UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

DIPARTIMENTO DI INFORMATICA
GIOVANNI DEGLI ANTONI

Corso di Laurea triennale in
Informatica

ANALISI DEI DATI PER PROBLEMI DI MEDICINA LEGALE

Relatore: Prof. Dario Malchiodi
Correlatore: Prof. Anna Maria Zanaboni

Tesi di Laurea di:
Alessandro Beranti
Matr. Nr. 855489

ANNO ACCADEMICO 2018-2019

Questo lavoro è dedicato a tutti gli studenti

"Io studio,
ma studiate pure voi,
che se studio solo io non serve a un c. . . o"

– Gli scarabocchi di Maicol & Mirco

"No tale is so good
that it can't be spoiled
in the telling"

– Proverbio

Ringraziamenti

Questa sezione, facoltativa, contiene i ringraziamenti.

Indice

Capitolo 1

Introduzione

Questo documento ha una duplice funzione: da un lato mostra un esempio completo di elaborato finale redatto in LaTeX e conforme allo standard PDF/A, e dall'altro contiene suggerimenti e risposte a domande frequenti poste dagli studenti. Se ne raccomanda, pertanto, un'attenta lettura.

Capitolo 2

Machine Learning

Il termine Machine Learning, o apprendimento automatico in italiano, si riferisce alla capacità dei computer di apprendere e agire senza essere programmati esplicitamente. Gli strumenti di machine learning si occupano di dotare i programmi della capacità di "apprendere" e adattarsi agli input forniti. Al giorno d'oggi siamo circondati da tecnologie basate sull'apprendimento automatico:

- software che rilevano lo spam a partire dai nostri messaggi e-mail,
- motori di ricerca che imparano a ordinare i risultati di una query di ricerca al fine di mostrare prima quelli più rilevanti,
- transazioni con carta di credito protette da un software che impara a rilevare le frodi,
- fotocamere digitali che imparano a rilevare i volti,
- assistenti digitali che imparano a riconoscere i comandi vocali,
- veicoli autonomi addestrati a guidare senza intervento umano,
- applicazioni scientifiche come la bioinformatica, la medicina e l'astronomia.

2.1 Come funziona il Machine Learning

Nel Machine Learning vengono usati metodi statistici che applicano un procedimento di induzione a partire da dati che rappresentano istanze di un problema, accoppiate con una possibile soluzione, per migliorare le prestazioni di algoritmi o fare previsioni più accurate. La qualità e la dimensione dei dataset (collezione di dati), raccolti o resi disponibili e utilizzati nel processo sono fondamentali per il successo e l'accuratezza delle previsioni fatte.

Il processo parte da un domain set X, ovvero una raccolta arbitraria di dati, che rappresentano gli oggetti che si desidera etichettare. Essi possono essere già etichettati nel caso di apprendimento supervisionato come non esserlo nel caso non supervisionato. Questo set di dati rappresenta l'input dell'algoritmo di apprendimento che si è scelto di usare. Il risultato è un modello a cui è associato un errore di generalizzazione, ovvero la probabilità che non venga predetta l'etichetta corretta. Il domain set può essere diviso in due sottoinsiemi: il *training set*, insieme di dati che vengono scelti per compiere l'addestramento, e il *test set* che viene utilizzato per valutare le prestazioni del modello predittivo.

Una categorizzazione dei compiti del machine learning si ha quando si considera l'output desiderato del sistema che può essere di diversi tipi:

- classificazione, significa che il dato è categorico [1] e viene fatta una divisione dei dati in classi o etichette. Può essere una classificazione binaria, nel quale le etichette sono soltanto due, oppure multiclasse, etichette sono tre o più,
- regressione, usata per predire un valore assimilabile a una quantità che varia in un insieme continuo, per esempio il prezzo di una casa date la dimensione e la metratura, [1]
- clustering, in cui un insieme di input viene diviso in gruppi in modo che i singoli elementi siano simili agli altri punti dello stesso insieme e diversi dagli elementi degli altri; diversamente da quanto accade per la classificazione, i gruppi non sono noti prima, rendendolo tipicamente un compito non supervisionato. [2]

Gli algoritmi di apprendimento automatico vengono tipicamente organizzati in quattro categorie, a seconda della natura del "segnale" utilizzato per l'apprendimento o del "feedback" disponibile al sistema di apprendimento. Queste categorie, anche dette paradigmi, sono:

- Machine Learning supervisionato,
- Machine Learning non supervisionato,
- Machine Learning per rinforzo,
- Machine Learning semi-supervisionato.

2.1.1 Apprendimento supervisionato

Attraverso l'apprendimento supervisionato si cerca di costruire un modello partendo da dati di addestramento etichettati, a partire dai quali si tenta di fare previsioni

su dati non disponibili o futuri. Con il termine "supervisione" si intende quindi che nell'insieme dei campioni (o dataset), i segnali di output desiderati sono già noti poiché precedentemente etichettati. Nell'apprendimento supervisionato l'output può essere sia numerico che categorico, quindi può trattarsi rispettivamente di regressione e classificazione; siccome il tirocinio ha riguardato solo quest'ultima d'ora in avanti parleremo di classificazione.

