UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN Ingegneria Informatica

# METODI DI ONE CLASS CLASSIFICATION


Relatore:
Prof. Stefano Tomasin

Laureando:
Davide Furlan



ANNO ACCADEMICO: 2023/2024
Data di laurea: 18/07/2024

# Indice

**Sommario**

One Class Classification (OCC) è un caso particolare di classificazione multiclasse. Il suo obiettivo consiste nel creare un modello il quale riesca a riconoscere valori anomali all'interno di un dataset: per farlo OCC si basa su un allenamento specifico a una unica classe di oggetti chiamata classe positiva.

Viene applicata in numerosi contesti quali intelligenza artificiale, statistica, visione computerizzata e studi biometrici. Questa tesi inizia con una introduzione al mondo di OCC e il concetto di anomalia. Verranno successivamente analizzate nel dettaglio le strategie principali di OCC: descrizione dati con vettori di supporto (SVDD), foresta di isolamento (Isolation Forest), autocodificatore (Autoencoder) e analisi delle componenti principali (PCA).

# Capitolo 1

# Introduzione

Le reti neurali nell'ultimo decennio hanno mostrato un notevole livello di performance nella classificazione di oggetti: oggi ampi set di dati forniscono un eccellente allenamento per gli algoritmi di intelligenza artificiale. Gli algoritmi di classificazione si dividono in due macrocategorie: metodi di classificazione supervisionati e non supervisionati. Il risultato dei primi può essere confrontato con valori e conoscenze a priori forniti da operatori umani, mentre quello dei secondi non ha metodi di paragone.

Nel contesto dell'intelligenza artificiale, un ruolo sempre più importante è quello di identificare anomalie, ovvero comportamenti inusuali o dati anomali, all'interno di un set di dati: tale ruolo viene svolto da One Class Classification (OCC). OCC è una tecnica non supervisionata di classificazione la quale, partendo da un set di dati di allenamento per una sola classe di interesse (classe positiva), è in grado di identificare gli elementi appartenenti alla classe positiva scartandone il resto. A differenza di metodi multi-classificabili, OCC risulta più complessa a causa del suo addestramento: OCC si baserà unicamente su un set di allenamento specifico per la nostra classe di interesse, e di conseguenza dovrà trattare un numero maggiore di classi sconosciute. Ogni campione studiato apparterrà quindi alla classe positiva oppure alla sua complementare: la classe negativa. OCC viene impiegata in discipline quali la ricerca di anomalie e per questo svolge un ruolo determinante in scenari di controllo dello stato di centrali nucleari, predizione del fallimento di motori, ecc.

Esistono varie tecniche: andremo a analizzare approcci basati su machine learning, deep learning e metodi statistici. Tra i principali studieremo:

1. Descrizione dei dati con vettori di supporto (SVDD): un metodo di intelligenza artificiale di tipo non supervisionato che consente di determinare una sfera di raggio minore possibile al cui interno sono presenti

tutti i dati della classe di interesse, ponendo le anomalie fuori dalla sfera stessa.

2. Autocodificatore: un modello di rete neurale specializzato in allenamento non supervisionato che, mediante due strutture specializzate, codificatore e decodificatore, consente di codificare i dati di input e comprimerli in una rappresentazione a dimensione inferiore dell'originale. Una volta raggiunta una compressione sufficiente, il decodificatore converte tali informazioni compresse e tenta di ricostruire il dato originario.

3. Foresta di isolamento: un metodo ad albero particolare che si differenzia per un valore di soglia che consente di identificare le anomalie: più un nodo ha una profondità minore nella struttura dell'albero, più facilmente verrà considerato un'anomalia (considerando che le anomalie sono meno frequenti dei valori normali) e quindi gli verrà assegnato un punteggio di anomalia maggiore.

4. Analisi delle componenti principali (PCA): una tecnica per la semplificazione dei dati in cui viene ridotta la complessità dell'input considerando solo le variabili principali, cioè le variabili che descrivono l'insieme di dati in ingresso, limitando il più possibile la perdita di informazioni.

Questo documento è strutturato nel seguente modo: nella sezione 2 analizzeremo l'algoritmo principale della OCC SVDD, nella sezione 3 studieremo la struttura ad albero Isolation Forest, nella sezione 4 discuteremo della rete neurale AUTOENCODER e, infine, nella sezione 5 parleremo della tecnica PCA. Infine, nella sezione 6 concluderemo l'articolo con un breve riassunto e una conclusione.

