Sapienza University of Rome

Master in Artificial Intelligence and Robotics

Machine Learning

A.Y. 2025/2026

Prof. Luca Iocchi

1. Introduction

Luca Iocchi

Overview

- What is a Machine Learning problem
- Example: learning to play checkers
- Machine Learning problem formulations
- Learning as search in hypothesis space
- Concept learning
- Machine Learning issues

References
T. Mitchell. Machine Learning. Chapters 1, 2

Machine Learning

Machine learning is programming computers to improve a performance criterion using example data or past experience.

Machine learning (or data mining) is the task of producing knowledge from data.

Machine Learning useful when

- Human expertise does not exist
- Humans are unable to explain their expertise
- Solution needs to be adapted to particular cases

Machine Learning

Machine Learning exploits

- Recent progress in algorithms and theory
- Growing flood of on-line data (Big Data)
- Increasing computational power (GPU)

Machine Learning Applications

Almost in any domain

- Computer vision
- Speech interaction
- Document/text analysis
- Data analytics
- Robot control
- Games

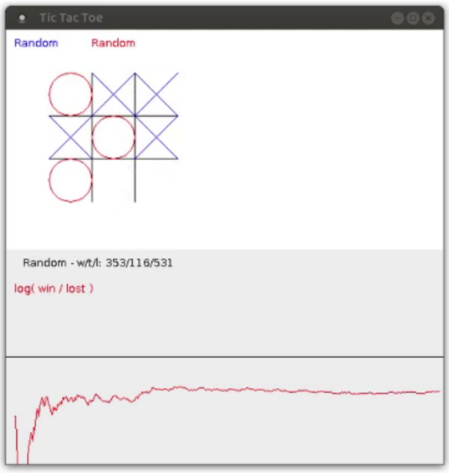
What is a Learning Problem?

Learning = Improving with experience at some task

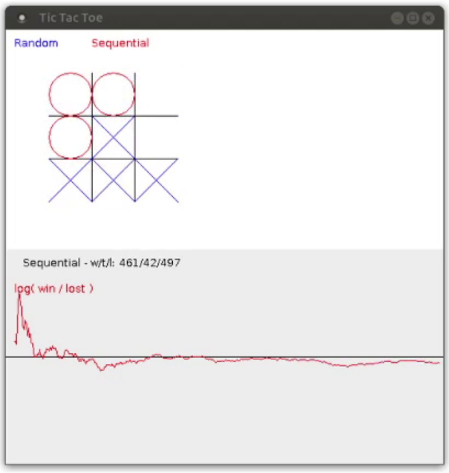
- Improve over task T,
- with respect to performance measure P,
- based on experience E.

Improving performance over time

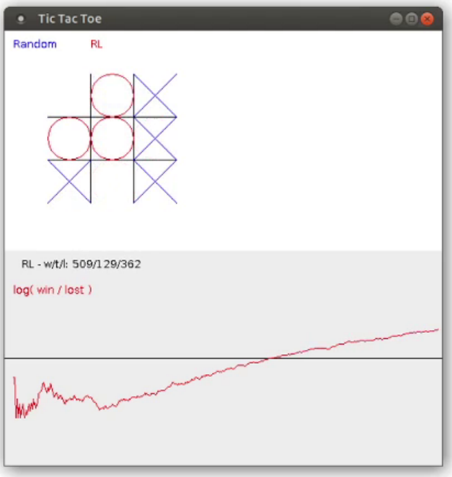
Example: Tic Tac Toe (`https://youtu.be/Bpocn3EHYOo`)



Random - w/t/l: 353/116/531



Sequential - w/t/l: 461/42/497



RL - w/t/l: 509/129/362

Example

Learn to play checkers

- T: Play checkers
- P: % of games won in a tournament
- E: opportunity to play against self

Learning to Play Checkers

- What experience?
- What exactly should be learned?
- How shall it be represented?
- Which algorithm to learn it?

Type of Training Experience

Several options:

- Human expert suggests optimal move for each configuration of the board
- Human expert evaluates each configuration of the board
- Computer plays against a human and automatically detects win/draw/loss configurations
- Computer plays against itself

Problem: is training experience representative of performance goal?

Choose the Target Function

- $ChooseMove : Board \rightarrow Move$
- $V : Board \rightarrow \Re$
- ...

