Machine learning

Foundations

Corso di Laurea Magistrale in Informatica

Università di Roma Tor Vergata

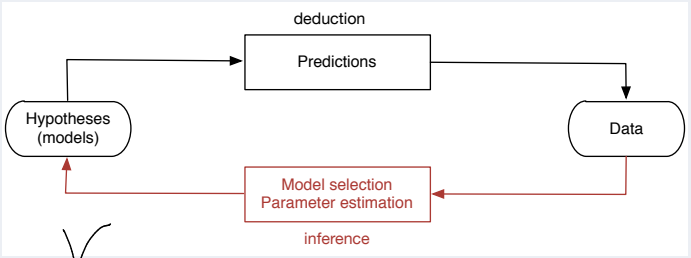
Prof. Giorgio Gambosi

a.a. 2021-2022

Machine learning: inductive approach

Learning of commonalities through analysis of a set of examples (training set), which is assumed to be available.

- A training set of n items is represented as a set of input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, used to derive a model.
- If the purpose is item classification with respect to a collection of predefined classes, the training set also includes a target vector $\mathbf{t} = \{t_1, \ldots, t_n\}$, where the class of each training set item is specified.



- Ognuno di questi d valori del vettore x_i è associato a qualche caratteristica, quindi ogni elemento del mondo considerato è rappresentato attraverso d sue caratteristiche numeriche.

- Esempio: abbiamo un insieme di persone e vogliamo predire il sesso sulla base di peso ed altezza.
Ogni persona sarà rappresentata con i valori di queste due caratteristiche, che sono le features e sono tutto quello che assumiamo di conoscere e tutta l'informazione per effettuare la predizione.

- Il sistema deve quindi essere in grado di fare quanto segue:
preso un insieme di vettori di 2 dimensioni, ognuno col il valore del target, viene utilizzato un metodo per derivare un altro metodo che, a partire dalle feature predirà se la persona è maschio o femmina, con una buona precisione.

- Dobbiamo inoltre avere una valutazione di quanto si comporta bene il metodo, va quindi sperimentato su dei dati, ad esempio vedendo quante volte sbaglia ma qual è l'insieme dei dati?
Inoltre Cosa vuol dire "quanto sbaglia"? Serve una funzione di errore.

Supervised learning

- We want to predict, given the values of a set (features) of an item $\mathbf{x}$, the unknown value of an additional feature target of the item
  - Target in $\mathbb{R}$: regression. Target in $\{1, \dots, K\}$: classification.
- General approach: defined (by means of learning from a set of examples) a model of the relation between feature and target values.
- The training set $\mathbf{X}$ $\mathbf{t}$ includes a feature vector $\mathbf{x}_i = \{x_{i1}, \dots, x_{im}\}$ and the corresponding target t_i for each item.
- The model could be:
  1. a function $y()$ which, for any item $\mathbf{x}$, returns a value $y(\mathbf{x})$ as an estimate of t
  2. a probability distribution which associates to each possible value $\overline{y}$ in the target domain, the corresponding probability $p(y = \overline{y}|\mathbf{x})$

Il training set è una matrice X $(n \times d)$

- Guardando il problema geometricamente, è come se avessimo un insieme di n punti in uno spazio di d dimensioni e a cui è associato un valore, vogliamo quindi trovare un modo per dividere lo spazio di d dimensioni: sempre nel caso sesso vs (altezza,peso), gli elementi saranno punti in uno spazio 2D, alcuni maschi e femmine. Vorremmo avere un metodo, ovvero una decomposizione dello spazio in regioni in cui ad ogni regione viene associato un valore del target così che quando arriva un nuovo elemento, in base alla regione dove esso cade ne possiamo determinare il valore del target associato.
Partizioniamo quindi lo spazio del dominio D in regioni, è solo un modo diverso di vedere la cosa.

- Nel caso in cui si utilizzi un approccio probabilistico, semrpe in merito all'esempio di sesso vs (altezza,peso): questa volta l'algoritmo restituirà, se la persona è alta 1.68 e pesa 57kg, una distribuzione di probabilità secondo cui con il 73% è femmina e col (100-73)% è maschio.
Questo caso è molto più informativo del primo, in quanto permette anche di stimare meglio l'affidabilità che è legata in qualche modo alla varianza della distribuzione. Inoltre, se il sistema da una distribuzione di probabilità dei possibili valori target, per poter dire Maschio/Femmina definitivamente serve un metodo che stabilisca il valore target da assegnare (serve quindi un passo in più).
Magari non conviene semplicemente assegnare in base alla probabilità più alta, ad esempio quando gli errori non pesano allo stesso modo: se prendiamo l'esempio del covid, se il sistema è probabilistico ci saranno degli errori che sono falsi positivi ed altri che sono falsi negativi. In questo contesto i due errori non pesano allo stesso modo. C'è quindi un discorso che ha quindi a che fare con la gestione del rischio, che rende possibile l'attuazione di politiche ulteriori e quindi rende questo approccio più informativo del primo.