Esistono molti algoritmi per svolgere apprendimento supervisionato, durante il tirocinio ho avuto modo di usare:

- Support Vector Machine,
- Decison Tree Classifier,
- Random Forest Classifier,
- Gaussian Naive Bayes,
- Linear Discriminant Analysis,
- Multi-Layer Perceptron Classifier.

2.1.1.1 Support Vector Machine

Il Support Vector Machine (SVM) è un algoritmo di apprendimento automatico supervisionato che può essere usato sia per scopi di classificazione che di regressione, si può applicare sia a problemi binari, sia a problemi multiclasse. Nel paragrafo seguente verrà descritto solo la versione per la classificazione binaria poichè è quella che ho usato durante il tirocinio.

L'SVM è basato sull'idea di riuscire a trovare un iperpiano che divida al meglio un set di elementi in due classi distinte [3]. Definiamo alcuni elementi che ci serviranno successivamente.

- Iperpiano: nel caso in cui si abbiano solo due dimensioni spaziali nel qualche si trovano le descrizioni degli oggetti da classificare x_1 e x_2, l'iperpiano è raffigurato come una linea che separa un insieme di dati [3]. Nel caso in cui le dimensioni siano 3, l'iperpiano è raffigurato come un piano, (cfr. Figura 1). Con più di 3 dimensioni viene definito "iperpiano".
- Support Vector: chiamati vettori di supporto in italiano, sono i punti che si trovano piu vicini all'iperpiano che divide i dati.
- Margine: è la distanza tra i vettori di supporto di due classi diverse. A metà di questa distanza viene tracciato l'iperpiano, (cfr. Figura 2).

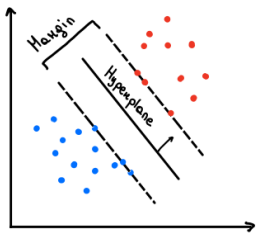

Figura 1: Iperpiano e margine

Il Support Vector Machine ha l'obiettivo di identificare l'iperpiano che divide meglio i vettori di supporto in classi. Esistono due algoritmi diversi a seconda se esista o meno l'iperpiano.

- Nel caso in cui esista e in particolare ce ne sia più di uno cerca quello con il margine più alto tra i vettori di supporto in modo da evidenziare meglio la divisione tra i dati. La massimizzazione del margine porta ad avere degli iperpiani che tendono a minimizzare le probabilità di errore quando classificano nuovi dati.

- Se l'iperpiano cercato non esiste, Support Vector Machine usa una mappatura non lineare per trasformare i dati di training X, in uno spazio di dimensione superiore rispetto a quello dei dati originali in modo che le immagini dei dati di due classi siano separati da un iperpiano, che sarà scelto per la suddivisione dei dati.

In dati linearmente separabili è possibile individuare un iperpiano in cui si possono distinguere due semispazi. Nella figura 3 è visibile come sia possibile disegnare un numero infinito di linee rette per separare i diversi elementi. Il problema è trovare quale tra le infinite rette risulti ottimale, ossia quella che, a fronte di una generica nuova osservazione, classifichi nel modo corretto l'osservazione.

Dato un training set etichettato:

$$(x_1, y_1), ..., (x_n, y_n)\ \forall i\ x_i \in R^d,\ y_i \in \{-1, +1\}$$

dove x_i sono i punti da classificare e y_i sono le etichette.

Un iperpiano è definito come

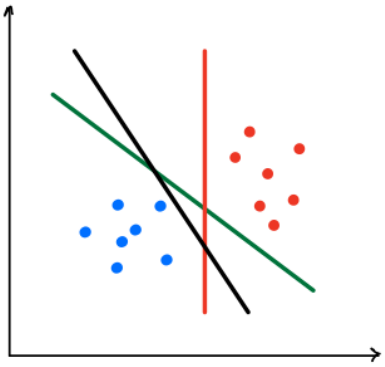

Figura 2: Infinite rette possono separare gli elementi

$$w_0 + w_1 z_1 + w_2 z_2 + ... + w_m z_m = 0,$$

dove ω è il vettore di peso, z è il vettore di caratteristiche di input e w_0 è il bias. In sostanza in m dimensioni un iperpiano di separazione è una combinazione lineare di tutte le dimensioni uguagliate a 0. Ragionando a due dimensioni per semplificare il problema abbiamo che

$$w_0 + w_1 z_1 + w_2 z_2 = 0.$$

I punti che stanno sopra l'iperpiano e che rappresentano un classe soddisfano la seguente condizione:

$$w_0 + w_1 z_1 + w_2 z_2 > 0,$$

mentre qualsiasi punto che si trova sotto l'iperpiano, appartiene all'altra classe, che è soddisfatta dalla seguente condizione :

$$w_0 + w_1 z_1 + w_2 z_2 < 0,$$

L'algoritmo di apprendimento per SVM, in realtà, riscrive queste condizioni in modo più stringente:

$$w_0 + w_1 z_1 + w_2 z_2 \geq 1, \ if \ y = 1,$$

$$w_0 + w_1 z_1 + w_2 z_2 \leq 1 \ if \ y = -1,$$

$$y \in (-1, +1)$$

Se il vettore dei pesi è indicato da w e $\| w \|$ è la sua norma, allora la dimensione del margine massimo è

$$\frac{1}{||w||} + \frac{1}{||w||} = \frac{2}{||w||},$$

ciò significa che minimizzando la norma del vettore peso w, avremo margine massimo che determina l'iperpiano ottimale.

