

Fondamenti

■ Dati e Pattern

- Numerici, Categorici, Sequenze

■ Problemi di Learning

- Classificazione
- Regressione
- Clustering
- Riduzione Dimensionalità
- Representation Learning

■ Tipi di Learning

- Supervisionato, Non supervisionato
- Batch, Incrementale, Naturale
- Reinforcement Learning

■ Training e Valutazione Prestazioni

- Funzione Obiettivo, Parametri, Iperparametri
- Misura delle Prestazioni
- Training, Validation, Test
- Convergenza, Generalizzazione e Overfitting

Dati e Pattern

- I dati sono un ingrediente fondamentale del machine learning, dove il comportamento degli algoritmi **non è pre-programmato** ma **appreso dai dati stessi**.
- Termini come **Data Science**, **Data Mining**, **Big Data** enfatizzano il ruolo dei dati.
- Utilizzeremo spesso il termine **Pattern** per riferirci ai dati
 - Pattern può essere tradotto in italiano in vari modi: *forma*, *campione*, *esempio*, *modello*, ecc. [meglio non tradurlo].
 - S. Watanabe definisce un pattern come l'opposto del caos e come un'entità vagamente definita cui può essere dato un nome.
 - Ad esempio un pattern può essere un volto, un carattere scritto a mano, un'impronta digitale, un segnale sonoro, un frammento di testo, l'andamento di un titolo di borsa.
- **Pattern Recognition** è la disciplina che studia il riconoscimento dei pattern (non solo con tecniche di learning ma anche con algoritmi pre-programmati). L'intersezione con il Machine Learning è molto ampia.

Tipi di Pattern

- **Numerici**: valori associati a caratteristiche misurabili o conteggi.
 - Tipicamente continui (ma anche discreti, es. interi), in ogni caso soggetti a ordinamento.
 - Rappresentabili naturalmente come vettori numerici nello spazio multidimensionale.
 - L'estrazione di caratteristiche da segnali (es., immagini, suoni) produce vettori numerici detti anche **feature vectors**.
 - Es. Persona: [*altezza, circonferenza toracica, circonferenza fianchi, lunghezza del piede*]
 - Principale tipologia di dati considerata in questo corso.
- **Categorici**: valori associati a caratteristiche qualitative e alla presenza/assenza di una caratteristica (yes/no value).
 - Non «semanticamente» mappabili in valori numerici.
 - Es. Persona: [*sexo, maggiorenne, colore occhi, gruppo sanguigno*].
 - Talvolta soggetti a ordinamento (ordinali): es. temperatura ambiente: *alta, media o bassa*.
 - Normalmente gestiti da sistemi a regole e alberi di classificazione.
 - Molto utilizzati nell'ambito del **data mining**, spesso insieme a dati numerici (mixed).

Sequenze e altri dati strutturati

- **Sequenze:** pattern sequenziali con relazioni spaziali o temporali.
 - Es. uno **stream audio** (sequenza di suoni) corrispondente alla pronuncia di una parola, una **frase** (sequenza di parole) in linguaggio naturale, un **video** (sequenza di frame).
 - Spesso a lunghezza variabile
 - La posizione nella sequenza e le relazioni con predecessori e successori sono importanti.
 - Critico trattare sequenze come pattern numerici.
 - Allineamento spaziale/temporale, e «memoria» per tener conto del passato.
 - Approcci:
 - Dynamic Time Warping (**DTW**), Hidden Markov Models (**HMM**)
 - Recurrent Neural Networks (**RNN**), Long Short-Term Memory (**LSTM**)
- **Altri dati strutturati:** output organizzati in strutture complesse quali alberi e grafi.
 - Applicazioni in Bioinformatics, Natural Language Processing, Speech recognition, ecc.
 - Esempio nella traduzione di una frase in linguaggio naturale, l'output desiderato è l'insieme dei *parse tree* plausibili.
 - Approcci: Structured SVMs, Bayesian Networks, HMM

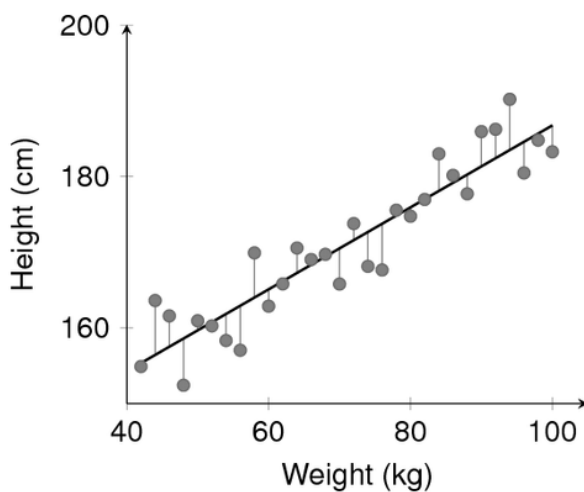
Classificazione

- **Classificazione:** assegna una **classe** a un pattern.
 - Necessario apprendere una funzione capace di eseguire il mapping dallo spazio dei pattern allo spazio delle classi
 - Si usa spesso anche il termine **riconoscimento**.
 - Nel caso di 2 sole classi si usa il termine **binary classification**, con più di due classi **multi-class classification**.
- **Classe:** insieme di pattern aventi proprietà comuni.
 - Es. i diversi modi in cui può essere scritto a mano libera il carattere **A**.
 - Il concetto di classe è semantico e dipende strettamente dall'applicazione:
 - 21 classi per il riconoscimento di lettere dell'alfabeto
 - 2 classi per distinguere le lettere dell'alfabeto italiano da quello cirillico
- **Esempi di problemi di classificazione:**
 - Spam detection
 - Credit Card fraud detection
 - Face recognition
 - Pedestrian classification
 - Medical diagnosis
 - Stock Trading

Quali sono i pattern e quali le classi ?

