Fondamenti

- Dati e Pattern
  - Numerici, Categorici, Sequenze
  - Dati Tabulari Eterogenei vs Dati Omogenei
- Problemi di Learning
  - Classificazione
  - Regressione
  - Clustering
  - Riduzione Dimensionalità
  - Representation Learning
  - Modelli Discriminativi vs Generativi
- 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
- Consigli pratici
  - Best (and bad) practices
  - Tool e Risorse

Dati e Pattern

- I dati sono un ingrediente fondamentale del machine learning, dove il comportamento dei modelli non è pre-programmato ma appreso dai dati stessi.
- Termini come Data Science, Data Mining, Big Data enfatizzano il ruolo dei dati.
- Generalmente un campione (multi-dimensionale) nel dominio di interesse è definito Data-point.
- Utilizzeremo spesso (come sinonimo di data-point) il termine Pattern:
  - Pattern può essere tradotto in italiano in vari modi: *forma*, *campione*, *esempio*, 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).
  - Es. Persona: [*sesso*, *maggiorenne*, *colore occhi*, *gruppo sanguigno*].
  - Talvolta soggetti a ordinamento (ordinali): es. temperatura ambiente: *alta*, *media* o *bassa*.
  - Naturalmente gestiti da sistemi a regole e alberi di classificazione.
  - Con tecniche di encoding (es. one-hot) o embedding è possibile mapparli su numeri, e quindi utilizzare modelli che operano solo su dati numerici. Attenzione a ordinal encoding arbitrari che non preservano semantica (tollerate solo da alcuni modelli).

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), l'andamento di un titolo di borsa (sequenza temporale del prezzo di chiusura).
  - 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 moderni: Deep-Learning: Long Short-Term Memory (LSTM), Transformers.
- 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: HMM, Bayesian Networks, Graph Neural Networks

Dati Tabulari (eterogenei)

- In molte applicazioni aziendali, i dati sono organizzati in una tabella (come nel modello relazionale dei DBMS), nelle cui colonne troviamo gli attributi (features) e nelle righe i record (data point).
  - Le colonne possono avere formato numerico o categorico (spesso sono presenti entrambi).
  - Le colonne sono eterogenee (es. l'età di una persona, il suo salario, la sua città di residenza)
  - I dati possono essere incompleti (presenza di null) e fortemente sbilanciati.
  - Non è semplice definire una metrica di similarità tra record.
- Le tecniche di deep learning hanno superato i metodi pre-esistenti nelle applicazioni che lavorano con dati omogenei non strutturati (immagini, audio, linguaggio), ma non sono (ancora) altrettanto competitive su dati tabulari:
  - Multi-classificatori basati su alberi (es. random-forest, gradient boosting) rimangono spesso più performanti [1].

[1] Deep Neural Networks and Tabular Data: A Survey

Encoding (esempio)

- Dato il campo materiale con i seguenti 5 valori possibili: {Acciaio, Cemento, Ghisa, Polietilene, PVC} come codificarlo in formato numerico?
  - One-hot encoding: si rimuove il campo e al suo posto si aggiungono tanti campi (o colonne) quanti sono i valori distinti (5 in questo caso). Ciascun campo è associato a un materiale diverso e può assume solo valore 0/1, dove 1 indica che il pattern è di quel materiale.

    Problema: e se i valori possibili sono centinaia o migliaia ?

  - Ordinal encoding: si trasforma il campo originale in campo numerico associando ai materiali dei valori ordinali, ad esempio: Acciaio = 1, Cemento = 2, Ghisa = 3, Polietilene = 4, PVC = 5.

    Le categorie più rare, per le quali il numero di esempi è limitato, potrebbero essere raggruppate con un valore corrispondente a «other»

    Problema: in base a quale criterio decidiamo l'ordine? Numeri vicini sono interpretati come materiali simili da alcuni modelli…

Fortunatamente i multi-classificatori basati su alberi tollerano bene l'ordinal encoding e spesso non è necessario ricorrere a one-hot encoding o altri embedding più complessi (come Target Encoding, Bayesian encoding che aumentano rischio di overfitting).

Classificazione

- Classificazione: assegnare 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:

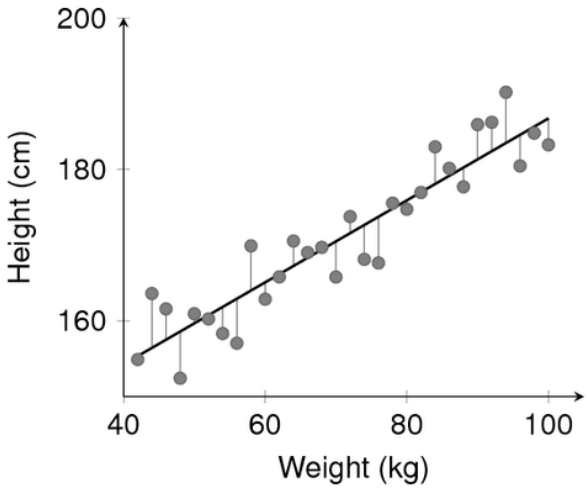
    - 21 classi per il riconoscimento di lettere dell'alfabeto

    - 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 diagnosys
  - Stock Trading

Quali sono i pattern e quali le classi ?

Regressione

- Regressione: assegnare 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.



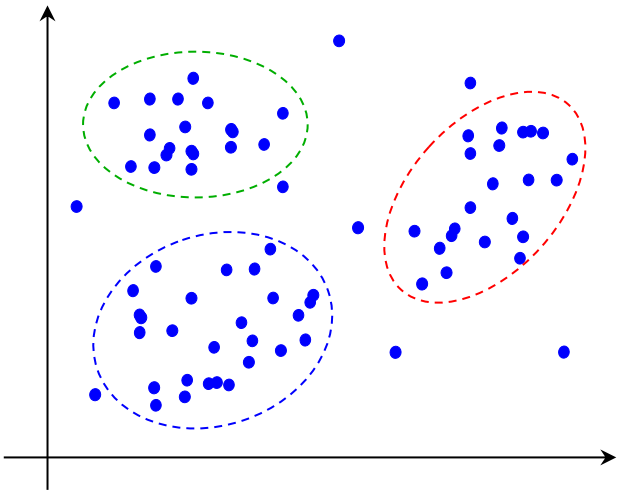
Es. *stima dell'altezza di una persona in base al peso*

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

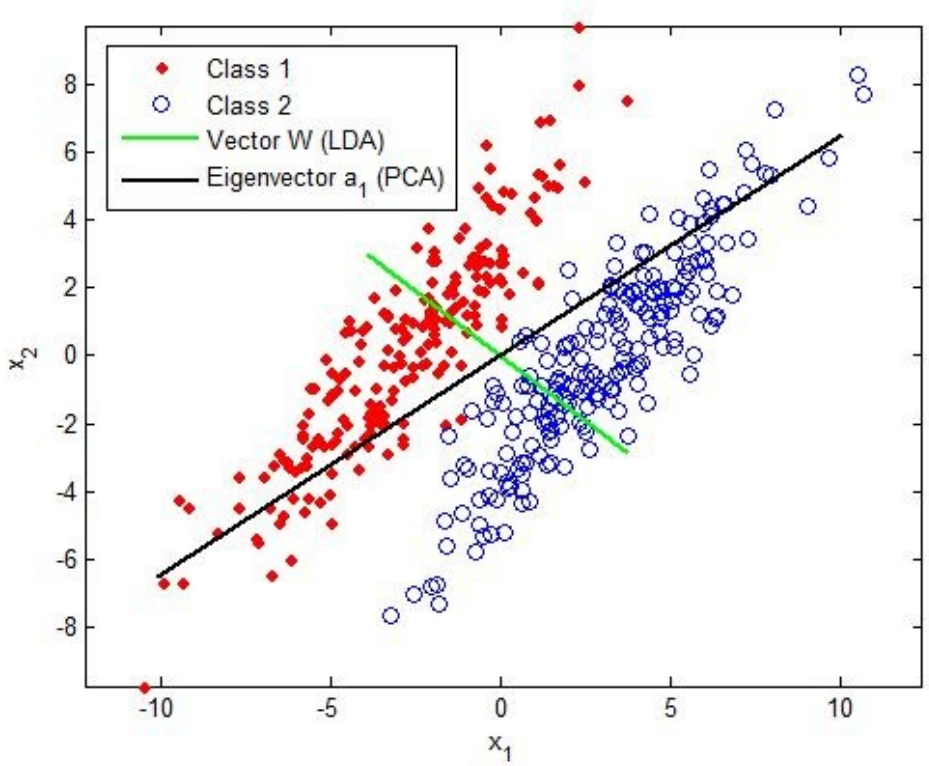
- Clustering: individuare gruppi (cluster) di pattern con caratteristiche simili.
  - Le classi del problema non sono note e i pattern non etichettati → la natura non supervisionata del problema lo rende più complesso della classificazione.
  - Spesso nemmeno il numero di cluster è noto a priori
  - 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 non supervisionata

Riduzione Dimensionalità

- Riduzione di dimensionalità: ridurre il numero di dimensioni dei pattern in input.
  - Consiste nell'apprendimento di un mapping da $\Re^d$ a $\Re^k$ (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 $d > 3$.