Non è però sempre possibile dividere i dati tramite un iperpiano, come esemplificato in Figura 4.

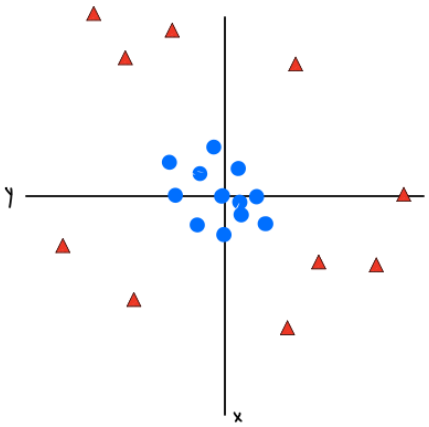

Figura 3: Non sempre è possibile dividere i dati linearmente

Per utilizzare la classificazione tramite iperpiani anche per dati che avrebbero bisogno di funzioni non lineari per essere separati, è necessario ricorrere alla tecnica degli spazi immagine (*feature spaces*). Questo metodo, che sta alla base della teoria delle SVM, consiste nel mappare i dati iniziali in uno spazio di dimensione superiore. Presupponendo quindi $m > n$, per la mappa si utilizza una funzione

$$\phi : \mathbb{R}^n \to \mathbb{R}^m, \tag{1}$$

attraverso la funzione ϕ i dati vengono mappati in uno spazio a più dimensioni, ciò comporta che i dati potrebbero essere linearmente separabili e quindi sarebbe possibile trovare un iperpiano che li separi. [3]

La tecnica degli spazi immagine è particolarmente interessante per algoritmi che utilizzano i dati di training x_i solo attraverso prodotti scalari $x_i \cdot x_j$. In questo caso nello spazio $\mathbb{R}^m$ non si devono trovare esplicitamente $\phi(x_i)$ e $\phi(x_j)$ ma basta calcolare il loro prodotto scalare $\phi(x_i) \cdot \phi(x_j)$. Per rendere semplice questo ultimo calcolo, che in spazi di dimensioni elevate diventa molto complicato, si utilizza una funzione detta *kernel* che restituisce direttamente il prodotto scalare delle immagini:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j), \tag{2}$$

Esistono svariati *kernel*, i più utilizzati sono:

- il kernel lineare: $K(x, y) = x \cdot y$,
- il kernel polinomiale: $K(x, y) = (x \cdot y)^d$ oppure $K(x, y) = (1 + x \cdot y)^d$,
- il kernel gaussiano: $K(x, y) = \exp(-||x - y|\,|^2)/(2\sigma 2)$,
- il kernel sigmoide: $K(x, y) = \tanh(kx \cdot y - \delta)$.

2.1.1.2 Decision Tree Classifier

Il Decision Tree Classifier è un metodo di apprendimento supervisionato usato sia per scopi di classificazione che di regressione. Utilizza un albero decisionale, *decision tree*, composto da [4]:

- nodi non terminali: rappresentano un test su uno o più caratteristiche,
- ramo: rappresenta un esito del test,
- foglia: rappresenta una possibile classe.

Dato un insieme di dati è possibile costruire un numero esponenziale di alberi di decisione. Alcuni alberi sono più efficienti di altri ma trovare l'albero ottimo è una cosa computazionalmente infattibile a causa dello spazio esponenziale nel quale bisogna cercare. Tuttavia esistono algoritmi che riescono, in un tempo ragionevole, a trovare un albero efficiente anche se non corrisponde all'ottimo. Questi algoritmi spesso implementano una strategia greedy che crea l'albero facendo una serie di decisioni localmente ottime su quale caratteristica dei dati usare per partizionare i dati. [2] Uno di questi algoritmi è l'algoritmo di Hunt nel quale l'albero di decisione viene costruito ricorsivamente partizionando il training set in sottoinsiemi sempre più piccoli. Sia D_t l'insieme dei dati di training che sono associati al nodo t e sia $y = y_1, y_2, ..., y_c$ le etichette. La definizione ricorsiva dell'algoritmo di Hunt è la seguente. [2]

- se tutti i dati in D_t appartengono alla stessa classe y_t, allora t è una foglia etichettata come y_t
- se D_t contiene dati che appartengono a più di una classe, viene selezionata una caratteristica per eseguire un test per partizionare i dati in sottoinsiemi più piccoli. Viene creato un nodo figlio per ogni risultato della condizione del test e i dati in D_t vengono divisi in base al risultato del test. L'algoritmo viene applicato in modo ricorsivo ad ogni nodo figlio.

