Grundlagen der künstlichen Intelligenz – Learning

Matthias Althoff

TU München

January 23, 2020

1/17
1/23

Organization

1. Types of Learning
2. Supervised Learning
3. Learning Decision Trees

The content is covered in the AI book by the section "Learning from Examples".

Learning Outcomes

- You can explain the difference between *unsupervised learning*, *reinforcement learning*, and *supervised learning*.
- You understand that *hypothesis selection* is crucial for *supervised learning*.
- You know the definition of a *decision tree*.
- Given a *decision tree*, you can explain how a decision based on that tree is made.
- You can explain why a *decision tree* can express any function of input attributes.
- You can apply the proposed *decision tree learning algorithm*.
- You can quantify the importance of attributes using *information-theoretic entropy*.

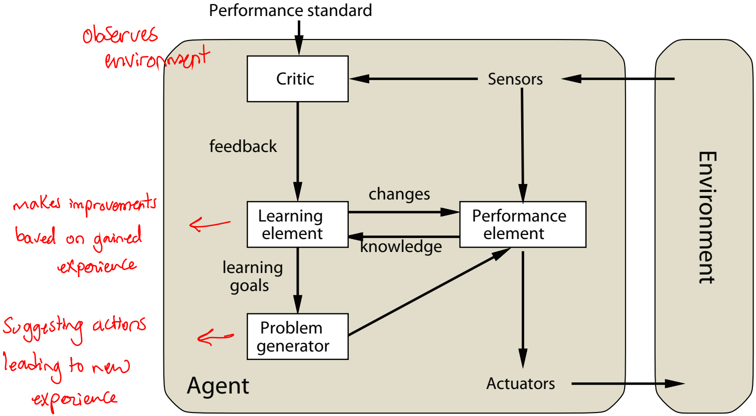
Introduction

An agent is **learning** if it improves its performance on future tasks after making observations about the world.

Motivation for learning

- Designers cannot anticipate all possible situations that an agent might face.
- Designers cannot anticipate all changes over time (e.g., stock market prediction).
- Human programmers might not have an idea how to program a solution themselves (e.g., face recognition).

Reminder: Learning Agent



- Any previous agent can be extended to a learning agent.
- The block **Performance element** is a placeholder for the whole of any of the previous agents.

Types of Learning

Unsupervised learning

- The agent learns patterns in the input even though no feedback is supplied.
- Most common technique is clustering, e.g., automated car might develop a concept for good and bad traffic without providing labeled examples.

Reinforcement learning

- The agent learns from a series of reinforcements – rewards or punishments.
- For instance, winning a chess game tells the agent it did something right.

Supervised learning (focus of this lecture)

- The agent observes some example input-output pairs and learns a function that maps from input to output.
- For instance, inputs for automated driving are sensor values, and outputs are instructions from a human (e.g., "brake").

Supervised Learning

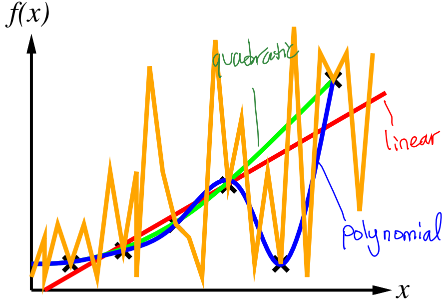
Task of supervised learning

Given a **training set** of N example input-output pairs

$$(x_1, y_1), (x_2, y_2), \dots (x_N, y_N),$$

where each y_i was generated by an unknown function $y = f(x)$, discover a function h that approximates f as well as possible.

E.g., curve fitting:



Ockhams razor: choose the simplest consistent hypothesis

Hypothesis Selection

Finding a good hypothesis $h(x) \in \mathcal{H}$ out of the set of possible hypothesis functions $\mathcal{H}$ is crucial. We wish to maximize the probability that a hypothesis belongs to a data set:

$$h^* = \arg\max_{h \in \mathcal{H}} P(h|data)$$

using Bayes' rule we obtain

$$h^* = \arg\max_{h \in \mathcal{H}} P(data|h)P(h).$$

We can say that simple hypotheses have high probability, e.g., the previous data points belong to a degree-1 or -2 polynomial and thus prevent the spiky solution.