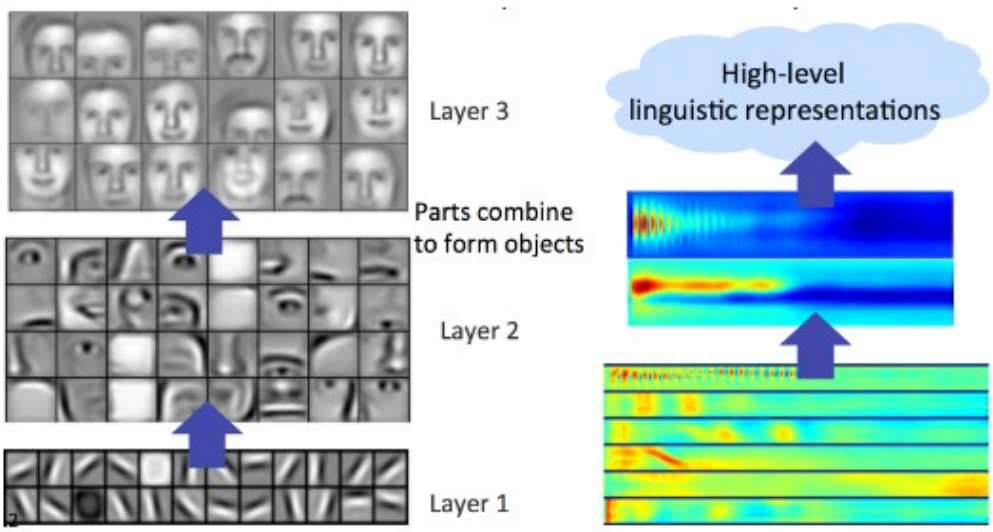


Representation Learning

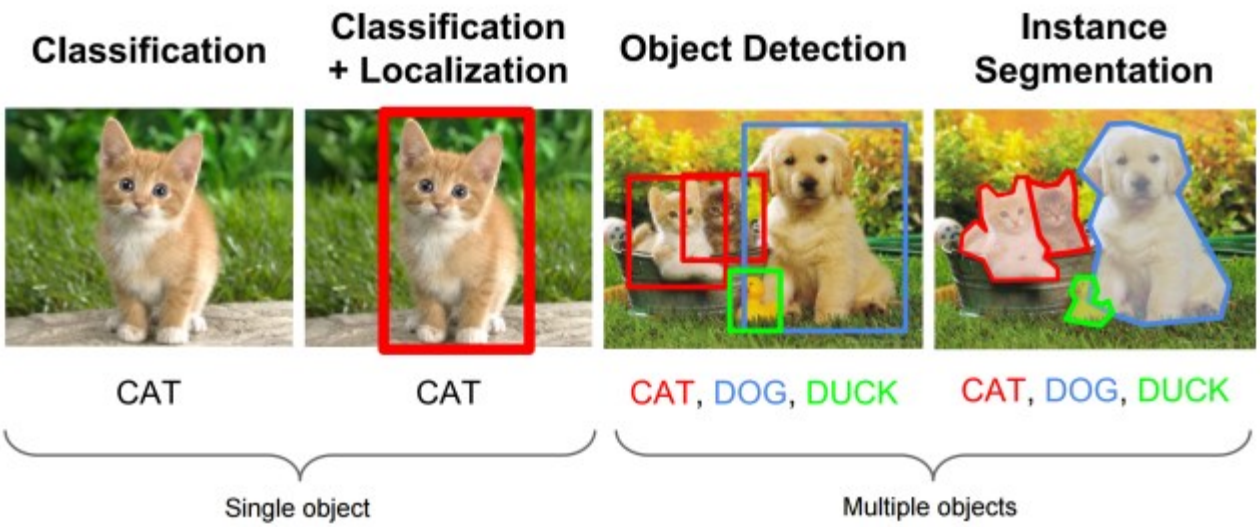
- Il successo di molte applicazioni di machine learning dipende dall'efficacia di rappresentazione dei pattern in termini di features.
  - La definizione di features ad-hoc (hand-crafted) per le diverse applicazioni prende il nome di feature engineering.
  - *Ad esempio per il riconoscimento di oggetti 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 raw 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.



Problemi nel dominio Visione



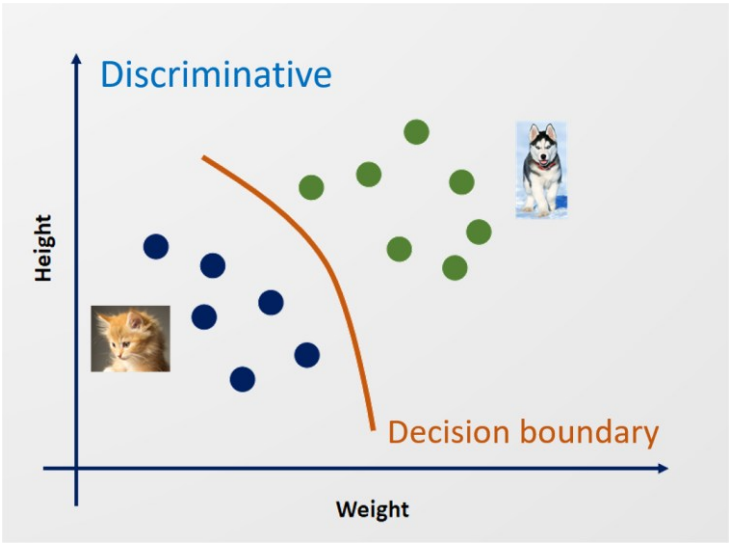
Assumiamo sia presente un solo oggetto o comunque un solo oggetto dominante

Possono essere presenti più oggetti

- Classification: determina la classe dell'oggetto.
- Localization: determina la classe dell'oggetto e la sua posizione nell'immagine (bounding box).
- Detection: per ogni oggetto presente determina la classe e la posizione nell'immagine (bounding box).
- Segmentation: al posto di una bounding box (rettangolare) etichetta i singoli pixel dell'immagine a classi con indici delle classi. Applicazioni in ambito *telerilevamento* (es. etichettare coltivazioni in immagini satellitari), immagini *mediche* (evidenziare patologie), *guida automatica* (evidenziare regione stradale).

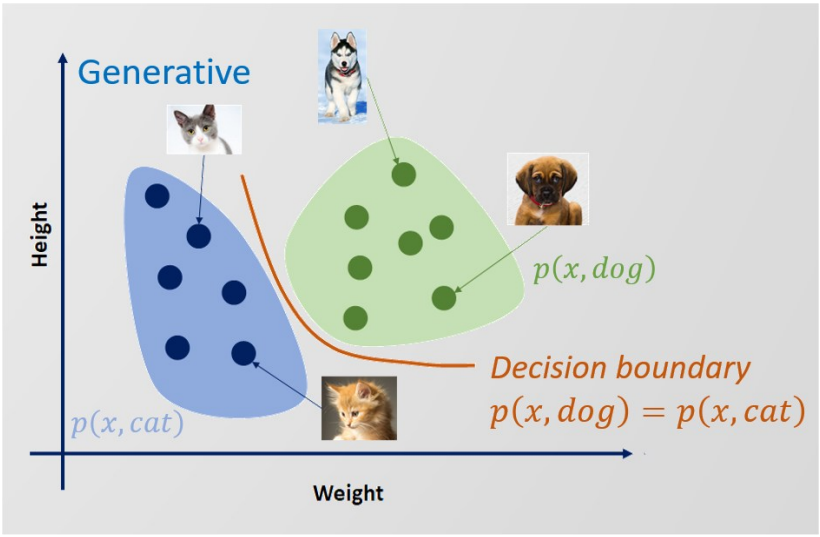
Modelli Discriminativi vs Generativi

- I modelli discriminativi (classificatori) hanno l'obiettivo di assegnare un nuovo datapoint a una classe. La cosa importante è apprendere il decision boundary che separa le classi.



Classificazione ≡ su quale lato del decision boundary cade il nuovo dato ?

- I modelli generativi apprendono (esplicitamente/implicitamente) la distribuzione probabilistica degli esempi usati per il loro addestramento. Dopo l'addestramento, possono:



- generare nuovi dati sintetici a partire da numeri random (anche in modo condizionato)
- modificare o trasformare l'input fornito
- classificare l'input comparando le probabilità che sia generato dalle diverse classi

Generative AI

- La moderna Generative AI (GenAI) fa largo uso di modelli generativi basati su reti neurali profonde (es. GAN, VAE, LLM).

- Attenzione: Il termine generative può essere ambiguo in quanto le tecniche di GenAI possono anche risolvere problemi che non richiedono la produzione creativa di contenuti (es. question/answering o classificazione di news). Pertanto generativo (da non confondere con creativo) si riferisce ai sottostanti modelli generativi usati.

- Esempio: Image generation by prompt conditioning e inpainting (Stable Diffusion)

Step 1
"Generate a purple dress"

Step 2
Select the image portion to change

Step 3
"Generate a foulard"

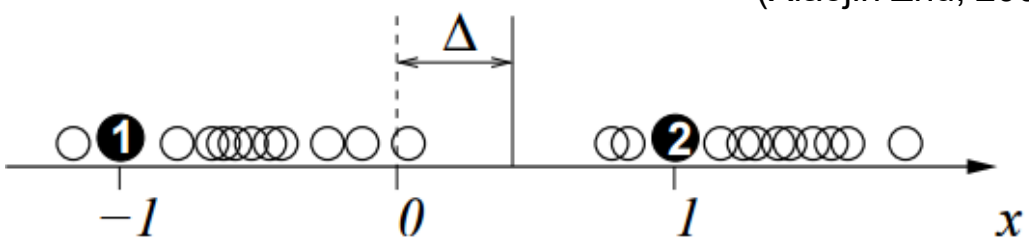
Apprendimento

- 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
  - *anomaly detection* è spesso considerata un'applicazione unsupervised (one-class classification): si hanno molti esempi di funzionamento normale, ma pochi casi di errore.

- 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.