# Capitolo 2

# SVDD

## 2.1 SVDD con dati positivi

Prima di introdurre l'argomento poniamo delle definizioni. Assumiamo che x sono vettori colonna e $x^2 = x \cdot x$. Abbiamo un set di allenamento $x_i$ con $i = 1, ..., N$ per il quale vogliamo ottenere una descrizione.

Il metodo Descrizione Dati con Vettori di Supporto (Support Vector Data Description, SVDD) dello studio [9] si basa sul trovare una sfera di raggio $R$ e centro a tali che riesca a contenere al suo interno tutti gli elementi del dataset $x_i$ positivi alla classe di interesse: i campioni che non rientrano in tale categoria verranno etichettati come valori anomali e saranno quindi posizionati fuori dalla sfera stessa. Il problema di ottimizzazione è

$$\min F(R, a) = \min R^2 \tag{2.1}$$

con vincoli

$$||x_i - a||^2 \leq R^2, \forall i \tag{2.2}$$

Così facendo però non si tiene conto delle anomalie: queste per ipotesi hanno un raggio maggiore rispetto a $R$. Si decide quindi di non rendere la distanza di $x_i$ dal centro a strettamente più piccola di $R^2$. Introduciamo variabili di tolleranza $\xi_i$ e $C$. $\xi_i \geq 0$ misura la deviazione del dato $x_i$ dalla superficie della sfera permettendo una piccola violazione dei vincoli a patto di migliorare la soluzione del problema: se $\xi_i = 0$ abbiamo a che fare con un oggetto posto dentro il raggio $R$ della classe positiva, altrimenti parliamo di un punto anomalo e quindi la sua distanza verrà penalizzata dal modello. Ricordando che con margine intendiamo la distanza tra i vettori di supporto $SV$ e il confine di decisione tra classe positiva e negativa, il parametro $C$ controlla il migliore compromesso tra il volume della sfera e gli errori $\xi_i$: più è grande il valore di $C$ allora maggiormente nel modello verranno penalizzati gli oggetti

con errore $\xi_i$ quindi si avrà un margine più ristretto. Un valore più piccolo invece risulterà in un modello più tollerante verso gli errori e questo comporta un volume maggiore della sfera. Il problema di ottimizzazione diventa:

$$\min F(R, a) = \min(R^2 + C \sum_i (\xi_i)) \tag{2.3}$$

con il vincolo che ogni oggetto $x_i$, escluse le anomalie, sarà dentro la sfera:

$$||x_i - a||^2 \leq R^2 + \xi_i, \xi_i \geq 0 \forall i \tag{2.4}$$

L'ultimo vincolo puo' venir incorporato in $F(R, a)$ usando i moltiplicatori di Lagrange:

$$L(R, a, \alpha_i, \gamma_i, \xi_i) = R^2 + C \sum_i \xi_i - \sum_i \alpha_i (R^2 + \xi_i) - ||x_i||^2 - 2ax_i + ||a||^2) - \sum_i \gamma_i \xi_i \tag{2.5}$$

con moltiplicatore lagrangiano $\alpha_i \geq 0$ che rappresenta il peso assegnato a ogni campione, $\gamma_i \geq 0$ rappresenta la variabile d'errore assegnata a ogni campione: se maggiore di zero vuol dire che abbiamo a che fare con una anomalia. $L$ dovrebbe venire minimizzata rispetto a $\xi_i, R, a$ e massimizzata rispetto $\alpha_i$ e $\gamma_i$. Ponendo le derivate parziali dei vincoli uguali a zero si ha:

$$\frac{\partial L}{\partial R} : \sum_i \alpha_i = 1 \tag{2.6}$$

$$\frac{\partial L}{\partial a} : a = (\sum_i \alpha_i x_i) / (\sum_i \alpha_i) = \sum_i \alpha_i x_i \tag{2.7}$$

$$\frac{\partial L}{\partial \xi_i} : C - \alpha_i - \gamma_i = 0 \tag{2.8}$$

Dalla ultima equazione $\alpha_i = C - \gamma_i$ ed essendo $\alpha_i \geq 0$ e $\gamma_i \geq 0$ possiamo eliminare $\gamma_i$ e quindi

$$0 \leq \alpha_i \leq C \tag{2.9}$$

Sostituendo:

$$L = \sum_i \alpha_i (x_i \cdot x_i) - \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_i) \tag{2.10}$$