Types of problems

Unsupervised learning

- We wish to extract, from a given collection of items dataset) $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, with no target associated, some synthetic information, such as:
  - subsets of similar items clustering)
  - the distribution of items in their domain (density estimation)
  - the projection, as informative as possible, of items on lower dimensional subspaces, that is, their characterization by means of a smaller set of features feature selection, feature extraction)
- A suitable model, of just the data features, is usually defined and applied also in the case of unsupervised learning.

Reinforcement learning

- We want to identify, in a given framework, a sequence of actions to be performed in order to maximize a certain profit
- As in supervised learning, no examples are given, but an environment is available which returns a profit in correspondance to the execution of any action

- Per l'unsupervised learning, possiamo andare ad identificare differenti aspetti inferiti dall'osservazione del dataset:
  - il fatto che gli elementi possono raggrupparsi in gruppi abbastanza separati fra loro. Possiamo osservare la tendenza a raggrupparsi in così detti cluster. Un analisi che cerca di capire se avviene tale raggruppamento è proprio il clustering: un esempio è cercare di raggruppare i dati in un certo numero di cluster, vedere poi in quanti cluster sta uno stesso elemento, l'idea è dire che gli elementi si raggruppano in qualche modo per via di una certa variabile che non vediamo e determina tale raggruppamento.

  - Possiamo osservare che gli elementi non sono completamente sparsi su tutto il dominio, ma sono introno ad una retta (ad esempio). Quindi i valori delle feature stanno più o meno intorno ad uno spazio di dimensione 1. C'è quindi una regolarità in quanto i valori delle feature non sono totalmente indipendenti fra loro, la variabile è quindi, in questo caso, una sola e che determina la posizione del punto sulla retta
    L'idea è che le feature con cui rappresentiamo gli elementi non sono fra loro indipendenti, quindi in realtà i gradi di libertà veri sono di meno rispetto al numero di feature considerate. Potremo avere una situazione in cui i punti non sono proprio sulla retta ma intorno ad essa, il che vuol dire che le feature dipendono anche da altro e non da una sola variabile. Questo si chiama feature selection o feature extraction.
    Caso limite: dei punti che hanno tutti lo stesso valore della y e diversi della x, andiamo quindi ad estrarre un sotto insieme dalle feature ottenendo dei vettori di dimensione d'< d.

    .

  - Selezione di outlayers: un outlayer è un valore molto distante da tutti gli altri, ad esempio quando si fa analisi di log cerchiamo di trovare outlayers.
    un nero in Cina e misuri il suo bel cazzone e quello degli altri (l'ha scritto Gian Marco).
    Definiamo una distribuzione di probabilità sui dati, per poi verificare la probabilità di ciascun elemento, se tale probabilità è bassa vuol dire che il punto è "strano".

Machine learning framework: domains

La dimensione è la grandezza del vettore x.

Domain set $\mathcal{X}$: Set of objects we may wish to label. Each object is modeled as a vector of features. The number of features is the dimensionality of the problem

Label set $\mathcal{Y}$: Set of possible label values associated to objects in $\mathcal{X}$.

- $\mathcal{Y}$ continuous: regression
- $\mathcal{Y}$ discrete: classification

Training set $\mathcal{T}$: A set of object-label pairs: $\mathcal{T} = \{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_n, t_n)\}$. We shall usually denote as $\mathbf{X}$ the matrix of objects (feature matrix), that is

$$\mathbf{X} = \begin{pmatrix} - & \mathbf{x}_1 & - \\ & \vdots & \\ - & \mathbf{x}_n & - \end{pmatrix}$$

rappresentano un campione sul dominio dei possibili valori.

and as $\mathbf{t}$ the vector of labels (target vector), that is

$$\mathbf{t} = \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix}$$

Machine learning framework: learner output

Learner output: The learner (an algorithm A) is requested to return, for a given training set $\mathcal{T}$, a prediction rule (classifier, regressor) $A(\mathcal{T}) = h : \mathcal{X} \mapsto \mathcal{Y}$

Training objects generation model: We assume that the objects observed in the training set are sampled from $\mathscr{X}$ according to some probability distribution $\mathscr{D}_1$. That is, for any $\mathbf{x} \in \mathscr{X}$, $p_{\mathscr{D}_1}(\mathbf{x})$ is the probability that $\mathbf{x}$ is the next object sampled in the training set

Training targets generation model: In the general case, we assume the labels associated to the objects in the training set are generated according to a probability distribution $\mathscr{D}_2$ conditional on $\mathscr{X}$. That is, for any $t \in \mathscr{Y}$, $p_{\mathscr{D}_2}(t|\mathbf{x})$ is the probability that the observed label of object $\mathbf{x}$ in the training set is t. For the moment, we shall assume that the relation between object and label is deterministic, that is it exists an unknown function f such that $t_i = f(\mathbf{x}_i)$

- L'estrazione dei vettori x_i sarà avvenuta usando una qualche distribuzione di probabilità che non conosciamo, che è la distribuzione degli elementi ed è come aver preso a caso gli elementi secondo tale distribuzione, quindi la probabilità di prendere l'elemento è proprio secondo quella della distribuzione:
$\forall x, p_{D_1}(x)$ è la probabilità di estrarre x data la distribuzione D_1
(ovvero la probabilità che il prossimo elemento campionato nel training set sia proprio x).