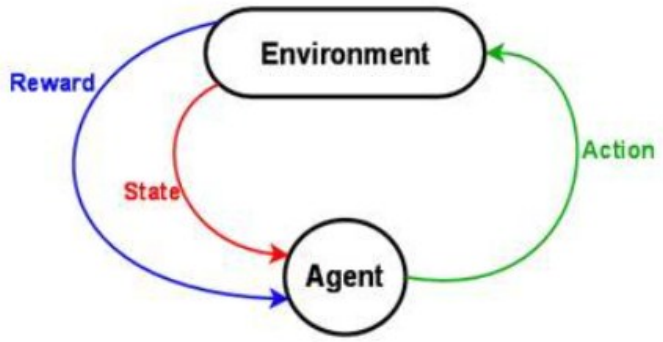
- Incrementale: a seguito dell'addestramento iniziale, sono possibili ulteriori sessioni di addestramento.
  - Scenari: Sequenze di Batch, Unsupervised Tuning.
  - Rischio: Catastrofic Forgetting (il sistema dimentica quello che ha appreso in precedenza).

- 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 subsequence 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 classici 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 generale, il comportamento di un modello M di machine learning è regolato da un set di parametri Θ (es. i pesi delle connessioni in una rete neurale). Per rendere esplicita questa dipendenza indichiamo il modello come $M(\Theta)$. 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 $f(Train, M(\Theta))$ può indicare:
  - l'ottimalità della soluzione (da massimizzare).

$$\Theta^* = argmax_\Theta\, f(Train, M(\Theta))$$

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

$$\Theta^* = argmin_\Theta\, f(Train, M(\Theta))$$

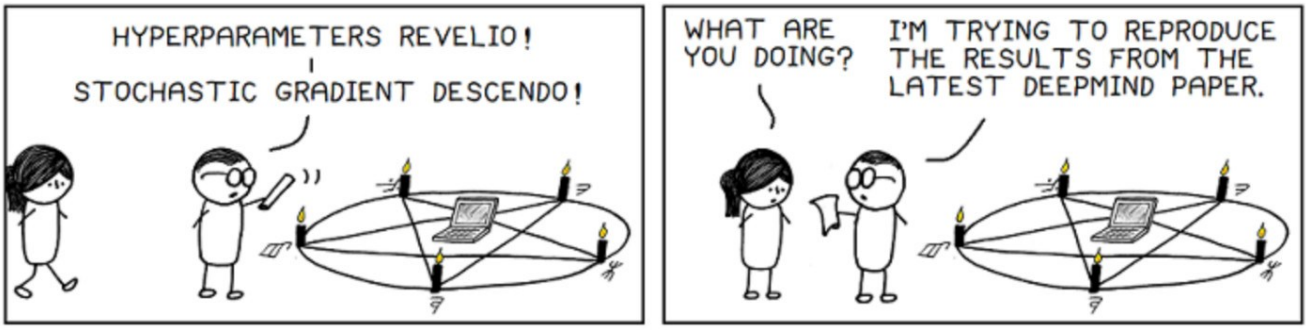
- $f(Train, M(\Theta))$ può essere ottimizzata:
  - esplicitamente, con metodi che operano a partire dalla sua definizione matematica.

    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



- Stabilito il modello da utilizzare, prima dell'apprendimento vero e proprio, deve essere definito il valore dei cosiddetti iperparametri. Gli iperparametri H definiscono i dettagli architetturali del modello e della corrispondente procedura di training. Per rendere esplicita anche questa dipendenza utilizziamo $M(H, \Theta)$
- 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.

Iperparametri (2)

- Per l'addestramento si procede con un approccio a due livelli. A livello esterno si fissano gli iperparametri H; a livello interno si esegue l'apprendimento e si valutano le prestazioni (su un Validation set disgiunto dal Training set, vedi slide successive). Al termine della procedura si scelgono gli iperparametri H^* che hanno fornito prestazioni migliori.

For H in Hyperparameter values

$\Theta^* = argmin_\Theta\, f(Train, M(H, \Theta))$

Evaluate $M(Valid, H, \Theta^*)$

Select best hyperparameters (H^*)

- Ma su quali dati si valutano le prestazioni, e con quali metriche?

Training, Validation, Test

- Il Training Set (Train) è l'insieme di pattern su cui addestrare il modello, 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.
  - Forte è la tentazione di tarare gli iperparametri e/o di scegliere il modello migliore (vedi cherry picking) sul test set. Operando in questo modo però si sovrastimano le prestazioni!
- Nei benchmark classifici di machine learning la suddivisione dei pattern in Train, Valid e Test è spesso predefinita, per rendere confrontabili i risultati. In caso contrario è compito del progettista definire la suddivisione. Si può optare per:
  - Set disgiunti
  - K-fold Cross-Validation

Set Disgiunti

- Se i dati sono sufficientemente numerosi per la complessità del problema e del modello scelto (ovvero non producono overfitting, vedi slide successive), si possono usare Set Disgiunti:

  Esempio. In problema di classificazione abbiamo a disposizione 15000 pattern; possiamo suddividerli in:

  - 10000 Training
  - 2500 Validation
  - 2500 Test

- Aumentare la partizione di Train consente di addestrare meglio il modello. D'altro canto Validation e Test troppo piccoli non permettono di misurare con sufficiente confidenza le prestazioni e la generalizzazione.

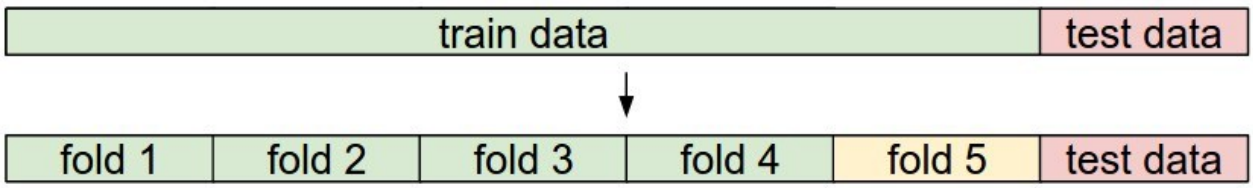
  - La dimensione del validation/test dipende anche dall'entità dell'errore atteso. Se l'errore di classificazione del sistema è del 10%, con un test set di 2500 esempi avremo 250 casi di errore. Se l'errore e lo 0.2% avremo solo 5 errori con conseguente scarsa affidabilità statistica.

- Lo split può essere random (shuffling) oppure mirato (es. per mantenere separati periodi temporali di train e test).

  - Nel caso di serie temporali, dove il sistema è addestrato su un periodo storico e produrre previsioni per un periodo successivo, uno split random non è corretto (porta a una sovrastima delle prestazioni) in quanto il training set include data point temporalmente vicini (molto simili) a quelli di test.

K-fold Cross-Validation

- Una scelta più robusta degli iperparametri (di fatto obbligatoria per dataset di piccole dimensioni) si ottiene con la procedura di k-fold Cross-Validation. Nell'ipotesi di 15000 pattern totali e k=5:
  - 5000 pattern sono scorporati a priori per il Test.
  - i 10000 pattern rimanenti (dopo averli mescolati) sono suddivisi in 5 gruppi (fold) da 2000 pattern ciascuno.



  - Per ogni combinazione di iperparametri H_i che si vuole valutare:
    - Si esegue 5 volte il training scegliendo uno dei fold come Valid e i 4 rimanenti come Train.
    - Si calcola l'accuratezza avg_acc_i come media/mediana delle 5 accuratezze sui rispettivi Valid
  - Si sceglie la combinazione di iperparametri con migliore *avg_acc*.
  - Scelti gli iperparametri ottimali si riaddestra il modello su tutto il training set (5 fold) e, solo a questo punto, si verificano le prestazioni sul test set.

- 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).

Cherry picking

Uno studio recente [1] ha condotto una revisione sistematica di 27 lavori scientifici che utilizzano il machine learning per il trading di azioni.

- Tutti i lavori considerati dichiarano predizioni accurate e strategie profittevoli.
- Al tempo stesso non si ha evidenza di casi reali di chiaro successo in ambito AI-driven trading.
- La quasi totalità dei lavori considerati esegue molteplici esperimenti per la scelta della strategia/modello migliore (pur eseguendo correttamente per ogni modello cross-validation per la scelta dei relativi iperparameri). I modelli sono poi "validati" sul test set, e viene scelto il migliore (cherry picking).
- In questo modo si ha overfitting del test set (che, non di rado, copre sole un breve periodo storico).
- Questa cattiva abitudine «accademica» (e non solo) può avere conseguenze nefaste in ambito business.

Morale: non usare il test set per orientare le scelte durante lo sviluppo di un sistema (strategia, features, modello), specialmente quando le dimensioni del test set sono limitate ed è forte il rischio di overfitting.

