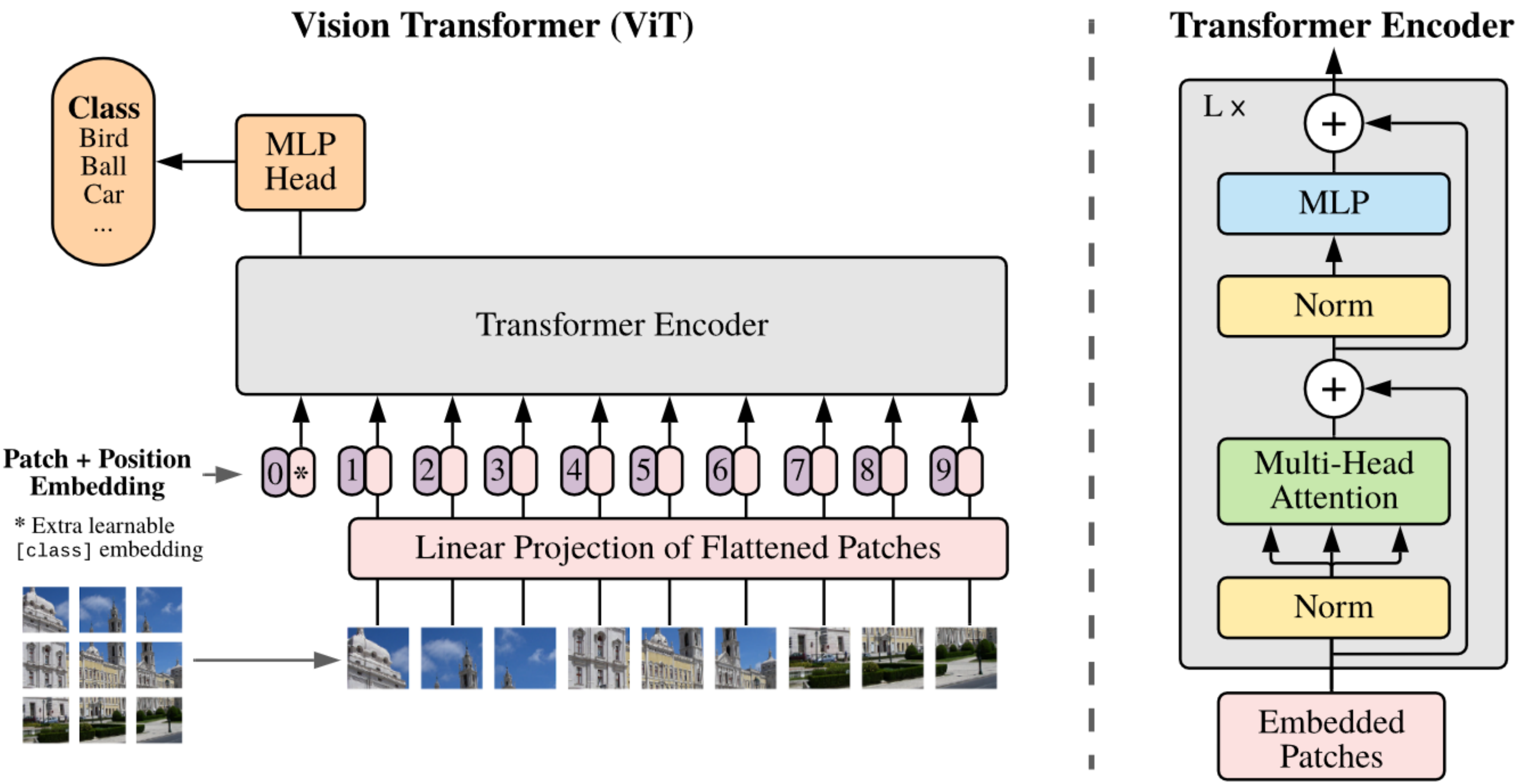
- A transformer block has
  - **Self Attention**: information exchange between tokens
  - **Feed forward network**: Information transform within tokens
    - E.g. a multi-layer perceptron with 1 hidden layer
  - **Normalization** (Layer normalization)
  - **Residual connection**



Recap: Transformer-based Large Language Model



Recap: Vision Transformer



Decision Tree

Recap: The XOR problem

Minsky and Papert (1969)

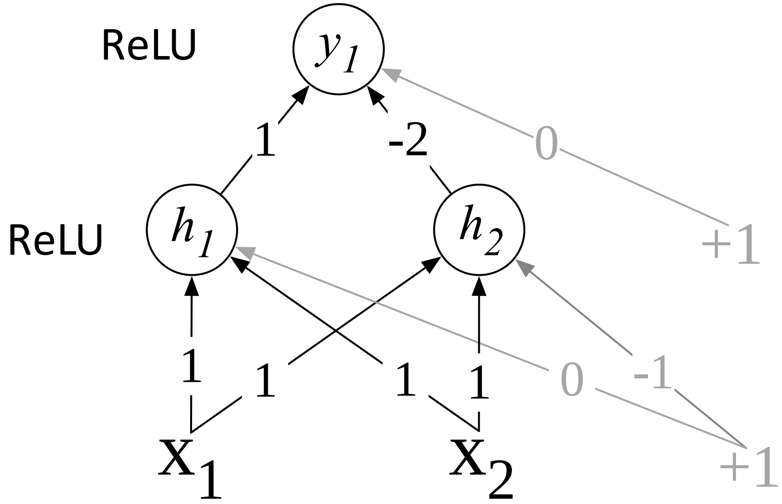
- Can perceptron compute simple functions of input?