Possible Definition for Target Function V

- if b is a final board state that is won, then $V(b) = 100$
- if b is a final board state that is lost, then $V(b) = -100$
- if b is a final board state that is drawn, then $V(b) = 0$
- if b is a not a final state in the game, then $V(b) = V(b')$, where b' is the best final board state that can be achieved starting from b and playing optimally until the end of the game.

Is it a correct function to learn the task?

Can we compute this function?

Possible Definition for Target Function V

- if b is a final board state that is won, then $V(b) = 100$
- if b is a final board state that is lost, then $V(b) = -100$
- if b is a final board state that is drawn, then $V(b) = 0$
- if b is a not a final state in the game, then $V(b) = V(b')$, where b' is the best final board state that can be achieved starting from b and playing optimally until the end of the game.

Is it a correct function to learn the task? **YES**

Can we compute this function? **NO**

Choose Representation for Target Function

- collection of rules
- neural network
- polynomial function of board features
- ...

A Representation for Function to Learn

$$\hat{V}(b) = w_0 + w_1 \cdot bp(b) + w_2 \cdot rp(b) + w_3 \cdot bk(b) + w_4 \cdot rk(b) + w_5 \cdot bt(b) + w_6 \cdot rt(b)$$

- $bp(b)$: number of black pieces on board b
- $rp(b)$: number of red pieces on b
- $bk(b)$: number of black kings on b
- $rk(b)$: number of red kings on b
- $bt(b)$: number of red pieces threatened by black (i.e., which can be taken on black's next turn)
- $rt(b)$: number of black pieces threatened by red

Is it a correct function to learn the task?

Can we compute this function?

A Representation for Function to Learn

$$\hat{V}(b) = w_0 + w_1 \cdot bp(b) + w_2 \cdot rp(b) + w_3 \cdot bk(b) + w_4 \cdot rk(b) + w_5 \cdot bt(b) + w_6 \cdot rt(b)$$

- $bp(b)$: number of black pieces on board b
- $rp(b)$: number of red pieces on b
- $bk(b)$: number of black kings on b
- $rk(b)$: number of red kings on b
- $bt(b)$: number of red pieces threatened by black (i.e., which can be taken on black's next turn)
- $rt(b)$: number of black pieces threatened by red

Is it a correct function to learn the task? **MAYBE**

Can we compute this function? **YES**

A Representation for Function to Learn

$$\hat{V}(b) = w_0 + w_1 \cdot bp(b) + w_2 \cdot rp(b) + w_3 \cdot bk(b) + w_4 \cdot rk(b) + w_5 \cdot bt(b) + w_6 \cdot rt(b)$$

- $bp(b)$: number of black pieces on board b
- $rp(b)$: number of red pieces on b
- $bk(b)$: number of black kings on b
- $rk(b)$: number of red kings on b
- $bt(b)$: number of red pieces threatened by black (i.e., which can be taken on black's next turn)
- $rt(b)$: number of black pieces threatened by red

Note: Features $f_i(b)$ known, Coefficients w_i unknown

Learning $\hat{V} \equiv$ estimating w_i

Obtaining Training Examples

Notation:

- $V(b)$: the true target function (always unknown)
- $\hat{V}(b)$: the learned function (approximation of $V(b)$ computed by the learning algorithm)
- $V_{train}(b)$: the training value obtained at $b \in$ training data

Dataset $D = \{(b_i, V_{train}(b_i))_{i=1}^n\}$

Estimating training values:

- $V_{train}(b_i) \leftarrow$ human expert
- $V_{train}(b_i) \leftarrow$ set of games
- ...