Regressione

- **Regressione:** assegna un **valore continuo** a un pattern.
- Utile per la predizione di valori continui.
- Risolvere un problema di regressione corrisponde ad apprendere una funzione approssimante delle coppie «input, output» date.



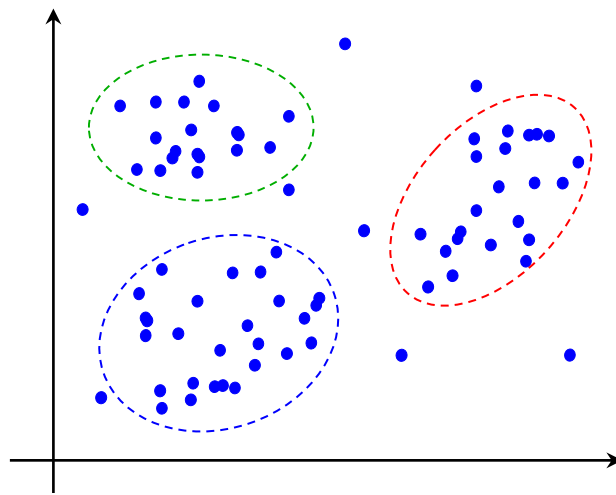
*Es. stima dell'altezza
di una persona in
base al peso*

Quale tipo di funzione? Con quale grado di regolarità?

- **Esempi di problemi di regressione:**
 - Stima prezzi vendita appartamenti nel mercato immobiliare
 - Stima del rischio per compagnie assicurative
 - Predizione energia prodotta da impianto fotovoltaico
 - Modelli sanitari di predizione dei costi
 - Object detection

Clustering

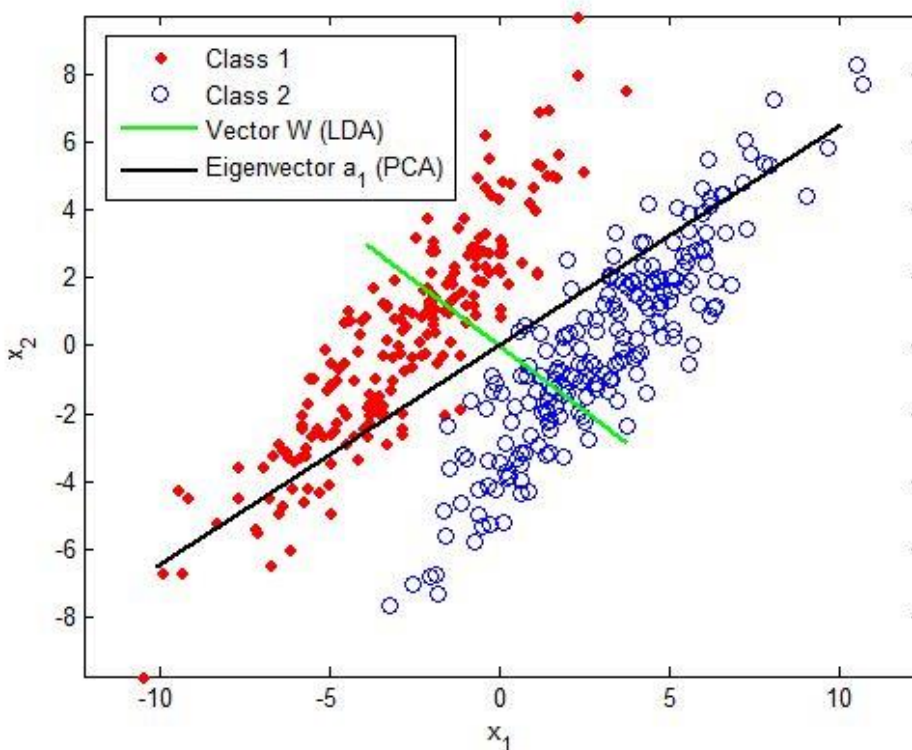
- **Clustering:** individua **gruppi** (cluster) di pattern con caratteristiche simili.
- Le classi del problema non sono note e i pattern non etichettati → la natura non supervisionata del problema lo rende più complesso della classificazione.
- Spesso nemmeno il numero di cluster è noto a priori
- I cluster individuati nell'apprendimento possono essere poi utilizzati come classi.



- **Esempi di problemi di clustering:**
 - Marketing: definizione di gruppi di utenti in base ai consumi
 - Genetica: raggruppamento individui sulla base analogie DNA
 - Bioinformatica: partizionamento geni in gruppi con simili caratteristiche
 - Visione: segmentazione immagini

Riduzione Dimensionalità

- **Riduzione di dimensionalità:** riduce il numero di dimensioni dei pattern in input.
- Consiste nell'apprendimento di un mapping da \mathbb{R}^d a \mathbb{R}^k (con $k < d$).
- L'operazione comporta una perdita di informazione. L'obiettivo è conservare le informazioni «**importanti**».
- La definizione formale di importanza dipende dall'applicazione.
- Molto utile per **rendere trattabili** problemi con dimensionalità molto elevata, per **scartare** informazioni ridondanti e/o instabili, e per **visualizzare** in 2D o 3D pattern con $d > 3$.