AND			OR			XOR		
x1	x2	y	x1	x2	y	x1	x2	y
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

Recap: Solution to the XOR problem

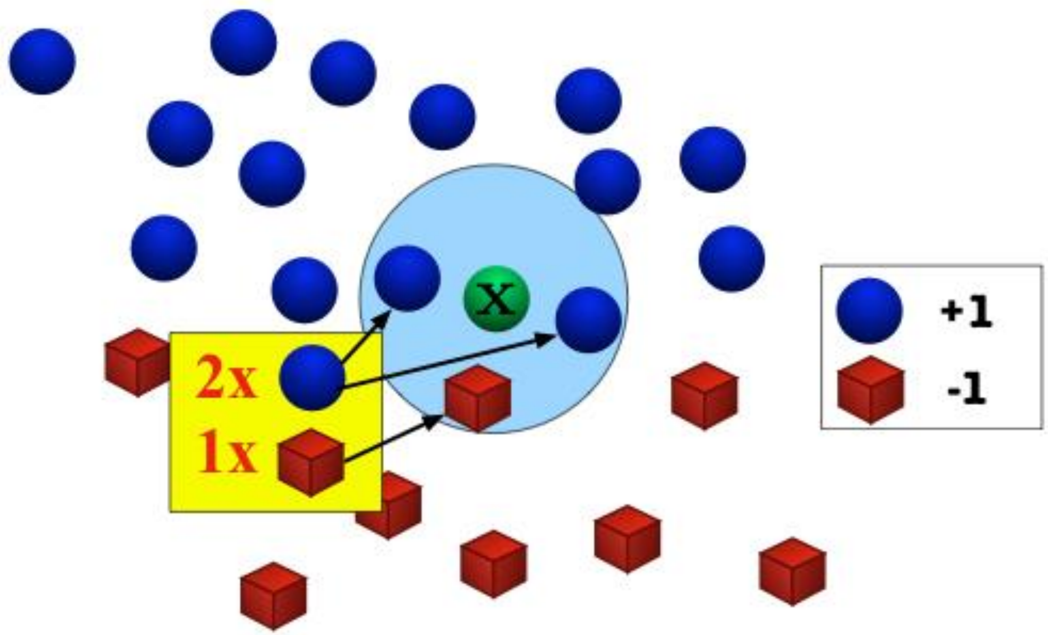
- XOR **can't** be calculated by a single perceptron
- XOR **can** be calculated by a layered network of units.

XOR		
x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



Recap: K nearest neighbor algorithm

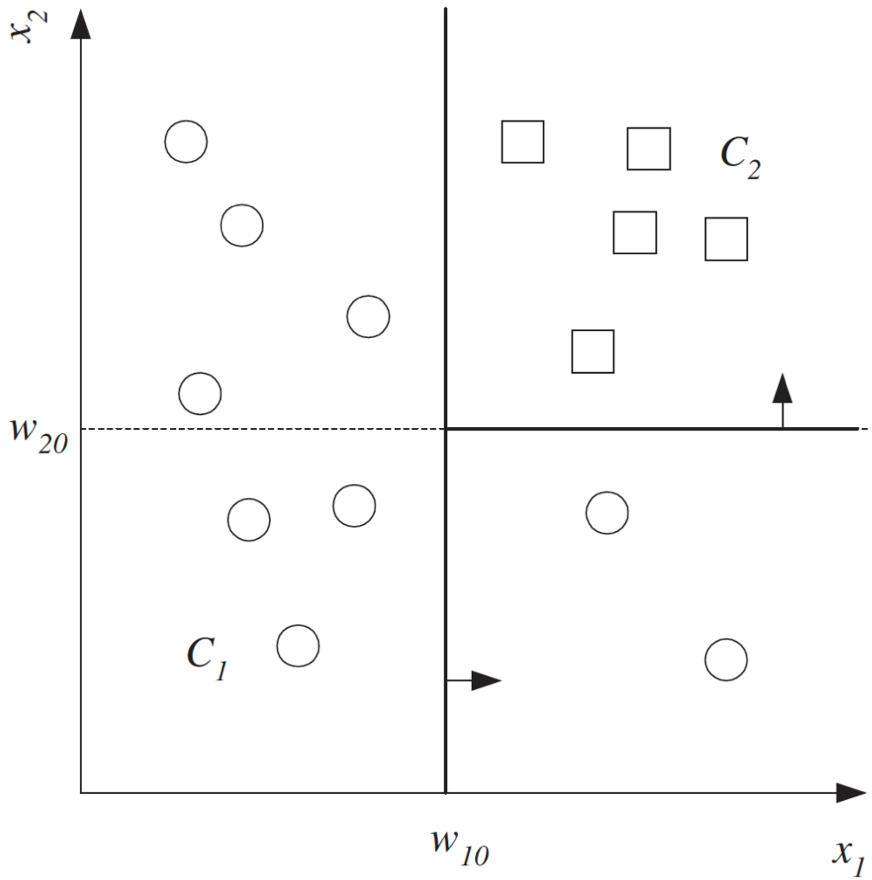
- Nearest neighbor often instable (noise)
- For a test input ***x***, assign the most common label amongst its k most similar training inputs