- Per la label: supponiamo di aver estratto x secondo D_1, allora il target relativo verrà estratto da un urna dove ci sono tutti i valori target, con una distribuzione di probabilità D_2 che sarà però condizionata ad x, quindi $p_{D_2}(t|\mathbf{x})$, questo per ciascuna $t \in Y$.

Machine learning framework: prediction risk

Given any element $\mathbf{x} \in \mathcal{X}$:

Error: The error of a predictor h derives from the comparison of its prediction $h(\mathbf{x})$ and the correct target label y.

Loss: The comparison is performed by applying a predefined loss function $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbf{R}$.

Risk of prediction: The error of a prediction $\hat{y}$ is defined in terms of prediction risk as given by applying the loss

$$\mathscr{R}(\hat{y}, y) = L(h(\mathbf{x}), y)$$

In the general case when only a probabilistic relation $p_{\mathscr{D}_2}(y|\mathbf{x})$ is assumed between label and target, this corresponds to

$$\mathscr{R}(\hat{y}, \mathbf{x}) = E_{\mathscr{D}_2}[L(\hat{y}, y)] = \int_{\mathcal{Y}} L(\hat{y}, y) \cdot p_{\mathscr{D}_2}(y|\mathbf{x}) dy$$

or, in the case of classification

$$\mathscr{R}(\hat{y}, \mathbf{x}) = E_{\mathscr{D}_2}[L(\hat{y}, y)] = \sum_{y \in \mathcal{Y}} L(\hat{y}, y) \cdot p_{\mathscr{D}_2}(y|\mathbf{x})$$

- Assumiamo di avere D_1 , D_2 che è condizionata, di aver preso le feature x da D_1 e per ognuno di essi di aver estratto la label da D_2, quindi per il momento supponiamo che ci sia una funzione incognita $f : t_i = f(\mathbf{x}_i)$.

Supponiamo di avere $x \in X$, questo x ha una sula label corretta che chiamiamo y. Supponiamo di fissare un predittore, ovvero (in questo caso) una funzione h che preso x fornisce una previsione. Possiamo computare $h(x)$ e confrontare il valore predetto con quello corretto e stabilire l'errore che stiamo commettendo (ovvero un NUMERELLO). Possiamo fare questo confronto nel training set, per stabilire quanto stiamo sbagliando usiamo una funzione predefinita che è la funzione di costo.

- più sarà elevato il valore dato dalla funzione loss e più stiamo sbagliando, consideriamo tutto come funzione di h : fissato il punto, cambiando h otteniamo una previsione diversa e quindi un valore diverso di loss per cui voglio scegliere la h tale per cui il valore di errore sia il più piccolo possibile. Il punto rimane fisso, vario h e scelgo quella con loss migliore e questa è l'ottimizzazione. Il rischio della predizione è il valore della funzione loss, ovvero il rischio che ci si assume nel fare la predizione $h(x_i)$ invece di y. Il problema è trovare, di tutte le funzioni di predizione quella "buona" e non basta considerare solo un punto, occorre sceglierne una che tenda a comportarsi bene per tutti i punti, ovvero una per cui la media del rischio è più bassa possibile, in modo che se estraggo punti a caso e la media è la più bassa, la funzione può essere quella buona.

In this framework, the optimal prediction is the one which minimizes the risk,

$$y^*(\mathbf{x}) = \underset{\hat{y}}{\operatorname{argmin}}\, \mathscr{R}(\hat{y}, \mathbf{x}) = \underset{\hat{y}}{\operatorname{argmin}}\, E_{\mathscr{D}_2}[L(\hat{y}, y)]$$

that is,

$$y^*(\mathbf{x}) = \underset{\hat{y}}{\operatorname{argmin}}\, L(\hat{y}, f(\mathbf{x})) \qquad \text{in the simpler case}$$

$$y^*(\mathbf{x}) = \underset{\hat{y}}{\operatorname{argmin}}\, E_{\mathscr{D}_2}[L(\hat{y}, y)] = \underset{\hat{y}}{\operatorname{argmin}} \int_{\mathscr{Y}} L(\hat{y}, y) \cdot p_{\mathscr{D}_2}(y|\mathbf{x}) dy \qquad \text{in the general case}$$

in the general case, this is denoted as Bayes estimator.

However, observe that this approach cannot be applied since both the function f and the distribution $\mathscr{D}_2$ of $p(y|\mathbf{x})$ are assumed unknown.