Generalization

We use a **test set** to check whether the hypothesis $h(x)$ **generalizes** well and predicts other values outside the training set (test set $\neq$ training set).

Decision Trees

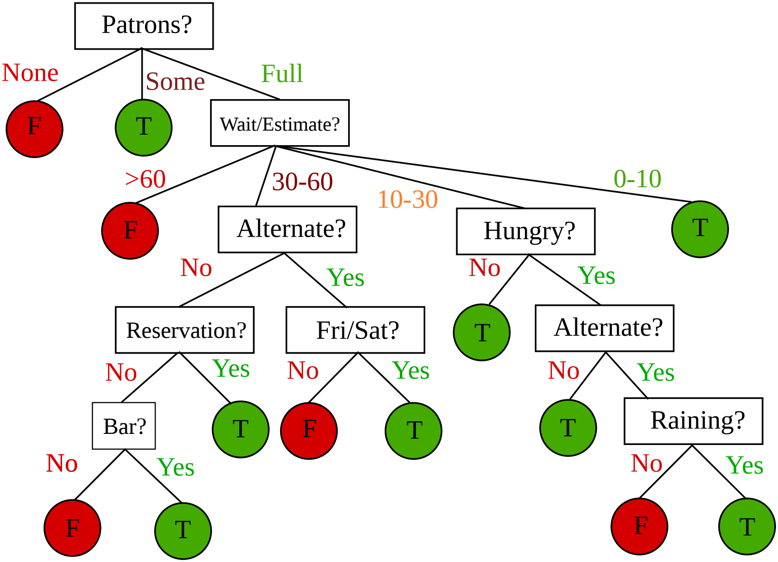
- A **decision tree** represents a function that takes a vector of attribute values as input and returns a "decision" – a single Boolean output value.
- We restrict ourselves to discrete inputs.
- A decision tree reaches its decision through a sequence of tests:
  - An internal node represents a test of a property;
  - Edges are annotated with the possible test values;
  - Each leaf node has the Boolean value which should be returned.
- Decision trees are natural for humans, e.g., "How To" manuals (e.g., for car repair) are often written in a decision tree structure.

Example: Restaurant

Goal predicate: *WillWait*

Patrons	:	How many guests? (none, some, full)
WaitEstimate	:	How long is the waiting time? ($0-10$, $10-30$, $30-60$, >60)
Alternate	:	Are there alternatives? (True/False)
Hungry	:	Am I hungry? (T/F)
Reservation	:	Do I have a reservation? (T/F)
Bar	:	Is there a bar where I can wait for my seat? (T/F)
Fri/Sat	:	Is it Friday or Saturday? (T/F)
Raining	:	Is it raining outside? (T/F)
Price	:	How high are the prices? ($, $$, $$$)
Type	:	Type of restaurant (French, Italian, Thai, burger)

Decision Tree for the Restaurant



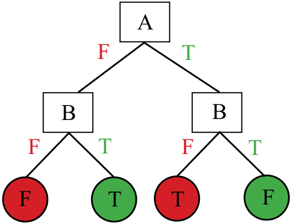
Expressiveness

Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row → path to leaf:

2^n rows

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F

2^n bit number



- There is a consistent decision tree for any training set with one path to a leaf for each example (unless f is nondeterministic in x), but it probably will not generalize to new examples.
- 2^n rows for n Boolean attributes results in 2^n bit number; n bits represent 2^n different numbers → 2^{2^n} possible decision trees.
- We prefer to find more *compact* decision trees.

Training Set for the Decision Tree

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
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

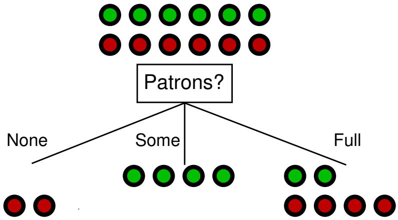
Inducing Decision Trees from Examples

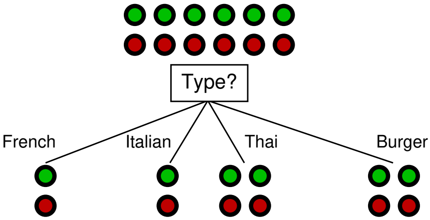
* It's always difficult to find optimal DT; one has to check all possible Solns. One valid soln is good heuristic.

