Machine Learning per la fisica applicata e fisica delle alte energie

Lezione 1: Mini sguardo d'insieme al Machine Learning.

1.1 GENERALITÀ SUL CORSO

48 h di lezioni (6 CFU) di cui
- 24 Dr. Nocera (12 + 12)
- 24 Prof. Badger (24)

Orario: Lun. 9-11; Mar. 14-16 sempre in AULA D

Obiettivi: familiarizzare con gli algoritmi e i software di Machine Learning in vista sia di una formazione di terzo livello sia di un successivo impiego nel settore ricerca e sviluppo.

Prerequisiti: familiarità con un moderno linguaggio di programmazione (Python, c++) come dal corso di Algoritmi Numerici per la Fisica.

Risultati dell'apprendimento:
- descrivere le tecniche di Machine Learning utilizzate nella risoluzione di problemi fisici complessi;
- individuare le caratteristiche di ciascuna tecnica e metterle in relazione con un problema fisico;
- applicare software specializzato agli scopi del punto precedente.

Programma: come da sito CAMPUSNET; flessibile; aggiornabile.

Modalità d'insegnamento: lezioni frontali + set di esercizi proposti a cadenza bisettimanale + discussione in classe di un paper a scelta tra quelli proposti in un database. Le attività non sono né obbligatorie, né sottoposte a giudizio, MA sono CALDAMENTE CONSIGLIATE.

Esame: ORALE. Colloquio di circa 30 minuti in cui lo studente espone la realizzazione di un progetto computazionale a sua scelta, ma concordato con i docenti a lezione, in cui una o più delle tecniche di Machine Learning sono applicate alla risoluzione di un problema fisico. Il problema fisico non deve essere necessariamente originale e deve essere possibilmente circoscritto. Le modalità di presentazione sono a scelta dello studente. La lingua (italiano o inglese) pure.

Bibliografia: sul sito CAMPUSNET

- Review
- Hastie, Tibshirani, Friedman "The elements of Statistical Learning" Data Mining, Inference and Prediction", Springer (2018) 2nd Ed.
- James, Witten, Hastie, Tibshirani, "An Introduction to Statistical Learning", Springer (2021) 2nd Ed.
- K. P. Murphy, "Probabilistic Machine Learning: An Introduction" MIT Press (2022).

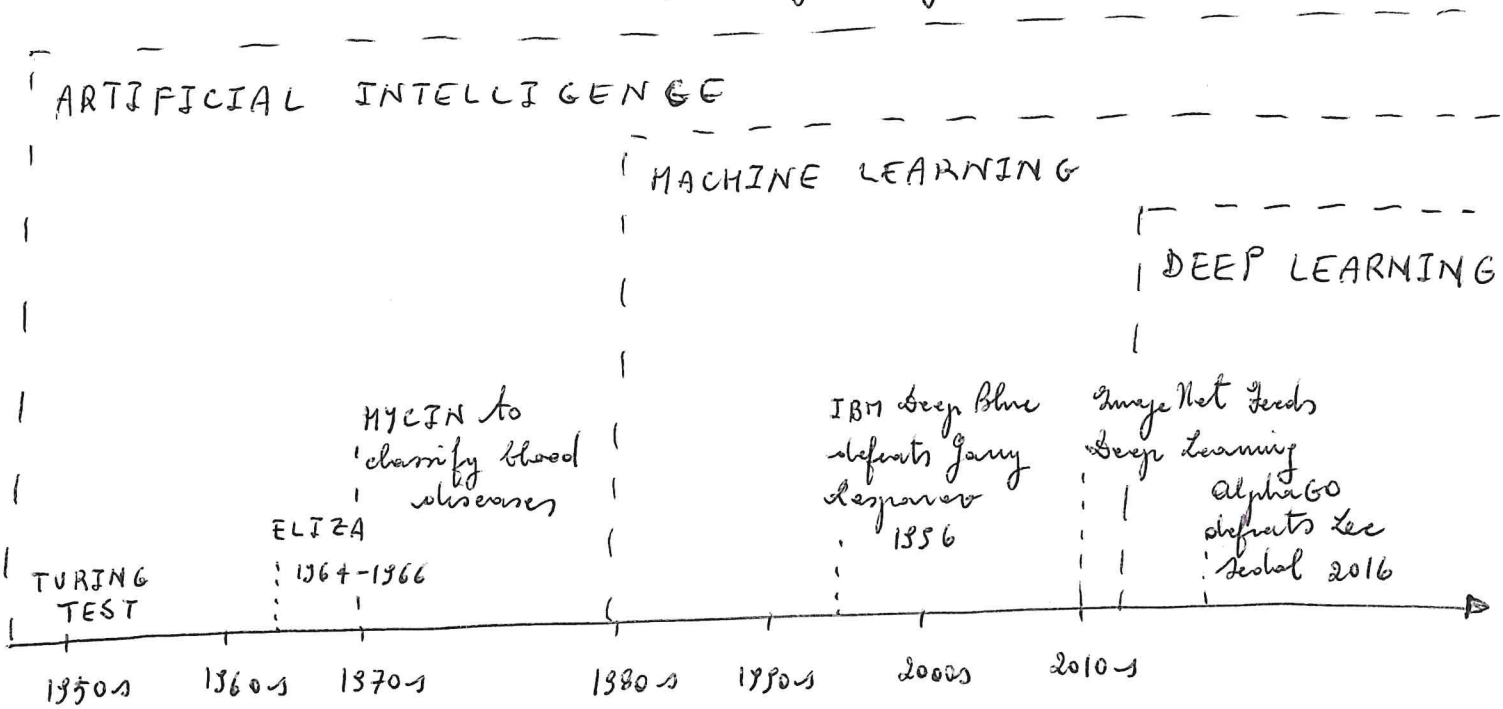
DOMANDE?

