10. Learning from Examples: Machine Learning

By: Alex S.

May, 2024

1. Machine Learning

- The agent cannot be autonomous if it cannot learn
- Machine learning is the induction problem. From examples → general ideas.

1.1. Machine Learning Approaches

- *Symbolic* - formally described knowledge
- *Example based* - does not fully represent the knowledge. Examples are classified, but they are not described
- *Evolution* - using many generations, optimizing the approach

1.2. Learning approaches

- New knowledge addition to the previously available knowledge
  - Knowledge analysis
  - Error resolution
- Useful regularity findings in data
  - Data mining
  - Data Science
  - kNN, decision tree, perceptrons
  - Evolution
- Supervised/Unsupervised learning

1.3. Agent structure

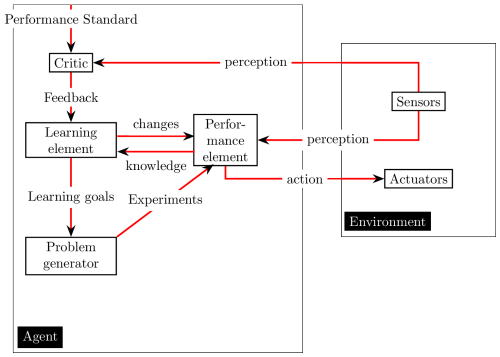


Figure 1: The learning agent general structure

In Figure 1 the problem generator is the generator of noise. The main 2 blocks are *learning* and *performance* elements.

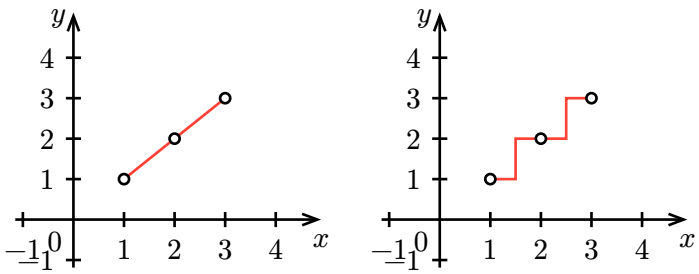
1.4. Learning element design aspects

1. Which performance element components should be modified?
   - Depends on the performance element
2. What knowledge representation type is used?
   - Depends on the project:
     - Frames, networks, logic, ...
3. What feedback is available?
   - Depends on the learning type supervised/ unsupervised
4. What primary knowledge is available?

2. Training

During training the main task of the agent is to find an approximation function by using examples: $(x, f(x))$

However, how do we know which approximation to use to get the best hypothesis?



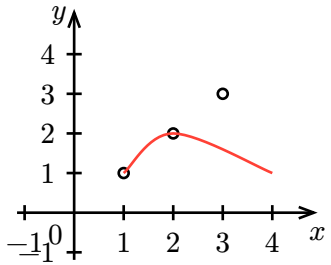


Figure 2: Several graphs that show that the function can be approximated in different ways. Example a) represents a linear relation. Example b) shows discrete relations. The c) example shows inconsistent behavior which cannot be used for hypothesis, because not all examples points are taken into account.

- We say hypothesis **generalizes** well if it correctly predicts the value of y for novel examples.
- To be compliant with hypothesis, every example must be included into it

2.1. How to choose the best hypothesis?

- *Ockham's razor* principle (1334) - there is no need to do bigger effort, if it is possible to use less effort
- Therefore we need to choose the simplest hypothesis
- There is a small possibility that the simplest hypothesis will be inconsistent or incorrect.
- The simplest hypothesis is easier to compute.
- And it better generalizes.

3. Training with decision trees

- Classified as symbolic training
- "White-box" method
- Algorithms to use:
  - ID3
    - More about the algorithm described in Russell & Norvig book in Chapter 18, page 702
  - C4.5, etc.
- Input is the set of attributes(X) and the output is the decision(y): $\{x_1, x_2, ..., x_n\} \in X \to y$
- **Classification** task trains a discrete function
- **Regression** task trains a continuous function

Decision tree is a graph where:
- Inner nodes proceeds with conditional checking
- No-way nodes defines a decision value

Decisions trees allow:
- describe one specific situation/object
- describe any boolean function

> **Note**
>
> there are 2^{2^n} boolean functions where n is the attribute amount

- hard to describe parity or majority functions

3.1. Example

Let's consider an example where we need to find a decision on waiting or not waiting a table in the restaurant. In Figure 3 the data used for the example is presented.