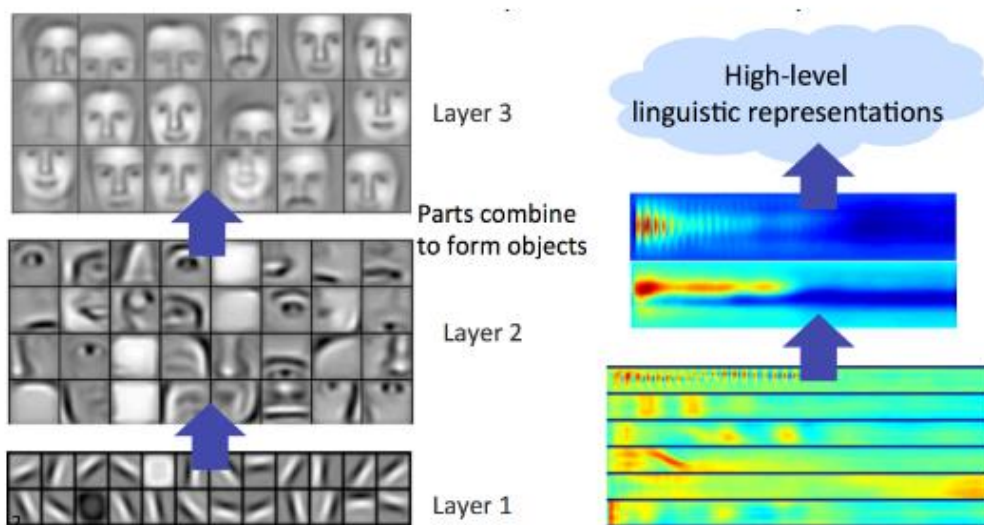


Representation Learning

- Il successo di molte applicazioni di machine learning dipende dall'efficacia di **rappresentazione** dei pattern in termini di **features**.
- La definizione di features ad-hoc (**hand-crafted**) per le diverse applicazioni prende il nome di **feature engineering**.
- *Ad esempio per il riconoscimento di oggetti esistono numerosi descrittori di forma, colore e tessitura che possiamo utilizzare per convertire immagini in vettori numerici.*

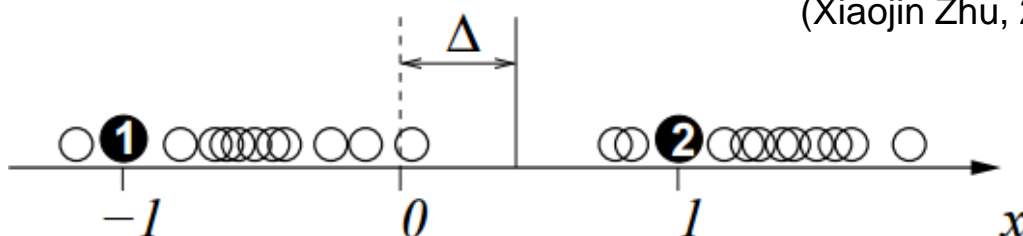
Representation Learning (o feature learning)

- Possiamo **apprendere** automaticamente feature efficaci a partire da **row data**? O analogamente, possiamo operare direttamente su row data (es. *intensità dei pixel di un'immagine, ampiezza di un segnale audio nel tempo*) senza utilizzare feature pre-definite ?
- Gran parte delle tecniche di **deep learning** (es. **convolutional neural networks**) operano in questo modo, utilizzando come input i raw data ed estraendo automaticamente da essi le feature necessarie per risolvere il problema di interesse.



Apprendimento

- **Supervisionato** (Supervised): sono note le classi dei pattern utilizzati per l'addestramento.
 - *il training set è etichettato.*
 - situazione tipica nella classificazione, regressione e in alcune tecniche di riduzione di dimensionalità (es. Linear Discriminant Analysis).
- **Non Supervisionato** (Unsupervised): non sono note le classi dei pattern utilizzati per l'addestramento.
 - *il training set non è etichettato.*
 - situazione tipica nel clustering e nella maggior parte di tecniche di riduzione di dimensionalità.
- **Semi-Supervisionato** (Semi-Supervised)
 - *il training set è etichettato parzialmente.*
 - la distribuzione dei pattern non etichettati può aiutare a ottimizzare la regola di classificazione.



