

Machine Learning: Classification and Regression

(From the Software Engineer Perspective)

Giovanni Ciatto
`giovanni.ciatto@unibo.it`

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
ALMA MATER STUDIORUM – Università di Bologna

PSLab Workshop Series
Cesena, Italy – December 18, 2019



Next in Line...

Overview

Regression and Classification Tasks

Learning Fundamentals

Machine Learning Workflow

Technological ecosystem



Focus on...

Overview

Perspective

Context

Regression and Classification Tasks

Definition

Representation

Learning Fundamentals

Overview

Learning by looking for hypotheses

Cost / performance evaluation

Learning algorithms

Machine learning workflows

Bias–Variance trade-off

Machine Learning Workflow

The lucky case

In practice

Technological ecosystem

(Opinionated) Purpose of the Talk

- ▶ Machine Learning (ML) is a **relevant & hyped** topic nowadays
- ▶ ML is a **wide** topic
- ▶ Our research interests only partially overlap with ML
- We as a group may benefit by **aligning** our backgrounds and skills on the topic
- We may need **more than one** talk to explore ML
 - ▶ possible progression:
Supervised → Unsupervised → Reinforcement → XAI

! XAI = eXplanable Artificial Intelligence

Focus on...

Overview

Perspective

Context

Regression and Classification Tasks

Definition

Representation

Learning Fundamentals

Overview

Learning by looking for hypotheses

Cost / performance evaluation

Learning algorithms

Machine learning workflows

Bias–Variance trade-off

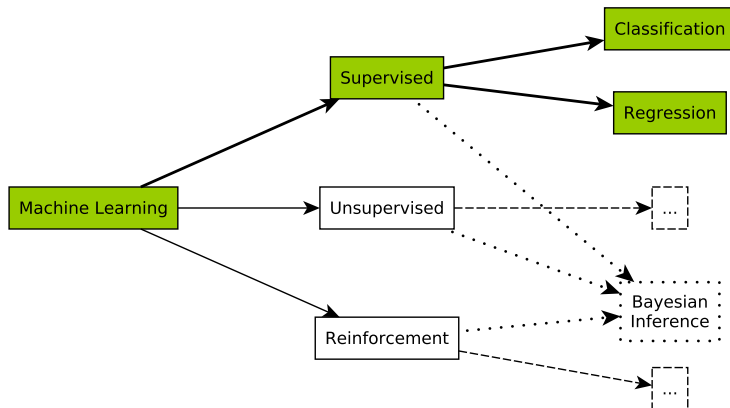
Machine Learning Workflow

The lucky case

In practice

Technological ecosystem

ML Taxonomy – We are here



- In this talk we mostly focus on the **classification** & **regression** tasks
ie the 2 most known & used tasks from **supervised ML**

Loosely defining ML

A computer program is said to learn from **experience** E with respect to some class of **tasks** T and **performance** measure P if its performance at tasks in T , as measured by P , improves with experience E

from [3]

This definition is very wide:

- ▶ what are the possible tasks? are there any constraints?
- ▶ how is performance measured in practice?
- ▶ how / when should experience be provided to tasks?
- ▶ how should the program learn?



Many application of ML

Machine Learning

▶ Supervised Learning

▶ Classification

- ▶ Image recognition
- ▶ Speech recognition
- ▶ ...

▶ Regression

- ▶ Curve fitting
- ▶ Value/Price estimation
- ▶ ...

▶ Unsupervised Learning

- ▶ Clustering
- ▶ ...

▶ Reinforcement Learning

- ▶ Best policy estimation
- ▶ ...

! research topic

! sub-category

! task

! application

! another application

! another task



The Software Engineer (SE) perspective

SE naturally reason in terms of **tasks** a the class of problems they solve:

- ▶ each ML task has a clear **interface** describing
 - ▶ what it accepts in input
 - ▶ what it produces as output
 - ▶ its expected behaviour
- ▶ several **methods** exist in ML to **implement** & **train** each task
 - ▶ ML courses usually focus on methods
 - ▶ we focus on interfaces in this talk
- ▶ **training** a program to perform some ML task requires data scientists to follow an articulated **workflow** with several choice points
 - eg analogy with development methodologies
 - ▶ the workflow only rarely fits ML courses...
 - ▶ ...it is usually acquired through experience by data scientists
 - presenting the full ML workflow along with its choice points is the main goal of this talk

Next in Line...

Overview

Regression and Classification Tasks

Learning Fundamentals

Machine Learning Workflow

Technological ecosystem



Focus on...

Overview

Perspective

Context

Regression and Classification Tasks

Definition

Representation

Learning Fundamentals

Overview

Learning by looking for hypotheses

Cost / performance evaluation

Learning algorithms

Machine learning workflows

Bias–Variance trade-off

Machine Learning Workflow

The lucky case

In practice

Technological ecosystem

Supervised Learning – Training

Given

- ▶ some finite set of **input data** $X = \{\bar{x}_i\} \subset \mathcal{X}$
 - ▶ from an input domain $\mathcal{X}$
- ▶ and its corresponding **expected output data** $Y = \{\bar{y}_i\} \subset \mathcal{Y}$
 - ▶ from an output domain $\mathcal{Y}$
- ▶ or, alternatively, a **dataset** $D = \{(\bar{x}_i, \bar{y}_i)\} \subset \mathcal{X} \times \mathcal{Y}$

for $i = 1, \dots, m$, a supervised learning task **assumes** a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ exists such that:

$$f(X) = Y$$

and it consists of finding a **good approximation** f^* of f by only leveraging on the information in D

