Refresher

Parte I

Indice

Capitolo 1

Basics

1.1 Introduzione al decision making

Il *decision making* è una parte integrante del ruolo di gestione; i dirigenti prendono centinaia di decisioni ogni giorno.
La *decision science* è una disciplina che usa tecniche quantitative per il processo decisionale, ovvero suggerisce quale decisione prendere in base a tecniche quantitative; è usata da ricerca operativa, statistica, inforamtica ...
Per poter applicare queste tecniche, spesso è richiesto una elaborazione dei dati grezzi in dati elaborati per ottenere un risultato.
Il decision making ha 3 proprietà:

- **Accuratezza**
- **Spiegabilità:** è la capacità di spiegare un risultato
- **Accettabilità:** la procedura per passare dal dato grezzo al risultato è accettata da tutte le parti coinvolte

1.1.1 Data exploration

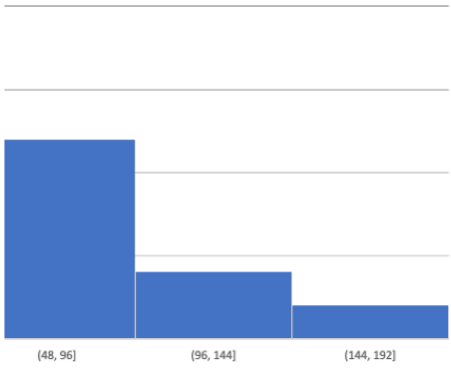
Supponiamo di avere un dataset di dipendenti, e di voler trarre una decisione sul licenziare o meno un dipendente in base all'assenteismo; vogliamo trarre delle informazioni dai dati usando delle semplici *statistiche descrittive*.
Nel nostro dataset troveremo sia dati **categorici** (etichette) che **numerici**.

Bucketizzazione

È un'operazione che permette di trasformare un dato numerico in uno categorico: l'insieme dei valori viene suddiviso in intervalli disgiunti; successivamente, ciascun valore viene sostituito con l'etichetta del suo intervallo.

Istrogammi e distribuzione

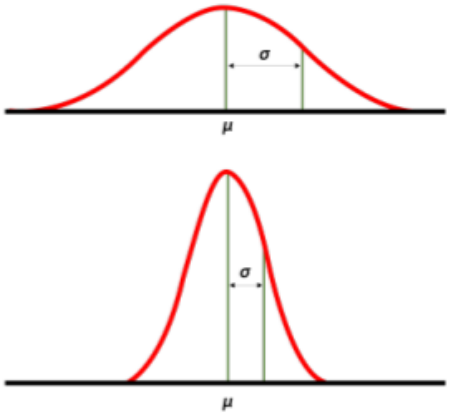
Una volta fatta la bucketizzazione, è più facile la visualizzazione grafica della distribuzione dei dati tramite istogrammi.



Deviazione standard campionaria

La deviazione standard misura quanto i valori numerici di una caratteristica sono dispersi rispetto alla media.
Se il grafico ha una forma a campana simmetrica:

- σ piccola indica valori vicini alla media
- σ grande indica valori lontani dalla media, valori dispersi



La formula prevede di calcolare la varianza di ciascun valore (distanza dalla media al quadrato) divisa per il numero di valori, il tutto sotto radice.

Mediana

La mediana è quel valore che divide l'insieme a metà; è quel valore per cui il 50% dei valori è più piccolo e l'altro 50% è più grande.

Quartili

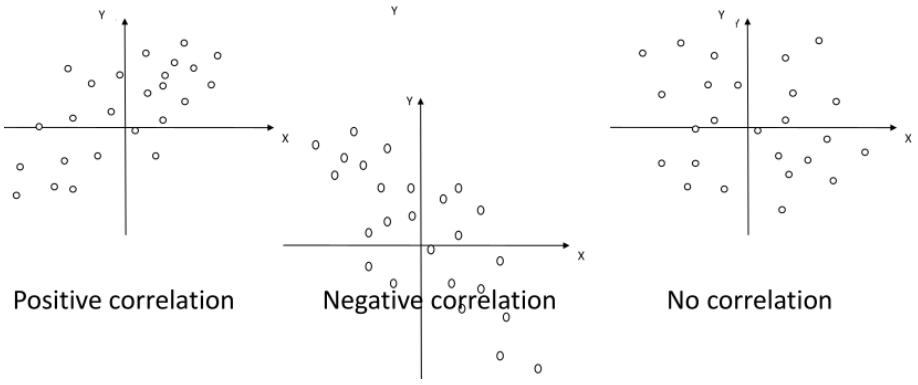
- **Primo quartile:** valore x tale che 25% dei valori è minore di x
- **Secondo quartile:** valore x tale che 50% dei valori è minore di x
- **Terzo quartile:** valore x tale che 75% dei valori è minore di x

$\Rightarrow$ il secondo quartile coincide con la mediana

La **differenza interquartile** (differenza tra terzo e primo quartile) è utile per fare analisi su dati la cui distribuzione non è a campana simmetrica, dato che la deviazione standard ha senso di essere usata solo su dati che seguono quella distribuzione (altrimeniti la media non sarebbe un valore tipico, non avrebbe senso misurare la distanza da essa).