[1] A review of machine learning experiments in equity investment decision making: why most published research findings do not live up to their promise in real life

Partizionare i dati

- Partizionare i dati in dati in Training, Validation e Test, richiede in genere di generare (random) gli indici degli elementi da assegnare ai diversi insiemi per poi procedere alla suddivisione facendo attenzione a dividere nello stesso modo anche le etichette (in classificazione) o valori target (nella regressione).
- La funzione train_test_split di Scikit Learn automatizza questa fase, a partire dalle proporzioni specificate in input, fornendo anche supporto per shuffling (ordinamento casuale dei pattern) e stratification (selezione bilanciata rispetto a certe categorie o insiemi di valori).
- Relativamente a Cross Validation Scikit Learn mette a disposizione la funzione cross_val_score che a partire da un unico training set e il numero K di fold, valuta K volte il sistema e ritorna un array di K score (es. accuratezze di classificazione).

https://scikit-learn.org/stable/modules/cross_validation.html

Selezione Automatica Iperparametri

La ricerca di valori ottimali per gli iperparametri può essere lunga e noiosa. Automatizzare quando possibile:

- Grid Search: per ogni iperparametro si definisce un insieme di valori da provare. Il sistema è valutato su tutte le combinazioni di valori di tutti gli iperparametri. Può essere molto costoso: raffinamento incrementale dei valori. Attenzione inoltre se vengono selezionati valori di bordo per uno o più iperparametri.
  - Esempio con funzione GridSearchCV di Scikit-Learn:

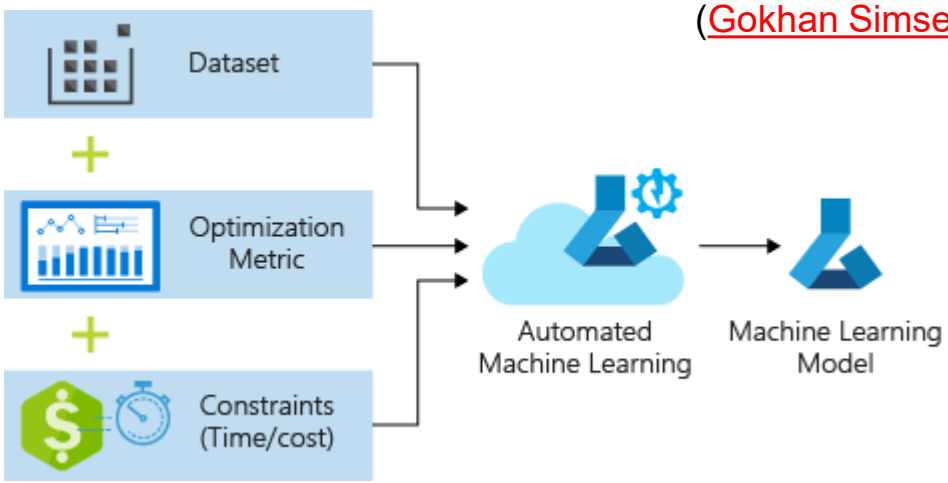
```
param_grid = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
 ]
```

  Ogni { } corrisponde a una grid search:
  - nella prima si valuta un classificatore SVM con kernel lineare e con 4 valori di C → 4 valutazioni;
  - nella seconda si valuta il classificare con kernel rbf, 4 valori di C e 2 valori di gamma → 8 valutazioni;
  - In totale 12 valutazioni, ciascuna delle quali fatta con Cross Validation (se richiesto).

- Random Search: sorteggia causalmente valori di iperparametri dai range/distribuzioni specificati/e, eseguendo un numero prefissato di iterazioni.
  - Funzione RandomizedSearchCV di Scikit-Learn:

Auto-ML

(Gokhan Simsek, 2019)



- I sistemi Auto-ML effettuano la scelta automatica del modello migliore (es. SVM, Random Forest, Rete Neutrale MLP) per la risoluzione di un problema. Oltre al modello possono essere scelte automaticamente tecniche di data/feature processing.
- L'approccio naïve prevede un ulteriore «ciclo esterno» è sul tipo di modello.

  For M_i in Models

  For H in Hyperparameter values for M_i

  $\Theta^* = argmin_\Theta\ f(Train, M_i(H, \Theta))$

  Evaluate $M_i(Valid, H, \Theta^*)$

  Select best model and hyperparameters (M^*, H^*)

- Auto-ML è solitamente eseguito su risorse remote in Cloud (dimensionate opportunamente), ma esistono versioni per computazione locale (es. Auto-Sklearn)
  - Auto-ML è efficace/efficiente?

Metriche per la misura di prestazioni

- Una possibilità consiste nell'utilizzare direttamente la funzione obiettivo per quantificare le prestazioni. In genere però si preferisce una misura legata direttamente 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.

$$Accuratezza = \frac{pattern\ correttamente\ classificati}{pattern\ classificati}$$

$$Errore = 100\% - Accuratezza$$

Attenzione ai problemi di classificazione con cardinalità classi molto sbilanciate:

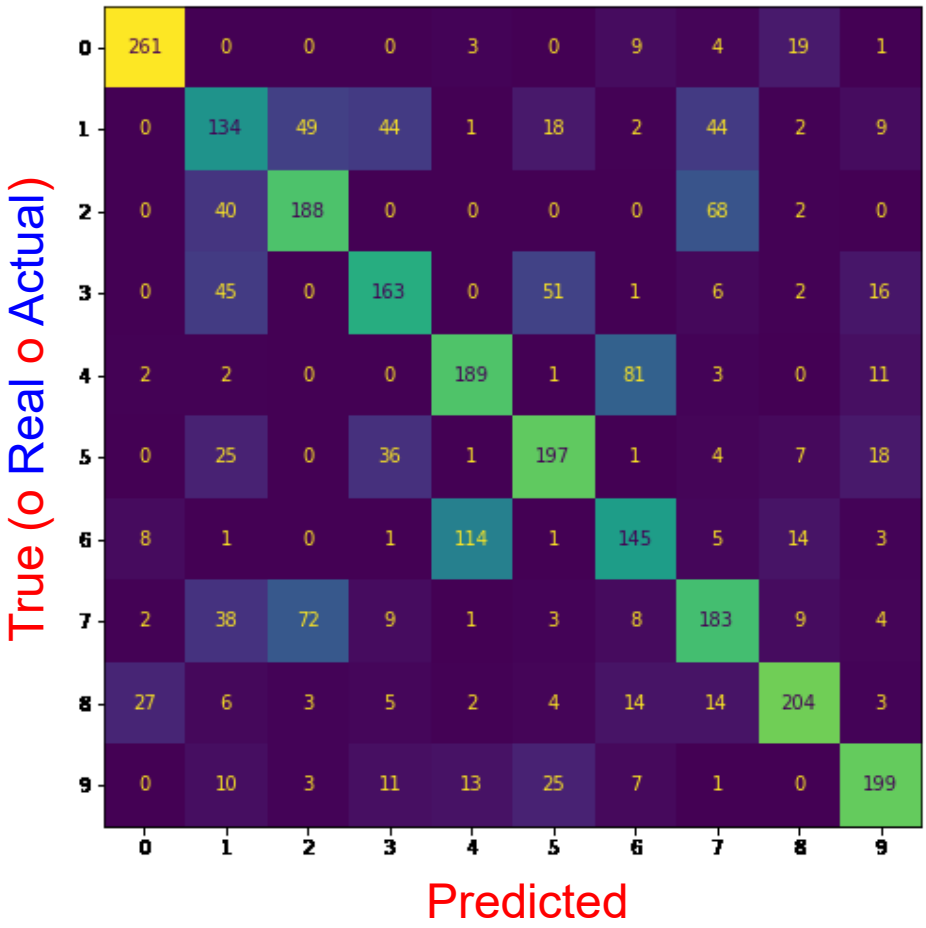
- Es. in un problema di classificazione binaria, se una delle due classi è rara, un classificatore «dummy» che non la predice mai potrebbe raggiungere un'accuratezza di classificazione vicina al 100%. Meglio usare Precision/Recall in questo caso (vedi slide successive).

- Nei problemi di Regressione, si valuta in genere l'RMSE (Root Mean Squared Error) ovvero la radice della media dei quadrati degli scostamenti tra valore vero e valore predetto (ulteriori approfondimenti nel seguito).