1.2 Che cos'è il Machine Learning?

T. Mitchell (1997)

"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".

Il termine è molto più vecchio e fu introdotto da Arthur Samuel nel contesto del computer gaming nel 1959.



L'impostazione del corso è di tipo probabilistico (statistical learning). Tutte le quantità non note sono trattate come variabili aleatorie (RANDOM VARIABLES) cui è associata una distribuzione di probabilità (PROBABILITY DISTRIBUTION) che descrive il set (pesato) di valori che la variabile può assumere.

MACHINE LEARNING

A SUPERVISED LEARNING B UNSUPERVISED LEARNING C REINFORCEMENT LEARNING

Il corso sviluppa essenzialmente A e B.

1.3 SUPERVISED LEARNING

Il compito T consiste nell'imparare una mappa f dagli inputs $x \in X$ agli outputs $y \in Y$; gli input x sono chiamati FEATURES (o COVARIATES o PREDICTORS) e sono generalmente costituiti da un vettore reale di dimensione finita: $X \equiv \mathbb{R}^D$

(per esempio l'altezza ed il peso di una persona). Gli output y sono noti come LABEL (o TARGET o RESPONSE). L'esperienza E è data come set di N coppie input-output: $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^{N}$ e si chiama TRAINING SET. N si chiama SAMPLE SIZE. La performance P dipende dal task T.

1.3.1. CLASSIFICAZIONE

In un problema di classificazione lo spazio degli output C è un set NON ordinato ed esclusivo di LABELS y, dette CLASSES, $\mathcal{Y} = \{1, 2, \dots, C\}$. Il problema di predire una classe dato un input è detto PATTERN RECOGNITION. Se gli output sono solo due, il problema è detto CLASSIFICAZIONE BINARIA.

Esempio: Classificare la specie di Iris (Setosa, Versicolor, Virginica)

In generale, in IMAGE CLASSIFICATION, gli input sono immagini

$$X = \mathbb{R}^D, \quad D = C \times D_1 \times D_2 \qquad C = 3 \text{ channels (RGB)}$$

$$f : X \longrightarrow \mathcal{Y} \qquad \text{(è l'immagine un cane? un gatto? oppure?)}$$

Nel caso delle specie di Iris i botanisti hanno individuato 4 caratteristiche numeriche: lunghezza e larghezza del sepalo e del petalo. Pertanto $X = \mathbb{R}^4$. Il training set è una collezione di 150 esempi delle 3 specie, 50 per ogni specie.

indice	sl [cm]	sw [cm]	pl [cm]	pw [cm]	label
0	5,1	3,5	1,4	0,2	Setosa
1	4,9	3,0	1,4	0,2	Setosa
⋮					
50	7,0	3,2	4,7	1,4	Versicolor
⋮					
150	5,9	3,0	5,1	1,8	Virginica
	D				

TABULAR DATA (DESIGN MATRIX)

BIG DATA: N grande $(N \gg D)$

WIDE DATA: D grande $(D \gg N)$

È una buona idea fare un'esplorazione dei dati (EXPLORATORY DATA ANALYSIS) per vedere se ci sono dei pattern ovvi, per esempio mediante grafi plots. Se basi di dati grandi, si può procedere mediante DIMENSIONALITY REDUCTION

$$f(\vec{x}, \vec{\vartheta}) = \begin{cases} \text{Setosa se p.l.} < 2{,}45 \text{ cm} \\ \text{Versicolor o Virginica altrimenti} \end{cases} \Bigg\} \text{DECISION TREE}$$

$$\begin{cases} \text{p.w.} < 1{,}75 \text{ cm Versicolor} \\ \text{Virginica altrimenti} \end{cases}$$