Learning algorithm

LMS Weight update rule:

Initialize w_i (e.g., small random value)
Do repeatedly:

- Select a training example b at random
  1. Compute $error(b)$:

  $$error(b) = V_{train}(b) - \hat{V}(b)$$

  2. For each board feature $f_i(b)$, update weight w_i:

  $$w_i \leftarrow w_i + c \cdot f_i \cdot error(b)$$

c is some small constant, say 0.1, to moderate the rate of learning

Testing and Evaluation

Learned function (after training)

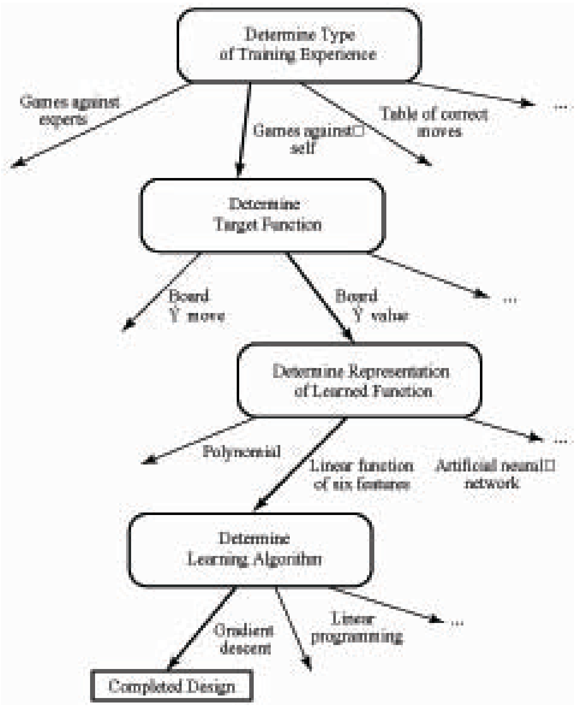
$$\hat{V}(b) = \hat{w}_0 + \hat{w}_1 \cdot bp(b) + \hat{w}_2 \cdot rp(b) + \hat{w}_3 \cdot bk(b) + \hat{w}_4 \cdot rk(b) + \hat{w}_5 \cdot bt(b) + \hat{w}_6 \cdot rt(b)$$

Use this function to play against human players or other computer programs.

How many games won against other players?

If we keep training (i.e., collecting more samples in the dataset), do we increase performance?

Design Choices


Determine Type of Training Experience
Games against experts
Games against□ self
Table of correct moves
...
Determine Target Function
Board Ÿ move
Board Ÿ value
...
Determine Representation of Learned Function
Polynomial
Linear function of six features
Artificial neural□ network
...
Determine Learning Algorithm
Gradient descent
Linear programming
...
Completed Design

Machine Learning Problems

- Supervised Learning
  - Classification
  - Regression
- Unsupervised Learning
- Reinforcement Learning

Machine Learning Problems

General machine learning problem:

Learning a function $f : X \rightarrow Y$, given a training set D containing information about f

Learning a function f means computing an approximated function $\hat{f}$ that returns values as close as possible to f, specially for samples x not present in the training set D.

$$\hat{f}(x) \approx f(x) \ \text{ for } \ x \in X \setminus X_D$$

Note: $X_D = \{x | x \text{ in } D\} \subset X$ with $|X_D| \lll |X|$

Machine Learning Problems

Different types of ML problems depending on
- type of input dataset

Learning a function $f : X \rightarrow Y$, given ...

- $D = \{(x_i, y_i)_{i=1}^n\}$ **(Supervised Learning)**
- $D = \{(x_i)_{i=1}^n\}$ **(Unsupervised Learning)**

Learning a behavior function $\pi : S \rightarrow A$, given ...

- $D = \{(< s_0, a_1, r_1, s_1, \ldots, a_n, r_n, s_n >_i)_{i=1}^n\}$ **(Reinforcement Learning)**

Supervised Learning

Different types of ML problems depending on
- type of function to be learned

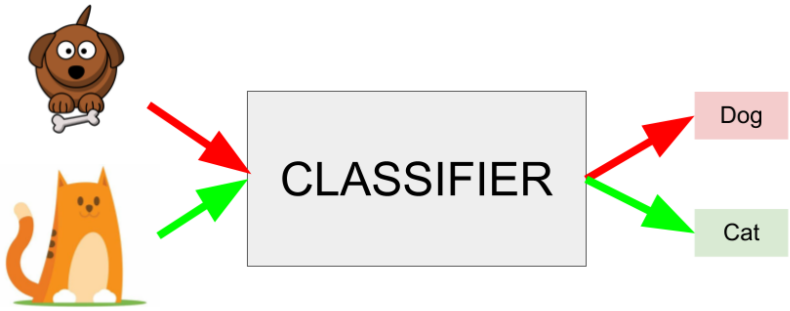
$$X \equiv \begin{cases} A_1 \times \ldots \times A_m,\ A_i \text{ finite sets } \textbf{(Discrete)} \\ \\ \Re^n \ \textbf{(Continuous)} \end{cases}$$

$$Y \equiv \begin{cases} \Re^k \ \ \textbf{(Regression)} \\ \\ \{C_1, \ldots, C_k\},\ \textbf{(Classification)} \end{cases}$$