Takeaway

Supervised learning $\approx$ function fitting from data

Supervised Learning – Focus on numeric data

- ▶ In practice, data is encoded into **numbers**
ie $\mathcal{X} \equiv \mathbb{R}^n$ and $\mathcal{Y} \equiv \mathbb{R}^k$
 - ▶ without loss of generality we will consider the case $k = 1$
- ▶ Consequently, the dataset is often represented through matrices

$$X = \begin{pmatrix} x_{1,1} & \cdots & x_{1,n} \\ x_{2,1} & \cdots & x_{2,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{pmatrix} \in \mathcal{M}_{m \times n}(\mathbb{R}) \quad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \in \mathbb{R}^m$$

where $\bar{x}_i = (x_{i,1}, \dots, x_{i,j}, \dots, x_{i,n}) \in \mathbb{R}^n$

- ▶ Thus, **rows** are used to represent **examples** of correct input/output instances, whereas **columns** represent **attributes** (a.k.a. **features**)
 - ▶ Features often come with **names** in case of structured data

Classification vs. Regression

Classification

In classification tasks, the **output** domain is **finite**, i.e. $|\mathcal{Y}| = \text{const}$

- ▶ thus, the possible output values are interpreted as **classes**

Regression

In regression tasks, the **output** domain is **infinite**, i.e. $|\mathcal{Y}| = \infty$

- ▶ and almost often **continuous**
- ▶ thus, the task is essentially a generalisation of **interpolation**

Learning vs. Usage

There are two phases in supervised learning:

Learning

Training a novel **predictor** out of the dataset

- ▶ The outcome of learning in classification is a **classifier**
- ▶ The outcome of learning in regression is a **regressor**
- ▶ The general term for classifiers and regressors is **predictor**

Usage

Providing **new** input data to some predictor & exploiting its output

- ▶ The outcome of a classifier is **class**
- ▶ The outcome of a regressor is **value**
- ▶ The outcome of a predictor is **prediction**

Dataset Example – Classification

The Iris dataset:

sepal length	sepal width	petal length	petal width	is setosa	is versicolor	is virginica
6.7	3.1	4.7	1.5	0	1	0
5.7	2.9	4.2	1.3	0	1	0
4.4	3.2	1.3	0.2	1	0	0
6.2	2.2	4.5	1.5	0	1	0
6.3	3.3	6.0	2.5	0	0	1
5.8	2.8	5.1	2.4	0	0	1
6.3	3.4	5.6	2.4	0	0	1
6.9	3.2	5.7	2.3	0	0	1
5.4	3.9	1.7	0.4	1	0	0
6.0	2.7	5.1	1.6	0	1	0
5.0	3.4	1.6	0.4	1	0	0
6.7	3.0	5.2	2.3	0	0	1
6.3	2.5	5.0	1.9	0	0	1
5.1	3.8	1.5	0.3	1	0	0
4.8	3.1	1.6	0.2	1	0	0
5.1	3.5	1.4	0.3	1	0	0
5.5	2.4	3.8	1.1	0	1	0
5.2	3.5	1.5	0.2	1	0	0

► output features: "is setosa", "is versicolor", "is virginica"

► $\mathcal{Y} = \mathbb{B}^3$

Dataset Example – Regression

The Boston Housing dataset:

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0.00632	18.00	2310	0	0.5380	6.5750	65.20	4.0900	1	296.0	15.30	396.90	4.98	24.00
0.02731	0.00	7070	0	0.4690	6.4210	78.90	4.9671	2	242.0	17.80	396.90	9.14	21.60
0.02729	0.00	7070	0	0.4690	7.1850	61.10	4.9671	2	242.0	17.80	392.83	4.03	34.70
0.03237	0.00	2180	0	0.4580	6.9980	45.80	6.0622	3	222.0	18.70	394.63	2.94	33.40
0.06905	0.00	2180	0	0.4580	7.1470	54.20	6.0622	3	222.0	18.70	396.90	5.33	36.20
0.02985	0.00	2180	0	0.4580	6.4300	58.70	6.0622	3	222.0	18.70	394.12	5.21	28.70
0.08829	12.50	7870	0	0.5240	6.0120	66.60	5.5605	5	311.0	15.20	395.60	12.43	22.90
0.14455	12.50	7870	0	0.5240	6.1720	96.10	5.9505	5	311.0	15.20	396.90	19.15	27.10
0.21124	12.50	7870	0	0.5240	5.6310	100.00	6.0821	5	311.0	15.20	386.63	29.93	16.50
0.17004	12.50	7870	0	0.5240	6.0040	85.90	6.5921	5	311.0	15.20	386.71	17.10	18.90
0.22489	12.50	7870	0	0.5240	6.3770	94.30	6.3467	5	311.0	15.20	392.52	20.45	15.00
0.11747	12.50	7870	0	0.5240	6.0090	82.90	6.2267	5	311.0	15.20	396.90	13.27	18.90
0.09378	12.50	7870	0	0.5240	5.8890	39.00	5.4509	5	311.0	15.20	390.50	15.71	21.70
0.62976	0.00	8140	0	0.5380	5.9490	61.80	4.7075	4	307.0	21.00	396.90	8.26	20.40
0.63796	0.00	8140	0	0.5380	6.0960	84.50	4.4619	4	307.0	21.00	380.02	10.26	18.20
0.62739	0.00	8140	0	0.5380	5.8340	56.50	4.4986	4	307.0	21.00	395.62	8.47	19.90
1.05393	0.00	8140	0	0.5380	5.9350	29.30	4.4986	4	307.0	21.00	386.85	6.58	23.10
0.78420	0.00	8140	0	0.5380	5.9900	81.70	4.2579	4	307.0	21.00	386.75	14.67	17.50
0.80271	0.00	8140	0	0.5380	5.4560	36.60	3.7965	4	307.0	21.00	288.99	11.69	20.20

► output feature: "MEDV"

► $\mathcal{Y} = \mathbb{R}$

Assumptions Recap

Common implicit assumptions in supervised ML

- ▶ Availability of a **fixed**, finite-size dataset
- ▶ Made up of a **fixed** amount of **numerical** inputs & outputs features
- ▶ Having no **missing** values
- ▶ Input data is **consistent**
ie $\nexists i, i' : \bar{x}_i = \bar{x}_{i'} \wedge \bar{y}_i \neq \bar{y}_{i'} \wedge i \neq i'$
- ▶ An (unknown) **function** mapping input to output data actually **exists**

Warning

Notice that all such conditions **rarely** fully hold for row data

Focus on. . .