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1..N}(pred_i - true_i)^2}$$

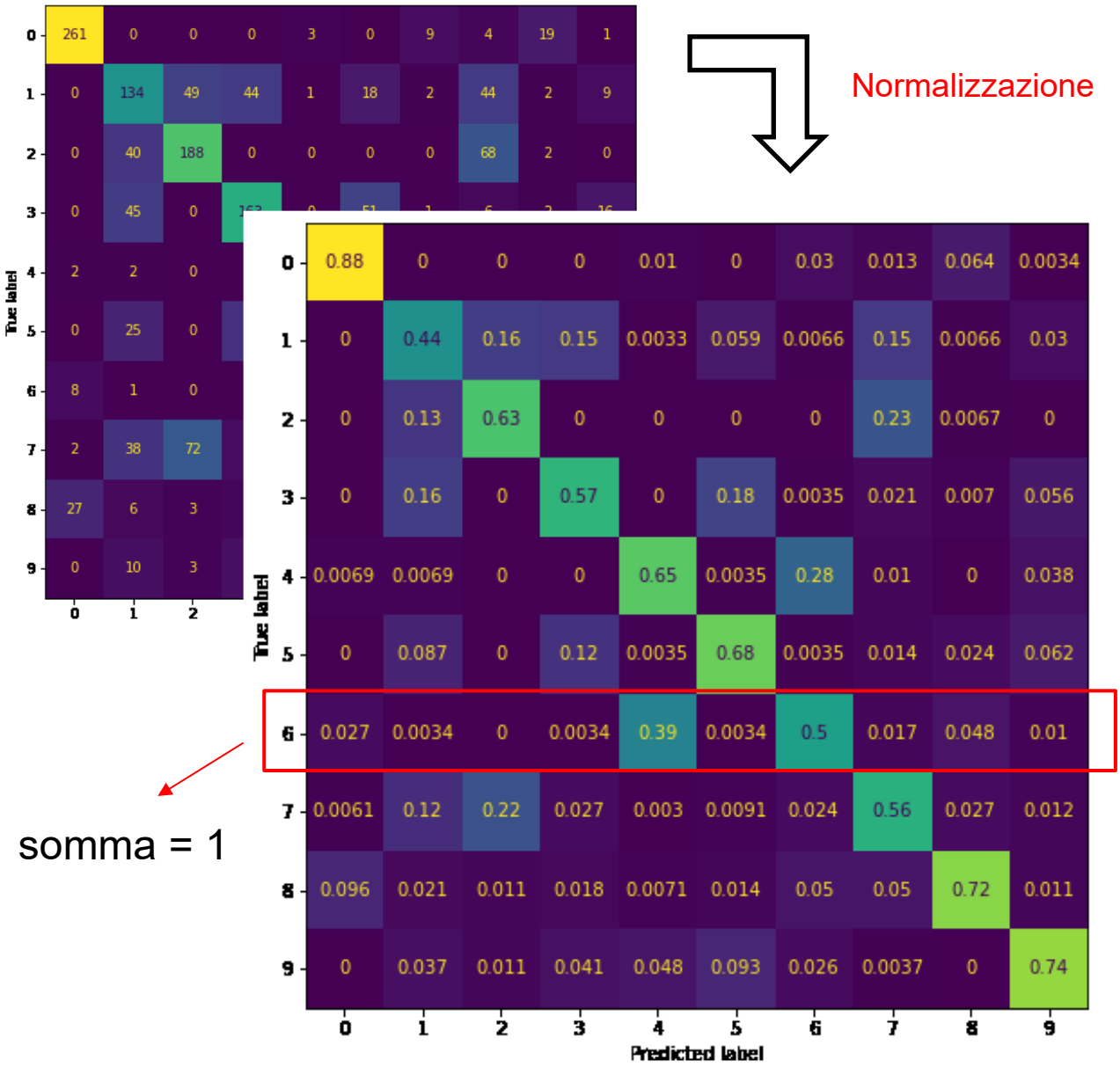
Matrice di Confusione

- La matrice di confusione (confusion matrix) è molto utile nei problemi di classificazione per capire come sono distribuiti gli errori.
- Nell'esempio un problema di classificazione digit (10 classi).
  - Assumiamo che sulle righe ci siano le classi True (o Real o Actual) e sulle colonne le classi Predicted (*non sempre cosi: a volte scambiate, fare attenzione!*)
  - Una cella (r,c) riporta il numero di casi in cui il sistema ha predetto di classe c un pattern di classe vera r.
  - Idealmente la matrice dovrebbe essere diagonale. Valori elevati (fuori diagonale) indicano casi di errore.



Matrice di Confusione (2)

- La matrice può essere normalizzata per righe (classi True) per passare da numero di errori a percentuali di errore.
- Dopo la normalizzazione i valori sulle righe hanno somma 1
- Funzione *plot_confusion_matrix* disponibile in scikit-learn



Classificazione Binaria

- Dato un classificatore binario e T = P + N pattern da classificare (P positivi e N negativi), 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.

- Passando da errori a frequenze/probabilità (che corrisponde a normalizzare per righe la confusion matrix, vedi slide successiva):

$$TPR\ (True\ Positive\ Rate) = \frac{TP}{P}$$

$$TNR\ (True\ Negative\ Rate) = \frac{TN}{N}$$

$$FPR\ (False\ Positive\ Rate) = \frac{FP}{N}$$

$$FNR\ (False\ Negative\ Rate) = \frac{FN}{P}$$

Con questa notazione l'accuratezza di classificazione può essere scritta come:

$$Accuracy = \frac{TP + TN}{T = P + N}$$

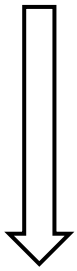
Classificazione Binaria

errori sulla matrice di confusione

- I 4 casi (TN, FP, FN, TN) sono facilmente collocabili sulle 4 celle della matrice di confusione (2x2).
- Le relative frequenze (TNR, FPR, FNR, TNR) occupano le stesse posizioni sulla matrice di confusione normalizzata.

		Predicted	
		False	True
Real	False	TN	FP
	True	FN	TP

Matrice di confusione non normalizzata



Infatti:
N = TN + FP
P = FN + TP

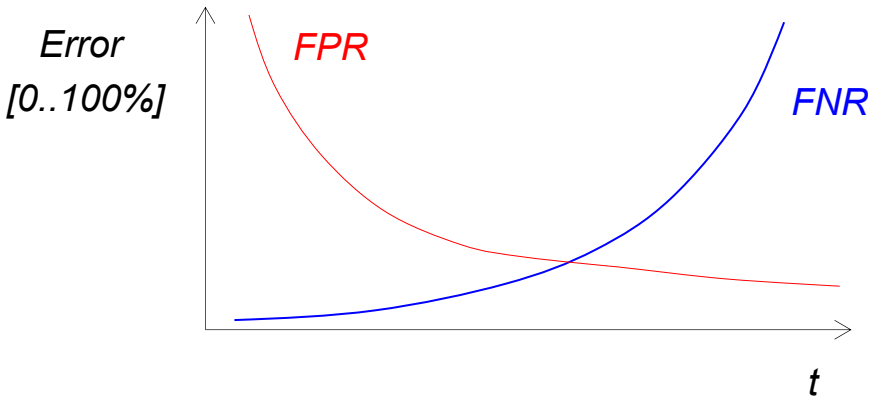
		Predicted	
		False	True
Real	False	TNR	FPR
	True	FNR	TPR

Matrice di confusione normalizzata (per righe)

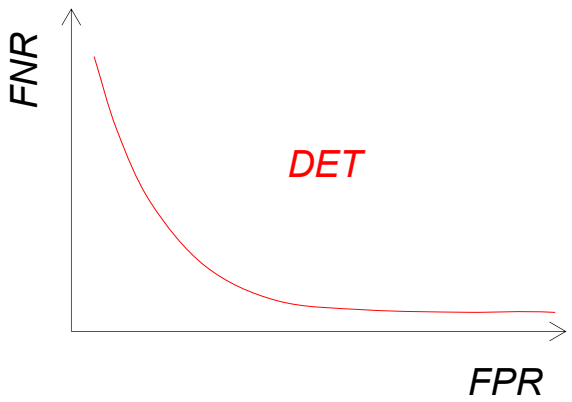
Attenzione: False e True sono talvolta scambiati di posto nella matrice

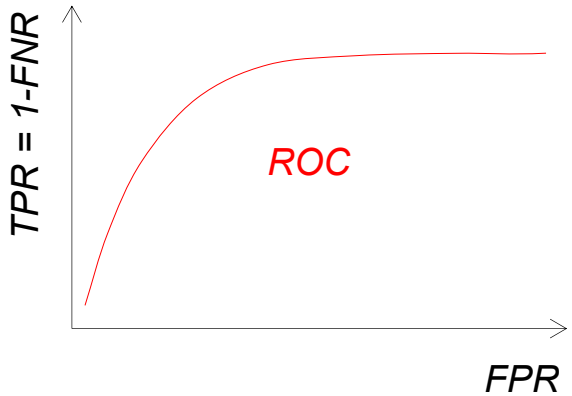
DET e ROC

- L'output di un classificatore è spesso probabilistico (valore continuo in 0…1). In una problema di classificazione binaria i pattern possono essere predetti come positivi (o negativi) confrontando l'output (della classe positiva) con una 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.



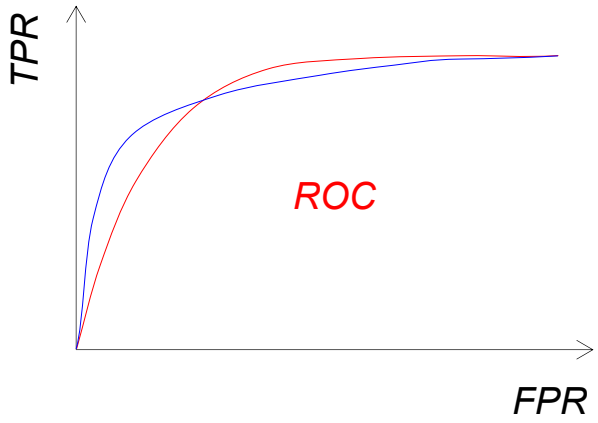
- Le due curve possono essere «condensate» in una curva DET (Detection Error Tradeoff) che nasconde la soglia. Piuttosto usata è anche la rappresentazione ROC (Receiver Operating Characteristic) che in ordinata riporta True positive invece di False negative (ROC è ribaltata verticalmente rispetto a DET).





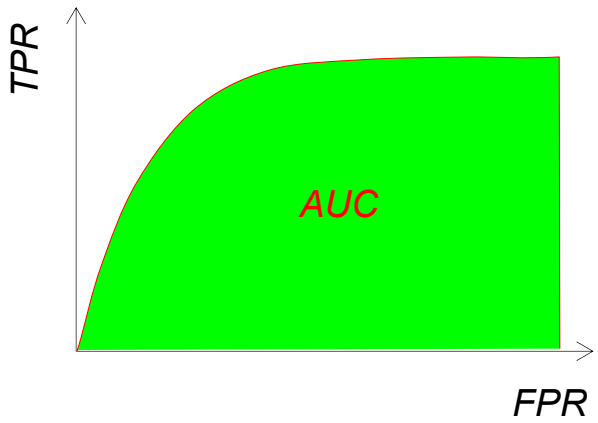
Area Under Curve (AUC)

- Quale dei due sistemi (rosso, blu) è migliore dell'altro?