Special case:
$X \equiv A_1 \times \ldots A_m$ (A_i finite) and $Y \equiv \{0, 1\}$ **(Concept Learning)**

Classification (aka Pattern Recognition)

Return the class to which a specific instance belong.

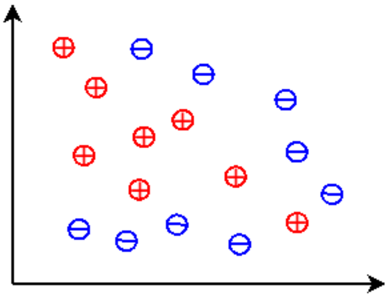


CLASSIFIER

Dog

Cat

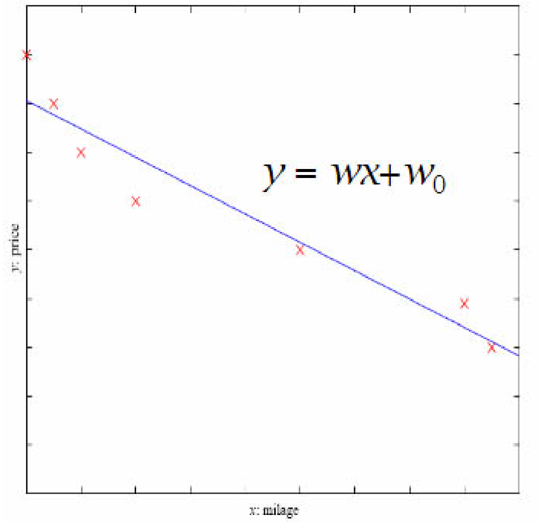
- Face/object/character recognition
- Speech/sound recognition
- Medical diagnosis
- Document classification



Regression

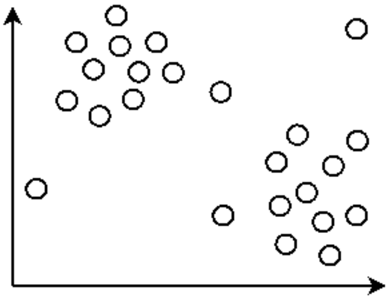
Approximate real-valued functions

Example



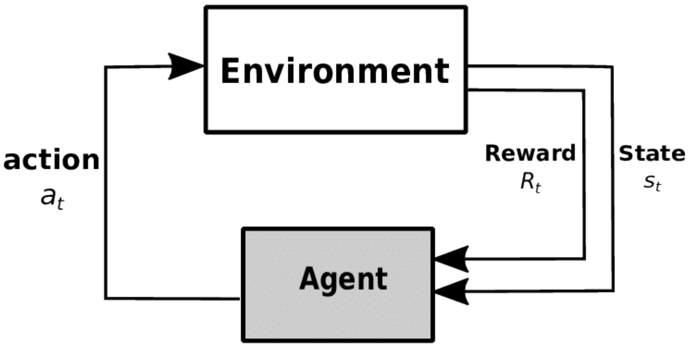
Unsupervised Learning

- Learning what normally happens
- No output available
- Clustering: grouping similar instances
- Example applications
  - Customer segmentation in CRM
  - Image compression: color quantization
  - Bioinformatics: learning motifs



Reinforcement Learning

- Learning a policy (state-action function)
- No supervised output available, only sparse and time-delayed rewards
- Example applications
  - Game playing
  - Robotic tasks
  - ... any dynamic system with unknown or partially known model



Concept Learning (aka Binary Classification)

Data set:

- Input: X (any kind)
- Output (labels): two classes (Yes/No, True/False, 0/1, +/-)

Target function: $c : X \rightarrow \{0, 1\}$

Example: *PlayTennis* - good time to play tennis

- Input: weather features X
- Output: $\{Yes, No\}$

PlayTennis Data Samples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

PlayTennis Prediction

Prediction on new samples (not in the training data)

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
Today	Sunny	Hot	Normal	Strong	**?**

A model trained with training data can predict a value of *PlayTennis* for Today, for example as Yes.