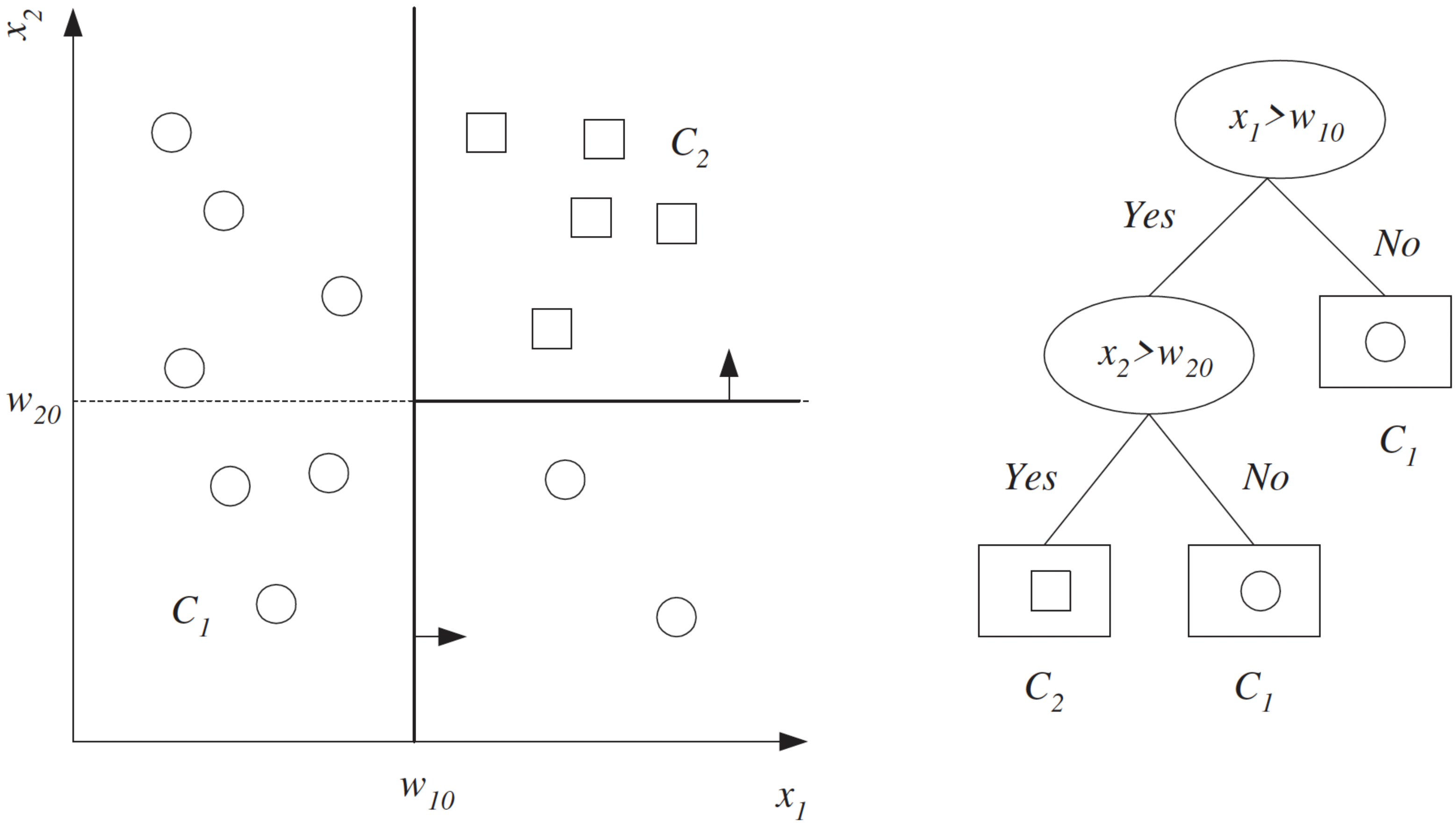
Decision Tree

- ❑Unlike neural network, decision tree is interpretable
- ❑Unlike KNN algorithm, decision tree doesn't store training data
- ❑Can be used for both classification and regression
- ❑Select feature automatically
- ❑Efficient at test-time

Binary Classification

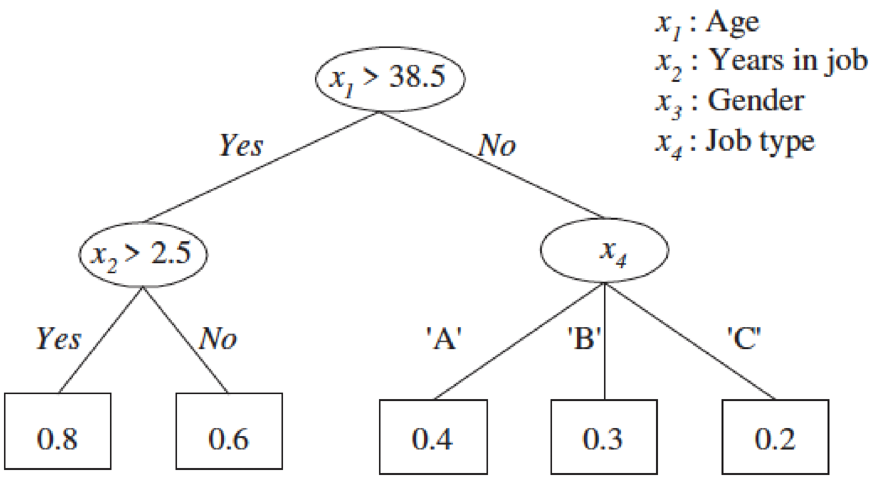


An example of Decision Tree



Rule extraction from decision tree

- A regression tree:



- R1: IF (age > 38.5) AND (years-in-job > 2.5) THEN y = 0.8
- R2: IF (age > 38.5) AND (years-in-job ≤ 2.5) THEN y = 0.6
- R3: IF (age ≤ 38.5) AND (job-type = ‘A’) THEN y = 0.4
- R4: IF (age ≤ 38.5) AND (job-type = ‘B’) THEN y = 0.3
- R5: IF (age ≤ 38.5) AND (job-type = ‘C’) THEN y = 0.2