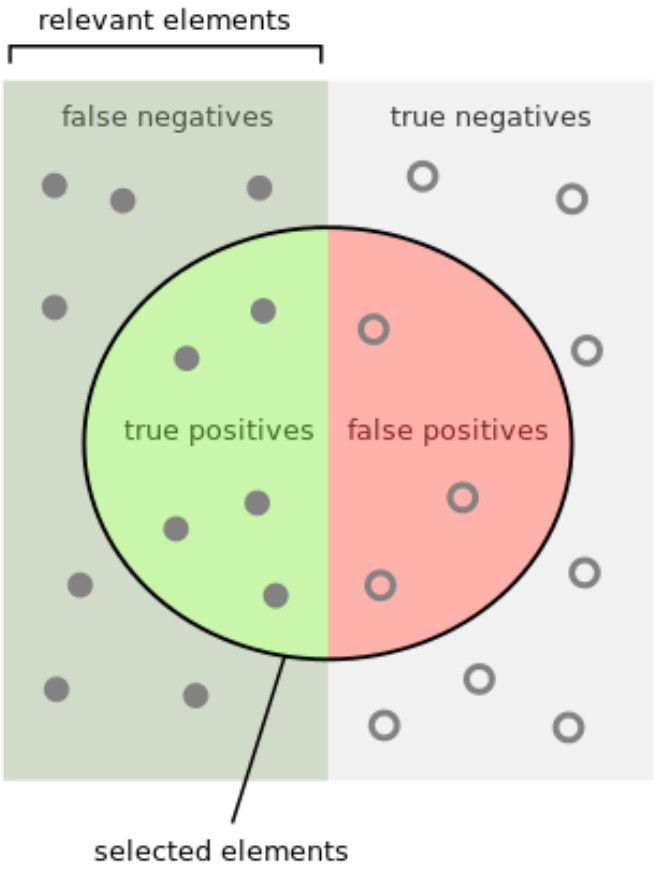
  - Il sistema migliore è quello la cui curva è più «alta», ma in questo caso le curve si intersecano … e l'ottimalità dipende dal punto di lavoro desiderato.

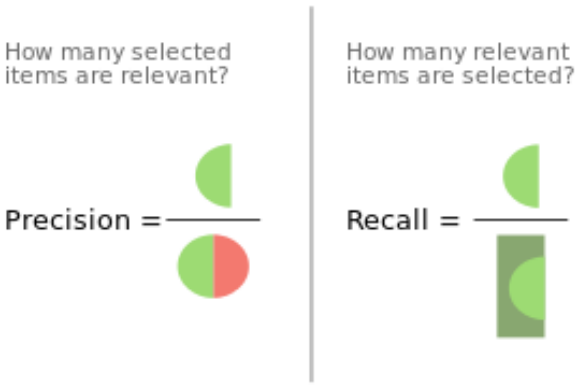
- Un confronto può essere fatto «mediando» sui diversi punti di lavoro del sistema. L'area sotto la curva ROC (AUC) è uno scalare in [0,1] che caratterizza la prestazione media (maggiore è, meglio è). Può essere calcolata come integrale numerico attraverso il metodo dei trapezi.



Precision - Recall

- Notazione molto usata in Information Retrieval e in generale nelle applicazioni di detection o di classificazione (con classi sbilanciate).
- Precision/Recall sono definiti a partire da TN,TP,FP,FN (classificazione binaria)





Precision indica quanto è accurato il sistema.

Che percentuale di documenti selezionati è pertinente?

$$Precision = \frac{TP}{TP + FP}$$

Recall quanto è selettivo.

Di tutti i documenti pertinenti quanti ne sono stati selezionati?

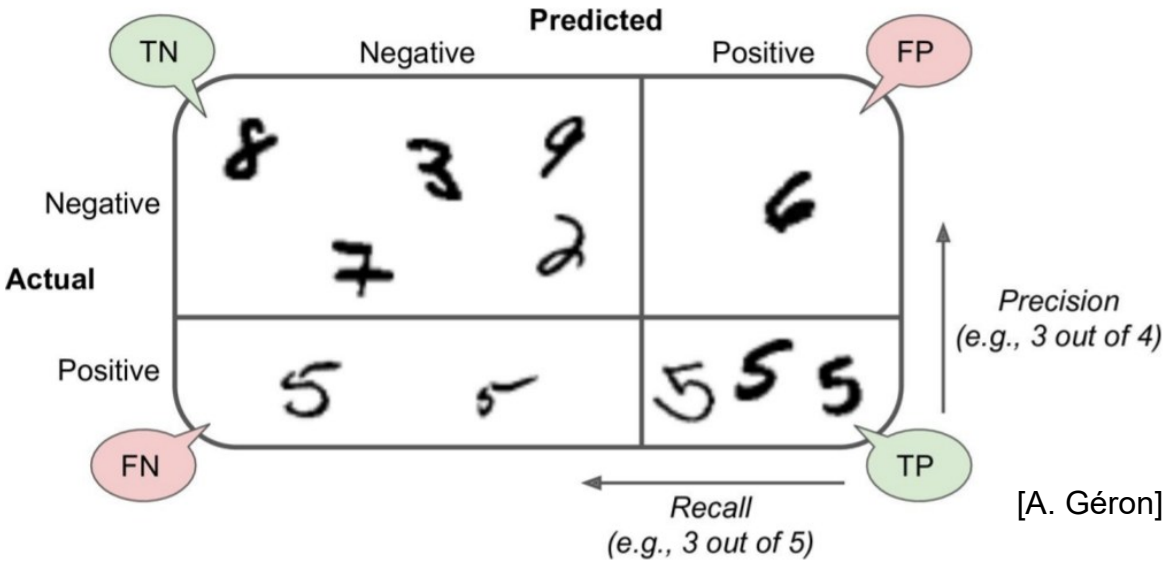
$$Recall = \frac{TP}{TP + FN} = \frac{TP}{P} = TPR$$

Recall già disponibile nella matrice di confusione normalizzata

Precision – Recall su matrice 2x2

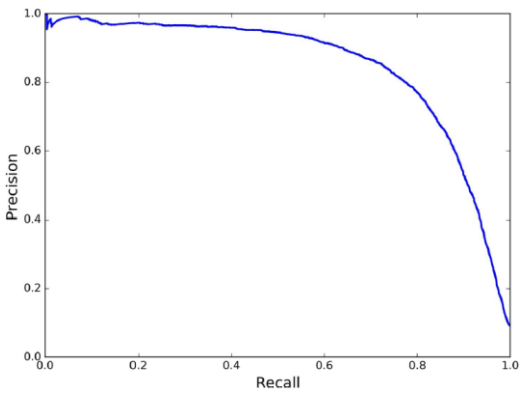
- La rappresentazione grafica più utile per comprendere Precision e Recall è la matrice di confusione 2x2 che evidenzia TN,TP,FP,FN:

- Il classificatore (binario) dell'esempio sotto ha come classe positiva il digit 5 e come classe negativa tutti gli altri digit.



[A. Géron]

Tipicamente anche Precision/Recall dipendono da soglia → grafico Recall/Precision



- Nella classificazione multi-classe, precision e recall si possono calcolare per ciascuna classe (considerando la classe come Positive e gli esempi di tutte le altre come Negative), come nell'esempio sopra.

Indicatori compatti: F1-score, AP

Indicatori compatti (a singolo valore) utili per comparare l'accuratezza di modelli diversi.

- F1-score (valori continui in 0..1) è calcolato come media armonica di Precision e Recall:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

  La media armonica:

  - Corrisponde alla media aritmetica se Precision = Recall
  - «Penalizzazione maggiore» rispetto alla media aritmetica quando i valori sono diversi.

- Average Precision (AP) è una sorta di AUC (Area Under Curve) sul grafico Recall/Precision:

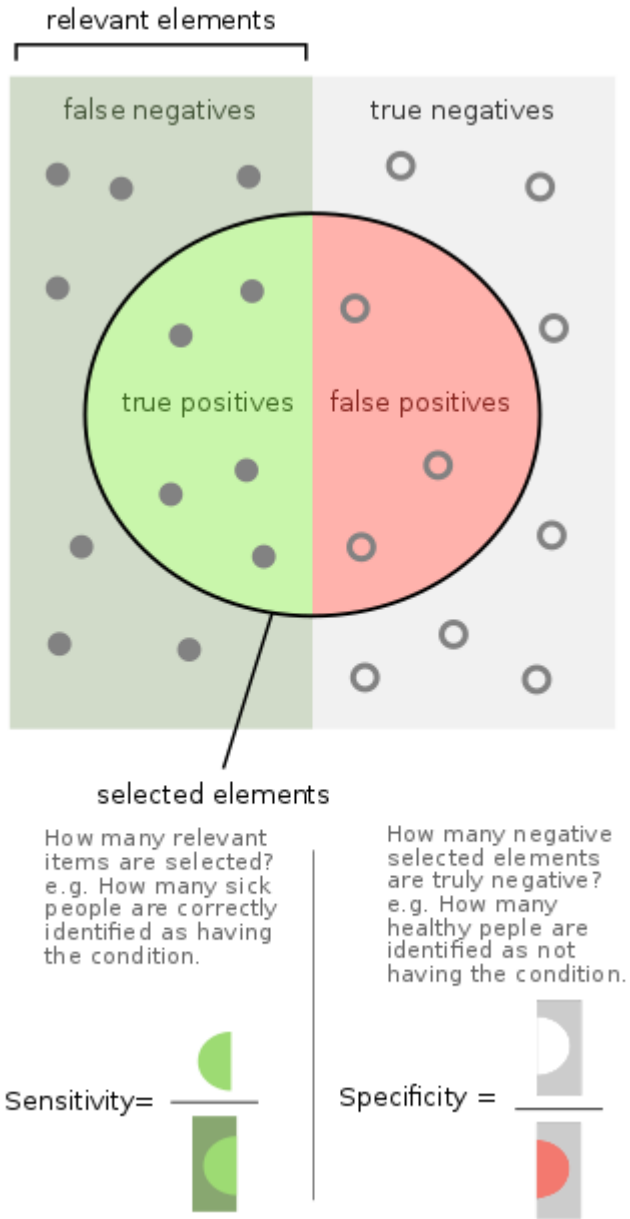
$$AP = \sum_n (R_n - R_{n-1})\, P_n$$

  dove P_n e R_n sono i valori di precision e recall alla soglia n. Differentemente da AUC si usa integrazione con rettangoli (e non trapezi) per evitare risultato over-ottimistico.