(Xiaojin Zhu, 2007)

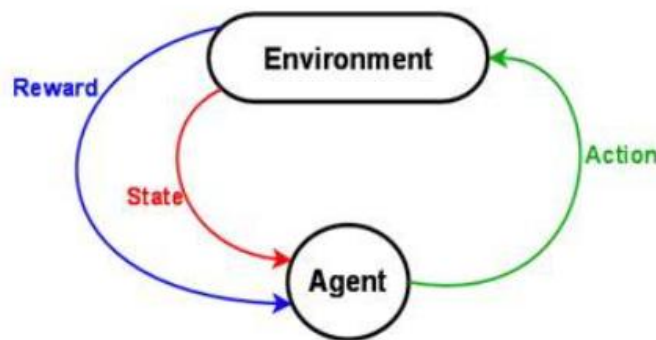
Batch, Incrementale, Naturale

- **Batch**: l'addestramento è effettuato una sola volta su un training set dato.
 - una volta terminato il training, il sistema passa in «working mode» e non è in grado di apprendere ulteriormente.
 - Attualmente, la maggior parte dei sistemi di machine learning opera in questo modo.
- **Incrementale**: a seguito dell'addestramento iniziale, sono possibili ulteriori sessioni di addestramento.
 - Scenari: Sequenze di Batch, Unsupervised Tuning.
 - Rischio: Catastrophic Forgetting (il sistema dimentica quello che ha appreso in precedenza).
- **Naturale**: addestramento continuo (per tutta la vita)
 - Addestramento attivo in working mode.
 - Coesistenza di approccio supervisionato e non supervisionato.

human-like learning involves an initial small amount of direct instruction (e.g. parental labeling of objects during childhood) combined with large amounts of subsequent unsupervised experience (e.g. self-interaction with objects)

Reinforcement Learning (RL)

- **Apprendere un comportamento:** l'obiettivo è apprendere un comportamento ottimale a partire dalle esperienze passate.
- un agente esegue **azioni** che modificano l'**ambiente**, provocando passaggi da uno **stato** all'altro. Quando l'agente ottiene risultati positivi riceve una ricompensa (**reward**) che però può essere temporalmente ritardata rispetto all'azione, o alla sequenza di azioni, che l'hanno determinata.
- Obiettivo è apprendere l'azione ottimale in ciascun stato, in modo da massimizzare la somma dei reward ottenuti nel lungo periodo.



- Nella pratica è molto difficile ottenere esempi che siano allo stesso tempo corretti e rappresentativi di tutte le situazioni in cui l'agente deve agire. Pertanto il classico approccio supervisionato non è facilmente applicabile.
- Numerose applicazioni in **Robotica** (es. object grasping, control, assembly, navigation).
- **Q learning** è uno degli approcci più noti e utilizzati.
- Una sua estensione (deep) denominata **Deep Reinforcement Learning** (DRL) è alla base dei successi ottenuti da Google DeepMind (Atari, AlphaGo).

Parametri e Funzione Obiettivo

- In genere, il comportamento di un algoritmo di Machine Learning è regolato da un set di **parametri** Θ (es. i pesi delle connessioni in una rete neurale). L'apprendimento consiste nel determinare il valore ottimo Θ^* di questi parametri.
- Dato un training set *Train* e un insieme di parametri, la **funzione obiettivo** $f(\text{Train}, \Theta)$ può indicare:

- l'**ottimalità** della soluzione (da **massimizzare**).

$$\Theta^* = \operatorname{argmax}_{\Theta} f(\text{Train}, \Theta)$$

- oppure l'**errore** o **perdita** (**loss-function**) da **minimizzare**.

$$\Theta^* = \operatorname{argmin}_{\Theta} f(\text{Train}, \Theta)$$

- $f(\text{Train}, \Theta)$ può essere ottimizzata:

- **esplicitamente**, con metodi che operano a partire dalla sua definizione matematica.

Es: si calcolano le derivate parziali di f rispetto ai parametri (gradiente), si eguaglia il gradiente a 0 (sistema di equazioni) e si risolve rispetto ai parametri.

- **implicitamente**, utilizzando euristici che modificano i parametri in modo coerente con f

Es: clustering con algoritmo k-means.

Iperparametri

- Molti algoritmi richiedono di definire, **prima** dell'apprendimento vero e proprio, il valore dei cosiddetti **iperparametri H**.
- Esempi di iperparametri:
 - Il numero di neuroni in una rete neurale.
 - Il numero di vicini k in un classificatore k -NN.
 - Il grado di un polinomio utilizzato in una regressione.
 - Il tipo di loss function.
- Si procede con un approccio a due livelli nel quale per ogni valore «**ragionevole**» degli iperparametri si esegue l'apprendimento, e al termine della procedura si scelgono gli iperparametri che hanno fornito **prestazioni migliori**.
- Ma come si **valutano le prestazioni**, e su **quali dati**?

Valutazione delle Prestazioni

- Un possibilità consiste nell'utilizzare direttamente la funzione obiettivo per quantificare le prestazioni. In genere però si preferisce una misura più diretta collegata alla semantica del problema.
- In un problema di classificazione, l'**accuratezza** di classificazione [0...100%] è la percentuale di pattern correttamente classificati. L'errore di classificazione è il complemento.

$$\text{accuratezza} = \frac{\text{pattern correttamente classificati}}{\text{pattern classificati}}$$

$$\text{errore} = 100\% - \text{accuratezza}$$

- Nel caso **più semplice** (e più comune nei benchmark di machine learning) si assume che il pattern da classificare appartenga a una delle classi note (**closed set**). Es. classificare le persone in {uomini, donne}.
- In molti casi reali invece i pattern da classificare possono appartenere a una delle classi note o a nessuna di queste (**open set**). Es. Classificare frutta in {mele, pere, banane}.

Due soluzioni:

- Si aggiunge alle classi un'ulteriore classe fittizia «**il resto del mondo**» e si aggiungono al training set i cosiddetti «**esempi negativi**».
- Si consente al sistema di non assegnare il pattern. A tal fine si definisce una **soglia** e si assegna il pattern alla classe più probabile solo quando la probabilità è superiore alla soglia.