Training Decision Tree

Restaurant Example

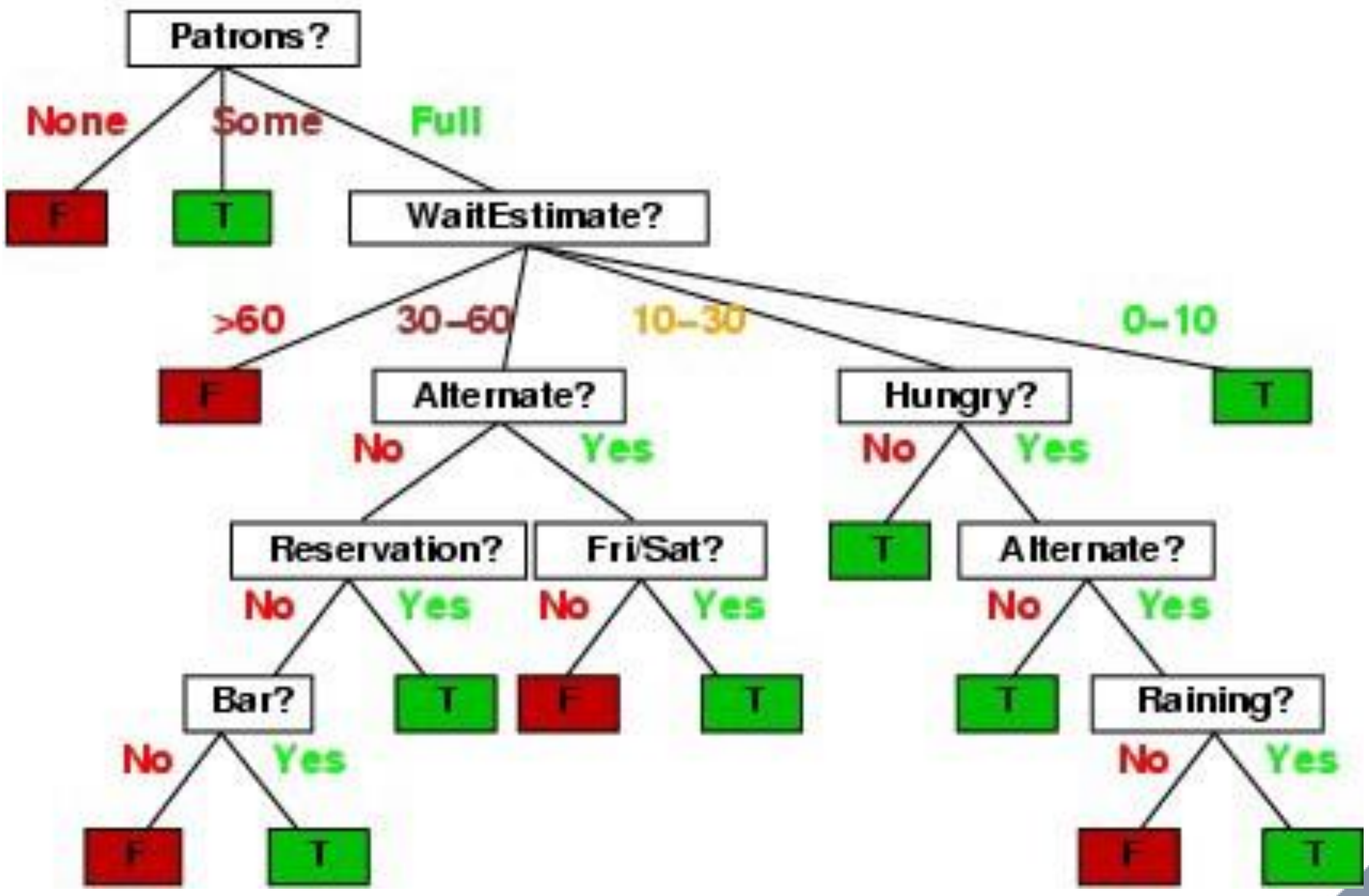
- Develop decision tree that customers make when deciding whether to wait for a table or leave
- Two classes: **wait**, **leave**
- Ten attributes:
    - Alternative available?
    - Bar in restaurant?
    - Is it Friday?
    - Are we hungry?
    - How full is restaurant?
    - How expensive?
    - Is it raining?
    - Do we have reservation?
    - What type of restaurant is it?
    - Estimated waiting time?

Training data

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Decision Tree

- How to decide whether to wait?



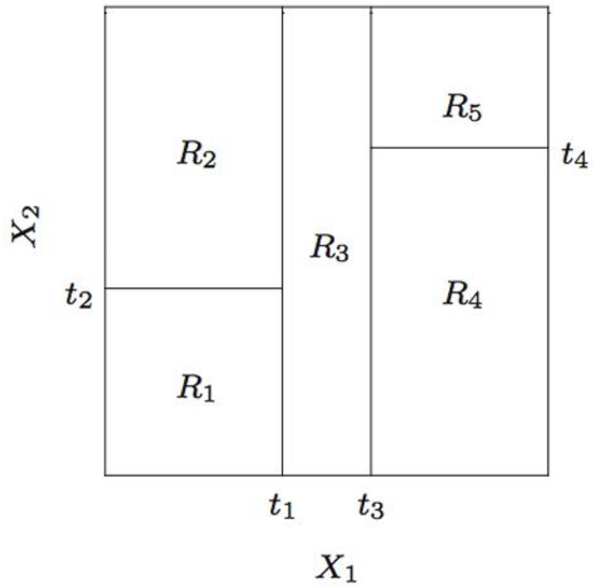
Discrete and continuous input

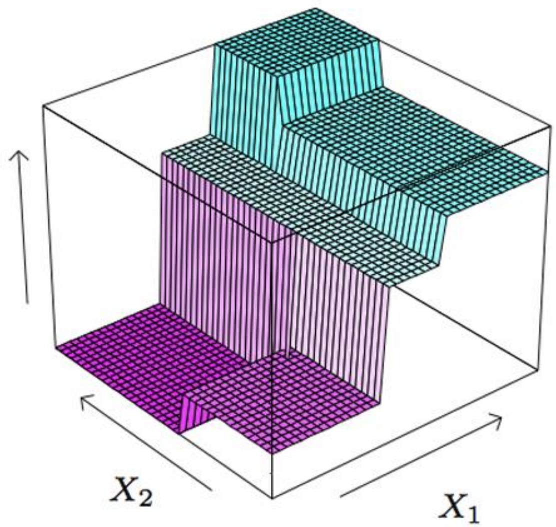
Simplest case: discrete inputs with small ranges (e.g., Boolean)
$\Rightarrow$ one branch for each value; attribute is “used up” (“complete split”)

Discrete and continuous input

Simplest case: discrete inputs with small ranges (e.g., Boolean)
$\Rightarrow$ one branch for each value; attribute is "used up" ("complete split")

For continuous attribute, test is $X_j > c$ for some split point c
$\Rightarrow$ two branches, attribute may be split further in each subtree