The error of a predictor h is defined in terms of risk expected loss on all objects in $\mathcal{X}$

$$\mathcal{R}(h) = E_{\mathcal{D}_1, f}[L(h(\mathbf{x}), f(\mathbf{x}))] = \int_{\mathcal{X}} L(h(\mathbf{x}), f(\mathbf{x})) \cdot p_{\mathcal{D}_1}(\mathbf{x}) d\mathbf{x}$$

In the general case,

$$\mathcal{R}(h) = E_{\mathcal{D}_1, \mathcal{D}_2}[L(h(\mathbf{x}), y)] = \int_{\mathcal{X}} \int_{\mathcal{Y}} L(h(\mathbf{x}), y) \cdot p_{\mathcal{D}_1}(\mathbf{x}) \cdot p_{\mathcal{D}_2}(y|\mathbf{x}) d\mathbf{x} dy$$

Since $\mathscr{D}_1$ and $\mathscr{D}_2$ (or f) are not known, the risk can only be estimated from the data available (the training set $\mathscr{T}$).

Empirical risk: The risk can be estimated from the training set by estimating the expectation of the loss function as the average loss on the set.

$$\overline{\mathscr{R}}_{\mathscr{T}}(h) = \frac{1}{|\mathscr{T}|} \sum_{(x,t)\in\mathscr{T}} L(h(x), t)$$

Machine learning framework: from learning to optimization

The fundamental approach in machine learning is deriving a predictor h which (at least approximately) minimizes the empirical risk computed on the available training set.

A learning problem is then reduced to a minimization problem in some functional space $\mathscr{H}$, the set of all possible predictors h.

$$h^* = \operatorname*{argmin}_{h \in \mathscr{H}} \overline{\mathscr{R}}_{\mathscr{T}}(h)$$

Here, $\mathscr{H}$ is the set of hypotheses or inductive bias

Issues related to the inductive bias

The choice of the set of hypotheses is an important issue in ML:

- what is the effect of the structure and size of $\mathscr{H}$?
- how to define $\mathscr{H}$ in such a way to make it feasible to compute h^*?

Choice of $\mathscr{H}$

- The hypotheses class $\mathscr{H}$ can be viewed as reflecting some prior knowledge that the learner has about the task
  - a belief that one of the members of the class $\mathscr{H}$ is a low-error model for the task
- A trivial way of pursuing this goal would be to define a very rich class, that is assuming that many possible functions belong to $\mathscr{H}$
- As a limit, $\mathscr{H}$ could be defined just as the set of all functions $f : \mathcal{X} \mapsto \mathcal{Y}$

- Ci sono una serie di considerazioni di carattere teorico, evidentemente la classe delle ipotesi H verrà presa per qualche motivo relativo magari a qualche conoscenza pregressa che dice quale insieme H usare, ma in realtà potremmo anche andare per tentativi: cerchiamo la migliore funzione in un certo contesto, magari poi cambiamo H e cerchiamo in un altro contesto e così via … per poi confrontare. Nei casi più diffusi, dopo avere definito H, individuare la h^* è qualcosa che avviene in maniera algoritmica (ci son librerie che lo fanno da un punto di vista sperimentale), ma se vogliamo applicare metodi di ML ciò che avverrà è che cambieremo H, dove questo può voler dire cambiare proprio la definizione delle funzioni, la loro struttura, oppure degli aspetti parametrici delle funzioni.

Ad esempio, per una regressione, potremmo considerare tutte le funzioni da $\mathbb{R} \to \mathbb{R}$, quindi potremmo:

- considerare solo polinomi di grado 1
- solo polinomi di grado 2, quindi funzioni della stessa classe, ma che variano per un parametro che caratterizzano la classe stessa
- usare funzioni trigonometriche, cambiando totalmente la struttura quindi

Scegliendo una certa H, ci si aspetta che lì ci sia un predittore che si comporta bene e allora la cosa migliore che si possa fare sembrerebbe essere definire l'insieme più ricco possibile, al limite prendere tutte le funzioni $f : Y \to X$.

Problem with large $\mathscr{H}$:

- Assume a binary classification problem with training set $\mathscr{T} = (\mathbf{X}, \mathbf{t})$, with $0/1$ loss

$$L(y,t) = \begin{cases} 0 & \text{if } y = t \\ 1 & \text{otherwise} \end{cases}$$

  that is, the loss is 1 if the item is misclassified, 0 otherwise. As a consequence, the risk is the expected number of classification errors, while the empirical risk is the fraction of items in the training set which are misclassified.
- Assume $p(t = 1|\mathbf{x}) = \frac{1}{2}$ for $\mathbf{x} \in \mathscr{X}$, that is, the two classes have same size in the population

Consider the classification function defined as:

$$h(x) = \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{x}_i \in \mathbf{X}, t_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

In loss $L(y,t) = \begin{cases} 0 & \text{se } y = t \\ 1 & \text{altr.} \end{cases}$ Sono quindi:

that is, h assigns to class 1 all items labeled as 1 in the training set. All other items are classified as 0.