Prestazioni nei problemi Open set

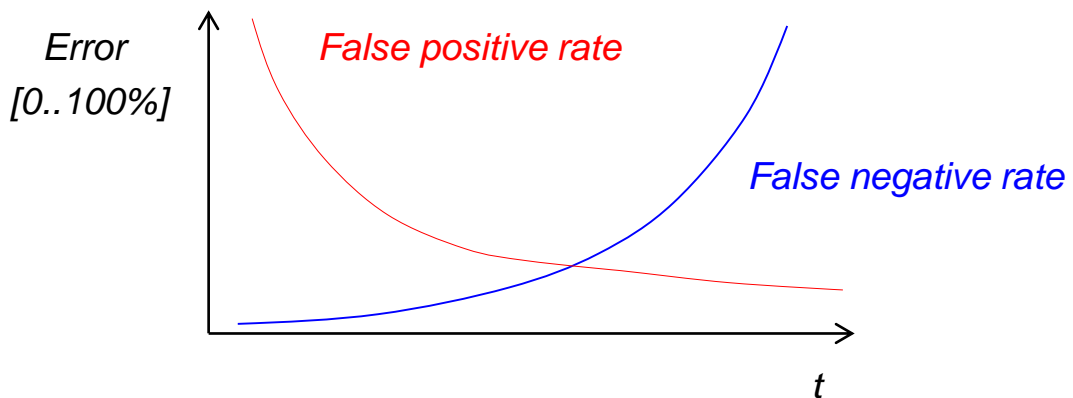
- Consideriamo un problema di classificazione **binario**, dove le due classi corrispondono a esempi **positivi** (vera classe) e **negativi** (classe fittizia resto del mondo).
 - *Es. Face detection. Esempi positivi: tutte le porzioni di immagine (finestre) in cui appaiono volti. Esempi negativi: finestre (scelte a caso) in cui non compaiono volti.*
- Analogamente possiamo considerare solo la classe **positivi** e un sistema (con soglia) in grado di calcolare la probabilità p di appartenenza di un pattern alla classe. Sia t il valore di soglia, allora il pattern viene classificato come positivo se $p > t$, come negativo in caso contrario.
- Dati N pattern da classificare, il risultato di ciascuno dei tentativi di classificazione può essere:
 - **True Positive (TP)**: un pattern positivo è stato correttamente assegnato ai positivi.
 - **True Negative (TN)**: un pattern negativo è stato correttamente assegnato ai negativi.
 - **False Positive (FP)**: un pattern negativo è stato erroneamente assegnato ai positivi. Detto anche errore di **Tipo I** o **False**.
 - **False Negative (FN)**: un pattern positivo è stato erroneamente assegnato ai negativi. Detto anche errore di **Tipo II** o **Miss**.
- Le frequenze dei due tipi di errore sono:

$$\text{False positive rate} = \frac{FP}{N}$$

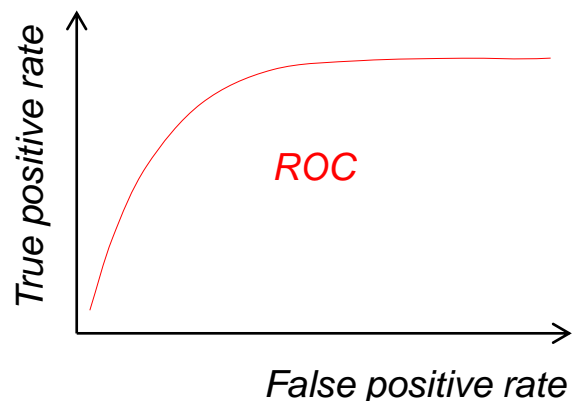
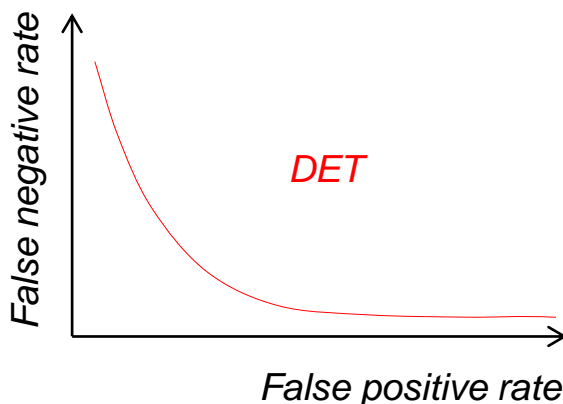
$$\text{False negative rate} = \frac{FN}{N}$$

Sistemi con soglia

- Nei sistemi con soglia false positive rate e false negative rate sono **funzione** della soglia t . Soglie restrittive (elevate) riducono i false positive a discapito dei false negative; viceversa soglie tolleranti (basse) riducono i false negative a discapito dei false positive.



- Le due curve possono essere «condensate» in una curva **DET** (Detection Error Tradeoff) che nasconde la soglia. Piuttosto usata è anche la rappresentazione **ROC** (Receiver Operating Characteristic) che in ordinata riporta True positive invece di False negative (ROC è ribaltata verticalmente rispetto a DET).