Also split large discrete ranges into two or more subsets

ID3 / C4.5 / J48 Algorithm

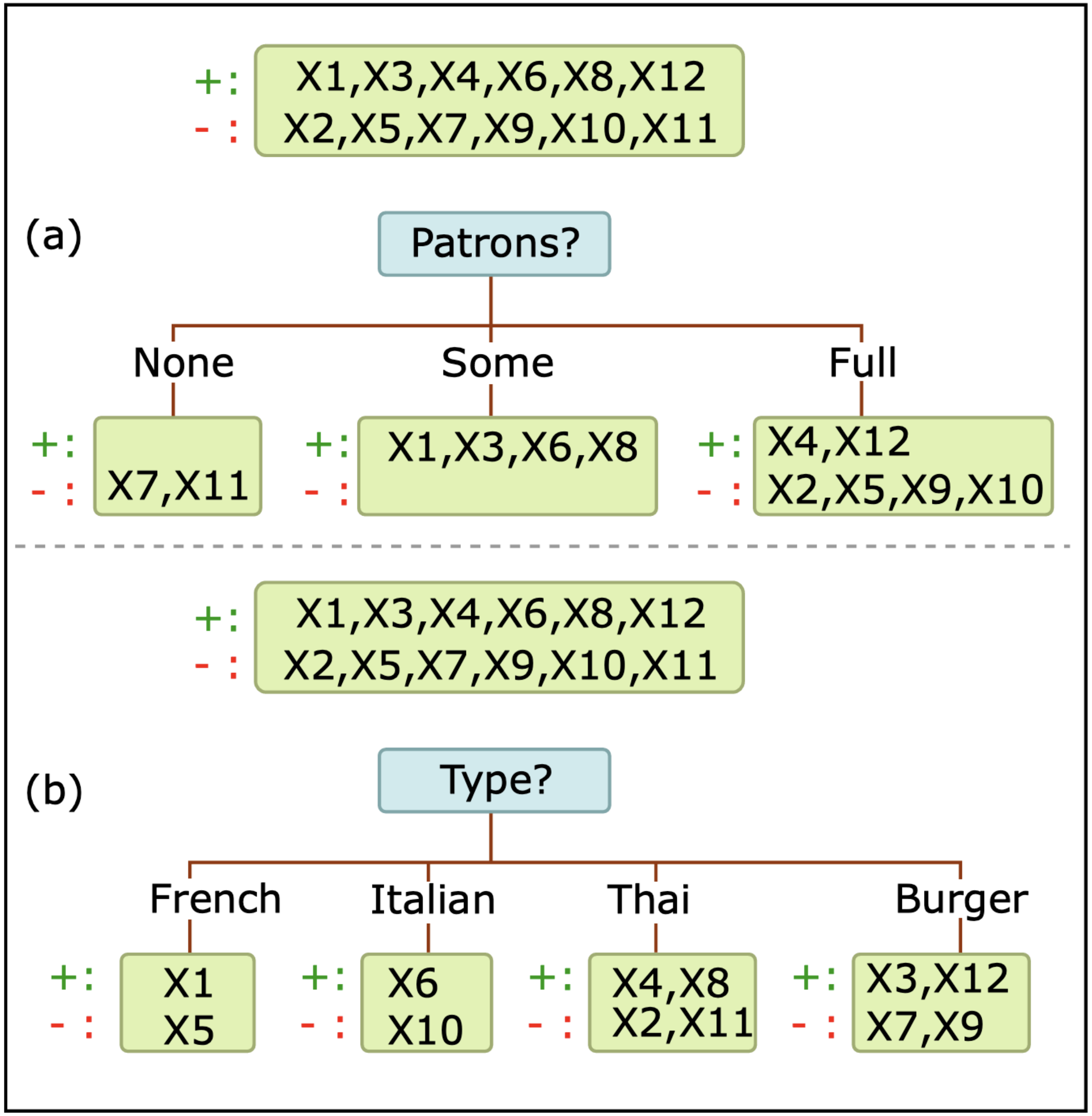
- Greedy algorithm developed by Ross Quinlan in 1987
- Top-down construction by recursively selecting best attribute to use at current node
  - Once attribute selected for current node, generate child nodes
- Partition examples using values of attribute
- Repeat for each child node until examples associated with a node are all positive or negative

Which feature/attribute to split first?

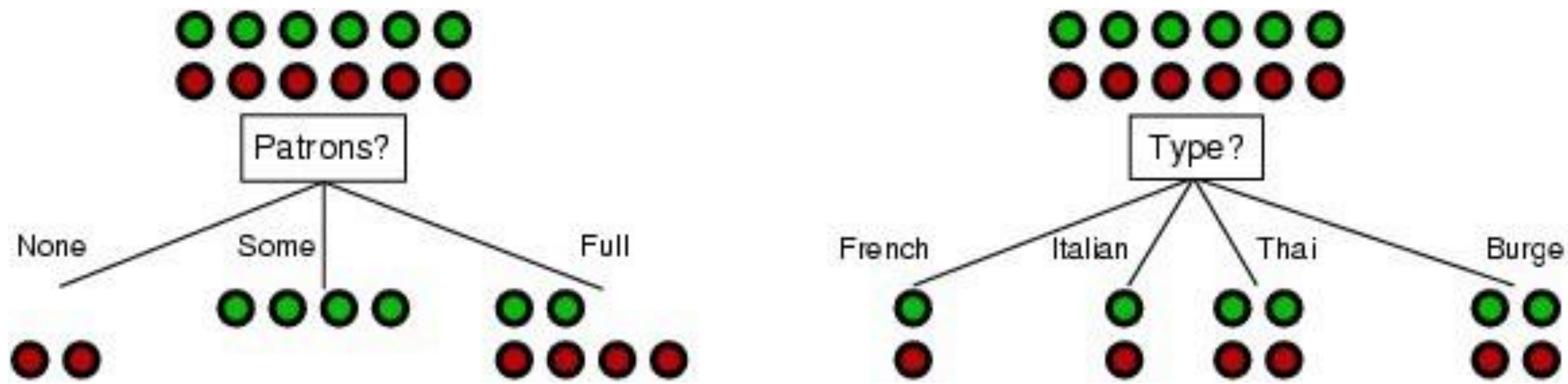
- ❑Probably Patron and Type

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Which feature/attribute to split first?