Clearly, the empirical risk here is 0 by definition, but the risk is $\approx \frac{1}{2}$. When applied to a dataset randomly sampled from the population, the quality of h^* is the same of a function which randomly assigns items to classes.

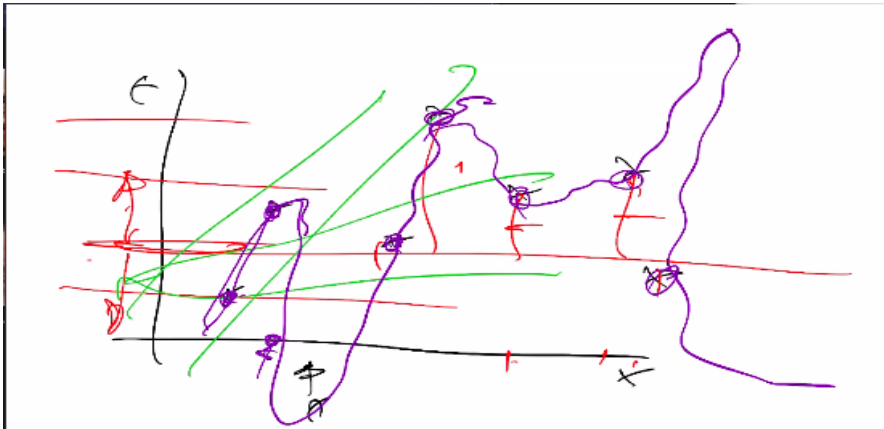
This is called overfitting: the classification method behaves well on the training set, but poorly on new data from the population.

→ ma ci bastiamo la questa!

- Il problema del predittore di sopra è quest'ultimo è troppo specifico per il training set, è stato costruito per rispondere bene sul training set. Potremmo quindi dire di rispondere sempre a caso, ma anche qui va male perché l'errore è sempre $\approx \frac{1}{2}$, non stiamo tenendo conto del training set questa volta.
Quindi non possiamo non tenere conto del training set ma nemmeno tenerne conto troppo, ad esempio nel secono caso manca la generalizzazione, ovvero considerare che il resto del mondo non è uguale a ciò che il classificatore sa già. Nel ML occorre tenere conto dei dati ma non tenerne troppo conto:
  - il fenomeno per cui il predittore si comporta troppo meglio su H che sui dati è l'overfitting.
    Il predittore predice molto bene cosa sa e molto male ciò che non sa;
  - nel caso opposto, ovvero quando si tiene poco conto dei dati, presumibilmente il predittore predirrà male tutto e si ha il fenomeno dell'underfitting.

- La funzione così definita non va bene perché la libertà su h è eccessiva, quindi "ci suggerisce" che se lo spazio delle ipotesi è molto vasto, troppo vasto allora si troverà una funzione che predice perfettamente il training set ma magari in questo modo è troppo specializzata e non riesce a predirre il resto. Ma se lo spazio delle ipotesi è troppo piccolo, magari la migliore funzione che tiro fuori predice male.

- esempio: vogliamo fare regressione, supponiamo che gli elementi siano questi:



se l'errore fosse la somma delle distanze, allora questa cambierà ma saremmo lì. È quindi una classe ristretta di funzioni, ovvero tutte le costanti. Possiamo passare a tutte le rette, la migliore retta sarà meglio della migliore costante (regressione lineare). Ancora meglio, le curve quadratiche: le distanze diminuiscono, perché abbiamo più gradi di libertà. I 7 punti, preso un polinomio di grado 6 li copro tutti e quindi il miglior polinomio di grado 6 è unico e passerà per tutti i punti ma avrà un andamento strano, ovvero dovrà molto "aggiustarsi" ma allora per un punto che non fa parte del training set è molto sballata. Con le rette siamo in underfitting, con i polinomi di "grado pari a" siamo in overfitting.

Choice of $\mathscr{H}$

With respect to $\mathscr{H}$, the following considerations can be done:

- If $\mathscr{H}$ is too large (complex), overfitting may occur: a function which behaves very well on the training set may be available which however performs poorly on new data
- If $\mathscr{H}$ is too small (simple), underfitting may occur: no function behaving in a satisfactory way, both on the training set and on new sets of data, is available in $\mathscr{H}$

This is related to the so-called bias variance tradeoff

Bias vs variance

The risk associated to the h^*, the predictor which minimizes the empirical risk, can be decomposed in two parts:

$$\mathscr{R}(h^*) = \epsilon_B + \epsilon_V$$

where:

- ϵ_B is the minimum risk achievable by any $h \in \mathscr{H}$: this is only determined by the inductive bias, and independent from the training set. It is a property of the class of hypotheses considered with respect to the prediction task. This is called bias
- ϵ_V is the difference between the above minimum risk in $\mathscr{H}$ and the risk associated to the best predictor in $\mathscr{H}$ with respect to the training set: it is related to the fact that empirical risk minimization only provides an estimate of the best predictor achievable for the given inductive bias. It is a measure of how well the predictor computed from a particular training set approximates the best possible one. Its expectation with respect to all possible training sets is a measure of how much a predictor derived from a random training set may result in poorer performances with respect to the best possible one. This is called variance