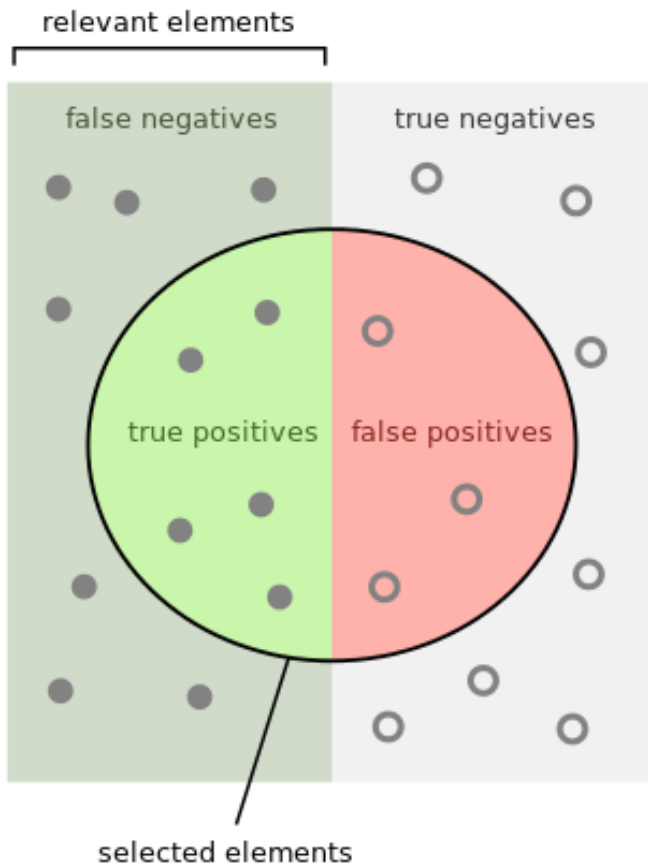
...ancora sui sistemi con soglia

- Un tipico esempio di sistemi con soglia sono i **sistemi biometrici** di **verifica** di identità:
 - Questa immagine dell'iride è del soggetto Q ?
 - False positive e False negative in questo caso prendono il nome di **False Match** e **False Non-Match**.
- E nel caso open set multi-classe?
 - In generale per la classificazione vedi [1]
 - Nella biometria si parla anche di **identificazione** distinguendola dalla verifica di identità e si definiscono FPIR (False Positive Identification Rate) e FNIR (False Negative Identification Rate).

[1] Sokolova & Lapalme (2009). A systematic analysis of performance measures for classification tasks. Information Processing and Management. <http://rali.iro.umontreal.ca/rali/sites/default/files/publis/SokolovaLapalme-JIPM09.pdf>

Precision - Recall

- Notazione molto usata in Information Retrieval.



Precision indica quanto è accurato il sistema.

Che percentuale di documenti selezionati è pertinente?

Recall quanto è selettivo.

Di tutti i documenti pertinenti quanti ne sono stati selezionati?

How many selected items are relevant?

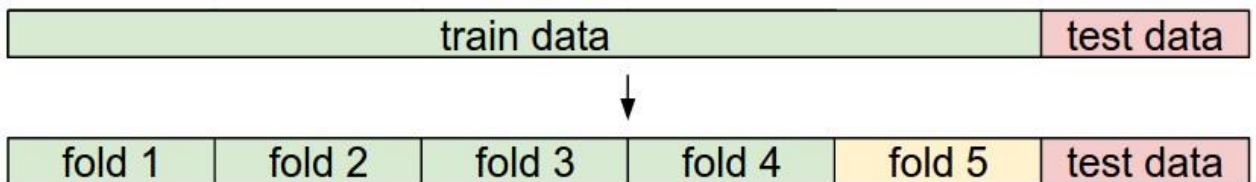
$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Training, Validation, Test

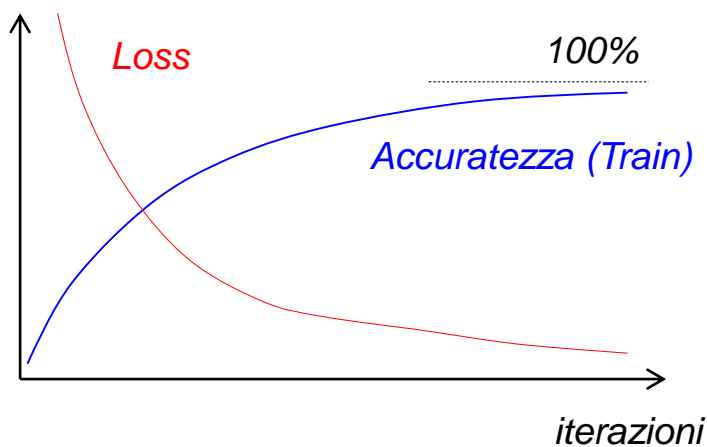
- Il **Training Set (Train)** è l'insieme di pattern su cui addestrare il sistema, trovando il valore ottimo per i parametri Θ .
- Il **Validation Set (Valid)** è l'insieme di pattern su cui tarare gli iperparametri H (ciclo esterno).
- Il **Test Set (Test)** è l'insieme di pattern su cui valutare le prestazioni finali del sistema. *Sempre forte è la tentazione di tarare gli iperparametri sul test set, ma questo dovrebbe essere evitato, pena sovrastima delle prestazioni.*
- Nei benchmark di machine learning la suddivisione dei pattern in Train, Valid e Test è spesso predefinita, per rendere confrontabili i risultati.
- In caso contrario come procedere? Es. dati 12000 pattern:
 - **Set disgiunti**: 10000 Train, 1000 Valid, 1000 Test.
 - **K-fold Cross-Validation**: 2000 Test, $K = 5$ partizioni (**fold**) da 2000 pattern. Si esegue cinque volte il training scegliendo uno dei fold come Valid e i 4 rimanenti come Train. La prestazione finale è la media/mediana delle 5 prestazioni (sul Test).