1.1.2 Relazione tra due feature

Ci si può chiedere se due colonne di un dataset siano correlate, ovvero se sono legate a tale punto che mi basta saperne una per conoscere anche l'altra.
Si possono mettere le due caratteristiche su un piano e rappresentarle graficamente per vedere se esiste una correlazione:



È possibile usare due metriche:

- **Varianza:** dà informazioni sulla variabilità interna di una variabile
- **Covarianza:** dà informazioni su come due variabili variano insieme; mi dice quanto il variare di una influenzi il variare dell'altra
  - *Positiva:* le variabile variano nella stessa direzione

- *Negativa:* le varaibili variano nella direzione opposta
- *Zero:* nessuna correlazione

La covarianza viene usata per ottenere un numero tra -1 e 1 (tramite il *coefficiente di correlazione di Pearson*) che mi dice quanto due variabili sono correlate tra loro.

- due variabili si dicono indipendenti se non condividono informazioni
- se sono dipendenti, la *quantità di informazione* condivisa si stima con la correlazione

Bisogna fare attenzione a fare una distinzione tra correlazione e causa. Potrebbe sembrare che due correlazioni siano correlate solo perchè sono correlate ad una terza grandezza; è il caso ad esempio di suicidi e mangiare aringhe.

Distribuzioni asimmetriche

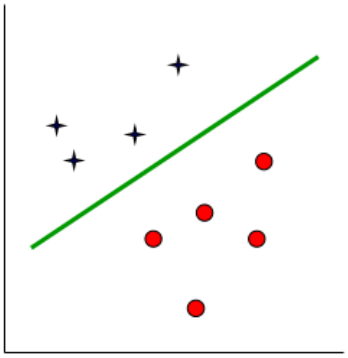
Quando un istogramma è costruito su valori che seguono una *distribuzione normale*, allora la forma delle colonne sarà *a campana*; tuttavia, in casi reali gli istogrammi sono spesso asimmetrici.

1.1.3 Decision Making via algorithms

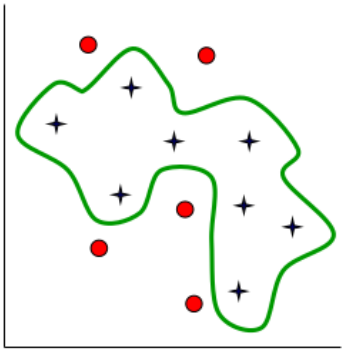
Per quale motivo si cerca di rispondere ad un problema tramite un modello supervisionato? Non basterebbe avere un algoritmo?
Gli algoritmi applicano una **teoria** alle grandezze (ad esempio la deteriorità di un motore elettrico dovuta alla temperatura); se ho una teoria, la posso **esplicitare con un algoritmo:** ho una funzione che lega input e output; non ho più bisogno di avere esempi, ma ho la teoria che mi dice che la deteriorità del motore è dovuta al numero di ore, ad una determinata temperatura...
Molti problemi non possono essere risolti in questo modo, dato che non c'è un'equazione esplicita; la devo *tirare fuori* dall'addestramento del modello.

1.1.4 Decision Making via Supervised Machine Learning

Non viene usata una formula e algoritmo che descrive in modo esplicito un fenomeno per derivare l'informazione per prendere una decisione. Viene usata una **formula generica con dei parametri, che vengono regolati per minimizzare l'errore**. Si tratta dunque di trovare i valori giusti dei parametri, viene fatto nella fase di addestramento.
Viene associato a ciascun valore un peso; alla somma pesata viene poi aggiunta una costante a cui *viene applicato un sogliatore*; in questo modo l'output è una funzione non lineare, invece che una semplice somma pesata.
La procedura di addestramento consiste nel **considerare l'errore come funzione dei pesi**, andando poi a modificarli per minimizzarli.
Un singolo neurone può approssimare solo una funzione lineare:



Una *rete neurale multi-livello* può teoricamente approssimare qualsiasi funzione dopo una certa soglia (*diventa Touring-complete*):

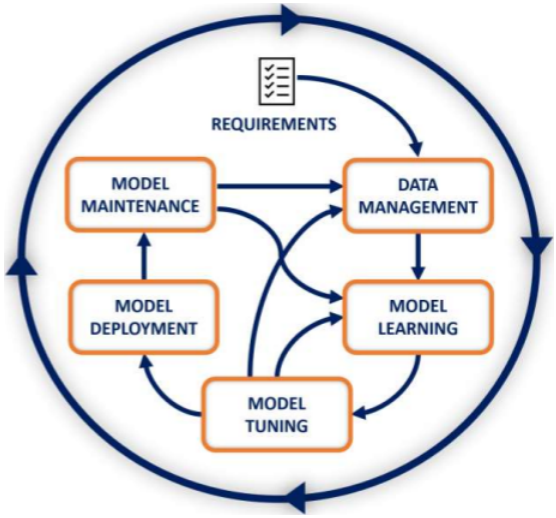


1.1.5 Ciclo di vita di un modello di ML

L'addestramento è solo una delle parti di vita di un modello di machine learning:

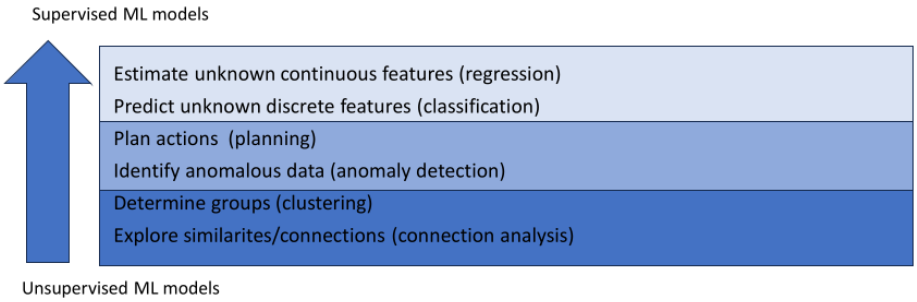
- **Data Management:** costruzione del dataset, controlli di vario tipo, eliminare colonne correlate tra loro ...
- **Model Learning:** addestramento del modello
- **Model Tuning:** fase post-addestramento in cui si danno al modello dati che non ha mai visto
- **Modelo Deployment:** il modello viene messo in produzione nel caso in cui soddisfi i requisiti; altrimenti, viene riportato indietro alle fasi precedenti
- **Model Maintenance:** si può avere un degradamento delle prestazioni perchè cambiano le distribuzioni dei dati in ingresso (è preferibile che siano gaussiane e stazionarie); devo fare gli opportuni controlli

Ciascuna di queste fasi può subire un attacco.