Bias vs variance

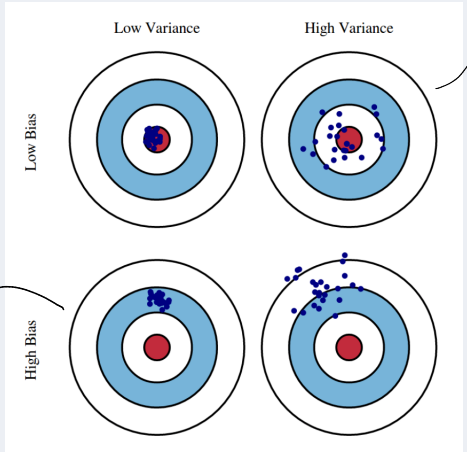
The choice of $\mathscr{H}$ is subject to a bias-variance tradeoff: higher bias tend to induce lower variance, and vice versa.

- High bias and low variance implies that all predictors which can be obtained from different training sets tend to behave similarly, with a similar risk (low variance). However, all of them then to behave poorly (high bias), since $\mathscr{H}$ is too poor to include a satisfactory predictor for the task considered. This results into underfitting
- Low bias and high variance implies that lot of predictors are available in $\mathscr{H}$, and among them a good one is usually avaliable (low bias). However, quite different predictors can be obtained from different training sets, which implies that it may easily happen that, while a very good performance can be obtained on the training set, the resulting predictor can behave quite differently and more poorly that the best possible one, which implies overfitting

- C'è quindi il trade-off fra bias e varianza: se dobbiamo effettuare apprendimento, definiamo un insieme di funzioni e poi deriviamo un training set a caso dalla popolazione e guardando ad esso cerchiamo di trovare la migliore funzione.Cosa può accadere:
  - caso banale, ho una sola funzione. Risponde sempre 0, il predittore migliore della classe (che è lui perché è da solo) predice sempre allo stesso modo e quindi questo è indipendente dal training set per quella classe. È un caso estremo, ma al variare del training set la migliore funzione può cambiare, nel caso delle rette orizzontali ne possiamo ottenere una diversa ma la qualità della predizione non sarà molto diversa. Quindi più o meno tutti i possibili predittori sbagliano un po' tutti allo stesso modo, il predittore può anche variare un po' ma lo sbaglio commesso sarà sempre molto simile e parliamo quindi di bias.
  - il caso opposto è quello in cui ho un polinomio di grado elevato, quindi il predittore scelto è fortemente dipendente dal training set. In caso di overfitting l'algoritmo di Machine Learning fornisce un predittore che cambia molto al variare del training set e parlo quindi di varianza. Ho sempre un insieme di predittori ampio in cui il predittore, per ogni nuovo training set, predice quest'ultimo molto bene ma cambia sempre la variare del training set.

- Queste due componenti ci sono sempre: il bias, che è l'errore sistematico ovvero quanto qualunque predittore che possiamo trovare sbaglia e la varianza, ovvero quanta differenza di prestazioni c'è fra due predittori derivati però da due training set diversi.

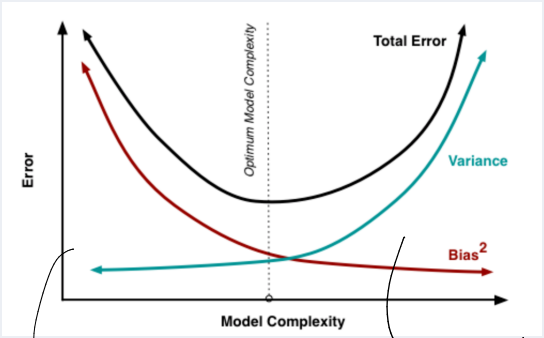
Bias vs variance



nel complesso, la freccia prende il centro ⇒ bias basso

In ML indica poca dipendenza dal training set.

Bias vs variance



caso di underfitting

aumenta la dimensione di H.

Computing h^*

- The optimization required to derive h^* can be complex in the general case, when a function must be derived in a function space.
- Usually, the situation is made easier by considering $\mathscr{H}$ as a space of functions parameterized by a suitable set of coefficients (for example, all polynomials of degree at most d, for a given d): this results in a minimization to be performed over a set of d-dimensional points.
- That is, $\mathscr{H} = \{h_\theta | \theta \in \Theta\}$, where Θ is the coefficients domain and h is a function template, parameterized by elements in Θ
- Minimizing the Empirical risk results into computing

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} \overline{\mathscr{R}}_{\mathcal{T}}(h_\theta)$$

es: per le rette: $y = w_1 x + w_0$
i parametri sono w_0 e w_1 ($d = 2$). Ogni funzione è, in generale un punto in uno spazio a k dimensioni.