Hypothesis space

Hypothesis h: *representation of the learned function (an approximation of the target function)*

Hypothesis space H: *set of all the functions that can be learnt (all possible approximations of the target function)*

Learning: *search in the hypothesis space* using dataset D

$$h^* = \mathrm{argmax}_{h \in H} \, Performance(h, D)$$

Concept Learning: Notation

c: target function $c : X \to \{0, 1\}$

X: instance space

$x \in X$: one instance

$D = \{(x_i, c(x_i))_{i=1}^n\}$: training set

$c(x)$: value of the target function over x (*true value* known only for instances in D)

H: hypothesis space

$h \in H$: one hypothesis (an approximation of c)

$h(x)$: estimation of h over x (*predicted or estimated value*)

Learning task

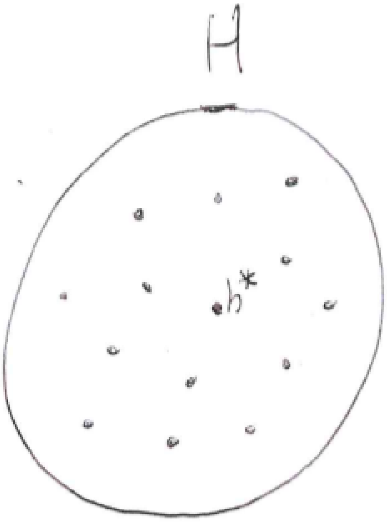
Given a training set $D = \{(x_i, c(x_i))\}$, find the *best* approximation $h^* \in H$ of the target function $c : X \to \{0, 1\}$.

Steps:

1. Define the hypothesis space H (i.e., a representation of the hypotheses)
2. Define a performance metric to determine the *best* approximation
3. Define an appropriate algorithm

Learning as a search problem

Given a representation of the hypothesis space H, search for the *best* hypothesis $h^* \in H$, according to a given performance measure.



Evaluating instances on hypotheses

$h(x)$ can be computed for every $x \in X$

$h(x_i) = c(x_i)$ can be verified only for instances x_i appearing in the data set D ($x_i \in D$), for which we know $c(x_i)$

> *A hypothesis h is* **consistent** *with a set of training examples D of target concept c if and only if* $h(x) = c(x)$ *for each training example* $(x, c(x))$ *in D.*

$$Consistent(h, D) \equiv (\forall x \in D)\ h(x) = c(x)$$

The *real goal* of a machine learning system is to find the *best* hypothesis h that predicts correct values of $h(x')$ for instances $x' \notin D$ with respect to the unknown values $c(x')$.

Performance measure

Given a training set $D = \{(x_i, c(x_i))\}$ and a hypothesis $h \in H$, a performance measure is based on evaluating $c(x_i) = h(x_i)$ for all $x_i \in D$.

Inductive learning hypothesis: Any hypothesis that approximates the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

Prototypical Concept Learning Task

Given:

- Instances X (e.g., possible days, each described by the attributes *Outlook, Temperature, Humidity, Wind*)
- Target function $c : X \rightarrow \{0, 1\}$ (e.g., $c = PlayTennis : X \rightarrow \{No, Yes\}$)
- Hypotheses H
- Training examples $D = \{(x_1, c(x_1)), \ldots, (x_n, c(x_n))\}$, positive and negative examples of the target function

Determine:

- A consistent hypothesis (i.e., $h \in H$ such that $h(x) = c(x), \forall x \in D$).