- It is intractable to find the smallest consistent tree from the training set (2^{2^n} possible decision trees).
- We need efficient heuristics!
- The `Decision-Tree-Learning` algorithm adopts a greedy divide-and-conquer strategy:
  1. Test the most important attribute first.
  2. The test divides the problem into two smaller subproblems that can be solved recursively.

Tweedback Question

What attribute would you choose?



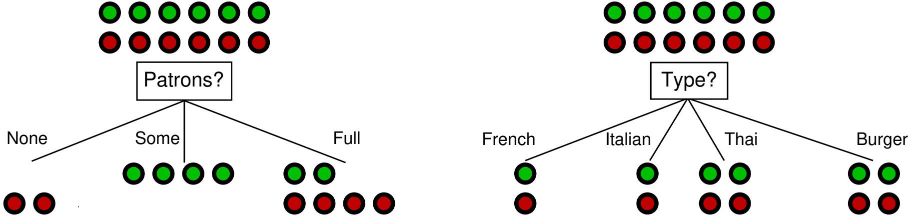


A Patrons

B Type

Choosing an Attribute

Idea: A good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



Patrons? is a better choice – gives *information* about the classification

Four Cases in Attribute Selection

1. If the remaining examples are all positive (or all negative), we can answer with *Yes* or *No* and are done.
2. If there are some positive and some negative examples, then choose the best attribute to split them.
3. If there are no examples left (no example for this combination of attribute values), we return a default value calculated from the plurality classification of all the examples that were used in constructing the node's parent (i.e., selects the most common output value among examples).
4. If there are no attributes left, but both positive and negative examples, there is a conflict. Reasons:
   - noise in the data;
   - domain is nondeterministic;
   - we cannot observe an attribute that would distinguish the examples.

   The best we can do is return the plurality classification of the remaining examples.

majority choice for parent node

Decision Tree Learning Algorithm

function DecisionTreeLearning(*examples*, *attributes*, *parent_examples*) **returns** tree

if *examples* is empty **then return** `Plurality-Value`(*parent_examples*)
else if all *examples* have the same classification **then return** the classification
else if *attributes* is empty **then return** `Plurality-Value`(*examples*)
else
    $A \leftarrow \arg\max_{a \in attributes}$ `Importance`(a, *examples*)
    tree $\leftarrow$ a new decision tree with root test A
    for each value v_k of A **do**
        $exs \leftarrow \{e | e \in examples$ **and** $e.A = v_k\}$ current
        subtree $\leftarrow$ `DecisionTreeLearning`(exs, $attributes \setminus A$, $examples$) parent
        add a branch to *tree* with label $(A = v_k)$ and subtree *subtree*
    return *tree*

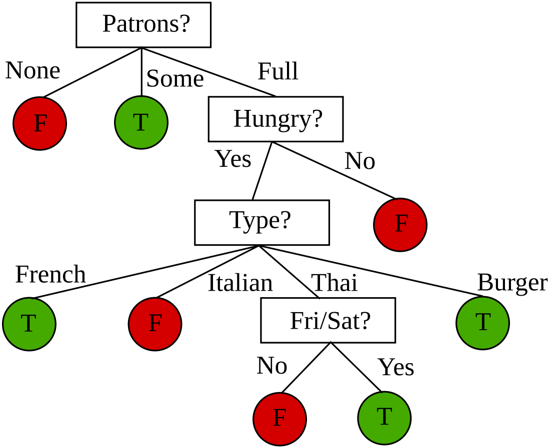
A: input attribute;
v_k: possible value of input attribute A;
`Plurality-Value`: selects the most common output value among examples.

Result from the Training Set

Decision tree learned from the 12 examples:

More Compact

less attributes needed



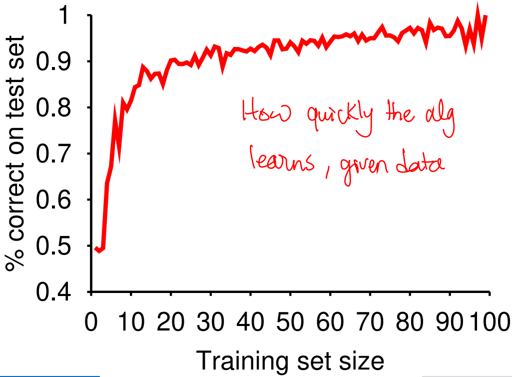
Substantially simpler than "true" tree – a more complex hypothesis is not justified by small amount of data.