- Nelle applicazioni dove è importante operare a Precision (o Recall) definiti meglio derivare esplicitamente dal grafico Recall/Precision:
  - Precision @ Recall = X
  - Recall @ Precision = X

Sensitivity - Specificity

- In diagnostica medica (es. immunologia) si usano i termini Sensitivity e Specificity per caratterizzare la bontà di un test (es. tampone Covid-19).
- La classe negative corrisponde ai soggetti «sani/non-infetti», la classe positive ai soggetti «malati/infetti»



Sensitivity ≡ Recall ≡ TPR

Tra tutti i sottoposti al test quanti infetti sono correttamente identificati come tali?

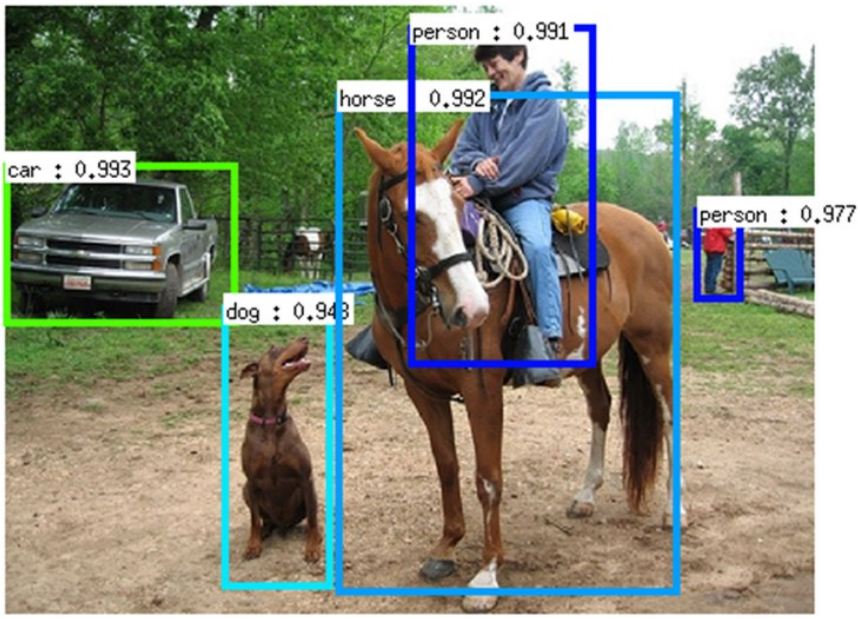
Specificity ≡ TNR indica quanto è accurato il sistema.

Che percentuale di non-infetti sottoposti a test sono stati dichiarati sani?

La Specificity corrisponde alla Recall della classe negativa

Entrambi già disponibili sulla diagonale della matrice di confusione normalizzata

Object Detection: AP e mAP



Detection:

per ogni oggetto presente determina la probabilità di appartenenza alla classe più probabile e la corrispondente bounding box.

- Possibili diversi tipi di errore, tra cui: oggetto non trovato, classe errata, bounding box sbagliata o imprecisa. Come quantificare il tutto con uno scalare per rendere sistemi confrontabili?
  - Average Precision (AP) è calcolata per oggetti di una singola classe su tutto il database.
  - mean Average Precision (mAP) è la media di AP su tutte le classi.
  - Per calcolare TP, FP (a una certa confidenza) si considera una prediction corretta quando la classe è giusta e la Intersection over Union (IoU) delle due bounding box (rilevata e vera) è maggiore di un valore dato (es. 0.5).
  - Per maggiori dettagli:

    https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

Problemi Closed e Open set

- 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 tutta la 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.

- 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. Positivi: tutte le porzioni di immagine (finestre) in cui appaiono volti. 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.

Sistemi Biometrici

- Un sistema biometrico [1] esegue la verifica di identità (1:1) o identificazione (1:N) di un soggetto sulla base di una sua caratteristica fisica (es. impronta digitale, volto, iride) o comportamentale (es. stile di battitura):

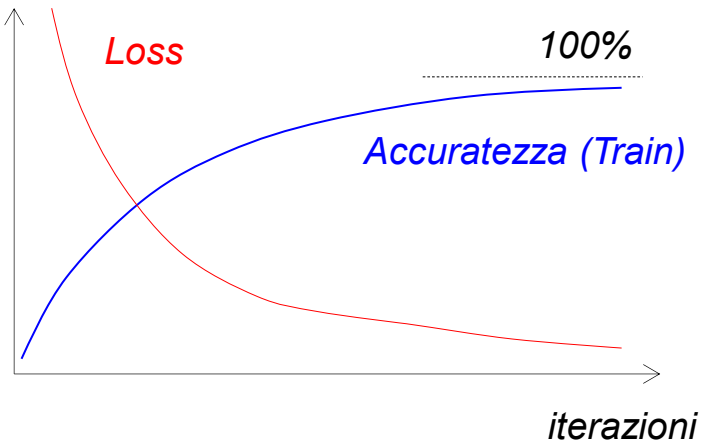
- La verifica di identità (es. questa immagine del volto è del soggetto Q?) può essere assimilata a un problema di classificazione binario:
  - False Positive e False Negative in questo caso prendono il nome di False Acceptance e False Rejection.
  - Pertanto False Positive Rate (FPR) e False Negative Rate (FNR) prendono il nome di False Acceptance Rate (FAR) e False Rejection Rate (FRR).

- L'identificazione (es. questa impronta digitale è nel database delle impronte dei criminali?) è un problema open set multi-classe.
  - Gli errori sono FPIR (False Positive Identification Rate) e FNIR (False Negative Identification Rate).

[1] D. Maltoni et. al, Handbook of Fingerprint Recognition, Springer 2022

Convergenza

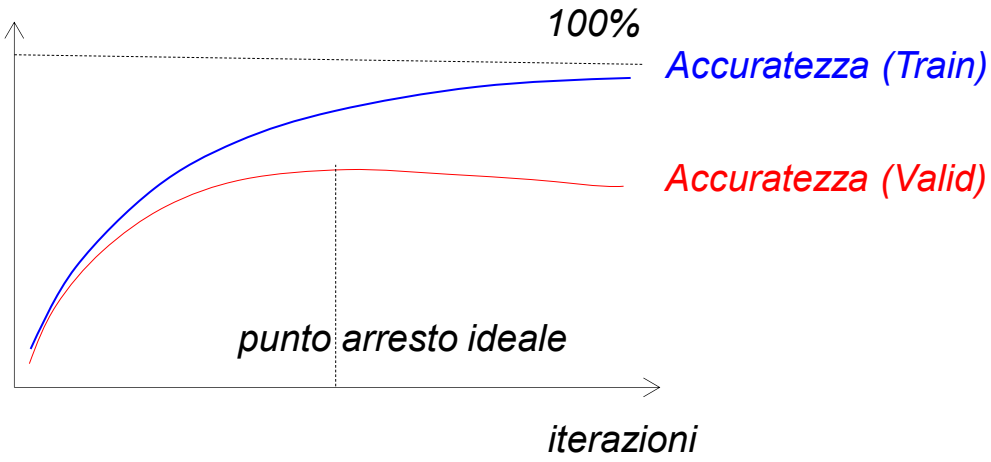
- Il primo obiettivo da perseguire durante l'addestramento è la convergenza sul Train set. Consideriamo un classificatore il cui addestramento prevede un processo iterativo. Si ha convergenza quando:
  - La loss (output della loss function) ha andamento decrescente.
  - L'accuratezza ha andamento crescente.



- Note:
  - Se la 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 la loss descresce 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 liberta 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 (early stopping).