1.2 Introduzione a Machine Learning

- I modelli supervisionati vanno addestrati; sono soggetti ad attacco in fase di inferenza ed in fase di training
- I modelli non supervisionati processano i dati sulla base delle loro proprietà, non su esempi che hanno visto; ad esempio, le tecniche di clustering sono basate sulla definizione di distanze intragruppo e intergruppo. Non esiste una fase di training in cui possono essere attaccati



Regressore e classificatore fanno parte delle task di completamento.

1.2.1 Paradigma Supervised Learning

Il problema è **imparare una relazione tra input e output a partire da degli esempi**; viene prima addestrato il modello e poi se ne ottiene uno (adde-

strato) che può essere messo in produzione. In sede di addestramento il modello viene regolato per cercare di **minimizzare l'errore**.
Può essere attaccato, per cercare di violare alcune proprietà (accuratezza, ma non solo), posso cercare di fargli sbagliare un input in particolare, posso fare violazioni di privacy cercando se un determinato elemento appartiene al training set ...
L'idea per creare un modello è:

- viene dato un input, e il modello produrrà un output sicuramente sbagliato
- si va a modificare i pesi per cercare di ridurre l'errore
- si continua così fino a che ottengo dei buoni risultati che minimizzano l'errore
- una volta finito questo processo, il modello non ancora pronto, perchè funziona bene sui dati che ha già visto

  $\rightarrow$ devo fare una fase di testing sui dei valori che non ha mai visto (*test set*)

Supervised ML Training in a nutshell

- Il modello interno più semplice è una somma pesata; si ha un input multivariato, dove ogni variabile viene moltiplicata per un peso per poi fare la sommatoria
- A questa somma pesata viene poi applicata una soglia (ad esempio, se è sopra la soglia considero il risultato benigno mentre se è sotto considero maligno)
- Quando ottengo il risultato, viene confrontato con quello reale (siamo nella fase di training), e se ho fatto un errore si va a modificare i pesi cercando di diminuire l'errore
- Si continua tante volte in questo modo sul training set, sperando che l'errore continui a diminuire sempre di più fino a fermarmi nel punto di minimo ($gradiente = 0$, ovvero la derivata delle funzioni multivariate)

All'input viene aggiunta una costante che prende il nome di **bias**, che aiuta a migliorare l'output riducendo la varianza e beneficiando all'accuratezza. Il training *decide la quantità di bias* che massimizza l'accuratezza.
Mettendo più strati di neuroni (l'output di uno va in input ad un altro) si riesce a fare un addestramento più rapido; con 3 o più strati si dice che la rete è *Touring completa.*

Error backpropagation

Uno dei problemi della rete neurale tradizonale è quello di non riuscire a fare una corretta *error backpropagation*, ovvero non si riesce a distribuire l'errore anche sugli strati interni.
Per arginare questo problema si usano i primi strati per modificare la rappresentazione degli input, comprimendoli in una parte più piccola, mentre sono solamente gli ultimi quelle che vengono effettivamente addestrati.

Overfitting

Bisogna fare attenzione all'*overfitting*: significa che il modello commette errore pari a 0 sui dati del training set, ma che su dati mai visti prima commette un grande errore; questo accade perché il modello non è in grado di generalizzare.
Per evitare che un modello vada in overfitting, generalmente un il training viene fermato prima di raggiungere un'accuratezza perfetta sui dati di training.

1.2.2 Diversi modelli di Machine Learning

- **Classificazione:** mappa vettori di caratteristiche in *categorie* o *classi*
- **Anomaly Detection:** serve a capire se un input è anomalo rispetto a quello atteso
- **Predizione:** un classificatore può anche agire da predittore, come ad esempio prevedere l'esito di un azione. Per fare il training gli do sole la parte iniziale dei dati e li faccio classificare come successo/insuccesso
- **Planning:** sono adatti in cui non riesco a calcolare l'errore per ogni singolo input; ad esempio, se gioco a scacchi è difficile classificare una singola mossa come buona o cattiva...

  $\rightarrow$ si usa un sistema di *reward*, non calcolato sulla singola mossa ma su una sequenza di mosse
- **Connection Analytics:** analizza la relazione tra utenti/sistemi; fondamentale per la sicurezza (ad esempio, individuare nodi centrali o vulnerabili)

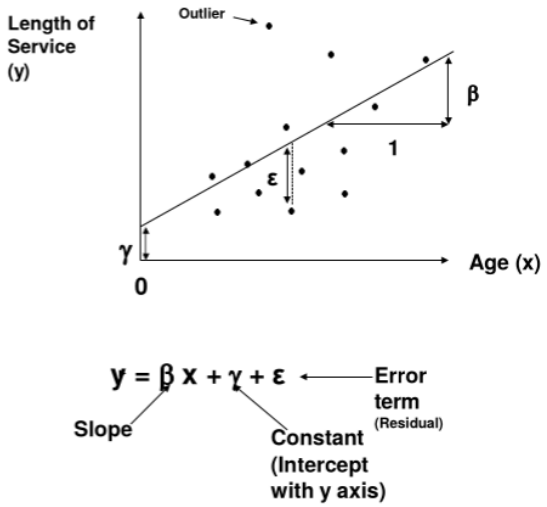
$\Rightarrow$ lo stesso problema può essere risolto con modelli diversi; la scelta dipende dai dati disponibili

Capitolo 2

AI-Based Decision Making

2.1 Regressore

Il regressore è una sorta di *via di mezzo* tra un classificatore e un anomaly detector; utilizza il residuo quadratico per restituire valori continui invece che una singola eticheta (data una x che non ha mai visto prima, mi restituisce la sua y).
Fare la regressione significa trovare la retta (o iperpiano) che minimizzi la somma dei quadrati delle distanze dei punti dalla retta.