THRESHOLD PARAMETER

Lo scopo del supervised learning è quello di ottenere modelli di classificazione (tra le altre cose). La PERFORMANCE può essere misurata come:

$$\mathcal{L}(\vartheta) \triangleq \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}\left(y_n \neq f(\vec{x}_n, \vec{\vartheta})\right) \quad \text{MISCLASSIFICATION RATE}$$

dove $\mathbb{I}(e)$ è l'indicatore binario $\mathbb{I}(e) = \begin{cases} 1 \text{ se } e \text{ è vero} \\ 0 \text{ se } e \text{ è falso} \end{cases}$

Nel caso in cui alcuni errori di classificazione siano più dannosi di altri, posso definire una loss function $l(y, \hat{y})$ e

$$\mathcal{L}(\vartheta) \triangleq \frac{1}{N} \sum_{n=1}^{N} l\left(y_n, f(\vec{x}_n, \vec{\vartheta})\right) \quad \text{EMPIRICAL RISK}$$

Una maniera per definire il TRAINING (o MODEL FITTING) è minimizzare il rischio empirico

$$\hat{\vartheta} = \arg\min_{\vartheta} \mathcal{L}(\vartheta) = \arg\min_{\vartheta} \frac{1}{N} \sum_{n=1}^{N} l\left(y_n, f(\vec{x}_n, \vec{\vartheta})\right)$$

sebbene lo scopo sia di considerare dati che NON conosciamo.

Nella determinazione di modelli parametrici è comune usare la probabilità logaritmica negativa come loss function

$$l\left(y, f(\vec{x};\vec{\theta})\right) = -\ln p\left(y / f(\vec{x};\vec{\theta})\right)$$

RAGIONE: un buon modello (con un loss basso) assegna un'alta probabilità all'output vero y $\forall$ input x. La media è

$$NLL(\vec{\theta}) = -\frac{1}{N}\sum_{n=1}^{N} \ln p\left(y_n / f(\vec{x}_n;\vec{\theta})\right) \quad \text{NEGATIVE LOG-LIKELIHOOD}$$

Minimizzando NLL, si calcola

$$\hat{\theta}_{MLE} = \arg\min_{\vec{\theta}} NLL(\vec{\theta}) \quad \text{MAXIMUM LIKELIHOOD ESTIMATE}$$

1.3.2 REGRESSIONE

In un problema di <u>regressione</u> lo spazio degli output è $y \in \mathbb{R}$. Esempio: y potrebbe essere il grado di tossicità di un fungo mangiato da una mucca al pascolo.

In tal caso, occorre definire una loss function differente. Una scelta comune è il QUADRATIC LOSS (o l_2):

$$l_2(y, \hat{y}) = (y - \hat{y})^2$$

segue che

$$MSE(\vec{\theta}) = \frac{1}{N}\sum_{n=1}^{N} \left(y_n - f(\vec{x}_n;\vec{\theta})\right)^2$$

supponendo che la distribuzione di probabilità degli output sia Gaussiana

$$\mathcal{N}(y/\mu,\sigma^2) \triangleq \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y-\mu)^2}$$

allora:

$$p(y/\vec{x};\vec{\theta}) = \mathcal{N}(y / f(\vec{x};\vec{\theta}), \sigma^2)$$

e pertanto

$$NLL(\vec{\vartheta}) = -\frac{1}{N}\sum_{n=1}^{N_D} \ln\left[\left(\frac{1}{2\pi\sigma^2}\right)^{1/2} \exp\left(-\frac{1}{2\sigma^2}(y_n - f(x_n;\vartheta))^2\right)\right]$$

$$= \frac{1}{2\sigma^2} MSE(\vec{\vartheta}) + cost$$

Pertanto MLE mi permette minimizza l_2.

1.3.2.1 REGRESSIONE LINEARE

$$f(x, \vec{\vartheta}) = wx + b \qquad \vec{\vartheta} = (w, b)$$

(slope: w, offset: b)

$$\hat{\vartheta} = \arg\min_{\vec{\vartheta}} MSE(\vec{\vartheta}) \quad \text{(LEAST SQUARE SOLUTION)}$$

In generale $f(\vec{x}, \vec{\vartheta}) = b + w_1 x_1 + w_2 x_2 + \dots + w_D x_D = b + \vec{w}^T \cdot \vec{x}$

(MULTIPLE LINEAR REGRESSION)

1.3.2.2 REGRESSIONE POLINOMIALE

$$f(x; \vec{w}) = \vec{w}^T \Phi(x) \text{ con } \Phi(x) = [1, x, x^2, \dots, x^D]$$

Se $D = N-1$ allora si parla di INTERPOLATION ($MSE = 0$)

$$f(\vec{x}; \vec{w}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + \dots$$