La misura di selezione degli attributi è un'euristica per selezionare il criterio di suddivisione che divide i dati in modo da massimizzare l'omogeneità dei sottoinsiemi di dati. I più famosi sono l'entropia e l'indice di Gini. [4]

2.1.1.3 Random Forest Classifier

Il Random Forest Classifier è un algoritmo di apprendimento supervisionato. Esso considera più alberi di decisione che operano come un insieme. Ogni albero della foresta genera una previsione e quella che compare più frequentemente diventa la previsione del modello. [5]

Il concetto fondamentale dietro una foresta casuale risiede nel fatto che un grande numero di alberi non correlati tra loro che operano insieme tendono a essere più efficienti di un singolo albero.

La bassa correlazione è la chiave, modelli non correlati possono produrre previsioni d'insieme più accurate di qualsiasi singola previsione. La ragione risiede nel fatto che, anche se alcuni alberi potrebbero sbagliare, molti altri avranno ottenuto la previsione corretta.

2.1.1.4 Gaussian Naive Bayes

Il Gaussian Naive Bayes è uno dei più semplici algoritmi di apprendimento supervisionato, e si basa sul teorema di Bayes. L'algoritmo assume che l'effetto di una particolare caratteristica in una classe sia indipendente dalle altre caratteristiche. Anche se le caratteristiche sono interdipendenti, esse vengono comunque considerate in modo indipendente[6]. Questa ipotesi, a cui ci si riferisce come indipendenza condizionale di classe, semplifica il calcolo. Applicando il teorema di Bayes si ha

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}, \tag{3}$$

dove y è un'etichetta e $X = (z_1, z_2, z_3, ..., z_n)$ è il corrispondente vettore di caratteristiche.

Siccome abbiamo assunto che le caratteristiche siano indipendenti possiamo dire che:

$$P(y|z_1, ..., z_n) = \frac{P(z_1|y)P(z_2|y)...P(z_n|y)P(y)}{P(z_1)P(z_2)...P(z_n)}. \tag{4}$$

Poiché il denominatore rimane costante per un determinato input possiamo rimuoverlo e scrivere:

$$P(y|z_1, ..., z_n) \propto P(y) \prod_{i=1}^{n} P(z_i|y). \tag{5}$$

Ora dobbiamo creare un modello per classificare. Lo facciamo trovando la probabilità di un dato insieme di input per tutti i possibili valori di y e prendendo l'output con la massima probabilità:

$$y = \arg\max P(y) \prod_{i=1}^{n} P(z_i|y). \tag{6}$$

Rimane solo da calcolare $P(y)$ e $P(z_i|y)$, che sono ricavabili dal dataset dato in input al sistema.

Nel Gaussian Naive Bayes si presume che i valori continui associati a ciascuna caratteristica siano distribuiti secondo un modello gaussiano.

Sulla base di questa assunzione è quindi possibile scrivere le probabilità condizionate che compaiono in (6) nel modo seguente

$$P(z_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(z_i-\mu_y)^2}{2\sigma_y^2}}, \tag{7}$$

dove μ_y è la media e σ_y la deviazione standard.

2.1.1.5 Linear Discriminat Analysis

Il Linear Discriminat Analysis è un algoritmo di apprendimento supervisionato il cui scopo è quello di trovare una combinazione lineare di caratteristiche che caratterizza o separa due o più classi di oggetti. La combinazione risultante può essere utilizzata come classificatore o anche per la riduzione della dimensionalità. 3.3

Il processo prevede di proiettare i dati in input su un sottospazio lineare dalle direzioni che massimizzano la divisione tra le classi. La dimensione dell'output è necessariamente inferiore al numero di classi, quindi questa è, in generale, una riduzione della dimensionalità piuttosto forte, e ha senso solo in un ambiente multiclasse.

2.1.1.6 Reti neurali multi-strato

Le reti neurali multi-strato sono un modello che utilizza l'algoritmo "backpropagation", basato sulla minimizzazione dell'errore, per compiere apprendimento supervisionato. L'idea di base è quella del neurone umano e della rete di neuroni che compone il nostro cervello. Il componente principale di una rete neurale è detto neurone o percettrone. Esso è identificato da n pesi reali $w_1, w_2, ..., w_n$ e quando riceve una serie di input $x_1, x_2, ..., x_n$ li moltiplica per i pesi corrispondenti, viene così prodotto un valore v sommando i prodotti ottenuti a cui viene sommato un termine di bias. [7] L'output della rete è ottenuto calcolando una particolare funzione, detta funzione di attivazione, usando v come argomento.