Qualità del regressore

Se mi accorgo che il regressore restituisce costantemente valori *troppo più alti (o bassi)*, allora significa che l'intercetta con l'asse x del regresspre è posizionata troppo in alto (o in basso) rispetto ai punti futuri. Si dice in questo caso

che *sto usando un bias*; si può *aggiustare* la retta aggiungendo i nuovi punti o rimuovendo quelli più vecchi.

Validazione

Per evitare overfitting e prendere il miglior regressore possibile, si ha bisogno di due set:

- **Training set:** usato per minimizzare l'errore quadratico
- **Test set:** verifico che errore ho ottenuto e guardo se c'è un bias

Bias-Varianza trade off

L'errore dipende sia dalla varianza che dal bias; si cerca di minimizzare entrambi, ovvero che sbaglio poco (varianza) e quel poco che sbaglio è ben distribuito (bias).
Purtroppo però quando muovo la retta per minimizzare l'errore, non riesco a minimizzare entrambi: al diminuire di uno cresce l'altro.

2.1.1 Regressione Ridge e Lasso

Sono dei metodi di regressione che introducono dei vincoli sui coefficienti del regressore, portando ad avere un regressore più sofisticato; cercano di trovare un compromesso tra varianza (quanto sbaglia) e bias (in che direzione sbaglia).

Regressione Ridge

Cerca di limitare la dimensione dei coefficienti, definendo un parametro λ pari alla somma dei quadrati dei coefficienti.
L'obiettivo è cercare di impedire ai punti di training di generare un predittore con un bias troppo alto per aver voluto minimizzare l'errore. Inoltre, la tutti i coefficienti devono avere valori diversi da 0, perchè voglio che il regressore prenda in considerazione tutte le variabili.
Il parametro λ viene scelto usando un set di validazione: lo scelgo sui dei dati selezionati di cui mi fido, per poi usarlo su tutto il training set.

Regressione Lasso

È simile alla *ridge regression*, con la differenze che è permesso mettere i valori dei coefficienti pari a 0. La regressione Lasso impone che la somma dei valori assoluti dei coefficienti sia minore di un certo valore, portando alcuni coefficienti ad essere pari a 0.
Questo può portare ad avere dei modelli più semplici che non includono alcune features.

2.1.2 Regressore multivariato

Si passa ad un modello multivariato; l'output (la y) dipende da una combinazione lineare degli input.
Le condizioni per aver un buon regressore multivariato sono che:

- gli errori siano indipendenti
- la distribuzione degli errori è gaussiana
- la varianza è costante

Una volta ottenuto un predittore, lo posso usare e sarà molto più rapido di un'inferenza tramite rete neurale.

2.1.3 Coefficiente di determinazione multipla

Rappresenta la spiegabilità del risultato di un regressore, ovvero fornire una descrizione dei fattori che hanno influenzato la decisione.
Il modo più semplice è fare l'analisi di quali feature hanno influenzato di più la decisione: viene fatto tramite il coefficiente r^2 calcolabile nel seguente modo:

$$r^2 = \frac{SSR}{SST} = \frac{\text{regression sum of squares}}{\text{total sum of squares}}$$

Questo coefficiente mi dice quanto è collettiva l'influenza delle variabili: magari ce n'è una che conta tantissimo e un'altra che è ininfluente. Mi dice quanto è collettiva la decisione, se dipende da tutte le variabili o se solo da un loro sottoinsieme.
Esistono test statistici per scartare le variabili che sono poco influenti: se un coefficiente è basso vuol dire che influenza poco, lo potrei scartare per avere un regressore più semplice.

2.1.4 Interazione tra variabili

Avviene quando sospetto che l'influenza di una variabile dipende dal valore delle altre: ad esempio, sospetto che l'influenza su y di x_2 dipenda dal valore di x_1.
Si può modellare introducendo i **prodotti incrociati**; è un'interazione tra le variabili di input, si ottiene una combinazione non lineare.

$$\hat{Y} = b_0 + b_1X_1 + b_2X_2 + b_3X_3$$
$$= b_0 + b_1X_1 + b_2X_2 + b_3(X_1X_2)$$

2.2 Basic ML models

L'obiettivo di un modello di machine learning è aprossimare una funzione f che non conosciamo. Ciò che cambia è il dominio e codominio:

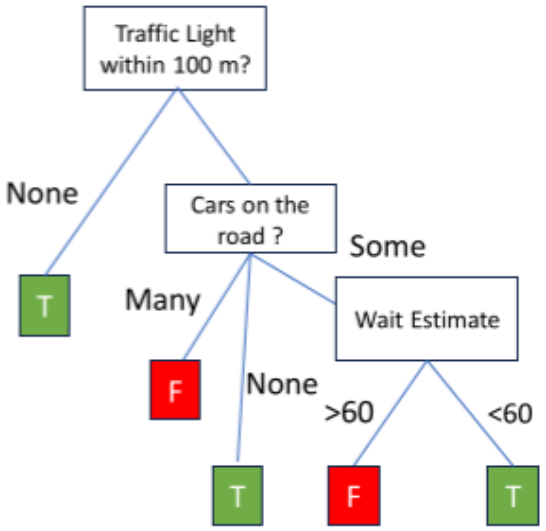
- se il codominio è un set finito di etichette ⇒ classificatore
- se il codominio è un numero reale ⇒ regressore
- se il codominio è un valore ⇒ predittore

2.2.1 Alberi di decisione

Gli alberi di decisione sono un modello di regressione o classificazione sotto forma di albero. L'albero viene costruito dividendo il dataset in sottoinsiemi sempre più piccoli, fino ad ottenere un albero composto da:

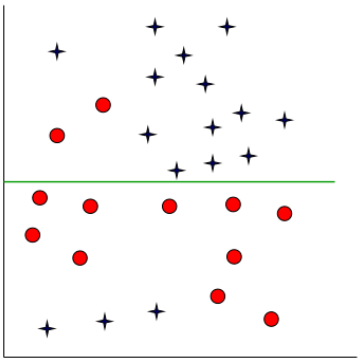
- **nodi decisionali**
  - hanno due o più rami, ciascuno corrispondente a un valore della feature in esame
- **nodi foglia**
  - rappresentano l'esito finale della decisione, cioè la predizione di un valore numerico (nel caso di regressore) o della classe (nel caso di classificatore)

Possono gestire sia dati numerici che categorici.

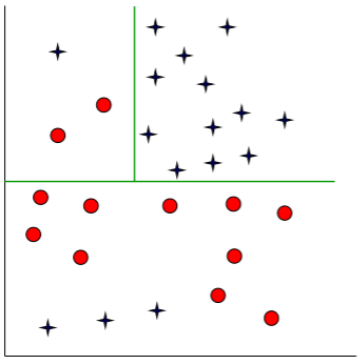


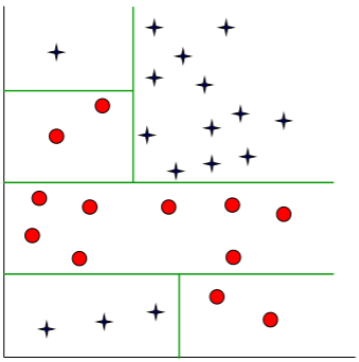
Un modo per costruire l'albero ed ordinare gli attributi è quella di farlo per **entropia massima**: dovrei cercare di prendere per primo quello con l'entropia più alta. Quando invece faccio la bucketizzazione, dovrei cercare di minimizzare l'entropia.

Esempio animato



Ciasuno dei due insiemi che ho ottenuto sono il più possibile *tutti uguali* $\rightarrow$ ho ridotto l'entropia

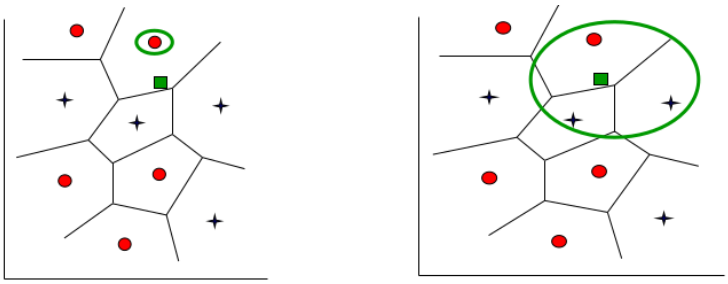




Ciascun sottoinsieme ha dati con la stessa etichetta; in questo caso l'albero di decisione fa zero errori sul training set, anche se devo stare attento all'overfitting.

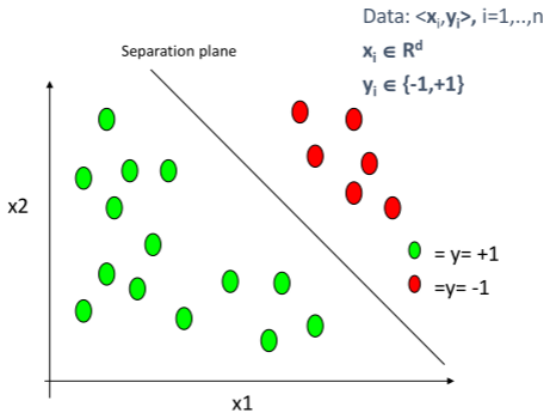
2.2.2 Nearest Neighbor Models

Si può dire che il modello è il training set, dato che non devo addestrare niente ma una nuova etichetta viene stimata a partire dal training set; quando mi arriva un nuovo punto (ad esempio un dipendente con età e anni di servizio), vado a vedere il punto più vicino a lui nello spazio multidimensionale, e uso la sua stessa etichetta.
Una estensione è quella di usare i k punti più vicini.
Nell'esempio in figura, il nuovo punto è il quadrato; nel caso a sinistra dirò che è un cerchio, mentre nel caso a destra ($k = 3$) dirò che è una croce.



2.2.3 Linear Support Vector Machines

È un metodo di classificazione in cui prendo il training set e cerco se trovo un iperpiano (nell'esempio una retta) che divida in due training set. Vengono calcolati i vettori di supporto (distanze delle coppie rosso-verde con distanza minima) e viene calcolato l'iperpiano che li taglia nel modo più ortagonale possibile.