Controllare l'Overfitting

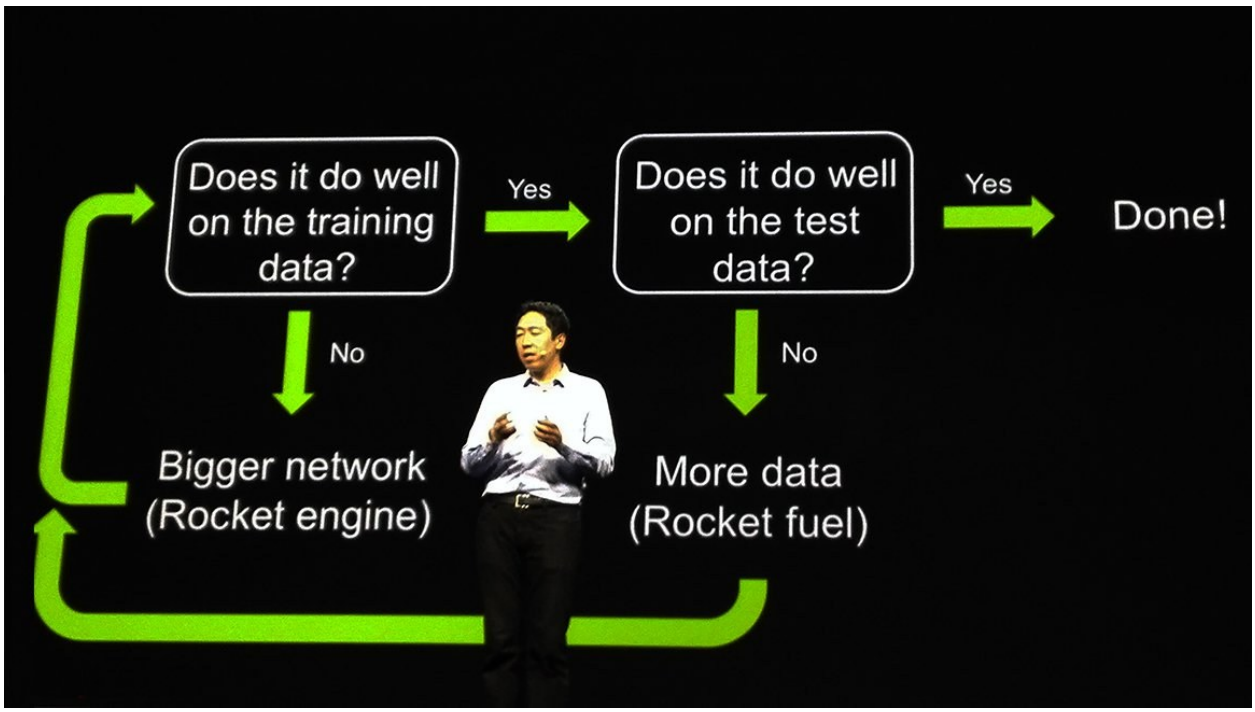
per massimizzare la Generalizzazione

- I gradi di liberta del classificatore non devono essere eccessivi, ma adeguati rispetto alla complessità del problema.
  - Buona norma partire con 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 Einsten

- I gradi di liberta 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/

Explainable AI (XAI)

- Una delle critiche maggiori rivolte ai modelli di machine learning (in particolare nel deep learning) è il loro funzionamento «black box» che non permette di capire il perché di certe decisioni.
- Se da un lato è vero che la complessità dei grandi modelli rende complessa l'analisi dell'elaborazione che ha portato a una certa decisione, dall'altro sono oggi disponibili tecniche e tool efficaci per interpretare le decisioni in diversi settori applicativi.
- Alcuni modelli (es. modelli lineari, alberi decisionali) rendono disponibili nativamente metriche (post-training) per quantificare l'importanza delle feature usate a livello globale (cioè indipendentemente dal singolo data point).
- Tecniche come Lime e Shap permettono un'analisi locale (model independent) del singolo data point in inference, ovvero permettono di capire l'importanza delle singole feature nella sua classificazione.
- Per la classificazione di immagini esistono tecniche (tipo Grad-CAM) che permettono di evidenziare i pixel che hanno avuto ruolo determinante nella classificazione.

Fonte: https://towardsdatascience.com/understand-your-algorithm-with-grad-cam-d3b62fce353

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 (preferibilmente utilizzando Cross-Validation).
  - evitate di concentrarvi solo su casi troppo semplici (ad esempio rimuovendo iterativamente i pattern che il vostro approccio non riesce a gestire).
  - evitate overfitting del test set (cherry picking)
- 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 dubug incrementale e unit testing. Gli algoritmi di Machine Learning non sono «esatti» e trovare bug nel codice può essere molto difficile!

Come si vince una competizione?

- Kaggle è una delle più popolari piattaforme dove sono condivisi benckmark di machine learning e si organizzano competizioni su domini diversi (anche con ingenti premi in denaro).
- Il più vincente tra i partecipanti della community (Kaggler #1 nel 2018) ha svelato in un'intervista le basi del suo approccio (sistematico):
  - Read the overview and data description of the competition carefully
  - Find similar Kaggle competitions. As a relatively new comer, I have collected and done a basic analysis of all Kaggle competitions.
  - Read solutions of similar competitions.
  - Read papers to make sure I don't miss any progress in the field.
  - Analyze the data and build a stable CV (Cross-Validation).
  - Data pre-processing, feature engineering, model training.
  - Result analysis such as prediction distribution, error analysis, hard examples.
  - Elaborate models or design a new model based on the analysis.
  - Based on data analysis and result analysis, design models to add diversities or solve hard samples.
  - Ensemble (e.g. multi-classifiers).
  - Return to a former step if necessary.

Model vs Data Optimization

- Molto tempo viene speso dai progettisti di applicazioni di ML per l'ottimizzazione del modello e dei suoi iperparametri (probabilmente perché questa è la fase più divertente).
- Nelle applicazioni pratiche però è più conveniente dedicare tempo all'ottimizzazione dei dati. A parità di tempo investito i benefici sono maggiori.

  Systematic improvement of data quality on a basic model is better than chasing the state-of-the-art models with low-quality data [1]

- Per miglioramento della qualità dei dati si intende:
  - La loro pulizia (data cleaning)
  - L'identificazione dei failure del sistema e la collezione di nuovi dati per i casi critici.
  - Tecniche di data augmentation (anche con generazione di dati sintetici).
  - Tecniche di supporto all'etichettatura automatica dei dati (etichettatura incrementale, self-labelling).
  - Uso/sviluppo di tool efficienti per la collezione, pulizia ed etichettatura dei dati.

[1] Big Data To Good Data: Andrew Ng Urges ML Community To Be More Data-Centric And Less Model-Centric

Analisi dei Requisiti

[tradotto da: Andrew Ng - The Batch - May 2022]

- Un cliente (non esperto) di machine learning, durante la fase di analisi dei requisiti, potrebbe sorprendersi quando gli dite che:
  - Non sappiamo a priori quanto sarà accurato il sistema.
  - Potrà essere necessaria una fase iniziale (costosa) di data collection.
  - Successivamente potrebbe essere necessario acquisire ulteriori dati (anche iterativamente).
  - Il sistema iniziale sviluppato potrebbe performare bene in laboratorio, ma non generalizzare in produzione (data drift, concept-drift)
  - Le prestazioni del sistema sviluppato potrebbero peggiorare nel tempo (data drift): è possibile dover re-investire per aggiornarlo (importante monitorare).
  - Il sistema potrebbe mostrare bias (es. comportamenti disomogenei per certe categorie, anche eticamente antipatici) difficili da prevedere e rilevare.
  - Può essere difficile capire perché il sistema ha fornito un certo output in corrispondenza di un dato input (black-box).
  - Nonostante il generoso budget del cliente, probabilmente non si raggiungerà AGI (Artificial General Intelligence)

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.

I principali framework per il deep learning (tutti con interfaccia Python) sono:

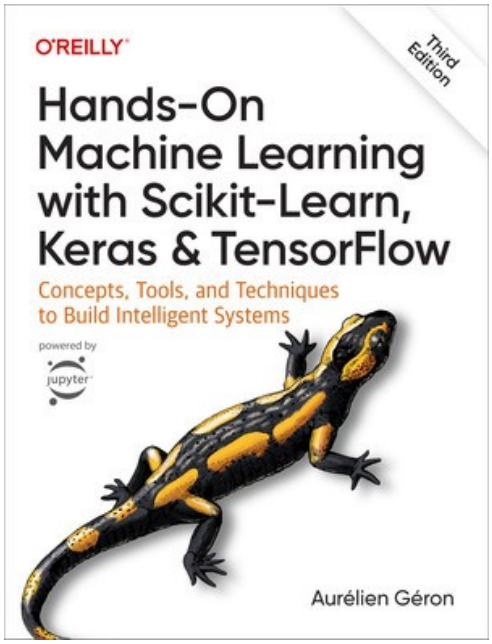
- Tensorflow* (Google). *Il più noto e diffuso in ambito business*
- PyTorch (Facebook). *Molto apprezzato in ambito ricerca*
- Caffe (Berkeley). *Superato, ma efficiente per Visione Artificiale*

** utilizzati per le esercitazioni di questo corso*

Altre risorse

- Nessun libro di testo ufficiale per la parte teorica del corso:
  - le slide coprono tutto il programma
  - approfondimenti e link a siti web forniti di volta in volta
- Per la parte pratica (esercitazioni):
  - testo consigliato:

nel seguito: [A. Géron]



Terza edizione del libro: evitare se possibile edizioni precedenti

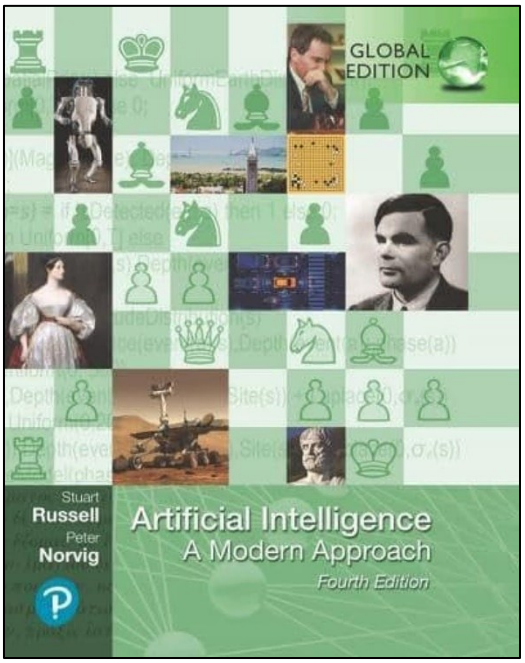
link Github esempi del libro

- Lacune in algebra lineare e probabilità ?
  - vedi cap. 2 e 3 del libro «Deep Learning» disponibile online a: https://www.deeplearningbook.org/

Altre risorse (2)

- Sulle tematiche di AI classica (AI simbolica), il testo didattico più noto e usato è:

Artificial Intelligence: A Modern Approach, Global Edition



Quarta edizione del libro, consigliata versione inglese.

Nella versione italiana suddivisione in due libri (primo su AI classica, secondo su machine learning)