Sapienza University of Rome

Master in Artificial Intelligence and Robotics
Master in Engineering in Computer Science

Machine Learning

A.Y. 2019/2020

Prof. L. Iocchi, F. Patrizi, V. Ntouskos

1. Introduction

L. Iocchi, F. Patrizi, V. Ntouskos

Overview

- What is a Machine Learning problem
- Example: learning to play checkers
- Machine Learning problem formulations
- Learning as search in hyphotesis 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 (navigating on Mars),
- Humans are unable to explain their expertise (speech recognition)
- Solution changes in time (routing on a computer network)
- Solution needs to be adapted to particular cases (user biometrics)

Machine Learning

Machine Learning exploits

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

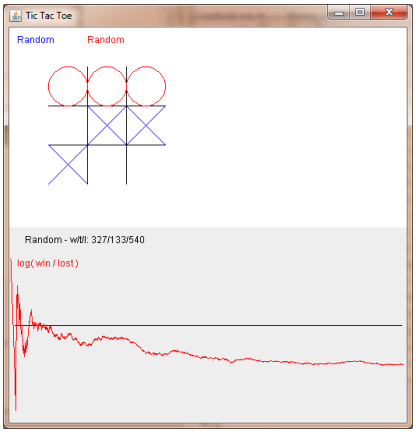
What is a Learning Problem?

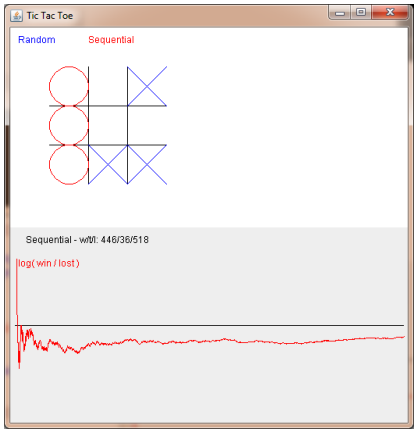
Learning = Improving with experience at some task

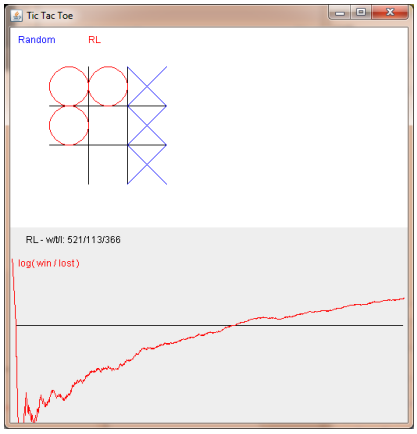
- Improve over task T,
- with respect to performance measure P,
- based on experience E.

Improving performance over time

Example: Tic Tac Toe







Example

Learn to play checkers

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

Learning to Play Checkers

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

Type of Training Experience

- 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

A 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.

This gives correct values, but is not operational!

Choose Representation for Target Function

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

A Representation for Learned Function

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

- $bp(b)$: 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: Unknown w_i Learning $V \equiv$ estimating w_i

Learning algorithm

LMS Weight update rule:

Initialize w_i
Do repeatedly:

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

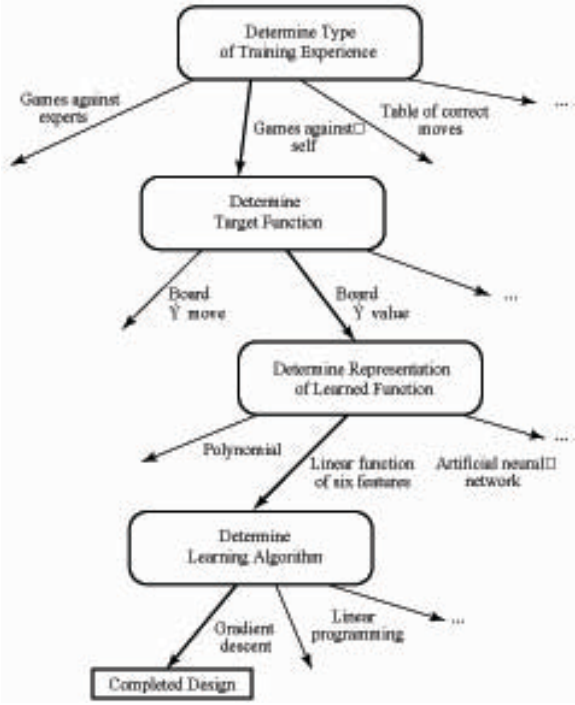
$$error(b) = V_{train}(b) - \hat{V}(b)$$

  2. For each board feature f_i, 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

Design Choices



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 sampled 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.

$X_D = \{x | x \in D\} \subset X$ and $|X_D| \lll |X|$

Machine Learning Problems

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

- $D = \{\langle x_i, y_i \rangle\}$ **(Supervised Learning)**
- $D = \{x_i\}$ **(Unsupervised Learning)**

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

- $D = \{< a_i^{(1)}, \ldots, a_i^{(n)}, r_i >\}$ **(Reinforcement Learning)**

Supervised Learning

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

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

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

Classification (aka Pattern Recognition)

Return the class to which a specific instance belong.