- **Leave-one-out**: caso estremo di cross-validation dove i fold hanno dimensione 1. Visto il costo computazionale, si utilizza quando i pattern sono pochi (es. < 100).

Convergenza

- Il primo obiettivo da perseguire durante l'addestramento è la **convergenza** sul Train set. Consideriamo un classificatore il cui addestramento prevede un processo **iterativo**. Si ha convergenza quando:
 - Il **loss** (output della loss function) ha andamento **decescente**.
 - L'**accuratezza** ha andamento **crescente**.

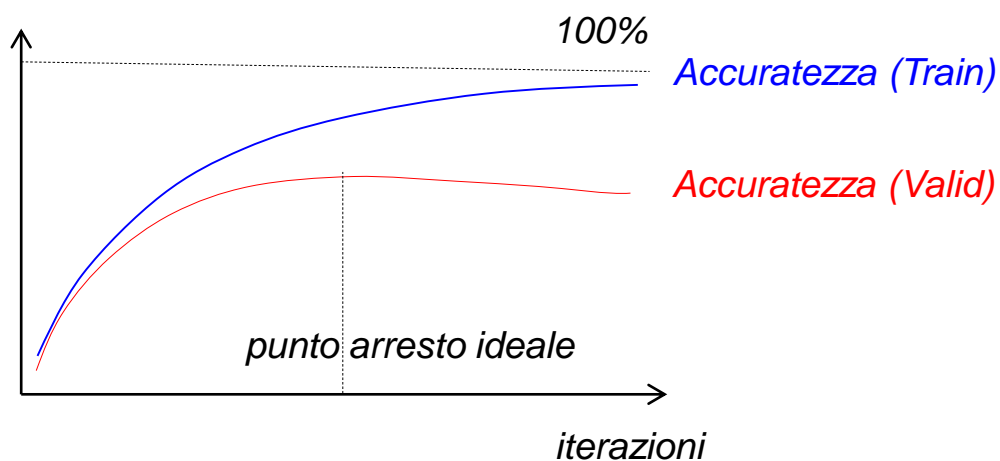


■ Note:

- Se il **loss non decresce** (o **oscilla** significativamente) il sistema non converge: il metodo di ottimizzazione non è efficace, gli iperparametri sono fuori range, il learning rate è inadeguato, ci sono errori di implementazione, ecc.
- Se il **loss decresce ma l'accuratezza non cresce**, probabilmente è stata scelta una loss-function errata.
- Se l'**accuratezza non si avvicina al 100% sul Train**, i gradi di libertà del classificatore non sono sufficienti per gestire la complessità del problema.

Generalizzazione e Overfitting

- Ricordiamoci che il nostro obiettivo è massimizzare l'accuratezza su **Test**. Nell'ipotesi che **Valid** sia rappresentativo di **Test**, ci poniamo l'obiettivo di massimizzare l'accuratezza su **Valid**.
- Per **generalizzazione** intendiamo la capacità di **trasferire** l'elevata accuratezza raggiunta su **Train** a **Valid**.
- Se i gradi di libertà del classificatore sono eccessivi, si raggiunge elevata accuratezza su **Train**, ma non su **Valid** (scarsa generalizzazione). In questo caso si parla di **overfitting** di **Train**. Questa situazione si verifica molto facilmente quando **Train** è di piccole dimensioni.



- Nei processi di addestramento iterativo, tipicamente dopo un certo numero di iterazioni l'accuratezza su **Valid** non aumenta più (e può iniziare a decrescere) a causa dell'overfitting. Monitorando l'andamento si può **arrestare l'addestramento** nel punto ideale.