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Figure 3: Data for decision trees. Taken from Russell & Norvig book.

Specific examples can be described with predicate or first-order logic:

1. $$\text{Restaurant} = \text{Full} \land \text{Waiting time} = 0 \text{ - } 10 \land \neg \text{ Hungry} \to \text{Wait}$$
2. $$\forall r \ (\text{Regular guests}(r, \text{Full}) \land \text{Waiting time}(r, 0 \text{ - } 10) \land \neg \text{ Hungry}(r) \to \text{Wait}(r))$$
3. $$\forall r \ (\text{Wait}(r) = P_1(r) \lor P_2(r) \lor ... \lor P_n(r))$$

   where $P_i(r)$ is one the branches of the tree

3.2. Decision tree construction

- We can create a tree, just by going through the examples one by one
  - However, this tree will be big, 12 levels deep
  - It will be overfitted, i.e. it will be able to predict only test data
  - No way it could generalize good enough
  - It breaks the Ockham's razor principle of simplicity
- Therefore, we use different approach:

Let's take a *Bar* to analyze its examples:

Bar			
No		**Yes**	
Negative	**Positive**	**Negative**	**Positive**
x_2	x_1	x_7	x_3
x_5	x_4	x_9	x_6
x_{11}	x_8	x_{10}	x_{12}

It does not fit, since there is no obvious examples for positive or negative results.

Let's take *Hungry*:

Hungry			
No		**Yes**	
Negative	**Positive**	**Negative**	**Positive**
x_5	x_3	x_2	x_1
x_7		x_{10}	x_4
x_{11}			x_6
x_9			x_8

It is closer, but still no explicit positives/negatives.

Let's take *Regular guests*:

Regular guests					
No one		**Some**		**Full**	
Neg	**Pos**	**Neg**	**Pos**	**Neg**	**Pos**
x_7			x_1	x_2	x_4
x_{11}			x_3	x_5	x_{12}
			x_6	x_9	
			x_8	x_{10}	

Here we can certainly see a nice tendency that *No one* and *Some* has explicitly only negative and positive examples. This is the best attribute, therefore it will go to the root of the tree.

The next step is to take all the data left with *Full* attribute, and do the same actions with the data left-overs.

The final tree in Figure 4

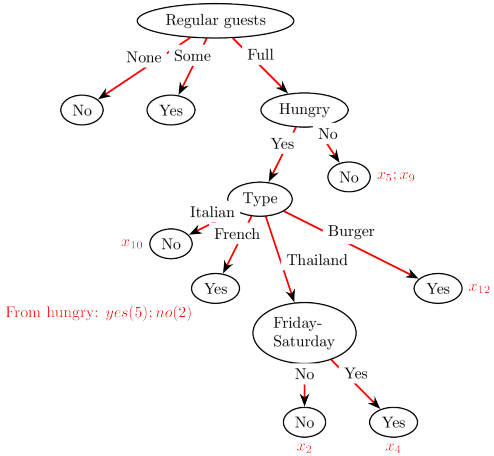


Figure 4: The final decision tree

> **Note**
>
> The value of *French* edge was not determined. It was determined by the majority of values from the parent node *Hungry*.

When doing a construction of the tree, it is needed to keep in mind following properties.

Properties:

1. If there are some positive or some negative examples, we choose the best attribute
2. If there are all positive or all negative examples, the node is fully decided and solved
3. If there are no values at the node, we return a default value. E.g. getting the majority from the parent values. As it was in the example of *French*, where majority of the parent was *Yes*.
4. If there are no attributes, but there is data, then there is no way, but to search for a new attribute

3.3. Choosing attribute tests

- To choose the attribute Shannon and Weaver method (1949) for information amount calculation is used.
- The less is known about the result, the more information is received.
- It is described with the **information** or **entropy** formula:

$$I(p(v_1), ..., p(v_n)) = \sum_{i=1}^{n} -p(v_i) \cdot \log_2 p(v_i)$$

where I is the information amount in bits and $p(v_i)$ is the probability of the random variable $v_i \in V$ to occur.

For example, the fair coin where it has a 50% probability of both results is calculated:

$$I\left(\frac{1}{2}, \frac{1}{2}\right) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1 \text{ bit}$$

- We have 1 bit of information where we certainly know “heads” or “tails”
- This is called as a “perfect” attribute that equally divides the information
- Every attribute divide a dataset into subsets. E.g. attribute *Hungry* from Figure 4 divides the set into *Yes* and *No*

When working with subsets, we have the following formula:

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

where p and n represent the total number of positive and negative examples.