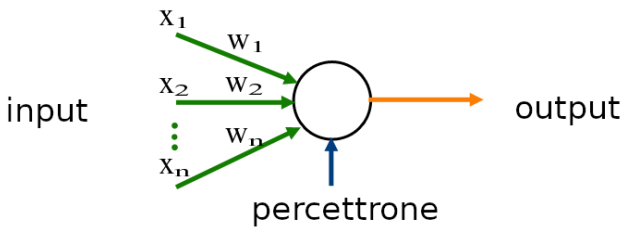

Figura 4: Visualizzazione di un percettrone

Una rete con un solo percettrone è chiamata a singolo livello. Esistono poi le reti multistrato, sono reti i cui percettroni sono disposti su più livelli, come indicato in Fgigura 8. Esistono tre tipi di livelli:

- livello input: costituito da un insieme di neuroni che rappresentano le caratteristiche in input,
- livello output: riceve i valori dall'ultimo livello hidden presente e li trasforma in valori di output,
- livello hidden: livelli intermedi tra input e output,

In questa rete ogni nodo, esclusi quelli di input, usa una funzione di attivazione non lineare per modellare il comportamento.

Esistono diverse funzioni di attivazione:

- Identity: $f(x) = x$
- Logistic: $f(x) = \frac{1}{1+e^{-x}}$
- Tanh: $f(x) = \tanh(x)$

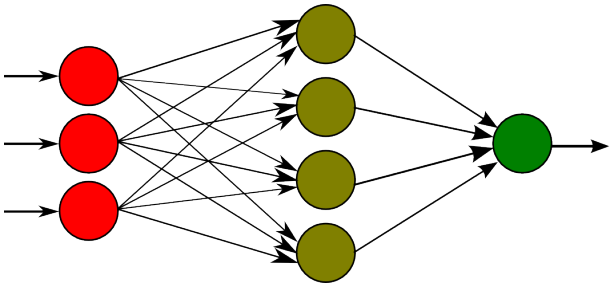

Figura 5: Visualizzazione di una rete multistrato; in rosso il livello di input, al centro il livello hidden, in verde a destra il livello di output

- Relu: $f(x) = \max(0, x)$

L'output y viene calcolato nel seguente modo:

$$y = \Phi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \Phi(w^T x + b) \tag{8}$$

dove w è il vettore di pesi, x è il vettore di input, b è il bias e Φ è la funzione di attivazione.

Il processo di addestramento del Multi-Layer Perceptron avviene mediante una continua regolazione dei pesi delle connessioni dopo l'elaborazione di ogni oggetti nel dataset a disposizione. Questa regolazione si basa sull'errore nell'output ed dà vita ad un processo di apprendimento chiamato "backpropagation". Tale processo continua fino a che non si verifica un criterio di fine apprendimento (per esempio, dopo un numero finito di iterazioni, o quando l'errore non scende sotto una soglia prefissata). Il processo continua fino a quando l'errore non raggiunge il valore più basso. [7]

2.1.2 Apprendimento non supervisionato

Nell'apprendimento non supervisionato, al contrario di quello supervisionato abbiamo dei dati senza etichetta. Le tecniche di apprendimento non supervisionato mirano a estrarre, in modo automatico, della conoscenza a partire da basi di dati, e questo avviene senza una specifica conoscenza dei contenuti da analizzare. Un esempio tipico di questi algoritmi lo si ha nei motori di ricerca. Questi programmi, data una o più parole chiave, sono in grado di creare una lista di link rimandanti alle pagine che l'algoritmo di ricerca ritiene attinenti alla ricerca effettuata. [8] I principali algoritmi utilizzati in ambito non supervisionato fanno riferimento a tecniche di clustering e a regole di associazione.

2.1.3 Apprendimento per rinforzo

Il terzo tipo di apprendimento automatico è l'apprendimento per rinforzo. L'obiettivo di questo tipo di apprendimento è quello di costruire un sistema che attraverso le interazioni con l'ambiente migliori le proprie performance. [8]

Per poter migliorare le funzionalità del sistema vengono introdotti dei rinforzi, ovvero segnali di ricompensa. Questo rinforzo non è dato dalle etichette, ma è una misurazione sulla qualità delle azioni intraprese dal sistema. Per questo motivo non può essere assimilato all'apprendimento supervisionato. Potremmo trovare questo tipo di apprendimento ad esempio nell'addestramento di un sistema per il gioco degli scacchi.

Inizialmente le mosse saranno del tutto casuali e senza una logica. Dal momento in cui il sistema riceverà dei feedback positivi, come ad esempio nel caso in cui mangi una pedina avversaria, allora riceverà un peso maggiore e conseguentemente un rinforzo positivo su quell'azione. Contrariamente in caso di azione negativa, il valore dei pesi su quell'azione andrà in decremento.

Conseguentemente a questi rinforzi, il sistema darà maggior peso alle mosse che gli hanno portato maggiori benefici e tenderà a replicare lo stesso comportamento su nuove mosse future.

2.1.4 Apprendimento semi supervisionato

Può essere visto come un quarto tipo di apprendimento automatico. In questo caso, al contrario dell'apprendimento non supervisionato, abbiamo che di tutti i dati presenti nel training set, solo pochi di essi sono stati etichettati.[9]

Capitolo 3

Dataset

Un dataset è una collezione di dati nel quale ogni colonna della tabella corrisponde a una caratteristica e ogni riga ad una singola osservazione o istanza. Essi sono il "cibo" per gli algoritmi di machine learning. Quanto più un dataset è ricco di osservazioni tanto più sarà in grado di fornire prestazioni e accuratezza in output.