Overview

- Perspective
- Context

Regression and Classification Tasks

- Definition
- Representation**

Learning Fundamentals

- Overview

- Learning by looking for hypotheses

- Cost / performance evaluation

- Learning algorithms

- Machine learning workflows

- Bias–Variance trade-off

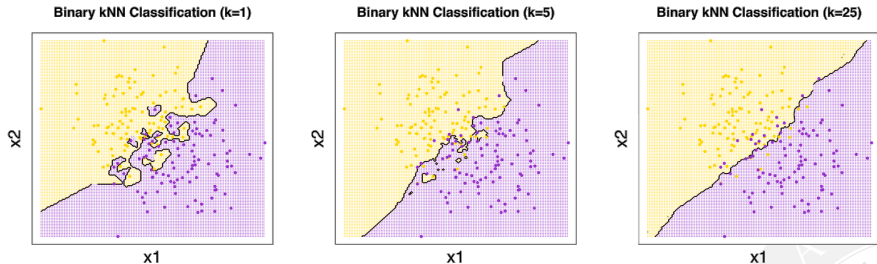
Machine Learning Workflow

- The lucky case

- In practice

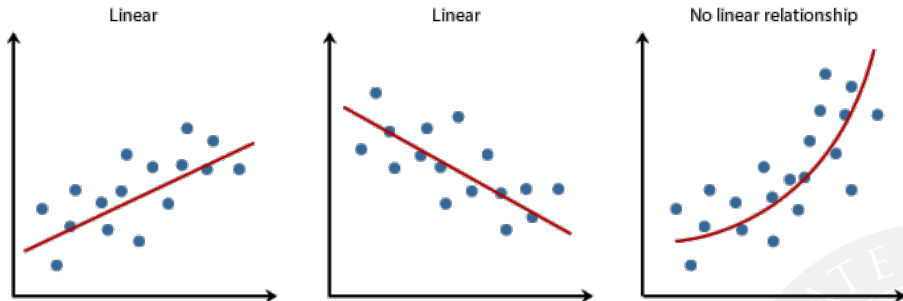
Technological ecosystem

Graphical Representation – Classification



- ▶ **Decision surfaces** are a nice way to represent a 2D-input classifier
- ▶ Most often, classifiers only represent **decision boundaries** internally

Graphical Representation – Regression



- ▶ 1D-input regressors can be represented by **curves** on the plane
- ▶ 2D-input regressors can be represented by **surfaces** in the space
- ▶ N D-input regressors can be conceived as **$(N - 1)$ -dimensional manifolds** in the N -dimensional hyperspace

About equivalence and generality

- ▶ In what follows we mostly provide examples & discussion concerning
 - ▶ **binary** classification—i.e. predictors where $\mathcal{Y} = \mathbb{B}$
 - ▶ **mono-dimensional** regression—i.e. predictors where $\mathcal{Y} = \mathbb{R}$
- ▶ This harms generality in no way, since
 - ▶ **k -class** classification can be attained through **k binary classifiers**
 - ▶ **k -dimensional** regression can be attained through **k independent regressors**
- ▶ Furthermore, **k -class classification** can be **reduced** to **k -dimensional regression**:
 1. the k -class classification task can be modelled as k binary classifiers
 2. each binary classifier can be modelled as a regressor where $\mathcal{Y} = [0, 1]$
 3. the outcome of each regressor can be **discretised**

Next in Line...

Overview

Regression and Classification Tasks

Learning Fundamentals

Machine Learning Workflow

Technological ecosystem



Focus on...

Overview

- Perspective
- Context

Regression and Classification Tasks

- Definition
- Representation

Learning Fundamentals

- Overview

- Learning by looking for hypotheses

- Cost / performance evaluation

- Learning algorithms

- Machine learning workflows

- Bias–Variance trade-off

Machine Learning Workflow

- The lucky case

- In practice

Technological ecosystem

Learning as function fitting

- ▶ In the end of the day, **training** a predictor means **fitting** a function
- ▶ This is usually performed automatically through a **learning** algorithm
 - ▶ “computing” the to-be-fitted function out of the dataset

But what does “computing a function” means in the general case?

It is difficult to answer to this question

- ▶ without relying on some **assumption**
- ▶ constraining the **nature/shape/structure** of the to-be-fitted function

The ML approach

- ▶ Let's draw some **hypothesis** on such nature/shape/structure

Focus on...