I punti di training usati per generare il modello sono relativamente rispetto ad altri modelli, come le reti neurali.
Dato che non è sempre possibile determinare una separazione netta delle feature, è possibile definire nuove variabili in funzione delle altre con funzioni non lineari. Ad esempio, posso usare una funzione non lineare per trasformare le variabili in uno spazio nuovo (ad esempio da due a tre dimensioni), dove è più facile calcolare l'iperpiano di seperazione.

Capitolo 3

Gestione dei dati

3.1 Managing Data for AI

Nel procedimento per passare dai dati grezzi a quelli per il training ci sono tutta una serie di operazioni da fare; assumono particolare importanza quelle di *compliance*, tra cui quella più importante è quella per la privacy. Vengono fatte sia per migliorare la qualità del dato che per introddure rumore per questioni di privacy.
Alcune tra le principali fasi di una *data lake*, ovvero una infrastruttura per la gestione di informazioni da dare in pasto a modelli di ML, sono:

- **Acquisizione** dei dati
- **Integrazione** dei dati; ad esempio, se mancano dei dati vado a riempire usando dei dati plausibili (posso prendere un punto vicino per stimare quello che manca), per rendere i miei dati più completi. Viene anche introdotto rumore per questioni di privacy
- **Storaging**
- **Analytics**

3.2 Migliorare la qualità dei dati

3.2.1 Ingegneria delle feature

I modelli neurali di grandi dimensioni riescono a lavorare sui dati grezzi, mentre i modelli semplici (come quelli visti fino ad adesso) lavorano su feature che hanno subito un **processo di analisi**: diminuire la covarianza, togliere feature spurie . . . operazioni finalizzati ad ottenere delle features con una **distribuzione gaussiana**; lavorano su dati con meno dimensioni, le cui dimensioni sono state selezionate a mano.

Se non faccio un processo di *feature engeneering* il training sarà molto più oneroso, mentre se viene fatta il training sarà molto più efficacie.

Alcuni metodi di selezione delle features

- **Filtraggio:** vengono tolte le features che si possono togliere
- **Wrapping:** viene usato un modello per generare delle coordinate in più da affiancare a quelle originali; data una feature, il modello mi dice una pseudofeature che poi verrà data in pasto al modello semplice.

  → Più features vengono **aggregate** in una pseudofeature, in modo che sarà più semplice da addestrare; il modello da addestrare in questo modo avrà una dimensionalità più bassa del problema
- **Embedding:** vengono scelte solo alcune feature, scartandone altre sulla base dei valori (ad esempio sulla base di valori statistici come la covarianza); è una sorta di filtro dinamico

Filtraggio univariato

È un metodo di filtraggio usato per filtrare una caratteristica tra poche; non si presta a dei dati *multidimensionali*.
Il procedimento è il seguente:

- calcola la correlazione tra tutte le possibili coppie
- per tutte le coppie la cui correlazione è alta, scarta una delle due caratteristiche in modo casuale
- si ripete il procedimento

Non è efficiente nel caso feature elevate perchè si devono calcolare tutte le possibili combinazioni.

3.2.2 Valori mancanti

Il modello deve avere colonne con la stessa densità; non ci devono essere colonne troppo vuote o troppo piene rispetto alle altre.
Nel caso di valori mancanti, si possono modelli come il *k nearest neighbor* per stimarli.

Capitolo 4

Classificazione e Predizione

4.1 Problemi di classificazione

i modelli di classificazione generalmente sono modelli neurali, dove l'input è una codifica vettoriale e l'output è la classe a cui l'input appartiene; durante il training i parametri del modello vengono adattati per ottenere un errore accettabile.
Tuttavia, per valutare un modello non basta considerare solo l'errore, ma bisogna tenere in considerazione altri fattori come:

- dimensione e complessità del classificatore ottenuto
- quantità di dati di training

Confusion Matrix

Viene utilizzata per mostrare per ciascuna riga e colonna gli errori nella classificazione:

classified as	Sameplace	Different Place	<--
Sameplace	40	10	
Differentplace	6	24	

Total error = 16/80 = 20%

Curve ROC

Si ottiene riportando falsi positivi e falsi negativi su un piano (quando si tratta di un classificatore binario). Dove è preferibile posizionarsi su questa curva nella fase di training dipende dal contesto e dall'impatto di ciascuno dei due errori (gli errori non hanno tutti lo stesso peso...).

Cross-Validation

Consiste nel dividere il dataset in un 80% usato per il training, e il restante 20% per fare validation; questo processo viene ripetuto più volte fino al raggiungimento di un'accuratezza uniforme. È un modo per *approssimare* la capacità del modello di generalizzare, anche se bisogna tenere conto che i punti del validation test sono stati punti del training set in una fase precedente.
Con la metodologia *held-out* si usa un set mai visto prima per fare validation.

Precision e Recall

Precision equivale a dire *"quanti sono classificati correttamente tra tutti quelli classificati come X?"*

p = precision: rate of o correctly classified as o = 48/(48+3) = 94.12 %

confusion matrix	classified as o	classified as +
o	48	2
+	3	47

Recall equivale a dire *"tra gli elementi della classe X, quanti sono stati classificati come X?"*

confusion matrix	classified as o	classified as +
o	48	2
+	3	47

r = **recall** (or sensitivity to) = rate of o's detected = 48/50 = 96%

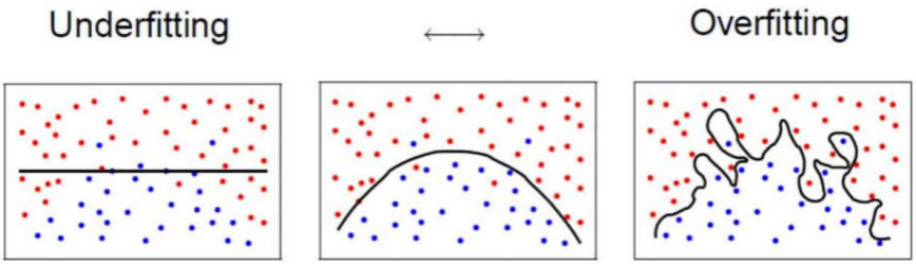
⇒ la differenza sta nel chiedersi *classificati correttamente rispetto a cosa? a quelli totali o a quelli della classe?*
Tipicamente, all'aumentare di una diminuisce l'altra; dato che si è interessati nel massimizzare entrambe le metriche, si può usare il parametro *F-score*, che è una sorta di media armonica delle due.