Il rischio associato ad ogni punto sarà: $\mathscr{R} : \mathbb{R}^k \to \mathbb{R}$

Computing h^*

- In most cases, $\Theta = \mathbb{R}^d$ for some $d > 0$: in this case, the minimization of $\overline{\mathcal{R}}_{\mathcal{T}}(h_\theta)$ is unconstrained and a (at least local) minimum could be computed setting all partial derivatives to 0

$$\frac{\partial}{\partial \theta_i} \overline{\mathcal{R}}_{\mathcal{T}}(h_\theta) = 0$$

  that is, setting to zero the gradient of the empirical risk with respect to the vector of parameters θ

$$\nabla_\theta \overline{\mathcal{R}}_{\mathcal{T}}(h_\theta) = \mathbf{0}$$

- The analytical solution of this set of equations is usually quite hard
- Numerical methods can be applied

- Abbiamo ancora il problema di dover minimizzare una funzione di k variabili: per farlo, si va a derivare ma abbiamo dei problemi:
  - i punti in cui la derivata è nulla possono essere flessi, max o di min;
  - inoltre, non è detto che otteniamo il minimo assoluto

Possiamo quindi fare la derivata parziale ed annullarla, magari useremo funzioni che per come sono fatte sappiamo che lì dove è nulla la derivata troviamo un minimo. A k variabili annulliamo le k derivate parziali, (ovvero il gradiente) abbiamo quindi un sistema kxk. Il sistema si risolve se lineare e questo dipende dalla funzione di partenza, ma l'approccio basato sull'analisi non si applica nella maggior parte dei casi (e per fortuna aggiungerei), allora si può procedere con i metodi numerici che è quello che tipicamente si applica in ML: si cerca di trovare un minimo locale nella funzione mediante un metodo iterativo.

Gradient descent

- Gradient descent performs minimization of a function $J(\theta)$ through iterative updates of the current value of θ (starting from an initial value $\theta^{(0)}$) in the opposite direction to the one specified by the current value of the gradient $J'(\theta) = \nabla_\theta J(\theta)$

$$\theta^{(k+1)} = \theta^{(k)} - \eta \nabla_\theta J(\theta)\,|_{\theta=\theta^{(k)}}$$

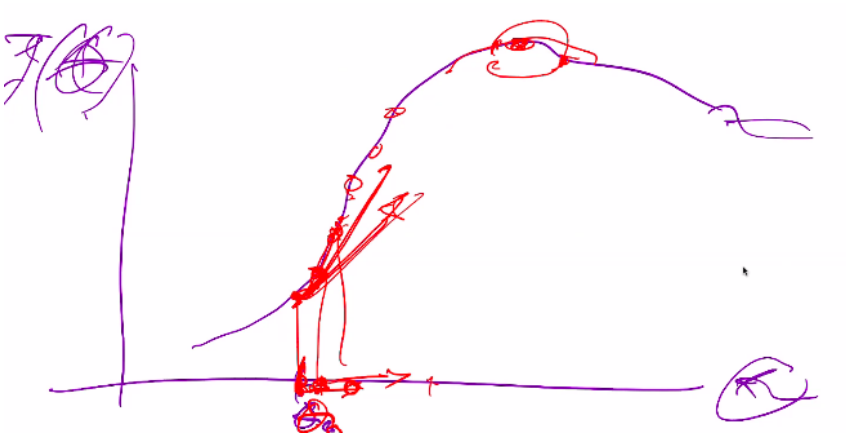
in quanto stiamo minimizzando

  that is, for each parameter θ_i

$$\theta^{(k+1)} = \theta^{(k)} - \eta \frac{\partial J(\theta)}{\partial \theta_i}|_{\theta=\theta^{(k)}}$$

- η is a tunable parameter, which controls the amount of update performed at each step

→ identifica il passo con cui si aggiorna.



- Partiamo da $\theta^{(0)}$, $\theta^{(1)}$ sarà dato da $\theta^{(0)}$ + il valore della derivata.
Se la derivata è elevata, quindi se c'è pendenza si farà un passo più lungo, altrimenti si avanzerà a passi più corti. Arrivati ad un punto di massimo la derivata è 0, il punto successivo è uguale al precedente e ci si ferma.

- Per il minimo, vale lo stesso ragionamento ma si sottrae.