Overview

- Perspective
- Context

Regression and Classification Tasks

- Definition
- Representation

Learning Fundamentals

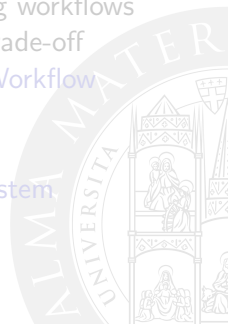
- Overview

Learning by looking for hypotheses

- Cost / performance evaluation
- Learning algorithms
- Machine learning workflows
- Bias–Variance trade-off

Machine Learning Workflow

- The lucky case
- In practice
- Technological ecosystem



Hypotheses & parameters I

The theory behind supervised ML can be synthesised as follows [1]:

1. Let $\Omega = \{\omega\}$ be a set of **hyper-parameters** ranged through by ω
 - ▶ Ω is the hyper-parameters space
 - ▶ ω is a particular assignment of the hyper-parameters
 - eg imagine each ω as a vector of real numbers
2. Let $\Theta = \{\theta\}$ be a set of **parameters** ranged through by θ
 - ▶ Θ is the parameters space
 - ▶ θ is a particular assignment of the parameters
 - eg imagine each θ as a real vector, as well
3. Let us denote by $\pi : \Omega \rightarrow 2^\Theta$ the function mapping each hyper-parameters assignment to some parameters subspace

Hypotheses & parameters II

4. Let us denote by

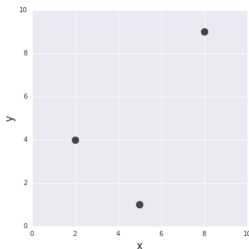
$$\mathcal{H}_\omega = \{h_\theta \mid h_\theta : \mathcal{X} \rightarrow \mathcal{Y}, \forall \theta \in \pi(\omega)\}$$

the **hypothesis space**, that is, the set of all possible **parametric** functions from the input domain $\mathcal{X}$ to the output domain $\mathcal{Y}$ whose possible parameters assignments are in $\pi(\omega)$

5. We thus say each h_θ is an **hypothesis** in the hypothesis space $\mathcal{H}_\omega$
6. We finally define supervised learning as the **automatic process** of searching for the **best hypothesis** $h_\theta^* \in \mathcal{H}_\omega$ w.r.t. the available data D
! assuming $\mathcal{H}_\omega$, Ω , π , and D are given
7. Supervised learning problems are tackled through learning **algorithms**
▶ selecting the best **hypothesis** given some smart choices of $\mathcal{H}_\omega$

Hypotheses & parameters – Example I

- Imagine we have the following 2-dimensional, 3-sized dataset D



- We want to find a regressor for D



Hypotheses & parameters – Example II

- ▶ We arbitrarily choose to focus on the hypothesis space

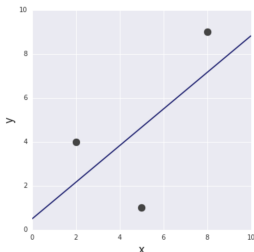
$$\mathcal{H}_{1,1} = \{h_{\theta}(x) = \theta_0 + \theta_1 \cdot x \mid \theta = (\theta_0, \theta_1) \in \mathbb{R}^2\}$$

containing all the possible hypotheses consisting of a **polynomial** function of **degree 1** in **1 variable**—i.e. all possible lines on the plane

- ▶ notice that here $\omega = (1, 1)$
- ▶ In this case:
 - $\mathcal{H}$ dictates the **shape** of the functions (polynomial)
 - ω **constrains** the shape of the functions (1 variable, 1 as degree)
 - θ **identifies** an hypothesis in the hypotheses space
- ▶ We may have chosen different hypothesis **spaces**
 - eg $\mathcal{H}_{p,1}$, i.e., the space of polynomial functions of degree p in 1 variable

Hypotheses & parameters – Example III

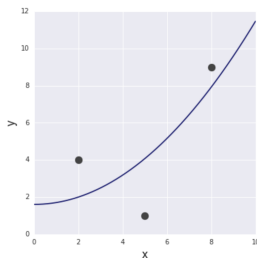
- ▶ We then let an automatic **learning** algorithm **find** the a good hypothesis $h^* \in \mathcal{H}_{1,1}$ among the many



- ▶ The algorithm must essentially compute a good assignment for the **parameters**, given the dataset D
 - ▶ hyper-parameters and data **do not** change

Hypotheses & parameters – Example IV

- Have we chosen the $\mathcal{H}_{2,1}$, one possible solution could have been



(i.e. one parabola)



Hypotheses & parameters – Summing up

Takeaways

- ▶ Learning = searching a good **hypothesis** among several possibilities
- ▶ **Automatic** learning algorithms **only** select parameters

Problems

- ▶ how are “good” hypotheses assessed?
- ▶ how do learning algorithms work?
- ▶ do they work for any possible $\mathcal{H}_\omega$?
- ▶ what about hyper-parameters? how are they assigned?

Focus on...

Overview

Perspective

Context

Regression and Classification Tasks

Definition

Representation

Learning Fundamentals

Overview

Learning by looking for
hypotheses

Cost / performance evaluation

Learning algorithms

Machine learning workflows

Bias–Variance trade-off

Machine Learning Workflow

The lucky case

In practice

Technological ecosystem

Evaluating Hypotheses

Problem

Learning algorithms select the best hypothesis in some hypothesis space

- ▶ how are “good” hypotheses assessed?

- ▶ Learning algorithms select the hypothesis h which
 - ▶ **minimises** the error $\varepsilon(h(X), Y)$, or
 - ▶ **maximises** the predictive performance $\rho(h(X), Y)$of the hypothesis w.r.t the available dataset $D = (X, Y)$
- ▶ This is attained by means of
 - ▶ a **cost** measure $\varepsilon : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$
 - ▶ or, a **performance** measure $\rho : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\leq 1}$



Evaluating Hypotheses – Measures I

- ▶ Several different cost/performance **measures** exists
- ▶ The **choice** depends on the several factors
 - ▶ the target **task**
 - ▶ the **data** at hand and their distribution
 - ▶ the nature of the **learning algorithm**



Evaluating Hypotheses – Measures II

Some possible measures

For regression

- ▶ Evaluating cost
 - ▶ mean squared error (MSE)
 - ▶ mean absolute error (MAE)
- ▶ Evaluating predictive performance
 - ▶ R^2

! good for gradient descent

For classification

- ▶ Evaluating predictive performance
 - ▶ Multi-class
 - ▶ Accuracy
 - ▶ Binary
 - ▶ F1-score

Focus on...