Se $x_i$ è un campione positivo rientrerà all'interno della sfera e quindi soddisferà

$$||x_i - a|| \leq R^2 + \xi_i \tag{2.11}$$

e quindi $\alpha_i = 0$. In particolare:

$$||x_i - a|| < R^2 \iff \alpha_i = 0, \gamma_i = 0 \tag{2.12}$$

$$||x_i - a|| = R^2 \iff 0 < \alpha_i < C, \gamma_i = 0 \tag{2.13}$$

$$||x_i - a|| > R^2 \iff \alpha_i = C, \gamma_i > 0 \tag{2.14}$$

Infatti qualora $\alpha_i = 0$ allora $x_i$ sarà interno alla sfera, se $\alpha_i < C$ sarà un oggetto al confine tra la regione degli oggetti positivi e quella delle anomalie, altrimenti verrà considerato una anomalia. Da notare che il valore da considerare è $\alpha_i$ in quanto non solo ci permette di assegnare un peso a $x_i$ ma anche perchè da (2.7) il centro a viene descritto come combinazione lineare di oggetti $x_i$. Per questo motivo considereremo soltanto gli oggetti con $0 < \alpha_i < C$ come i nostri vettori di supporto (SV) in quanto utili a determinare il confine migliore tra classe positiva e negativa. Volendo testare un nuovo oggetto $z$ per valutarne la sua natura, quindi capire se si tratta di un punto anomalo o positivo alla classe di interesse, calcoliamo la sua distanza dal centro della sfera (considerando tale distanza inferiore al raggio R):

$$||z - a||^2 = (z \cdot z) - 2\sum_i \alpha_i (z \cdot x_i) + \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) \leq R^2 \tag{2.15}$$

Poichè $R^2$ è la massima distanza dei punti non anomalia dal centro della sfera ($\alpha_i < C$), escluderemo i punti anomalia ($\alpha_i = C$). Quindi

$$R^2 = (x_k \cdot x_k) - 2\sum_i \alpha_i (x_k \cdot x_i) + \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) \tag{2.16}$$

per ogni $x_k \in SV$ con $\alpha_k < C$.

## 2.2 SVDD con esempi negativi

Qualora dovessero essere presenti elementi appartenenti alla classe negativa nel dataset questi possono venire sfruttati, anziché ignorati, all'interno dell'algoritmo: ovviamente a differenza degli elementi appartenenti alla classe positiva che stanno dentro la sfera questi dovrebbero restare fuori. Per distinguere le due tipologie di campioni useremo come convenzione per dati positivi indici $i, j$ mentre per dati anomali $l, m$. Inoltre cataloghereremo gli elementi della classe di interesse con un parametro $y_i = 1$ mentre per le anomalie $y_l = -1$. Quindi la funzione da minimizzare, tenendo conto anche degli elementi negativi e ai fattori di tolleranza per elementi positivi e anomalie, $\xi_i$ e $\xi_l$, diventa:

$$F(R, a, \xi_i, \xi_j) = R^2 + C_1 \sum_i \xi_i + C_2 \sum_l \xi_l \tag{2.17}$$

e i vincoli

$$||x_i - a||^2 \leq R^2 + \xi_i, ||x_l - a||^2 \geq R^2 - \xi_l, \xi_i \geq 0, \xi_l \geq 0 \forall i, l \tag{2.18}$$