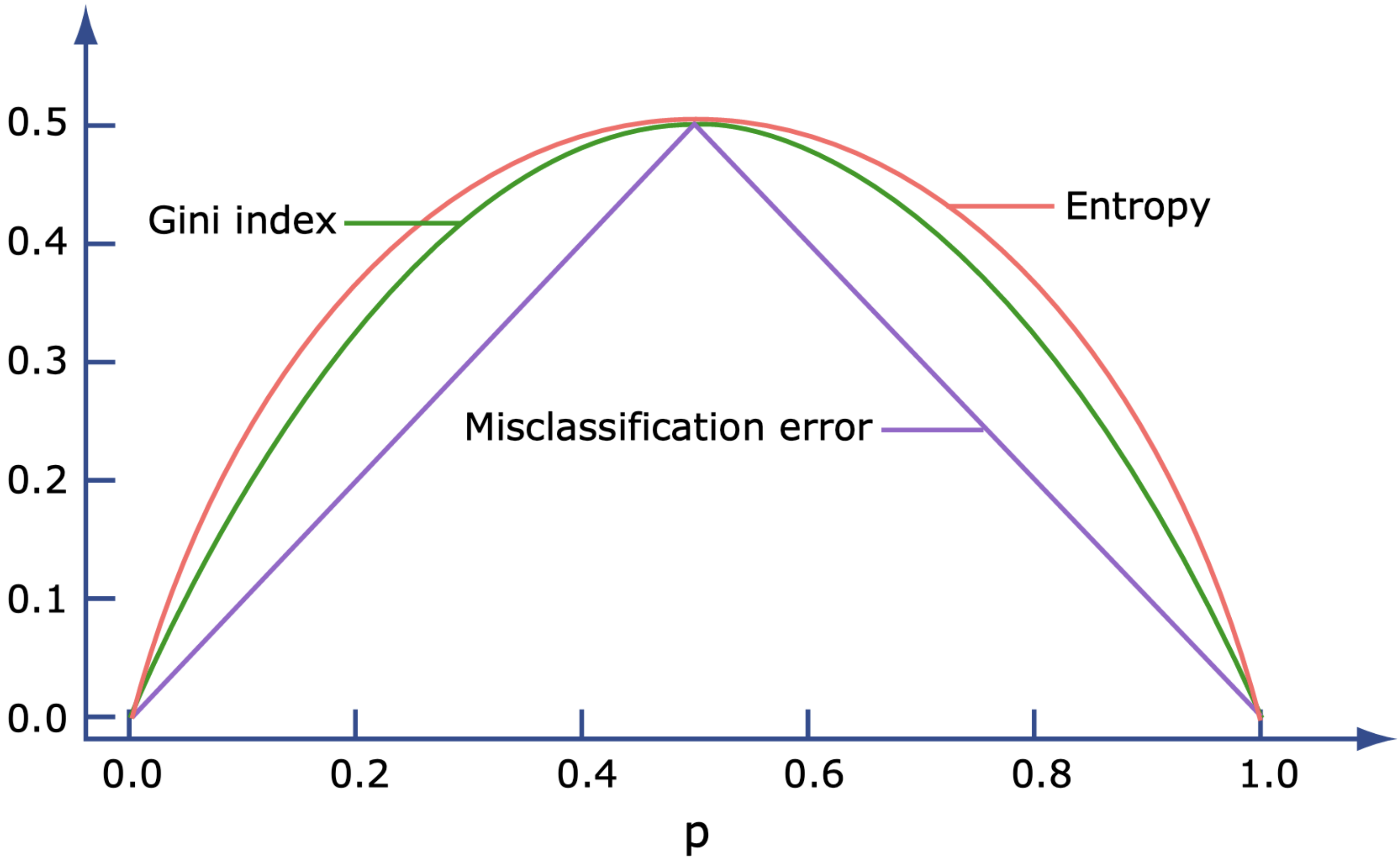
Which feature/attribute to split first?



- ❑Idea: good attribute splits examples into subsets that are (ideally) *all positive* or *all negative*

Purity Criterion

- A node is pure if it contains instances of the same class



Information and Entropy

Information answers questions

The more clueless I am about the answer initially, the more information is contained in the answer

Scale: 1 bit $=$ answer to Boolean question with prior $\langle 0.5, 0.5 \rangle$

Information in an answer when prior is $\langle P_1, \ldots, P_n \rangle$ is

$$H(\langle P_1, \ldots, P_n \rangle) = \Sigma_{i=1}^{n} - P_i \log_2 P_i$$

(also called entropy of the prior)

Convenient notation: $B(p) = H(\langle p, 1-p \rangle)$

Information before split

Suppose we have p positive and n negative examples at the root

$\Rightarrow \quad B(p/(p+n)$ bits needed to classify a new example

E.g., for 12 restaurant examples, $p = n = 6$ so we need 1 bit

Information after split

Suppose we have p positive and n negative examples at the root

$\Rightarrow \quad B(p/(p+n)$ bits needed to classify a new example

E.g., for 12 restaurant examples, $p = n = 6$ so we need 1 bit

An attribute splits the examples E into subsets E_k, each of which (we hope) needs less information to complete the classification

Let E_k have p_k positive and n_k negative examples

$\Rightarrow \quad B(p_k/(p_k+n_k)$ bits needed to classify a new example

$\Rightarrow$ **expected** number of bits per example over all branches is

$$\Sigma_i \frac{p_i + n_i}{p+n} B(p_k/(p_k+n_k))$$

For $Patrons$, this is 0.459 bits, for $Type$ this is (still) 1 bit

$\Rightarrow$ choose the attribute that minimizes the remaining information needed

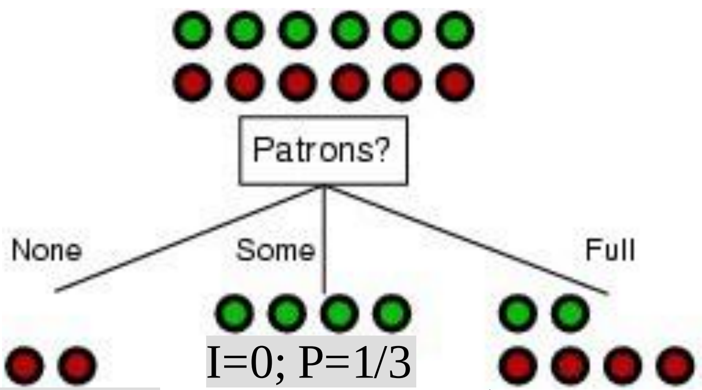
Information Gain

- Gain(X,T) = Info(T) - Info(X,T) is difference of
  - info needed to identify element of T and
  - info needed to identify element of T after attribute X known
- This is gain in information due to attribute X
- Used to rank attributes and build DT