Overview

- Perspective
- Context

Regression and Classification Tasks

- Definition
- Representation

Learning Fundamentals

- Overview

- Learning by looking for hypotheses

- Cost / performance evaluation

Learning algorithms

- Machine learning workflows

- Bias–Variance trade-off

Machine Learning Workflow

- The lucky case

- In practice

Technological ecosystem

The (supervised) learning problem

Learning algorithms are essentially **methods** for solving the problem

$$\min \varepsilon(h_{\theta}(X), Y) \quad \text{s.t. } \theta \in \Theta$$

or, alternatively:

$$\max \rho(h_{\theta}(X), Y) \quad \text{s.t. } \theta \in \Theta$$



About learning algorithms

Problem

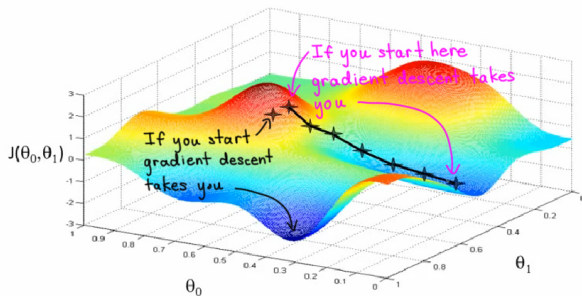
Learning algorithms select the best hypothesis in some hypothesis space

- ▶ how do learning algorithms work?
 - ▶ do they work for any possible $\mathcal{H}_\omega$?
-
- ▶ Learning algorithms are usually **specific** for a particular functional form $\mathcal{H}$
 - ! gradient descent is a notable exception
 - ▶ the nature of the functional form constrains the admissible hyper-parameters Ω
 - ▶ They often come with some **learning parameters** which are not automatically tuned by the algorithm
 - ▶ They are usually **greedy** thus provide no optimality guarantee
 - ! linear regression as a notable exception
 - ▶ this is not a limiting issue in practice

Sorts of learning algorithms

- ▶ Supervised learning algorithms can be categorised in several ways
- ▶ We here propose the following categorization:
 - ▶ **exact** algorithms, providing an **optimal**, exact solution to the learning problem
 - eg Linear Regression, or Support Vector Machines
 - ▶ **non-parametric** algorithms, leveraging on a **lazy**, **local** rules prediction
 - eg k-Nearest-Neighbours + variants
 - ▶ **gradient-descent-based** algorithms, supporting virtually any **differentiable** functional form
 - eg Neural Networks, or Generalised Linear Models + Regularization
 - ▶ **split-based algorithms**, partitioning the input domain through iterative splits
 - eg Decision Trees / Rules
 - ▶ **ensemble algorithms**, combining several predictors into a better one
 - eg Random Forests, Boosting, Bagging

About Gradient Descent (GD)



GD peculiar properties

- ▶ Greedy approach
- ▶ Applicable to any h_θ for which $\varepsilon(h_\theta(X), Y)$ is **differentiable** w.r.t θ
- ▶ Can handle any amount of data
- ▶ Incremental & flexible (i.e. supports modifications to the dataset)

Focus on...

Overview

- Perspective
- Context

Regression and Classification Tasks

- Definition
- Representation

Learning Fundamentals

- Overview

- Learning by looking for hypotheses

- Cost / performance evaluation

- Learning algorithms

Machine learning workflows

- Bias–Variance trade-off

Machine Learning Workflow

- The lucky case

- In practice

Technological ecosystem

From Learning Algorithms to ML Workflows I

Problem

Learning algorithms *automatically* select the best parameters

▶ what about *hyper*-parameters? how are they selected?

- ▶ Estimation of **hyper**-parameters is one of many **poorly** automatic tasks
- ▶ Learning algorithms are just **methods** automatising the selection of **parameters**
 - ▶ assuming several other **choices** have been already performed
- ▶ While pursuing the goal of selecting the best hypotheses for the data, **data scientists**:
 - ▶ leverage on an articulated **workflow**...
 - ▶ ... which only partially rely on **automatic** procedures
 - ▶ ... and whose steps may vary depending on the data & task at hand

From Learning Algorithms to ML Workflows II

- ▶ We will present the workflow later in this talk

- ▶ The main point here is:

*the workflow states **data scientists** must select hyper-parameters*

To do so:

1. they **choose** some functional forms $\mathcal{H}$, $\mathcal{H}'$, $\mathcal{H}''$
 - ▶ according to their **experience** and the data at hand
2. they then estimate a reasonable hyper-parameters **subspace** for each $\mathcal{H}$
 - ▶ again, according to their experience and the data at hand
3. they then **sample** the many hyper-parameters subspaces
 - ▶ **granularity** varies w.r.t. the available **time** & computational resources
4. they finally **compare** the best hypotheses for each functional form $\mathcal{H}$
 - ▶ and select the best one among them
 - ▶ **comparison** is performed through standard means

From Learning Algorithms to ML Workflows III

- ▶ This is better discussed later in this talk

Takeaways

- ▶ Humans must lead the search for the best **hyper-parameters**
- ▶ Data scientists follow an articulated **methodology**
 - ▶ a.k.a. ML workflow
- ▶ The methodology includes several **choice points** and requires **experience**

Focus on...