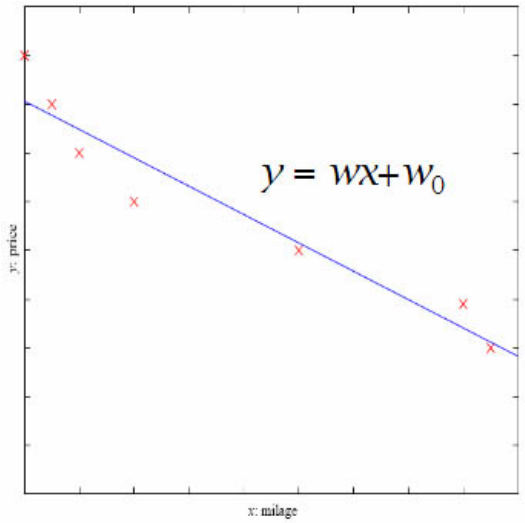
Examples:

- Face recognition: pose, lighting, occlusion (glasses, beard), make-up, hair style.
- Character recognition: different handwriting styles.
- Speech recognition: temporal dependency, multi-modality (dictionary, visual and acoustic)
- Medical diagnosis: from symptoms to illnesses

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: a sequence of outputs
- No supervised output available, only sparse and time-delayed rewards
- Example applications
  - Game playing
  - Robotic tasks
  - Multiple agents
  - ... (any dynamic system with unknown or partially known evolution model)

Open questions in Machine Learning

- What algorithms can approximate functions well (and when)?
- How does number of training examples influence performance?
- How does complexity of hypothesis representation impact performance?
- How does noisy data influence performance?
- What are the theoretical limits of learnability?
- How can prior knowledge of learner help?
- How can systems alter their own representations?
- How can systems learn continuously?
- What clues can we get from biological learning systems?
- ...

Notation

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

X: instance space

$x \in X$: one instance

$D = \{\langle x_i, c(x_i)\rangle_{i=1}^{m}\}$: 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

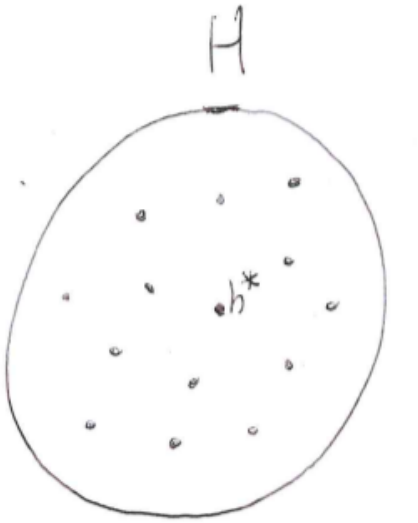
Given a training set $D = \{\langle x_i, c(x_i)\rangle\}$, find the *best* approximation $h^* \in H$ of the target function $c : X \to \{0, 1\}$.

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 hypotysis 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 $\langle x, c(x) \rangle$ 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 $D = \{\langle x_i, c(x_i) \rangle\}$ 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.

Training Examples

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

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., $c = \mathit{PlayTennis} : X \rightarrow \{\mathit{No}, \mathit{Yes}\}$)
- Hypotheses H
- Training examples $D = \{\langle x_1, c(x_1)\rangle, \ldots, \langle x_m, c(x_m)\rangle\}$, positive and negative examples of the target function

Determine:

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

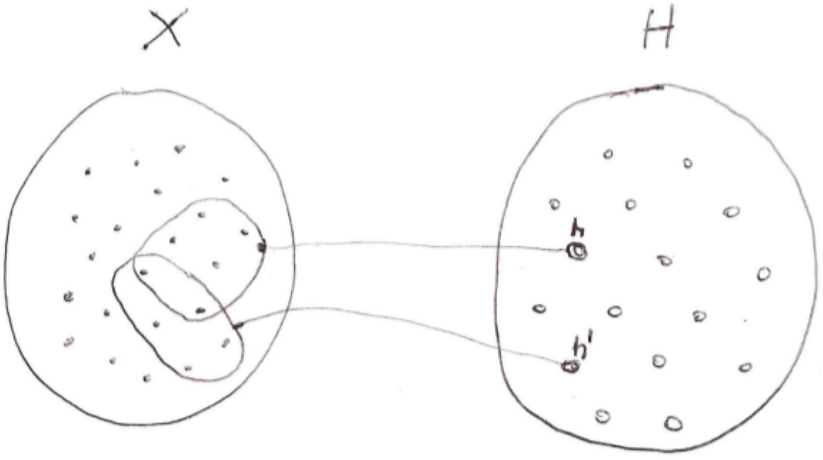
Representation of hypothesis space

... 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).

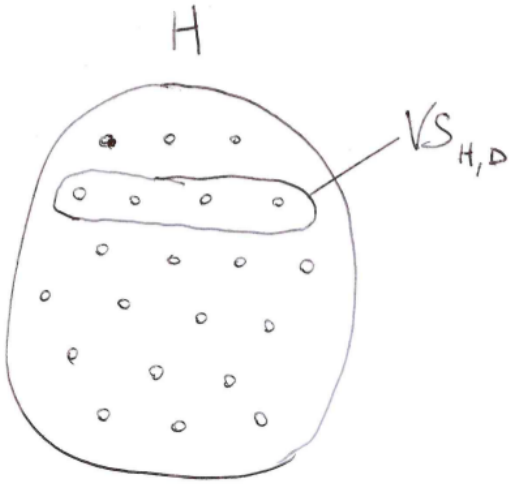
$$H \leftrightarrow 2^X$$



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* $\leftarrow$ a list containing every hypothesis in H
2. For each training example, $\langle x, c(x) \rangle$
   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

Instance space X: integer points in a 2D plane

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

Data set D: positive and negative examples of points belonging to a rectangle

Determine the version space of data set
$D = \{\langle(1,1),+\rangle, \langle(4,2),+\rangle, \langle(2,4),-\rangle, \langle(0,0),-\rangle, \langle(5,2),-\rangle\}$

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

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

(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 $h' \in H', h'(x') = +$, for some other $h' \in H', h'(x') = -$
2. $VS_{H,D} \Rightarrow \exists x' \notin D$, for some $h \in H, h(x') = +$, for some other $h \in H, h(x') = -$
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)
$D = \{\langle x_i, y_i \rangle\}$ 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!!!**