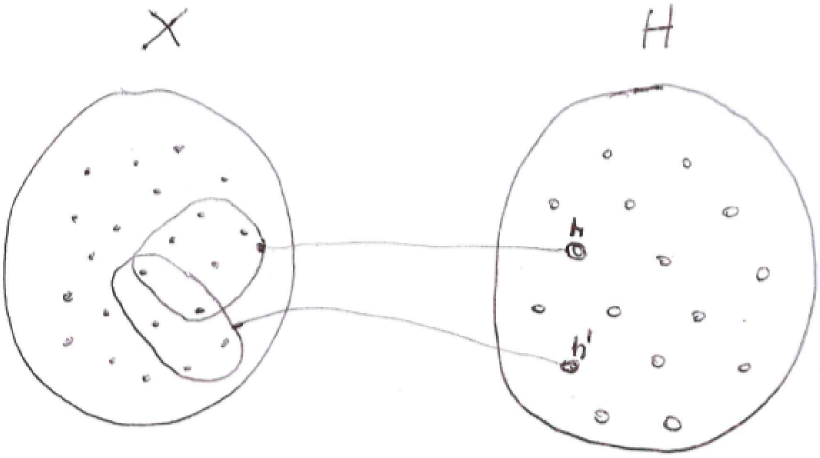
Representation of hypothesis space

... let's skip this part for the moment ...

Representation of hypothesis space

In concept learning (i.e., binary classification), every hypothesis is associated to a set of instances (i.e., all the instances that are classified as positive by such hypothesis).

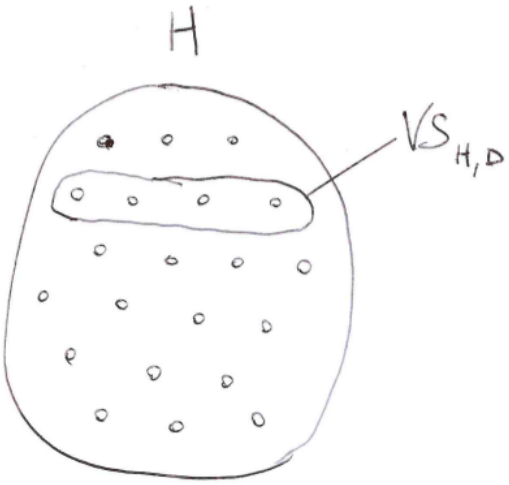
$$h \leftrightarrow S = \{x \in X | h(x) = 1\} \subseteq X \qquad H \leftrightarrow 2^X$$



Version Spaces

The **version space**, *$VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with all training examples in D.*

$$VS_{H,D} \equiv \{h \in H | Consistent(h, D)\}$$



LIST-THEN-ELIMINATE Algorithm

1. *VersionSpace* ← a list containing every hypothesis in H
2. For each training example, $(x, c(x))$
   remove from *VersionSpace* any hypothesis h for which $h(x) \neq c(x)$
3. Output the list of hypotheses in *VersionSpace*

Note: enumerating all the hypotheses!

Example 1

Target function $f : \mathbb{N} \to \{+, -\}$

Dataset $D = \{(1, +), (3, +), (5, +), (6, -), (8, -), (10, -)\}$

Hypothesis: $h \subseteq \mathbb{N}$

Hypothesis space: $H = 2^{\mathbb{N}}$ (set of subsets of $\mathbb{N}$)

Examples of hypotheses:

1. $h_1 =$ “$\{n \in \mathbb{N} \mid \ldots\}$”
2. $h_2 =$ “$\{n \in \mathbb{N} \mid \ldots\}$”
3. $h_3 =$ “$\{n \in \mathbb{N} \mid \ldots\}$”
4. $h_4 =$ “$\{n \in \mathbb{N} \mid \ldots\}$”

Example 1

Target function $f : \mathbb{N} \to \{+, -\}$

Dataset $D = \{(1, +), (3, +), (5, +), (6, -), (8, -), (10, -)\}$

Hypothesis space: $H = 2^{\mathbb{N}}$ (set of subsets of $\mathbb{N}$)

Prediction for new samples $n \notin D$ (e.g., $n = 11$)

1. $h_1(11) = ?$
2. $h_2(11) = ?$
3. $h_3(11) = ?$
4. $h_4(11) = ?$

Example 1

Target function $f : \mathbb{N} \to \{+, -\}$

Dataset $D = \{(1, +), (3, +), (5, +), (6, -), (8, -), (10, -)\}$

Representation power vs. Generalization power

Hypothesis space: $H' = 2^H = 2^{2^{\mathbb{N}}}$ (set of hypotheses / set of sets of subsets of $\mathbb{N}$)

One solution: $\theta = \{h_1, h_2, h_3, h_4\} \in H'$ (set of subsets of $\mathbb{N}$)
Prediction (using majority voting) $\theta(11) = ?$

$h_1 \in H$ has higher generalization power than $\theta \in 2^H$, since $h_1(11) = +$, $\theta(11) = ?$

Example 1

Target function $f : \mathbb{N} \to \{+, -\}$

Dataset $D = \{(1, +), (3, +), (5, +), (6, -), (8, -), (10, -)\}$

Hypothesis space: $H' = 2^H = 2^{2^{\mathbb{N}}}$

Useless solution (no generalization power)

$\exists\, \theta_0 \in H'$, s.t.