Questi vincoli vengono incorporati nella funzione F da minimizzare (2.17) e introducendo i moltiplicatori di Lagrange $\alpha_i, \alpha_l, \gamma_i, \gamma_l$ otteniamo la Lagrangiana:

$$\begin{aligned} L(R, a, \alpha_i, \gamma_i, \xi_i) = R^2 + C_1 \sum_i \xi_i + C_2 \sum_l \xi_l - \sum_i \gamma_i \xi_i - \sum_l \gamma_l \xi_l - \\ \sum_i \alpha_i (R^2 + \xi_i) - ||x_i||^2 - 2a * x_i + ||a||^2) \\ - \sum_l \alpha_l \alpha_l [(x_l - a)^2 - R^2 + \xi_l) \end{aligned} \tag{2.19}$$

con $\alpha_i \geq 0, \alpha_l \geq 0, \gamma_i \geq 0, \gamma_l \geq 0$.
Ponendo ora le derivate parziali di $L$ rispetto a $R$, a, $xi_i, xi_l$ uguali a zero ottengo i seguenti vincoli:

$$\frac{\partial L}{\partial R} : (\sum_i \alpha_i) - (\sum_l \alpha_l) = 1 \tag{2.20}$$

$$\frac{\partial L}{\partial a} : a = (\sum_i \alpha_i x_i) - (\sum_l \alpha_l x_l) \tag{2.21}$$

$$\frac{\partial L}{\partial \xi_i} : C - \alpha_i - \gamma_i = 0 \tag{2.22}$$

$$\frac{\partial L}{\partial \xi_l} : C - \alpha_l - \gamma_l = 0 \tag{2.23}$$

Così facendo sostituendo le equazioni dalla (2.20) alla (2.23) ottengo:

$$\begin{aligned} L = \sum_i \alpha_i (x_i \cdot x_i) - \sum_l \alpha_l (x_l \cdot x_l) - \sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j) \\ + \sum_{i,j} \alpha_l \alpha_j (x_l \cdot x_j) - \sum_{l,m} \alpha_l \alpha_m (x_l \cdot x_m) \end{aligned} \tag{2.24}$$

Possiamo notare che la formula è identica alla normale SVDD ponendo $\alpha'_i = y_i \alpha_i$ (l'indice i ora serve per catalogare tutti gli oggetti del dataset). I vincoli (2.20) e (2.21) diventano $\sum_i \alpha'_i = 1$ e $a = \sum_i \alpha'_i x_i$ e puo' di nuovo essere usata (2.15). Quindi quando incontreremo dati anomali useremo non più (2.10) ma (2.24).

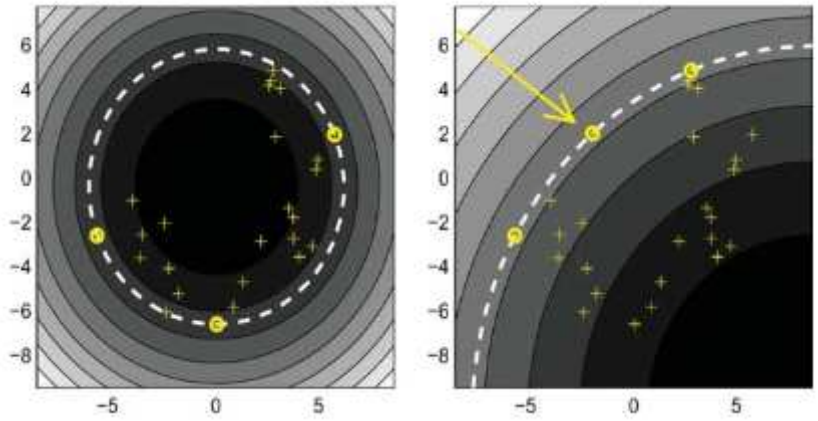


Figura 2.1: Differenza tra svdd con dati normali (figura 1) e con anomalie(figura 2): nella figura 1 si nota che, non essendoci punti anomali, tutti i dati rientrano nella sfera. Nella figura 2 vi è un dato anomalo posto sul bordo della sfera.

Nella figura 2.1 notiamo la presenza di una anomalia: tale punto verrà utilizzato come $SV$ per la classe anomala. Così facendo però dalla equazione (2.15) non si riesce più a distinguere vettori di supporto positivi da quelli della classe anomala: serve un approccio più flessibile che consenta di inserire la maggior parte dei dati dentro la sfera e posizionare fuori le restanti anomalie.

## 2.3 Approccio più flessibile

Calcolare il prodotto scalare tra due vettori può portare a algoritmi poco efficienti causa la complessità per la determinazione dei confini della classe di interesse: la funzione di Kernel $K(x_i, x_j)$ consente di rendere valori in ingresso, che prima risultavano non separabili linearmente, separabili in uno spazio multidimensionale. Con questo approccio poniamo valori della classe di interesse dentro una sfera ed eventuali anomalie fuori dalla stessa. Ne studieremo di due tipi: kernel polinomiale e gaussiano.