1.3.2.3 RETI NEURALI

I due esempi precedenti prevedono che f sia lineare in $\vec{w}$

$$f(\vec{x}, \vec{w}, \vec{V}) = \vec{w}^T \Phi(\vec{x}, \vec{V}) \qquad \vec{V} \text{ è un set di parametri per } \Phi(x)$$

$$f(\vec{x}, \vec{\vartheta}) = f_L(f_{L-1}(\dots(f_1(\vec{x}))\dots)) \text{ con } f_l(\vec{x}) = \vec{w}^T f_{1:L-1}(\vec{x})$$

NEURAL NETWORK

Overfitting and generalisation. Uno può riscrivere

$$\mathcal{L}(\vec{\vartheta}, \mathcal{D}_{train}) \triangleq \frac{1}{|\mathcal{D}_{train}|} \sum_{(\vec{x},\vec{y}) \in \mathcal{D}_{train}} \ell(\vec{y}, f(\vec{x}; \vec{\vartheta}))$$

rendendolo esplicito il set di training $\mathcal{D}_{train}$. Se $\mathcal{L}(\vec{\vartheta}, \mathcal{D}_{train})$ è tale per cui MSE = 0, allora si parla di OVERFITTING.
Per stabilire se un modello soffre di OVERFITTING, supponiamo di conoscere la "vera" distribuzione $p^*(\vec{x}, \vec{y})$ usata per generare il training set. Allora, posso calcolare il POPULATION RISK

$$\mathcal{L}(\vec{\vartheta}; p^*) \triangleq \mathbb{E}_{p^*(x,y)}[\ell(\vec{y}, f(\vec{x}, \vec{\vartheta}))]$$

La differenza $\mathcal{L}(\vec{\vartheta}; p^*) - \mathcal{L}(\vec{\vartheta}, \mathcal{D}_{train})$ si chiama GENERALISATION GAP: GG large $\Rightarrow$ OVERFITTING. In practice, partition the data in $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$

$$\mathcal{L}(\vec{\vartheta}; \mathcal{D}_{test}) \triangleq \frac{1}{|\mathcal{D}_{test}|} \sum_{(\vec{x},\vec{y}) \in \mathcal{D}_{test}} \ell(\vec{y}, f(\vec{x}, \vec{\vartheta})) \simeq \text{POPULATION RISK}$$

How do we choose our model? Hyperoptimisation.

No free lunch theorem. "All models are wrong, but some are useful" (George Box, 1987). There is no single best model that works optimally for all kinds of problems.

1.4. UNSUPERVISED LEARNING

Il compito T consiste nel trovare certe caratteristiche degli inputs $x \in X$ senza conoscere gli output y. Da un punto di vista probabilistico, si tratta di fittare un modello $p(\vec{x})$ non condizionato (per contro, il supervised learning fitta $p(\vec{y}/\vec{x})$).

Differenze tra SL e US

US: non c'è necessità di collezionare grandi training sets

US: non c'è necessità di partizionare X
US: obbligo il modello a "spiegare" gli inputs

1.4.1 CLUSTERING and LATENT FACTORS

Il problema consiste nel trovare partizioni in X che corrispondono a x simili. Esempio: classificazione degli Iris (CLUSTERING).
Il problema consiste nel proiettare un dato set "high-dimensional" in caratteristiche "low dimensional". Esempio: supponiamo che $x_n \in \mathbb{R}^D$ sia generato da low-dimensional LATENT FACTORS $z_n \in \mathbb{R}^K$: $z_n \to x_n$. Assumiamo un prior fattorizzato $p(z_n)$

$$p(\vec{x}_n / \vec{z}_n; \vec{\theta}) = \mathcal{N}(\vec{x}_n / \vec{W}\vec{z}_n + \vec{\mu}, \vec{\Sigma}) \quad \text{FACTOR ANALYSIS}$$

(se $\vec{\Sigma} = \sigma^2 \vec{I}$, PRINCIPAL COMPONENT ANALYSIS)

se il modello è non-lineare, allora si parla di VARIATIONAL AUTOENCODER

Come valutare la bontà dell'unsupervised learning?

Misurare la probabilità assegnata dal modello a test samples non viste dal modello

$$\mathcal{L}(\vec{\theta}; \mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ln p(\vec{x} / \vec{\theta})$$

cioè un "buon modello" non si farà sorprendere da dati samples che obbediscono al modello.

1.5 REINFORCEMENT LEARNING

In questa classe di problemi, il sistema o AGENTE, deve imparare ad interagire con l'ambiente. Questo può essere realizzato mediante una POLICY $\vec{a} = \pi(\vec{x})$ che specifica che cosa l'agente deve fare $\forall$ possibile input $\vec{x}$ dell'ambiente. L'agente riceve una ricompensa occasionale (o una punizione) per le sue azioni. È simile ad imparare con un critico (anziché con un insegnante)