Overview

- Perspective
- Context

Regression and Classification Tasks

- Definition
- Representation

Learning Fundamentals

- Overview

- Learning by looking for hypotheses

- Cost / performance evaluation

- Learning algorithms

- Machine learning workflows

- Bias–Variance trade-off**

- Machine Learning Workflow

- The lucky case

- In practice

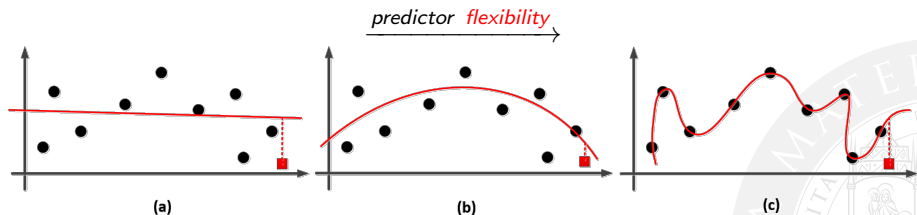
- Technological ecosystem

Subtleties of function fitting

Notice that

It is **always** possible to **perfectly** fit the data

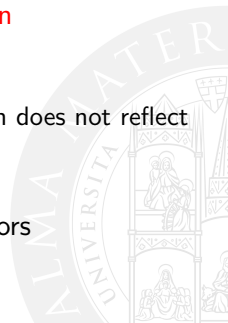
- ▶ Unless the dataset is **inconsistent**



- ▶ Is this what we really want?

Subtleties supervised learning

- ▶ The definitions provided so far are **oversimplified**
- ▶ The **real goal** of supervised learning is **not** to barely **fit** a function over data
- ▶ The actual goal is to **estimate** a function capturing the real input-output behaviour characterising some **population**
 - ▶ given a **sample** limited in size—i.e., the **dataset**
 - ▶ whose size may be **far lower** than the population size
 - ▶ which may expose **random peculiarities** the population does not reflect
 - ▶ which may be affected by some **random error**
- perfectly fitting a dataset may produce **fragile** predictors
 - ▶ which only work well for **that** dataset



Bias–Variance trade-off

The training process may find it self into **2 undesirable situations**

- ▶ high **bias** vs. high **variance**
- ▶ mutually exclusive & difficult to spot

High **bias** situation

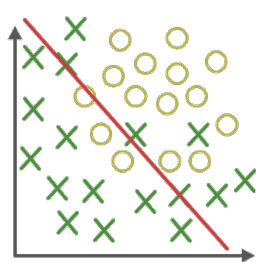
- ▶ The predictor under training is **too inflexible** w.r.t. the data
- ▶ We thus say the predictor is **under-fitting** the data
- More training would be **useless**

High **variance** situation

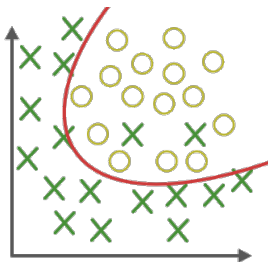
! here Variance $\neq$ the statistical operator

- ▶ The predictor under training is **too flexible** w.r.t. the data
- ▶ We thus say the predictor is **over-fitting** the data
- The predictor is learning too much peculiarities of the dataset
 - it won't **generalise** to the population

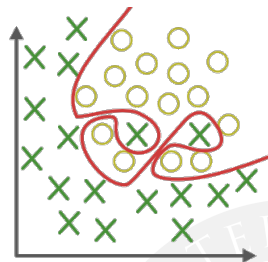
Bias-Variance trade-off — Classification



Under-fitting
(too simple to
explain the variance)



Appropriate-fitting

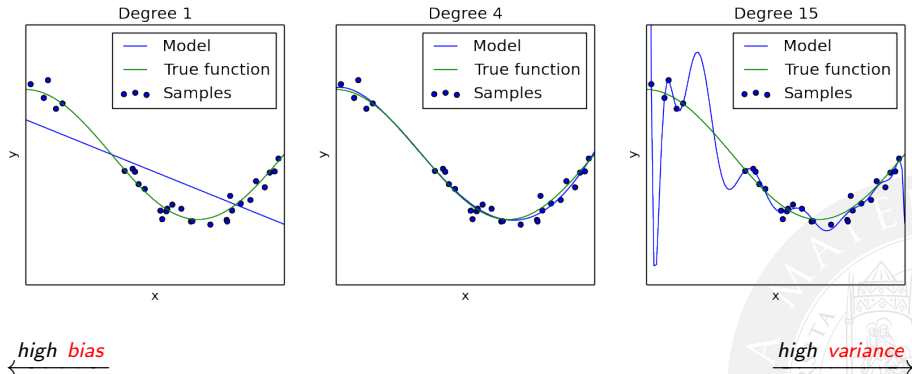


Over-fitting
(forcefitting--too
good to be true)

← *high bias*

high variance →

Bias-Variance trade-off — Regression



Bias-Variance trade-off — Consequences

Problem

- ▶ It is **very hard** to **recognise** if a high-bias/variance situation is in place
 - ← since no one knows the population & the real function
 - ← since high-dimensional scenarios cannot be **graphically represented**

How to control the problem

Two main means:

- ▶ **test-set separation** to assess the **final predictor** against the population
- ▶ **cross-validation** to assess predictors for **comparisons**

Test-set separation I

Purpose

- ▶ Makes it possible to **assess** the predictor selected by a ML workflow
- ▶ Answers the question “will it **generalise** on the population”?