## 2.4 Kernel polinomiale

Dato come parametro $d \in N^+$ il grado del kernel, il kernel polinomiale è: $K(x_i, x_j) = (x_i x_j)^d$. Quando i vettori oggetto non sono centrati attorno l'origine assumono norme di maggior valore e essendo $\theta_{i,j}$ l'angolo tra i vettori

oggetto $x_i$ otteniamo

$$x_i \cdot x_j = \cos(\theta_{i,j})||x_i|| \cdot ||x_j|| \tag{2.25}$$

Per $\theta_{i,j}$ piccolo, $\cos(\theta_{i,j}) \approx 1$, e per grandi gradi $d$ il kernel polinomiale diventa:

$$(x_i \cdot x_j)^d = \cos(\theta_{i,j})^d||x_i||^d \cdot ||x_j||^d \approx ||x_i||^d \cdot ||x_j||^d \tag{2.26}$$

Il problema è che oggetti con norma maggiore risulteranno più influenti nella formula del kernel polinomiale: una possibile soluzione può essere il kernel gaussiano.

## 2.5 Kernel gaussiano

Introduciamo il Kernel gaussiano dove denotiamo con s la deviazione standard. Se s risulta grande significa che oggetti anche lontani tra loro verranno considerati simili altrimenti se risulta piccolo un oggetto verrà considerato simile solo ad altri a lui vicini:

$$K(x_i, x_j) = \exp(-||x_i - x_j||^2/s^2) \tag{2.27}$$

Sostituendo nella (2.10) riesco a ricondurrmi all'equazione:

$$L = \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) - \sum_i \alpha_i (K(x_i, x_i)) \tag{2.28}$$

Tale Kernel è indipendente dalle norme dei vettori oggetto posizione dei valori rispetto all'origine $a$ della sfera bensì utilizza la distanza reciproca tra questi. Riprendendo la (2.15) il primo termine sarà uguale a 1 dato $||z - z|| = 0$ mentre il resto sarà una sommatoria pesata di gaussiane. In particolare volendo testare un oggetto del dataset $z$ e $C_R$ costante che dipende solo da vettori di supporto $x_i$ e non da $z$:

$$\sum_i \alpha_i \exp(-||z - x_i||^2/s^2) \geq -R^2/2 + C_R \tag{2.29}$$

Gli oggetti vengono mappati come vettori di norma unitari (dove $\phi(x_i)\phi(x_j) = 1$ con $\phi(x_i)$ mappatura degli oggetti $x_i$) in un nuovo spazio multidimensionale dove contano solo gli angoli tra i vettori oggetto. Per piccoli valori di s, $exp(-||z - x_i||^2/s^2) \approx 0$, $\forall i \neq j$ e l'equazione (2.10) viene ottimizzata quando tutti gli oggetti diventano oggetti di supporto con $\alpha_i = 1/N$. Per grandi valori invece la soluzione si approssima a una soluzione sferica. Questo può essere provato mediante una espansione di Taylor:

$$K(x_i, x_j) = 1 - ||x_i||^2/s^2 - ||x_j||^2/s^2 + 2(x_i x_j)/s^2 + ... \tag{2.30}$$

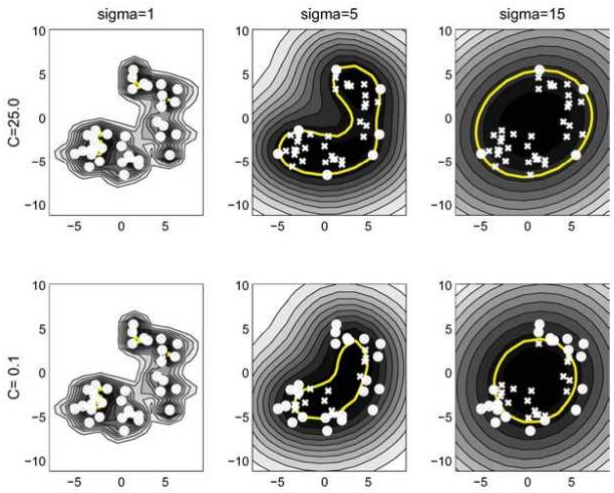


Figura 2.2: svdd allenata con kernel gaussiano su un set di dati con distribuzione dei valori a forma di banana. Date diverse lunghezze di banda (s=1,5,15) e valori $C$ diversi (C=40 e C=0.1). I vettori di supporto sono indicati in cerchi, il cerchio bianco è il confine tra regione positiva e negativa.

