Machine Learning

Computer Engineering

Fabio Vandin

October 4^{th}, 2022

A Formal Model (Statistical Learning)

We have a *learner* (us, or the machine) has access to:

1. **Domain set** $\mathcal{X}$: set of all possible objects to make predictions about
   - domain point $x \in \mathcal{X}$ = *instance*, usually represented by a vector of *features*
   - $\mathcal{X}$ is the *instance space*
2. **Label set** $\mathcal{Y}$: set of possible labels.
   - often two labels, e.g $\{-1, +1\}$ or $\{0, 1\}$

$\mathcal{X} = \{$all graduates in CE$\}$

$\vec{x} \in \mathcal{X}$, $\vec{x} \in \mathbb{R}^4$

$\mathcal{Y} = \{$FUN, NOT FUN$\}$

$= \{+1, -1\}$

Features for graduates:

- final grade
- ML grade
- age at graduation
- height

A Formal Model (Statistical Learning)

We have a *learner* (us, or the machine) has access to:

1. **Domain set** $\mathcal{X}$: set of all possible objects to make predictions about
   - domain point $x \in \mathcal{X}$ = *instance*, usually represented by a vector of *features*
   - $\mathcal{X}$ is the *instance space*
2. **Label set** $\mathcal{Y}$: set of possible labels.
   - often two labels, e.g $\{-1, +1\}$ or $\{0, 1\}$
3. **Training data** $S = ((x_1, y_1), \ldots, (x_m, y_m))$: finite sequence of labeled domain points, i.e. pairs in $\mathcal{X} \times \mathcal{Y}$
   - this is the learner's **input**
   - S: *training example* or *training set*

A Formal Model

4. **Learner's output** h: prediction rule $h : \mathcal{X} \rightarrow \mathcal{Y}$
   - also called *predictor*, *hypothesis*, or *classifier*, model
   - $A(S)$: prediction rule produced by learning algorithm A when training set S is given to it
   - sometimes $\hat{f}$ used instead of h

A Formal Model

4. **Learner's output** h: prediction rule $h : \mathcal{X} \to \mathcal{Y}$
   - also called *predictor*, *hypothesis*, or *classifier*
   - $A(S)$: prediction rule produced by learning algorithm A when training set S is given to it
   - sometimes $\hat{f}$ used instead of h

5. **Data-generation model**: instances are generated by some probability distribution and labeled according to a function

"random generator" for features →

   - $\mathcal{D}$: probability distribution over $\mathcal{X}$ (**NOT KNOWN TO THE LEARNER!**)

formula that we wrote →

   - labeling function $f : \mathcal{X} \to \mathcal{Y}$ (**NOT KNOWN TO THE LEARNER!**)
   - label y_i of instance x_i: $y_i = f(x_i)$, for all $i = 1, \ldots, m$
   - each point in training set S: first sample x_i according to $\mathcal{D}$, then label it as $y_i = f(x_i)$

A Formal Model

4. **Learner's output** h: prediction rule $h: \mathcal{X} \to \mathcal{Y}$
   - also called *predictor*, *hypothesis*, or *classifier*
   - $A(S)$: prediction rule produced by learning algorithm A when training set S is given to it
   - sometimes $\hat{f}$ used instead of h
5. **Data-generation model**: instances are generated by some probability distribution and labeled according to a function
   - $\mathcal{D}$: probability distribution over $\mathcal{X}$ (**NOT KNOWN TO THE LEARNER!**)
   - labeling function $f: \mathcal{X} \to \mathcal{Y}$ (**NOT KNOWN TO THE LEARNER!**)
   - label y_i of instance x_i: $y_i = f(x_i)$, for all $i = 1, \ldots, m$
   - each point in training set S: first sample x_i according to $\mathcal{D}$, then label it as $y_i = f(x_i)$
6. **Measures of success**: *error of a classifier* = probability it does not predict the correct label on a $\underbrace{\text{random data point}}_{\text{may be in } S \text{ or not}}$ generate by distribution $\mathcal{D}$

Loss

Given domain subset $A \subset \mathcal{X}$, $\mathcal{D}(A)$ = probability of observing a point $x \in A$.

In many cases, we refer to A as *event* and express it using a function $\pi : \mathcal{X} \to \{0,1\}$, that is:

$$A = \{x \in \mathcal{X} : \pi(x) = 1\}$$

In this case we have $\mathbb{P}_{x\sim\mathcal{D}}[\pi(x)] = \mathcal{D}(A)$

Error of prediction rule $h : \mathcal{X} \to \mathcal{Y}$ is

model

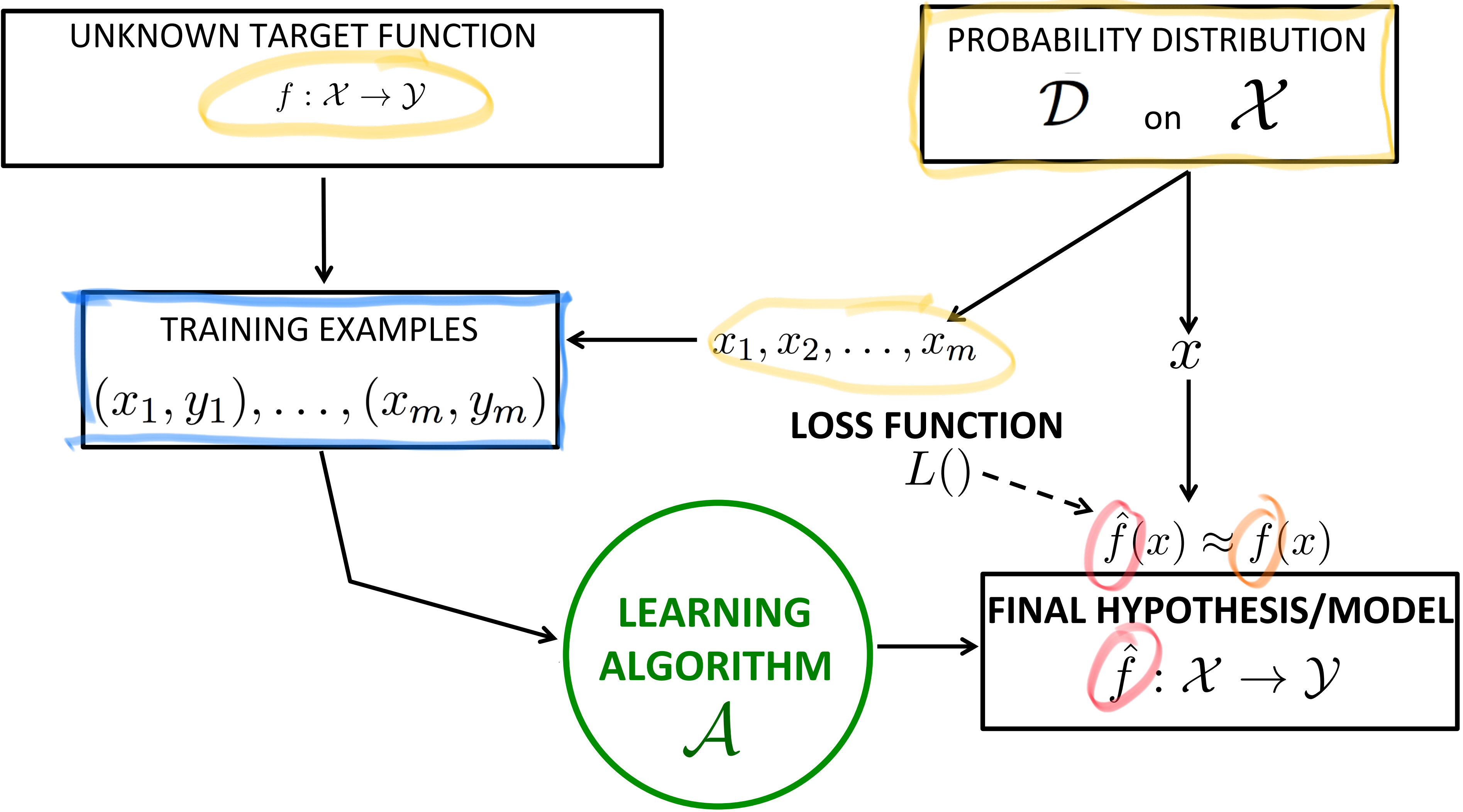
$$L_{\mathcal{D},f}(h) \stackrel{def}{=} \mathbb{P}_{x\sim\mathcal{D}}[h(x) \neq f(x)] \stackrel{def}{=} \mathcal{D}(\{x : h(x) \neq f(x)\})$$