How to

1. Data scientists must **partition** the dataset in **training-** & **test-set**
 - ▶ test-set size = **10-20%** of dataset size
 - ▶ this must be the **VERY FIRST** operation on the dataset
2. The test-set is used to **numerically evaluate** the final predictor. . .
 - ▶ at the **END** of a ML workflow

3. . . . by computing

$$\varepsilon(h_{final}(X_{test}), Y_{test}) \quad \text{or} \quad \rho(h_{final}(X_{test}), Y_{test})$$

Test-set separation II

There's a theory behind such practice:

Rationale

We don't want to **over-estimate** the predictor performances by assessing it through the data it was **trained upon**

Cross-validation (CV) I

Purpose

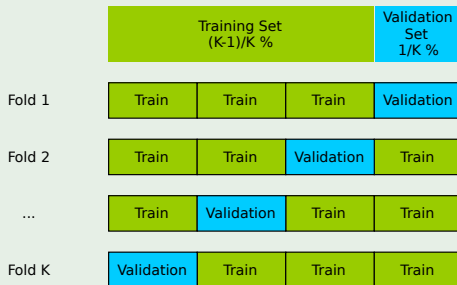
- ▶ Makes it possible to **compare** predictors within a ML workflow
- ▶ Assuming the same metric is used for cross-validating predictors

Rationale

1. Most learning algorithms are **stochastic & greedy** processes
 - different run may lead to (slightly?) different result
 2. How can we be sure the selected predictor is actually the best one?
 - ▶ or at least a good one?
- **Retry** the training several times and **take the best** predictor in K runs

Cross-validation (CV) II

K-Fold Cross-validation — How to



- ▶ The training set is further partitioned into training- and **validation**-set
- ▶ K predictors are produced by the **K-fold** CV
 - ▶ and **validated** against unseen data
- ▶ Many information may be inferred by analysing their performances

Next in Line...

Overview

Regression and Classification Tasks

Learning Fundamentals

Machine Learning Workflow

Technological ecosystem



Focus on...

Overview

- Perspective
- Context

Regression and Classification Tasks

- Definition
- Representation

Learning Fundamentals

- Overview

- Learning by looking for hypotheses

- Cost / performance evaluation

- Learning algorithms

- Machine learning workflows

- Bias–Variance trade-off

Machine Learning Workflow

- The lucky case

- In practice

Technological ecosystem

ML Workflow – The lucky case I

1. Assume a **well-formed** dataset $D = (X, Y)$ is available
 - ▶ are **expected** outputs (Y) available?
2. Perform training- and **test-set** separation

$$(X, Y) = (X_{train}, Y_{train}) \cup (X_{test}, Y_{test})$$

3. **Analyse** the training-set
 - ▶ type of **task**? dimension of the output?
 - ▶ **per-feature** data distribution?
 - ▶ **per-class** data distribution?

! lookahead for next steps

! classification only

ML Workflow – The lucky case II

4. Ensure the training-set data is **consistent** & **numeric**
5. Ensure it is **homogeneously** distributed among **features**
6. Ensure it is **homogeneously** distributed among **classes**
! classification only
7. Select the best hypothesis
 - ▶ perform comparisons through **cross-validation**
8. Finally, evaluate the hypothesis through the **test-set**
 - ▶ **NEVER** turn back



ML Workflow – The lucky case III

Notice

Many things may go wrong in this process



Focus on. . .

Overview

- Perspective
- Context

Regression and Classification Tasks

- Definition
- Representation

Learning Fundamentals

- Overview

- Learning by looking for hypotheses

- Cost / performance evaluation

- Learning algorithms

- Machine learning workflows

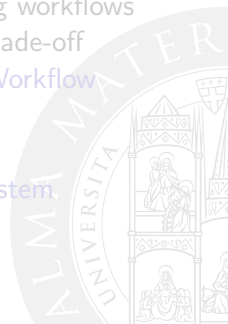
- Bias–Variance trade-off

Machine Learning Workflow

- The lucky case

- In practice

- Technological ecosystem



ML Workflow – In practice I

Step 1 – Assume a **well-formed** dataset is available

- ▶ Are the **expected** outputs Y actually available?
- ▶ Getting the expected outputs is an hard-to-automate task
- ▶ Manual labelling is often required for most classification tasks
- ▶ IoT sensing devices are a great source of examples



ML Workflow – In practice II

Step 2 – Perform training- and **test-set** separation

- ▶ There are are some **implicit** trade-off here
- ▶ Creating good predictors require a lot of **training** data
- ▶ Performing good evaluations of a predictor require a lot of **test** data
- ▶ But the **total** amount of data is usually fixed...
- ▶ ... plus, training and test data must be kept **disjoint**
- ▶ This is also why **BigData** had such a big impact



ML Workflow – In practice III

Remarks

- ▶ In real-world ML competitions the test data is **not** provided
- ▶ Short-circuit for paper reviews:
no/wrong test-set evaluation $\implies$ no results

ML Workflow – In practice IV

Step 4 – Ensure training data is **well formed**

- ▶ Is all data **numeric**?
- ▶ Is there any **missing** value?
- ▶ Is each input data vector **fixed-length**?
- ▶ Are all **input features** relevant?
- ▶ Is the amount of input feature **enough**?
- ▶ Is the amount of input feature **too much**?