3.1 Iris

Durante il tirocinio, inizialmente, per prendere familiarità con gli strumenti e algoritmi necessari a compiere lo studio, ho utilizzato il dataset Iris. Esso è un dataset multivariato introdotto da Ronald Fisher nel 1936. Consiste in 150 istanze di fiori iris e classificate secondo tre specie: Setosa, Virginica e Versicolor. Le variabili considerate sono lunghezza e larghezza di sepalo e petalo.

	sepal.length	sepal.width	petal.length	petal.width	variety
2	5.1	3.5	1.4	.2	Setosa
3	4.9	3	1.4	.2	Setosa
4	4.7	3.2	1.3	.2	Setosa
5	4.6	3.1	1.5	.2	Setosa
6	5	3.6	1.4	.2	Setosa
7	5.4	3.9	1.7	.4	Setosa

Figura 6: Visualizzazione di un estratto del dataset Iris

3.2 Incidenti Stradali

Apprese le basi del Machine Learning ho iniziato a lavorare con il dataset "Incidenti Stradali", questo dataset è stato fornito da medici legali e raccoglie, tristemente,

i decessi causati da incidenti stradali. Sono state raccolte 131 istanze e ognuna rappresenta una persona deceduta. Il dataset è organizzato in vari livelli di dettaglio. Un primo livello è composto dalle caratteristiche basilari:

- Numero del verbale, verrà usato come indice univoco
- Data del decesso
- Sesso
- Anni
- Peso
- Altezza
- BMI, indice di massa corporea

Successivamente sono raccolte tutte le ossa rotte durante l'incidente con etichette che vanno da 0, nessuna lesione, a 4, lesione massima. Successivamente sono descritti i totali delle rotture avvenute nei distretti di:

- Testa
- Torace
- Addome
- Scheletro

Ogni singolo distretto è poi diviso più nello specifico in singole ossa, come in figura 10

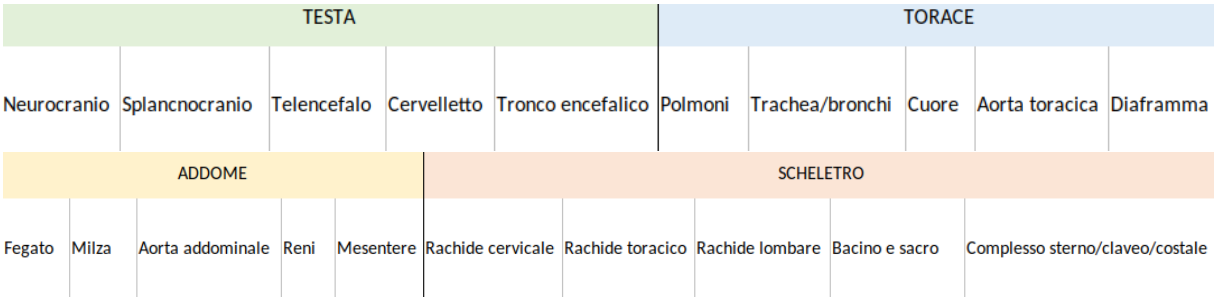

Figura 7: Suddivisione caratteristiche dettagliate

Il dataset ha inoltre due ulteriori livelli di dettaglio. Esso contiene infatti anche dati più specifici riguardanti la frattura o meno di singole ossa raggruppate in:

- Cranio
- Rachide
- Torace - Gabbia Toracica
- Bacino
- Arti Superiori
- Arti Inferiori

In ultimo è registrato il mezzo che ha investito la persona. Questo è il label set, la y dei modelli descritti al capitolo precedente. Tutti gli esperimenti effettuati in fase di studio sono incentrati sul cercare di classificare il tipo di mezzo, leggero ovvero auto, etichettato con 0, o pesante, con etichetta 1, che ha investito l'individuo.

Nel mio studio mi sono limitato a considerare le caratteristiche basilari, i totali e quelle riportate in figura 10.

3.3 Riduzione della Dimensionalità

Nei problemi di classificazione di Machine Learning ci sono spesso molte caratteristiche da tenere in considerazione, basti pensare che il dataset "Incidenti Stradali" al completo ha circa 350 caratteristiche. Per dataset con molte dimensioni, diciamo con un numero maggiore di 10, la riduzione della dimensionalità è un pratica importante svolta prima di applicare algoritmi di Machine Learning per evitare l'effetto chiamato *curse of dimensionality*, "maledizione della dimensionalità". Il problema sorge nel momento in cui la dimensionalità aumenta e lo spazio aumenta così rapidamente che i dati disponibili diventano radi, sparsi. In ambito statistico questa scarsità è problematica in quanto i dati necessari a supportare il risultato aumentano in modo esponenziale. Inoltre dati con molte dimensioni possono causare problemi di *overfitting*, ovvero il sistema in qualche modo impara e memorizza il risultato, si adatta (*fitting*) troppo bene (*over*) ai dati di training perdendo di generalità. Il modello sembra perfetto per i dati di training ma quando si prova ad applicarlo ai dati di test si verificano molti errori. Per questo motivo viene effettuata una riduzione della dimensionalità in modo da eliminare le caratteristiche più irrilevanti e lasciare spazio a quelle rilevanti.