$$F = 2\,\frac{precision \times recall}{precision + recall}$$

Considerazioni sull'addestramento

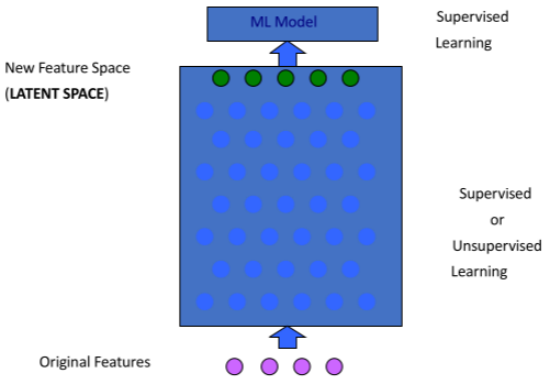
Conoscendo il costo dei diversi tipi di errori, potrei cercare di fare in modo di regolare i parametri per minimizzare il costo totale degli errori; bisogna sempre

fare attenzione a non andare in overfitting, ovvero quando il modello è troppo adatto al training set e non sarà in grado di generalizzare.



4.2 Deep Classifier Models

Quando si ha una rete a tanti livelli si ha il problema di dove ripartire l'errore sui vari pesi.
Un possibile approccio potrebbe essere quello del **training strato per strato:** vengono *bloccati* tutti gli strati tranne uno (nel senso che non vengono modificati i parametri), per poi pertubrare i pesi dello strato non bloccato nelle varie direzioni, scegliendo quella in cui l'errore diminuisce maggiormente.
Questo procedimento viene ripetuto più volte passando strato per strato, fino ad ottenere un modello che ha per parametri i vari pesi che sono stati addestrati separatamente; ovviamente, il procedimento deve essere ripetuto più volte, perchè magari quando modifico i parametri per uno strato sto generando errore *da un'altra parte*.
Questo metodo funziona discretamente con reti *shallow*, ovvero con pochi strati.
A metà anni 2000 è stato introdotto una nuova metodologia: consiste nel non fare training di tutti gli strati, ma di fare usare il training set con le label per addestrare solo gli ultimi due strati della rete; gli strati precedenti sono utilizzati per la **costruzione di uno spazio latente**. In altre parole, il loro scopo è **trovare un encoding dei punti di input che mantenga la loro correlazione statistica ma su un numero diverso di features**; non mira quindi a prevedere la label.
Qualunque sia la distribuzione di ingresso (si spera Gaussiana su più variabili), si va a cercare un encoding su meno features tale che sia anch'esso multivariato Gaussiano, e che la divergenza (la distanza tra le due distribuzioni) sia pari a 0: questo è ciò in cui consiste il training di tutti gli strati intermedi.
L'idea è che le features che escono (che sono meno) sono quelle su cui poi si addestrano gli ultimi strati.



Nella figura lo spazio latente è inflattivo, ovvero escono più features di quelle che sono entrate. Non deve essere per forza così, può essere anche deflattivo a seconda dei casi.
$\Rightarrow$ il deep learning consiste in:

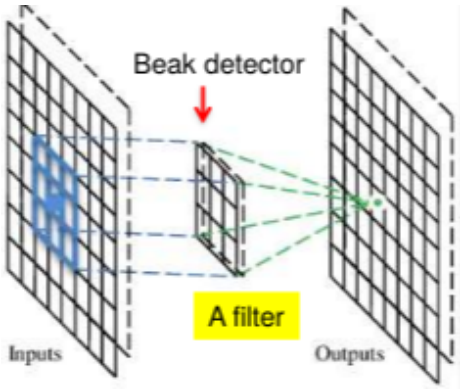
1. fase di pre-training degli strati iniziali (auto-encorders)
    - questa fase è non-supervisionata perchè non mi interessa delle etichette di classificazione, mi interessano solo le distribuzioni dei punti per calcolare la divergenza (sono parametri estraibili dal training set)
2. fase di training supervisionato sugli strati finali; è la classica classificazione con minimizzazione dell'errore

Unsupervised Pre-Training in a nutshell

Supponiamo di avere una rete di un solo strato con n input ed output, e k parametri intermedi con $k < n$
Viene fatto un training di *riproduzione esatta*, ovvero si addestra lo strato e riprodurre esattamente gli input; una volta addestrato il modello, si rimuove il layer di output.