distribution "true" function

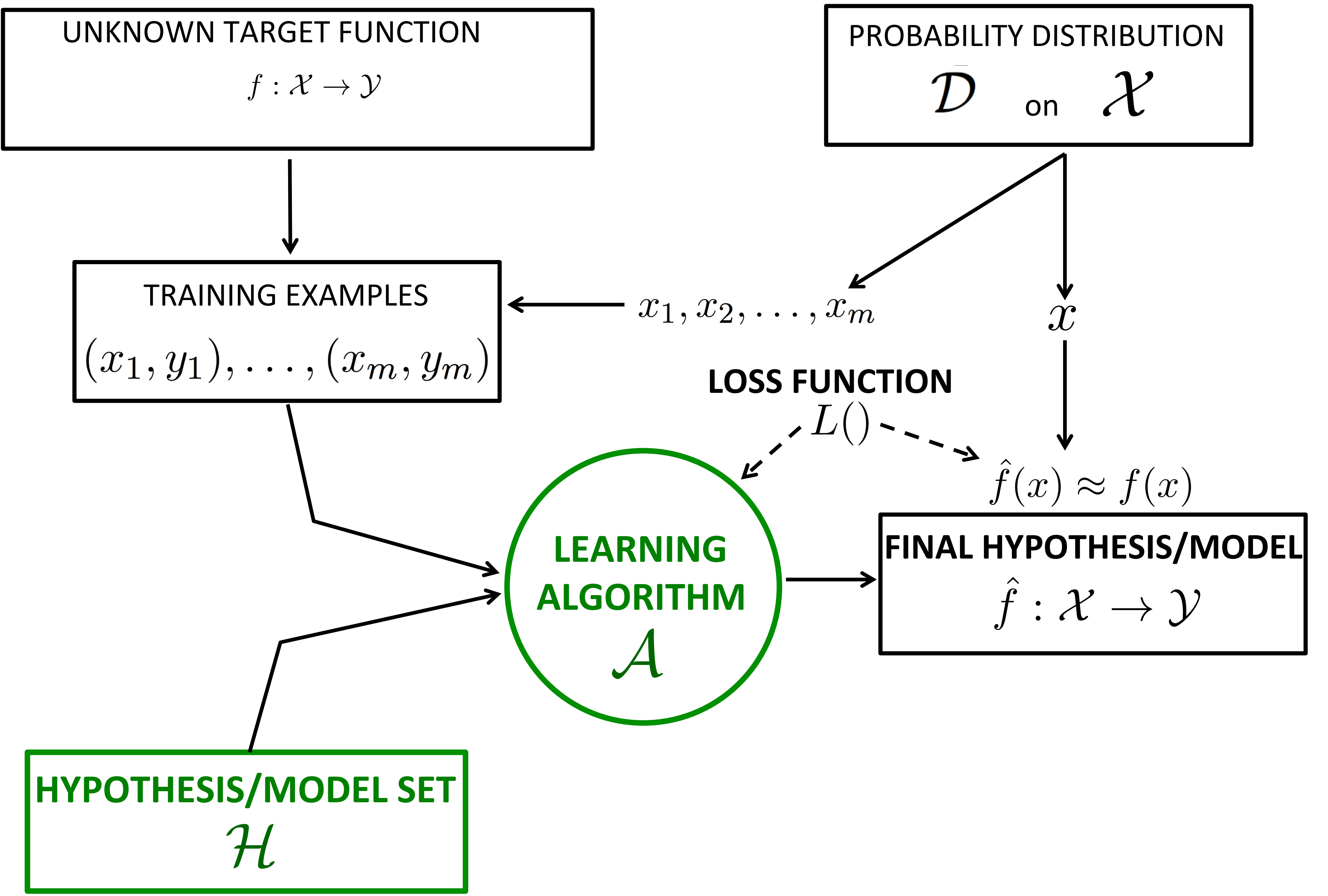
Notes:

- $L_{\mathcal{D},f}(h)$ has many different names: **generalization error**, *true error*, *risk*, **loss**, ...
- often f is obvious, so omitted: $L_{\mathcal{D}}(h)$

Learning Process (Simplified)



Learning Process (Simplified)



Types of Learning

y_i are known: **training set** $(x_1, y_1), \ldots, (x_m, y_m)$

⇒ **supervised learning**

Training set contains only $x_1, x_2, \ldots, x_m$

⇒ **unsupervised learning**

There can be different types of output:

- $\mathcal{Y}$ is **discrete**
- $\mathcal{Y}$ is **continuous**

Notes: we will see a more general learning model soon, main ideas are the same!

Types of Learning

$\mathcal{Y}$ is …

	y_i known ***Supervised Learning***	y_i not available ***Unsupervised Learning***
Discrete	classification	clustering dimensionality reduction …
Continuous	regression	

(Rough) Course Plan

PART I: Supervised Learning

Introduction

Probability Review

Learning Model: PAC Learning

Model Complexity and VC Dimension

Linear Models for Regression: least squares

Linear Models for Classification: Perceptron

Model Selection and Validation

Regularization and Feature Selection

(Rough) Course Plan

Support Vector Machines (SVM) for Classification and Regression

SVM and Kernels

Neural Networks for Classification and Regression

Deep Learning

Decision Trees and Random Forests

PART II: Unsupervised Learning

Hierarchical clustering

Cost based clustering: k-means

Principal Component Analysis (if time permits…)

Objectives

Provide the fundamentals and basic principles of the learning problem

Introduce the most common algorithms for regression and classification

Analytical and practical ability in using these tools for the solution of basic problems

Some hands-on experience

Course Prerequisites!

Calculus

Programming

Linear Algebra

Probability

Calculus

- derivatives
- minimization of functions
- partial derivatives of functions of multiple variables
- integrals

Programming

- You should know at least one programming language (e.g., Java)

  … learning Python will be easy!

Linear Algebra

- matrix factorization
- matrix inversion
- linear independence
- rank, column space, null space
- orthogonality, projections
- eigenvalues, eigenvectors
- symmetric positive definite matrices
- matrix differentiation

Probability

- discrete random variables (r.v.), moments, expectation
- joint, marginal, conditional distribution
- some famous distributions:
  - discrete: binomial
  - continuous: Gaussian
- Independence and conditional independence
- Bayes Theorem
- Law of large numbers

Useful but may not be required: continuous r.v.'s, probability density function (PDF), cumulative distribution function (CDF)

Useful link: *Seeing theory (visualization for probability, statistics, etc.)*
https://students.brown.edu/seeing-theory/basic-probability/index.html

See background material on "Useful links and other stuff"