Information Gain

stay
leave

$I = -.5*\log_2(.5) - .5*\log_2(.5) = 0.5+0.5 = 1$

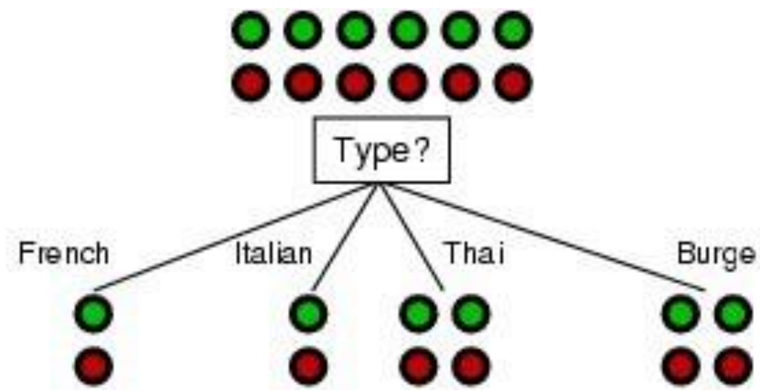


I=0; P=1/6

I=0; P=1/3

$I=-(1/3*\log_2(1/3)-2/3*\log_2(2/3); P=1/2$
$I*P=0.46$

Information gain = 1 - 0.46 = 0.54



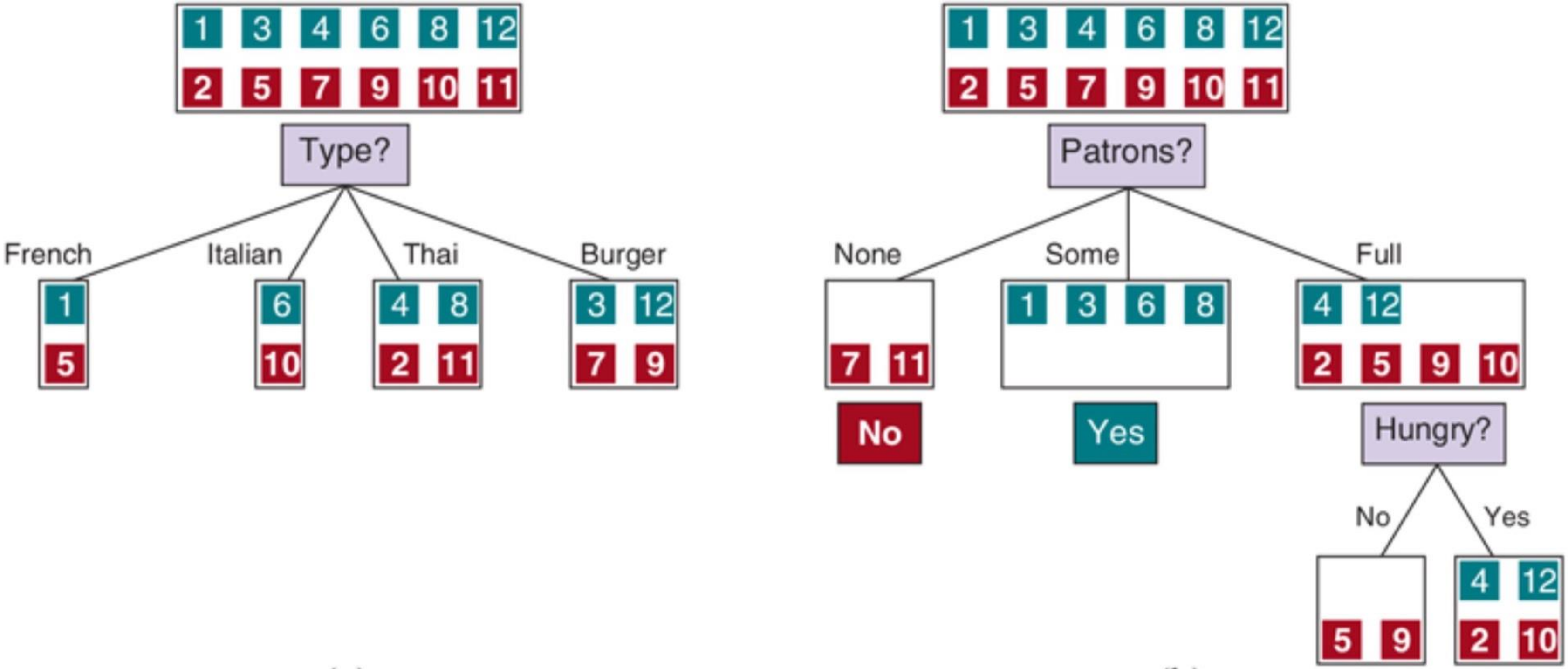
I=1;P=1/6 I=1; P=1/6 I=1; P=2/6 I=1; P=2/6