Learning Curve

- We can evaluate the accuracy of a learning algorithm with a **learning curve**.
- We have 100 examples, which we split into a training set and a test set.
- We start with a training set of size 1 and increase it up to size 99.
- For each training/test ratio, 20 random splits are used and averaged.

1:99
2:98
⋮
99:1

The larger the training set & the lower the testing set, the better results you obtain



How quickly the alg learns, given data

Quantifying Importance of Attributes (1)

So far we have not provided a clear approach for `Importance`(a, *examples*) on slide 18. We will use entropy (not in the thermodynamic sense):

Information-theoretic entropy

Entropy is a measure of the uncertainty of a random variable; acquisition of information corresponds to a reduction in entropy.

info $\propto$ 1/entropy

The entropy of a random variable V with values v_k is defined as

$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k).$$

Examples:

- Fair coin: $H(Fair) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1.$
- Loaded coin (99% heads):
  $H(Loaded) = -(0.99 \log_2 0.99 + 0.01 \log_2 0.01) \approx 0.08.$

The unit of entropy is *bit*: $H(Fair) = 1$ bit, $H(Loaded) \approx 0.08$ bit.

Quantifying Importance of Attributes (2)

We introduce $B(q)$ as the entropy of a Boolean random variable that is true with probability q:

$$B(q) = -(q \log_2 q + (1-q)\log_2(1-q)).$$

If a training set contains p positive examples and n negative examples, the entropy of the goal attribute on the whole set is

$$H(goal) = B\left(\frac{p}{p+n}\right).$$

The restaurant training set on slide 13 has $p = n = 6$ so that $H(Goal) = B(0.5) = 1$.

Quantifying Importance of Attributes (3)

- An attribute A with d values divides the training set E into subsets $E_1, \dots, E_d$.
- Each subset E_k has p_k positive and n_k negative examples, resulting in $B(p_k/(p_k + n_k))$ bits of information to answer the question.
- A randomly chosen example from the training set has the kth value with probability $(p_k + n_k)/(p + n)$, so the expected entropy remaining is

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right).$$

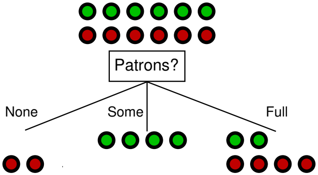
- The **information gain** from the attribute test on A is the expected reduction in entropy:

$$Gain(A) = B\left(\frac{p}{p + n}\right) - Remainder(A).$$

Quantifying Importance of Attributes (4)

$Gain(A)$ on the previous slide is exactly what is implemented in `Importance`(a, $examples$) on slide 18.

Examples:



- $Gain(Patrons) = 1 - [\frac{2}{12}B(\frac{0}{2}) + \frac{4}{12}B(\frac{4}{4}) + \frac{6}{12}B(\frac{2}{6})] \approx 0.541.$
- $Gain(Type) = 1 - [\frac{2}{12}B(\frac{1}{2}) + \frac{2}{12}B(\frac{1}{2}) + \frac{4}{12}B(\frac{2}{4}) + \frac{4}{12}B(\frac{2}{4})] = 0.$

This confirms that *Patrons* is a better attribute to split on. [highest gain]

Beyond Decision Tree Learning

So far, we have only focused on decision trees. Machine Learning has much more to offer, e.g.,

- classification with linear models;
- artificial neural networks;
- clustering techniques;
- support vector machines;
- ensemble learning, etc.

These topics are covered in machine learning lectures at our department.

Summary

- Learning is needed for unknown environments and lazy designers.
- Types of learning: Unsupervised learning, reinforcement learning, supervised learning.
- In supervised learning, one tries to learn a function $y = h(x)$ from input-output pairs.
- It is crucial to find a hypothesis that agrees well with the examples. **Ockhams razor** maximizes a combination of consistency and simplicity.
- **Decision trees** can represent all Boolean functions. The information-gain heuristic efficiently finds simple decision trees.
- The performance of a learning algorithm is measured by the **learning curve**, which shows the prediction accuracy on the **test set** as a function of the **training-set** size.