Layer convoluzionale

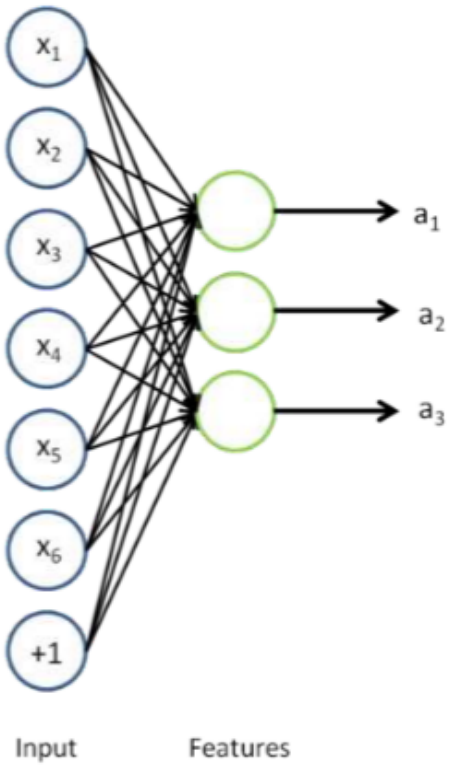
Una rete neurale convoluzionale è una rete neurale con alcuni strati convoluzionali: ciò che fanno è sommare dei gruppi di input per ottenere un layer a covarianza ridotta rispetto al layer originale.
Gli input sono collegati a dei filtri, e l'output del filtro (che è una sorta di somma pesata) è collegato al layer successivo. Si usano questi raggruppamenti di input come se fossero un encoder, ovvero si usano per ottenre in output un'altra rappresentazione dell'input.



Si usa una rete convoluzionale per eseguire un encoding; tuttavia, un encoding non è necessariamente convoluzionale.

4.3 Autoencoders

È un tipo di apprendimento non-supervisionato che trasforma l'*input raw* in *input latente*.



Si può che impara delle feature generiche dai dati di input, estraendo delle *macrofeatures*.
Possiamo fare una distinzione tra due tipi di encoder:

- **Deflattivi**, il cui scopo è comprimere l'input
- **Inflattivi**, il cui scopo è il aggiungere rumore ai dati di input

4.3.1 GAN - Generative Adversial Network

Sono modelli deep in grado di produrre dei sample X simili ai dati di input.
Si usa un **discriminatore** per valutare la qualità del dato generato da una GAN: restituisce 1 se lo considero vero, oppure 0 se lo riconosce come dato che è stato generato; avrò ottenuto un buon generatore quando sarà in grado di ingannare il discriminatore.

4.4 Modelli predittivi

Le serie temporali (dataset in cui una colonna è un timestamp) stanno alla base per poter fare un modello predittivo; consistono in dati misurati ad un tempo specifico, generalmente con intervalli regolari. L'analisi delle serie temporali cerca di **identificare fattori** che sono influenti sui valori della serie.
Una serie temportale si dice stazionaria se:

- la covarianza tra due punti è indipendente dal momento in cui sono stati osservati
- la media è costante nel tempo

Le serie stazionarie hanno i migliori predittori lineari, mentre per quelle non stazionarie risulta più complesso e lento implementare un predittore.

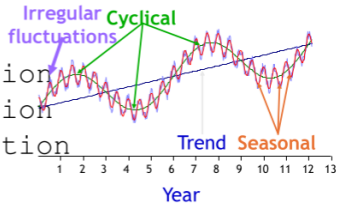
4.4.1 Trend

Il trend rappresenta l'andamento generale di una serie temporale su un arco di tempo sufficientemente lungo; è rilevabile con i dati disponibili solo se l'arco di tempo è abbastanza lungo da permetterlo. Una serie temporale può essere suddivisa in quattro componenti.

Trend secolare

Rappresenta l'andamento generale della serie nel lungo periodo; una volta individuato, permette di identificare le altre componenti più facilmente.

1. Secular trend
2. Seasonal variation
3. Cyclical variation
4. Irregular variation

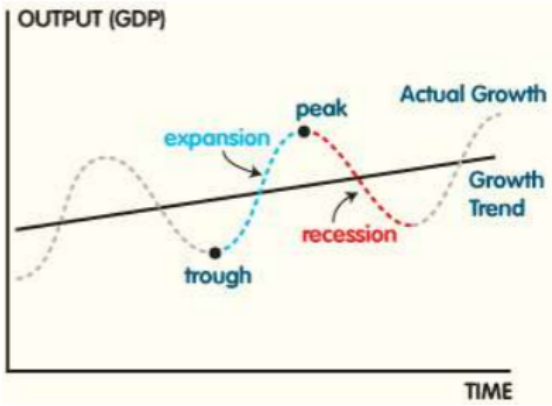


Variazione stagionale

È un pattern di cambiamenti che avviene ad intervallo di tempo regolari; sono spessi legati alle stagioni o ad eventi e festività, come ad esempio i casi di influenza in inverno.

Variazioni cicliche

Sono dei pattern che si ripetono ma con un intervallo di tempo *più irregolare* rispetto alle variazioni stagionali. Sono più complicati da individuare, ed è difficile prevedere per quanto si potrae il periodo di espansione o contrazione.



Variazioni irregolari

Sono delle variazioni casuali nella serie temporale, che non posso essere spiegate con i casi precedenti, come ad esempio dei crolli finanziari o catastrofi naturali.

4.4.2 Analisi dei trend

I trend passati stanno alla base per la predizione futura.

Semi-averages

Bisogna controllare prima che non ci siano outliers nei dati; la procedura è la seguente:

1. dividere i dati in due intervalli di tempo uguali
2. calcolare la media per ogni intervallo
3. tracciare una retta tra i due punti medi
4. estendere la retta nel futuro per fare previsioni

Medie mobili

L'idea è che se la medie è calcolata su intervallo di tempo sufficientemente lungo, l'effetto delle variazioni a breve termine sarà ridotto; in altre parole, le variazioni cicliche e stagionali saranno *smussate*, ottenendo un grafico più *morbido* che rappresenta il trend generale.
Il grado di smussamento può essere regolato selezionando quali valori includere nella media.