$I = 6/6*1 = 1$

Information gain = 1 - 1 = 0

- **Information gain** for asking Patrons is 0.54, for asking Type is 0

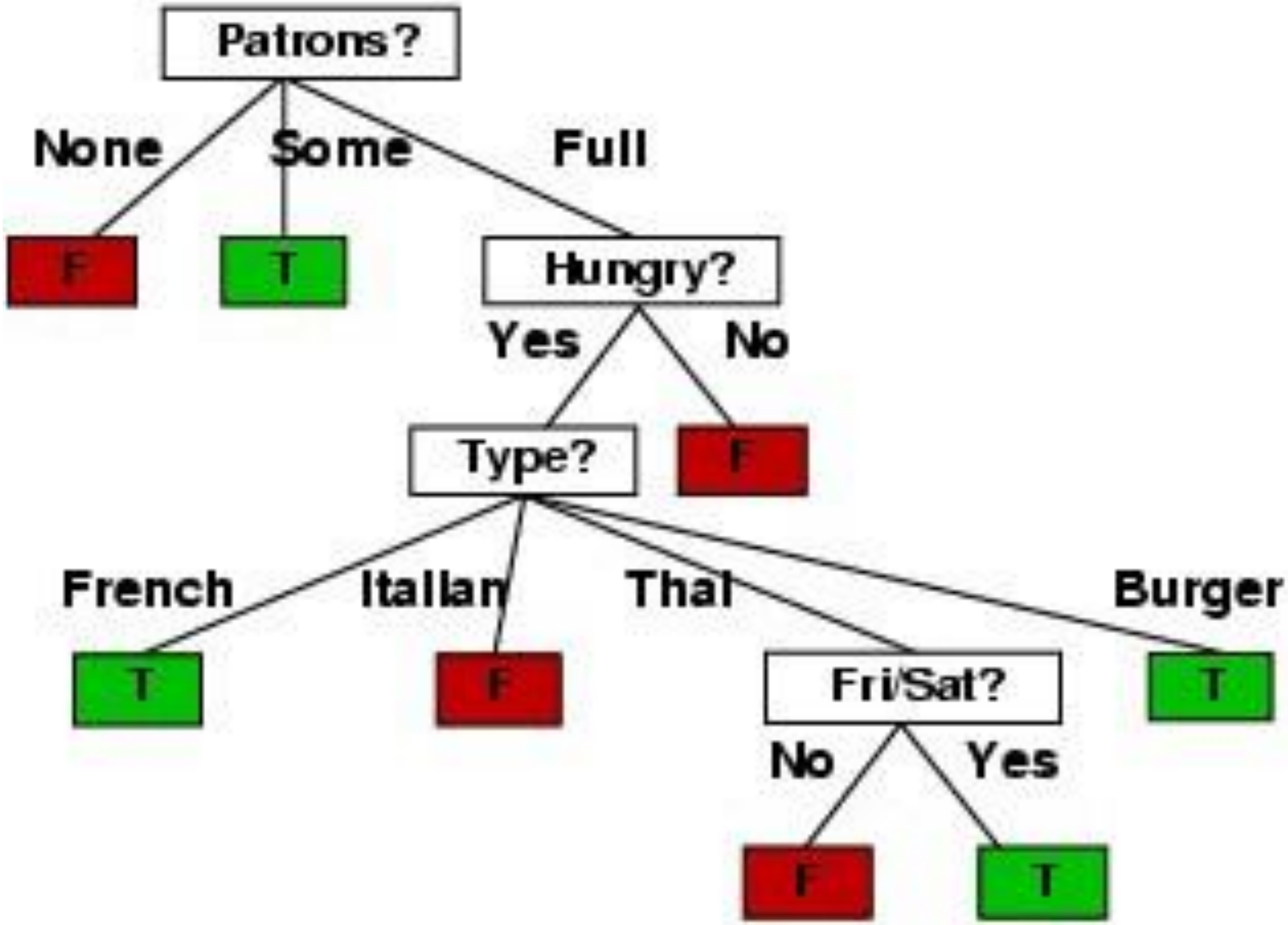
Choosing Patrons Yields more Information



(a) (b)

The ID3 algorithm used this to decide what attribute to ask bout next when building a decision tree

Decision tree by ID3 algorithm

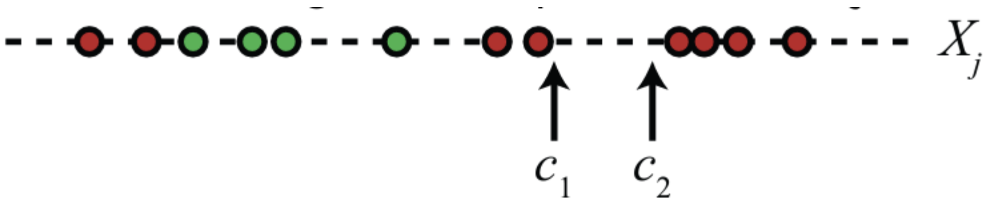


Optimal splits for continuous attributes

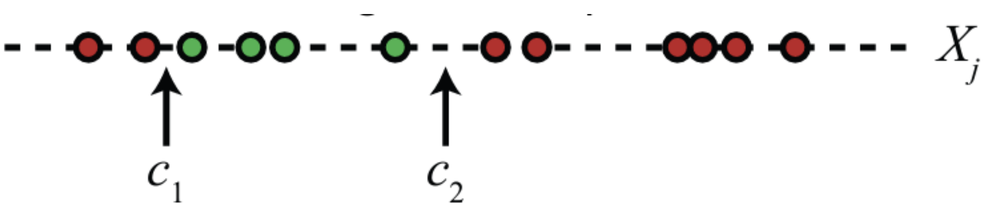
- Infinitely many possible split points c to define node test

$$X_j > c \ ?$$

- No! Moving split point along the empty space between two observed values has no effect on purity; so just use midpoint



- Moreover, only splits between examples from different classes can be optimal for purity



Regression Tree

- Purity criterion: variance E of values at a node
- We consider a node to be pure if $E \leq \theta$ for a threshold $\theta > 0$. In that case, we do not split it.
- If a node m is not pure, we split it.
- Rather than assigning a constant output value to a leaf, we can assign it a regression function.

Overfitting, Early Stopping, and Pruning

- Growing the tree until each leaf is pure will produce a large tree that overfits.
- *Early stopping*: we stop splitting if the impurity is below a user threshold $\theta > 0$.
- *Pruning*: we grow the tree in full until all leaves are pure and the training error is zero. Then, we find subtrees that cause overfitting and prune them

Summary

- Efficient learning algorithm
- Handle both discrete and continuous inputs and outputs
- Robust against any monotonic input transformation, also against outliers
- Automatically ignore irrelevant features: no need for feature selection
- Decision trees are usually interpretable

What is next?

- ❑Ensemble models that combines multiple learners
- ❑Bagging
- ❑Boosting