In questo contesto entrano in gioco algoritmi di riduzione della dimensionalità come PCA e TSNE.

3.3.1 PCA

Principal component analysis, o analisi delle componenti principali, è una tecnica che mira a ridurre il numero di variabili o caratteristiche che descrivono un insieme di dati ad un numero inferiore cercando di limitare la perdita di informazione. PCA prende un dataset contenente una lista di tuple che rappresentano punti nello spazio a molte dimensioni e trova le direzioni lungo le quali le tuple si allineano meglio. L'idea è di trattare le tuple come una matrice M e trovare gli autovettori per MM^T o M^TM.

La matrice di questi autovettori può essere vista come una rotazione nello spazio.

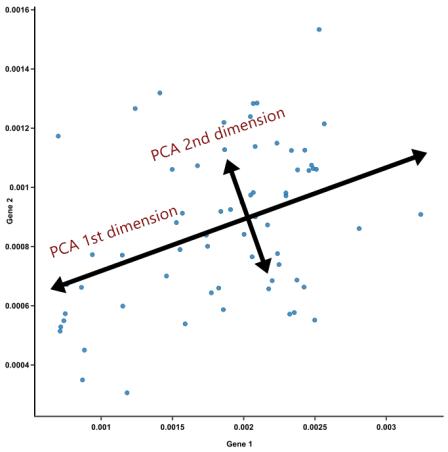

Figura 8: Dimensioni trovare tramite PCA

Quando si applica questa trasformazione dei dati, l'asse corrispondente all'autovettore principale è quello nel quale i dati sono più sparsi, ovvero l'asse nel quale la varianza dei dati è massimizzata.

Detto in altro modo, i punti possono essere osservati meglio se distesi lungo questo asse, con piccole deviazioni da esso. Allo stesso modo, l'asse corrispondente al secondo autovettore (l'autovettore corrispondente al secondo autovalore più grande) è l'asse lungo il quale la varianza delle distanze dal primo asse è maggiore, e così via fino al numero di dimensioni che si è scelto di tenere.

Ma come funziona esattamente? Per prima cosa si costruisce la matrice dei dati A nel quale ogni colonna corrisponde ad una caratteristica e ogni riga ad una istanza. Successivamente si calcola la media di ogni colonna e la si sottrae ad ogni elemento della matrice, e otteniamo così la matrice B.

A questo punto viene calcolata la matrice di covarianza. Nella diagonale principale è visualizzata la varianza mentre tutti gli altri elementi rappresentano la covarianza tra le caratteristiche. Ogni valore di covarianza assume un segno: se

positivo significa che c'è una correlazione positiva, al contrario, se negativa, c'è una correlazione inversa.

La matrice appena calcolata serve per poter calcolare gli autovalori, operatori che si basano sul fatto che moltiplicando o dividendo un vettore cambia solo la sua lunghezza e non la direzione. Un autovalore λ è tale se risolve $Av = \lambda v$, con A matrice quadrata e v vettore che viene definito autovettore.

Esempio: A=$\begin{bmatrix} 1 & 1 \\ 8 & 1 \end{bmatrix}$ e $v = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, allora 5 rappresenta un autovalore in quanto verifica l'equazione

Per calcolare gli autovalori di una matrice è possibile risolvere la seguente equazione

$$det(A - \lambda I) = 0 \tag{9}$$

dove I è la matrice identità e det rappresenta il determinante della matrice.

Allo stesso modo è possibile calcolare gli autovettori corrispondenti agli autovalori calcolati precedentemente risolvendo l'espressione e trovando v:

$$det(A - \lambda I)v = 0 \tag{10}$$

Gli autovettori trovati formeranno gli assi del nuovo sistema di riferimento di dimensioni più piccole rispetto a quello iniziale. Tuttavia gli autovettori definiscono solo le dimensioni dell'asse poiché hanno tutti lunghezza 1, di conseguenza per decidere quale autovettore vogliamo eliminare per il nostro sottospazio di dimensione inferiore dobbiamo vedere gli autovalori degli autovettori. Quello che faremo sarà eliminare gli autovettori con autovalori più bassi in quanto hanno il minor numero di informazioni sulla distribuzione dei dati al loro interno. Quello che viene fatto è ordinare gli autovalori dal più grande al più piccolo e prendere i primi k autovettori corrispondenti.

In ultimo dobbiamo formare le componenti principali, per farlo calcoliamo:

$$CP = W^T \cdot B^T \tag{11}$$

dove CP è la matrice costituita dalle componenti principali, W^T è la matrice formata usando gli autovettori che abbiamo scelto di mantenere precedentemente di cui prendiamo la trasposta, B^T è la trasposta della matrice B precedentemente calcolata. Quello che si ottiene sono i dati originali ma inseriti nello spazio considerando come assi di riferimento gli autovettori calcolati.