$\theta_0(1) = +$, $\theta_0(3) = +$, $\theta_0(5) = +$, $\theta_0(6) = -$, $\theta_0(8) = -$, $\theta_0(10) = -$
$\theta_0(n) = ?$, $\forall n \notin D$

Note: not possible for hypothesis space H

Example 1: Representation vs generalization power

Target function $f : \mathbb{N} \rightarrow \{+, -\}$

Hypotesis spaces: $H = 2^{\mathbb{N}}$ and $H' = 2^{2^{\mathbb{N}}}$

H' has higher representation power than H
- there exist hypothesis $\theta \in H'$ that cannot be represented in H

H' has lowher generalization power that H
- there exists hypothesis $\theta \in H'$ that cannot predict values of the target function for samples not in the dataset.

Example 2

Instance space X: integer points in a 2D plane

Output classes: $\{+, -\}$

Data set D: positive and negative examples in 2D plane

Hypotheses H: rectangles in the 2D plane with edges parallel to the axes ($h(x) = +$ iff x inside the rectangle)

Exercises:
1) Determine the version space of data set
$D = \{((3, 3), +), ((4, 2), +), ((2, 4), -), ((1, 1), -), ((5, 2), -)\}$
2) Draw a dataset for which the version space is empty (i.e., no consistent hypothesis can be determined)

Example 2

Now let's consider a different hypothesis space.

Hypotheses H': sets of rectangles in the 2D plane with edges parallel to the axes

Note: H' can represent any possible subset of X

$$\forall S \in 2^X, \exists h \in H', such\ that\ S = \{x \in X | h(x) = +\}$$

(while this property is not true for the hypothesis space H)

Output of a learning system

1. $VS_{H',D}$ (H' can represent all the subsets of X)
2. $VS_{H,D}$ (H can not represent all the subsets of X)
3. $h^* \in VS_{H,D}$ (h^* is one hypothesis chosen within the VS)

Classify new instances

Given $x' \notin D$, predict class $+$ or $-$.

1. $VS_{H',D} \Rightarrow \forall x' \notin D$, for some $h' \in VS_{H',D}, h'(x') = +$, for some other $h' \in VS_{H',D}, h'(x') = -$
2. $VS_{H,D} \Rightarrow \exists x' \notin D$, for some $h \in VS_{H,D}, h(x') = +$, for some other $h \in VS_{H,D}, h(x') = -$
3. $h^* \in VS_{H,D} \Rightarrow \forall x' \notin D$, $h^*(x')$ is either $+$ or $-$.

Machine Learning representation issue

1. $VS_{H',D} \Rightarrow$ cannot predict instances not in D (for all $x' \notin D$, it will answer *unknown*)
2. $VS_{H,D} \Rightarrow$ limited prediction instances not in D (for some $x' \notin D$, it will answer *unknown*)
3. $h^* \in VS_{H,D} \Rightarrow$ maximum prediction power on values not in D (for all $x' \notin D$, it will always return a predicted value)

If hypothesis space is too powerful (any subset of the instances can be represented in it), and the search is complete (all the solutions are computed), then the system is not able to classify new instances (no generalization power).

Machine Learning noisy data issue

Data set may contain noisy data (a typical case in real applications)
$D = \{(x_i, y_i)\}$ with $y_i \neq c(x_i)$ for some i

There may be no consistent hypotheses, i.e., $VS_{H,D} = \emptyset$

In case of noisy data set, statistical methods must be used to implement robust algorithms.

Summary

- Machine Learning can be seen as learning a function from samples.
- Learning as search requires definition of a hypothesis space and an algorithm to search solutions in this space
- Performance are based on consistency
- Two main issues: 1) representation vs. generalization power, 2) noisy data
- **Statistical methods are needed!!!**