We can also just use the information or entropy formula: $I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$ where i is the subset.

3.4. Information gain

We can calculate the information gain for each attribute in the following way:

1. Calculate the remainder entropy for the current subset:

$$R(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} \cdot I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

where R is a remainder, A is the attribute, v is the number of subsets in the attribute.

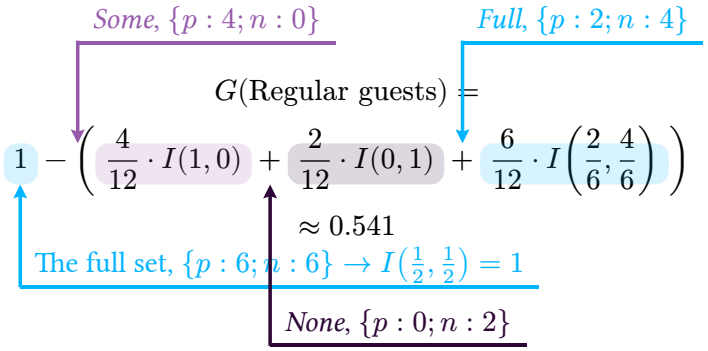
2. Calculate the gain for the attribute with:

$$G(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - R(A)$$

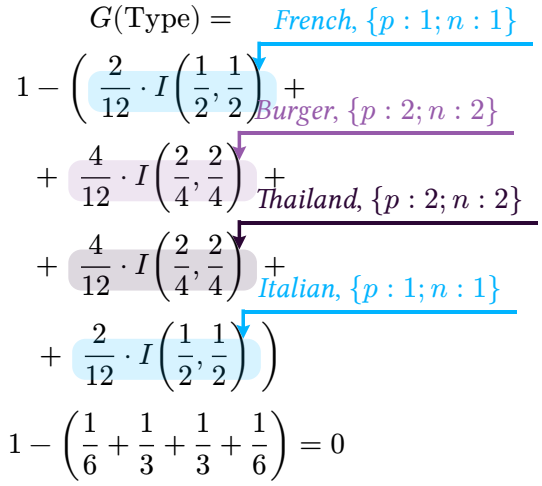
where G is the information gain for the attribute

3.4.1. Example

Let's calculate the information gain for *Regular guests*.



Let's calculate the information gain for attribute *Type*:



It looks obvious that $G(\text{Regular guests}) > G(\text{Type})$ therefore we use the attribute with the biggest information gain as the root node.

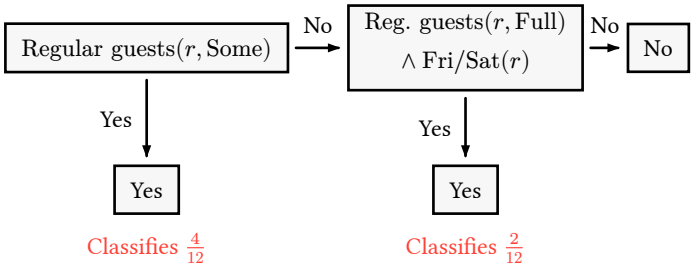
4. Evaluation of learning algorithm

1. Need to collect a big dataset with examples
2. Need to separate the dataset into 2 sets: **training** and **testing**
3. Need to use a **training** dataset to generate a hypothesis
4. Measure the amount of examples that are correctly classified
5. Amend the **training** dataset with more examples possible

5. Decision list learning

- Logical statement in the limited form that consists of the series of tests described with words and conjunctions. If the test is successful then the following tests are decided.

Example:

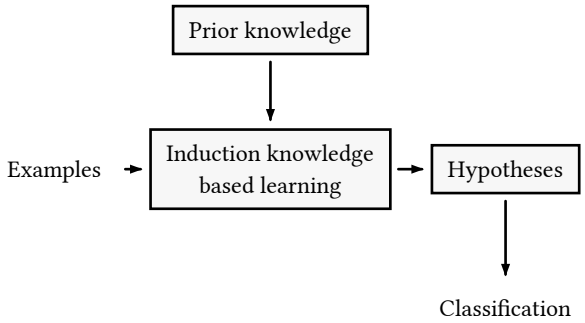


The diagram above can be described as:

$$\text{Wait} \equiv (\text{Restaurant} = \text{Some}) \vee (\text{Restaurant} = \text{Full} \wedge \text{Fri/Sat})$$

6. Usage of knowledge in learning

- Most of the methods researched previously do not use any prior domain knowledge
- The cumulative learning process can be displayed the following way:



- Prior knowledge is described with the first-order logic

6.1. Example

There is a logical description:

$$\text{Alternative}(X_1) \wedge \neg\ \text{Bar}(X_1) \wedge \wedge \neg\ \text{Fri/Sat}(X_1) \wedge \text{Hungry}(X_1)$$

The classification is:

$$\text{Wait}(X_1) \equiv Q(X_1)$$

where $Q(X_1)$ is the unary goal predicate

- The goal is to find the logical statement that will satisfy the $Q(X)$ for all examples.
- Every hypothesis describes a goal predicate candidate C_i
- Therefore every hypothesis H_i is a sentence described as:

$$\forall x\ Q(x) \equiv C_i(x)$$

- For the restaurant problem, the inducted decision tree suggests the following candidate(hypothesis):

$$\begin{aligned}\forall r \ \ &\text{Wait}(r) \equiv \text{Regular guests}(r, \text{Some}) \vee \\ &\vee (\text{Regular guests}(r, \text{Full}) \wedge \text{Hungry}(r) \wedge \\ &\qquad \wedge \text{Type}(r, \text{French})) \vee \\ &\vee (\text{Regular guests}(r, \text{Full}) \wedge \text{Hungry}(r) \wedge \\ &\qquad \wedge \text{Type}(r, \text{Burger})) \vee \\ &\vee (\text{Regular guests}(r, \text{Full}) \wedge \text{Hungry}(r) \wedge \\ &\qquad \wedge \text{Type}(r, \text{Thailand}) \wedge \text{Fri/Sat}(r))\end{aligned}$$

- If hypothesis is **consistent** with the whole dataset, then it is **consistent** with every example.
- The table of hypothesis consistence with examples:

		Actual	
		True	*False*
Predicted	*True*	True positive(TP)	False positive(FP)
	False	False negative(FN)	True negative(TN)

7. Searching the current-best-hypothesis

- For pseudocode use Russell & Norvig book on page 770
- The idea of the method is to keep one hypothesis and adjust it to keep the consistence
- The algorithm is greedy
- The process can be visualized as in Figure 5

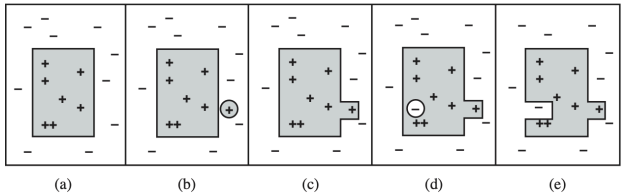


Figure 5: a) A consistent hypothesis. (b) A false negative. (c) The hypothesis is generalized. (d) A false positive. (e) The hypothesis is specialized.

- Generalization and specialization are defined as the operations which change the hypothesis extension
- If hypothesis H_1 with definition C_1 is the hypothesis' H_2 with definition C_2 generalization, then

$$\forall x \ C_2(x) \Rightarrow C_1(x)$$

I.e. C_1 is the generalization over C_2

E.g.

$$\forall r \ C_2(r) \equiv \text{Alternative}(r) \wedge \text{Regular guests}(r, \text{Some})$$
$$\forall r \ C_1(r) \equiv \text{Regular guests}(r, \text{Some})$$
$$C_2 \Rightarrow C_1$$

7.1. Example

- $H_1 : \forall r \ \ \text{Wait}(r) \equiv \text{Alternative}(r)$
  - 2nd example is false positive from Figure 3
  - Therefore we use specialization
- $H_2 : \forall r \ \ \text{Wait}(r) \equiv \text{Alternative}(r) \wedge \text{Regular guests}(r, \text{Some})$
  - 3rd example from Figure 3 is false negative
  - Therefore we generalize the hypothesis
- $H_3 : \forall r \ \ \text{Wait}(r) \equiv \text{Regular guests}(r, \text{Some})$
  - 4th example from Figure 3 is false negative
  - Therefore we generalize the hypothesis
  - The generalization removes the attribute, therefore we replace it with something else or use disjunction($\vee$)
- $H_4 : \forall r \ \ \text{Wait}(r) \equiv \text{Regular guests}(r, \text{Some}) \vee (\text{Regular guests}(r, \text{Full}) \wedge \text{Fri/Sat}(r))$
  - H_4 is consistent with all examples
  - It could also be different, e.g.
    - $H_4' : \forall r \ \ \text{Wait}(r) \equiv \neg \ \text{Waiting time}(r, 30 - 60)$
    - etc.