In Machine learning, minimization of the Empirical Risk is performed, hence gradient descent takes the form

$$\theta_i^{(k+1)} = \theta_i^{(k)} - \eta \frac{\partial}{\partial \theta_i} \frac{1}{|\mathcal{T}|} \sum_{(x,t)\in\mathcal{T}} L(h_\theta(x), t)|_{\theta=\theta^{(k)}}$$

$$= \theta_i^{(k)} - \frac{\eta}{|\mathcal{T}|} \sum_{(x,t)\in\mathcal{T}} \frac{\partial}{\partial \theta_i} L(h_\theta(x), t)|_{\theta=\theta^{(k)}}$$

This is called batch gradient descent: observe that, at each step, all items in the training set must be considered

↳ se il training set è grande, non scala

Stochastic gradient descent

Batch gradient descent can be modified by performing the update, at each step, on the basis of the evaluation at a single item of the training set for single parameters,

$$\theta_i^{(k+1)} = \theta_i^{(k)} - \eta \frac{\partial}{\partial \theta_i} L(h_\theta(x_j), t_j)|_{\theta=\theta^{(k)}}$$

Mini-batch gradient descent

An intermediate case is the one when a subset of the items in the training is considered at each step

$$\theta_i^{(k+1)} = \theta_i^{(k)} - \frac{\eta}{|B_r|} \sum_{(x,t)\in B_r} \frac{\partial}{\partial \theta_i} L(h_\theta(x), t)|_{\theta=\theta^{(k)}}$$

This is called mini-batch gradient descent

Calcoliamo ogni volta considerando un sottoinsieme degli elementi del training set.

Probabilistic approaches

As done before, we assume that the observed dataset (features and target) has been derived by randomly sampling:

- $\mathscr{X}$ according to the probability distribution $p_{\mathscr{D}_1}(x)$ (usually the uniform distribution)
- $\mathscr{Y}$ according to the conditional distribution $p_{\mathscr{D}_2}(y|x)$

1. we may then consider a class of possible conditional distributions $\mathscr{P}$ and
2. select (infer) the "best" conditional distribution $p^* \in \mathscr{P}$ from the available knowledge (that is, the dataset), according to some measure q
3. given any new item x, apply $p^*(y|\mathbf{x})$ to assign probabilities for each possible value of the corresponding target
4. an independent decision strategy must be applied to $p^*(y|\mathbf{x})$ to return a specific prediction $h(\mathbf{x})$

- In molti casi vogliamo un metodo che ci dia una distribuzione per ogni valore del target, ovvero per ogni valore avere delle probabilità. Quindi non vogliamo una $h(x)$ ma una $p(t|x)$, questi sono approcci che danno più informazione ed a cui possiamo poi attaccare delle regole di decisione a posteriori.
L'idea è più o meno la stessa, quindi consideriamo una classe di possibili distribuzioni condizionali e cerchiamo di capire qual è la migliore secondo una misura. A questo punto, trovata la p^*, dato un nuovo elemento x calcoliamo $p^*(y|\mathbf{x})$. Come per le funzioni, dobbiamo definire una classe delle possibili distribuzioni condizionate ed i problemi sono gli stessi di prima:
  - come definire la classe;
  - come definire la misura della qualità della distribuzione.

Potremmo pensare di effettuare la predizione magari mettendo insieme le predizioni di più predittori, ad esempio quelle di tutti quelli possibili. Servirà quindi un modo per comporre le risposte e per trovarne una singola: è l'approccio che viene chiamato ensamble, notare che per fare le cose correttamente occorrerebbe pesare la predizione fatta da ogni predittore sulla base dell'affidabilità fatta da ognuno di essi.
C'è quindi un aspetto di composizione delle predizioni fatte da ognuno dei singoli modelli, ognuna pesata dalla qualità predittiva sempre pesata in base al training set. Questo approccio segue la stessa idea, anche se sviluppata in maniera diversa, ovvero dell'apprendimento Bayesiano ma con una formulazione matematica più elegante.

Inferring a best distribution

- how to define the class of possible conditional distributions $p(y|\mathbf{x})$?
  - usually, parametric approach: distributions defined by a common (arbitrary) structure and a set of parameters
- what is a measure $q(p, \mathcal{T})$ of the quality of the distribution (given the dataset $\mathcal{T} = (\mathbf{X}, \mathbf{t})$)?
  - this is related to how a dataset generated by randomly sampling from $\mathcal{D}_1$ (usually uniform) and $\mathcal{D}_2$ could be similar to the available dataset $\mathcal{T}$

- q è simile al concetto della verosim empirica.
- le distribuzioni p saranno pesate con q, ottenendo quindi un prodotto di probabilità:
  $q(p, \mathcal{T}) \cdot p(y|x)$ => approccio Bayesiano.

A different approach

Instead of finding a best distribution $p^* \in \mathscr{P}$ and use it to predict target probabilities as $p^*(y|\mathbf{x})$ for any element $\mathbf{x}$, we could

- consider for each possible conditional distribution $p \in \mathscr{P}$ its quality $q(p, \mathscr{T})$
- compose all conditional distributions $p(y|\mathbf{x})$ each weighted by its quality $q(p, \mathscr{T})$ (for example by means of a weighted averaging)
- apply the resulting distribution

Different strategies

Assume q takes the form of a probability distribution (of probability distribution)

- first approach: take the modal value (the distribution of maximum quality) and apply it to perform predictions
- second approach: compute the expectation of the distributions, wrt the probability distribution q