Sebbene PCA sia uno degli algoritmi più famosi e usati poiché è veloce, facile da usare e intuitivo ha un problema: è una tecnica lineare, questo significa che non è in grado di trovare delle dipendenze non lineari che gli consentono di tornare ad uno stato precedente. Una volta passati da N a M dimensioni, non è più possibile tornare ad N partendo da M.

t-SNE non è limitato da proiezioni lineari il che significa che può adattarsi ad ogni tipo di dataset.

3.3.2 t-SNE

t-SNE, o t-Distributed Stochastic Neighbor Embedding, è una tecnica di riduzione della dimensionalità non lineare. Utilizza una struttura locale nel quale punti aventi molte dimensioni vengono mappati nello spazio usando un numero inferiore di dimensioni in modo che le distanze tra i punti rimangano quasi le stesse.

L'algoritmo prevede tre steps:

- Misura le somiglianze tra i punti nello spazio a molte dimensioni e, per ogni punto x_i, centra una distribuzione gaussiana sopra di esso. Viene misurata la densità di tutti gli altri punti x_j sotto questa distribuzione. Successivamente viene normalizzata la curva per tutti gli altri punti.

  Questo ci dà un set di probabilità (Pij) che sono proporzionali alle somiglianze. Questo significa che se i punti x_1 e x_2 hanno valori uguali sotto questa distribuzione gaussiana, le loro proporzioni e somiglianze sono uguali. La distribuzione può essere manipolata usando quella che viene chiamata perplessità, la quale influenza la varianza della distribuzione, ovvero quanto è ampia la curva, che a sua volta influenza il numero dei vicini di ogni punto. Il normale range della perplessità è tra 5 e 50.

- Simile al primo ma anziché usare una distribuzione gaussiana usa Student (allega foto). Essa ci fornisce un secondo set di probabilità Q_ij in uno spazio ridotto. Come si può vedere dalla figura 11 la t-distribution ha code più lunghe rispetto alla gaussiana, questo fornisce una migliore visualizzazione della distanza dei punti.

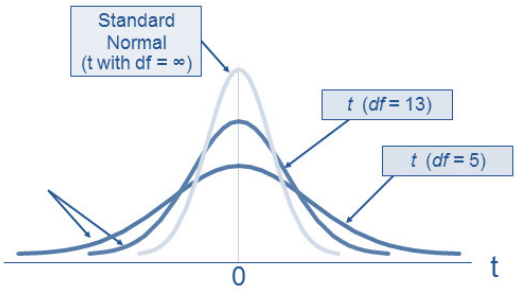

Figura 9: Distribuzione gaussiana vs t-Student

- Vogliamo ora che le probabilità Q_{ij} calcolate nello spazio ridotto riflettano al meglio possibile le P_{ij} calcolate usando lo spazio con più dimensioni. Misuriamo la differenza tra le distribuzioni di probabilità degli spazi delle due

dimensioni usando la divergenza di Kullback-Liebler (KL) che confronta efficacemente valori di P_{ij} e Q_{ij} di grandi dimensioni. Infine, usiamo la tecnica della discesa gradiente per ridurre al minimo la funzione di costo KL.

Nel capitolo 3 vedremo concretamente come utilizzare PCA e t-SNE per ridurre la dimensione del dataset Incidenti Stradali.

Capitolo 4

Esperimenti

4.1 Model Selection

4.1.1 Scelta degli Iperparametri

$$x_i(n) = a_{i1}u_1(n) + a_{i2}u_2(n) + \cdots + a_{iJ}u_J(n)\,. \tag{12}$$

4.1.2 Scalare i dati

4.2 Errore di Generalizzazione

4.2.1 Training

4.2.2 Cross Validation

4.3 Risultati Ottenuti

Capitolo 5

Conclusioni

Nelle conclusioni si tirano le somme di quanto realizzato, facendo un riassunto stringato del lavoro svolto. In particolare vanno dichiarati punti di forza e criticità della ricerca effettuata, nonché quali aspetti dello stato dell'arte siano stati superati dal lavoro in oggetto.

Bibliografia

[1] Fredrik Lindsten Thomas B. Schön Andreas Lindholm, Niklas Wahlström. *Supervised Machine Learning*. Cambridge University Press, 2020.

[2] Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, and Vipin Kumar. *Introduction to Data Mining (2nd Edition)*. Pearson, 2nd edition, 2018.

[3] Andrew Ng. Cs229 lecture notes. url http://cs229.stanford.edu/notes/cs229-notes3.pdf/, October 2018.

[4] Oded Maimon and Lior Rokach. *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag, Berlin, Heidelberg, 2005.

[5] Breiman Leo. *Random Forests*.

[6] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[7] K.-L Du and M.N.s Swamy. *Neural Networks and Statistical Learning*. 01 2019.

[8] Zoubin Ghahramani. Unsupervised learning. In *Advanced Lectures on Machine Learning*, pages 72–112. Springer-Verlag, 2004.

[9] Philippe Thomas. Semi-supervised learning by olivier chapelle, bernhard schölkopf, and alexander zien (review). *IEEE Trans. Neural Networks*, 20(3):542, 2009.