! usually yes

Numerifying the dataset

- ▶ There are several means to convert **non-numeric** data into **numeric**
 - ▶ the means to be used in practice depends on the **nature** of the data
 - eg **categorical** data → **one-hot** encoding
 - eg **ordinal** data → **integer** encoding

Color	Red	Yellow	Green
Red			
Red	1	0	0
Yellow	1	0	0
Green	0	1	0
Yellow	0	0	1

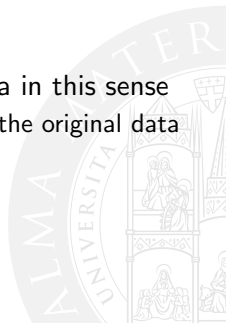
workclass	workclass
State-gov	0
Self-emp-not-inc	1
Private	2
Private	2
Private	2

Imputing missing values

- ▶ There are several means to fill **missing values** into a data set
eg a dataset containing NaN values
- ▶ Any method for such purpose is called **imputer**
- ▶ Imputers **look at similar data** and try to infer the missing values
! this implies imputers must operate **AFTER** test-set separation
- ▶ This is an highly **domain-specific** activity

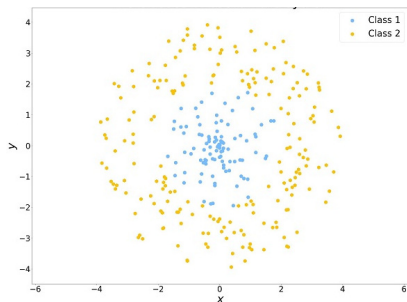
Fixed- vs. Variable-length data

- ▶ Most supervised learning algorithms require **fixed-length** input data
 - ! unless the **kernel-trick** is used in **instance-based** algorithms
- ▶ If the available data is **variable in length**, it must be manipulated
 - ▶ this is a form of **feature engineering**
eg **time-related** data is usually variable in length
- ▶ There are a plethora of methods for manipulating data in this sense
 - ▶ they draw a fixed amount of **statistical features** from the original data
eg **aggregate** data **per period**
- ▶ This is again an highly **domain-specific** activity



Feature Engineering

- ▶ Some times features are too **abstract** / **fine-grained** for learning algorithms to properly work
- ▶ Feature engineering $\approx$ extracting **new** features from data
 - ▶ the new features are usually **less** and aimed at **easing** learning
- ▶ Two notable cases:



Dimensionality Reduction

- ▶ **High-dimensional** data is trouble some in data science
 - ▶ for both computational and statistical reasons
 - eg **Curse of Dimensionality** [2]
- ▶ High dimensionality $\implies$ huge hypothesis space
 - $\rightarrow$ troubles in supervised learning (likely to **overfit**)
- ▶ Data scientists often try **decreased** the amount of feature in the dataset
 - ▶ through methods from the **Dimensionality Reduction** umbrella
- ▶ Examples of methods:
 - ▶ Principal Component Analysis (**PCA**)
 - ▶ Feature Subset Selection
 - ▶ Auto-encoders
 - ▶ Discriminant Analysis
 - ! Feature Engineering may actually reduce dimensionality



ML Workflow – In practice V

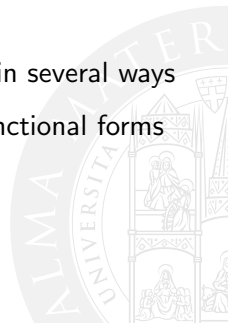
Step 5 – Ensure data is **homogeneously** distributed among **features**

- ▶ what if they are not?
- ▶ **Homogeneous distributions** of data within features **eases** the work of most algorithms
 - better to run learning algorithms on **normalised** data
 - ! especially **gradient-descent**-based algorithms
- ▶ Several **normalisation** means:
 - eg normalisation w.r.t. the variational **range** ($Max - min$)
 - eg normalisation w.r.t. the **standard deviation**
- ! Normalisation must be performed **AFTER** test-set separation
- ▶ Normalisation should be (re-)performed **every time** a new training-set is selected
 - eg in **any fold** of a cross-validation process

ML Workflow – In practice VI

Step 5 – Ensure data is **homogeneously** distributed among **classes**

- ! this only make sense in **classification** tasks
- ▶ what if they are not?
- ▶ **Balanced distributions** of data within classes **eases** the work of most algorithms
- ▶ The “**unbalanced classes**” case affects ML workflows in several ways
- ▶ It affects the choice w.r.t. **learning algorithms** and functional forms
 - ▶ **split-based** algorithms are less prone to such issues
eg decision trees
- ▶ It affects the choice w.r.t. **performance measures**
eg **f1-score** is less prone to such issues



Next in Line...

Overview

Regression and Classification Tasks

Learning Fundamentals

Machine Learning Workflow

Technological ecosystem



Python for ML

- ▶ **Python** is the most used platform for ML & data science in general
 - ▶ in particular the **Anaconda** distribution
 - ▶ perfect level of detail for SE
- ▶ Pool of **interoperable** Python libraries for data science
 - numpy** — algebra, statistic, and calculus
 - pandas** — datasets manipulation, visualisation, and analytics
 - matplotlib** — advanced, programmatic plotting facilities
 - scikit-learn** — general purpose ML, programmatically
 - ▶ very good architecture, implementation & **documentation**
 - tensorflow** — symbolic differentiation of arbitrary functions
 - keras** — Tensorflow front-end for neural-networks
- ▶ Pycharm as IDE or **Jupyter Notebooks**

Machine Learning: Classification and Regression

(From the Software Engineer Perspective)

Giovanni Ciatto
`giovanni.ciatto@unibo.it`

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
ALMA MATER STUDIORUM – Università di Bologna

PSLab Workshop Series
Cesena, Italy – December 18, 2019



References I

- [1] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin.
Learning From Data – A Short Course.
AMLbook.com, 2012.
- [2] Trevor Hastie, Robert Tibshirani, and Jerome Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.
- [3] Thomas M. Mitchell.
Machine Learning.
McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