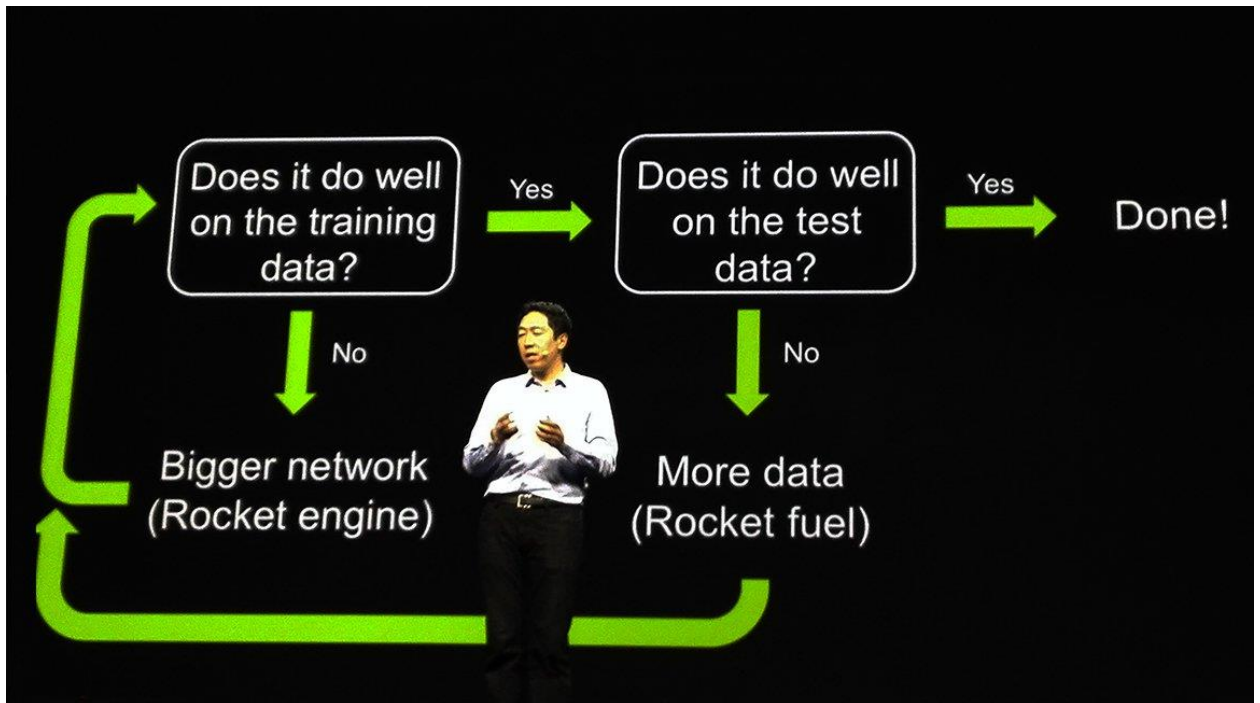
Controllare l'Overfitting

per massimizzare la Generalizzazione

- I gradi di libertà del classificatore non devono essere eccessivi, ma adeguati rispetto alla complessità del problema.
- Buona norma partire **con un pochi gradi di libertà** (controllabili attraverso iperparametri) e **via via aumentarli** monitorando accuratezza su Train e Valid.
 - *A parità di fattori la spiegazione più semplice è da preferire* Guglielmo di Occam (Occam Razor).
 - *Everything Should Be Made as Simple as Possible, But Not Simpler* Albert Einstein
- I gradi di libertà influenzano la **regolarità** della soluzione appresa (esempio regolarità del **decision boundary** nella classificazione o regolarità di una **funzione approssimante** nella regressione).
- La regolarità della soluzione può essere talvolta controllata aggiungendo un **fattore regolarizzante** alla loss-function che penalizza soluzioni irregolari.

Es. in una rete neurale si possono favorire soluzioni prive di pesi con valori elevati (L_2 regularization), oppure dove solo una parte dei pesi sono diversi da 0 (L_1 regularization \rightarrow sparsity).

La «ricetta» in una slide



Andrew Ng (Stanford, Baidu, Coursera)

<http://www.gizmodo.com.au/2015/04/the-basic-recipe-for-machine-learning-explained-in-a-single-powerpoint-slide/>

In Pratica

- Non utilizzate approcci di Machine Learning per problemi sui quali **non avete a disposizione sufficienti esempi** per il Training e il Test.
- Collezionare esempi (ed **etichettarli**) può richiedere **ingenti sforzi**, a meno che non siate in grado di reperire i pattern in rete, e/o non possiate pagare qualcuno per collezionarli/etichettarli al posto vostro (es. *Crowdsourcing via Amazon Mechanical Turk per ImageNet*).
- Collezionate pattern **rappresentativi** del problema da risolvere e distribuiteli adeguatamente tra Train, Valid e Test.
 - evitate di concentrarvi solo su casi troppo semplici (ad esempio **rimuovendo** iterativamente i pattern che il vostro approccio non riesce a gestire).
- **Automatizzate** «subito» al meglio le procedure di valutazione delle prestazioni, le eseguirete molte volte ... e alla fine avrete risparmiato un sacco di tempo.
- **Confrontate** le prestazioni del vostro sistema solo con altri addestrati sullo stesso dataset e con lo stesso protocollo.
- Attenzione all'**affidabilità statistica** dei risultati su set di piccole dimensioni. **Intervalli di confidenza** e **simulazioni su più Run** (al variare delle condizioni iniziali) possono aiutarvi.
- Infine (**ma estremamente importante**): scrivete **codice** strutturato, ordinato, eseguite debug incrementale e unit testing. Gli algoritmi di Machine Learning non sono «**esatti**» e trovare bug nel codice può essere molto difficile!

Tool per il Machine Learning

Nel corso degli anni ricercatori, sviluppatori indipendenti e imprese hanno sviluppato numerosi **tool software** (**librerie**, **framework**, **simulatori**), gran parte dei quali open-source.

La **scelta** del tool (e relativo linguaggio di programmazione) dipende dagli obiettivi del progetto e dalla preferenze dello sviluppatore.

Tra i tool più noti, ricordiamo:

- **Scikit-learn** (Python). *General Purpose*
- **OpenCV** (C++). *Molto utilizzato in ambito Visione Artificiale*
- **Weka** (Java). *Molto utilizzato in ambito Data Mining*
- **R and Caret**. *Dalla Statistica al Machine Learning*

Per un elenco più dettagliato:

<https://github.com/josephmisiti/awesome-machine-learning>.

Tool specifici per il **deep learning** saranno introdotti in seguito.