sostituendo (2.30) a (2.28) ottengo:

$$\begin{aligned} L &= \sum_i \alpha_i (1 - ||x_i||^2/s^2 - ||x_j||^2/s^2 + 2(x_i x_j)/s^2 + ...) \\ &- \sum_{i,j} \alpha_i \alpha_j (1 - ||x_i||^2/s^2 - ||x_j||^2/s^2 + 2(x_i x_j)/s^2 + ...) \\ &= 1 - 1 - 2\sum_i \alpha_i ||x_i||^2/s^2 - 2\sum_{i,j} \alpha_i \alpha_j (x_i \cdot x_j)/s^2 + ... \end{aligned} \tag{2.31}$$

Escluso $2/s^2$ ottengo l'equivalente di (2.10). Quando si utilizza un nuovo parametro di regolarizzazione $C' = 2C/s^2$ ottengo la stessa soluzione.

Come osservabile in figura 2.2 utilizzo sigma $s = 1.0$, $s = 5$ e $s = 15$: solo nel caso di $s = 15$ ottengo una sfera. Notiamo che all'aumentare di s il numero di vettori di supporto diminuisce. Usando un kernel gaussiano ottengo una sfera di raggio minore rispetto all'utilizzo di modelli sferici o basati sul kernel polinomiale.

# Capitolo 3

# Foresta di isolamento

La foresta di isolamento (isolation Forest, IForest) secondo [5] è un metodo non supervisionato che si basa, per la determinazione di eventuali punti anomali dentro un dataset, su un "punteggio di anomalia": se vicino a 1 l'elemento viene considerato anomalo, se vicino a 0 appartiene alla classe di interesse. In questa sezione parleremo di isolamento come "separare una istanza dal resto delle istanze" di un dataset. Viene proposto per i suoi diversi vantaggi [3]:

1. complessità computazionale limitata grazie all'assenza di calcoli complessi quali determinazione di distanza o densità di un nodo

2. minimi requisiti di memoria tali da consentire il salvataggio della struttura ad albero

3. analisi di dataset di capienza elevata e complessità temporale dipendente linearmente dalla dimensione dell'albero stesso

L'idea principale è che, essendo in un tipo di allenamento non supervisionato, i punti anomali sono meno frequenti rispetto ai punti positivi alla classe di interesse, quindi siano più facili da separare dal dataset. Per separare le anomalie l'algoritmo genera partizioni ricorsive del campione, seleziona randomicamente un attributo e un valore di questi compreso tra un valore minimo e massimo: minore è il numero di partizioni associato ad un nodo maggiore è la sua probabilità di venir considerato anomalia. Il partizionamento ricorsivo può essere rappresentato mediante un albero di isolamento ITree (struttura ad albero binario) mentre il numero di ripartizioni necessarie a isolare un punto corrisponderà alla sua profondità dentro l'albero.

Ponendo $X = \{x_1, ..., x_n\}$ set di punti d-dimensionali e $X' \in X$, un ITree è definito come un albero binario. L'ITree è un albero binario in cui per

ogni nodo $T$ questi è un figlio o interno dell'albero con associati due figli ($T_l$ e $T_r$) e un test o una foglia (ogni istanza dell'albero è unica). Definiamo inoltre un test per un nodo $T$: consiste nello scegliere un attributo $q$ e un valore di divisione $p$ tale che il test $q < p$ determina la selezione tra $T_l$ e $T_r$. Effettuando una serie di test riusciamo a isolarci ogni istanza del dataset

In particolare l'algoritmo IForest divide ricorsivamente $X$ selezionando randomicamente un attributo $q$ e un valore di divisione $p$ finché non si raggiungerà una delle seguenti condizioni:

1. il nodo raggiunge una altezza limite
2. il nodo presenta una sola istanza
3. tutti i valori di un nodo assumono lo stesso valore (non si potrebbe più selezionare un valore casuale di split)

In figura 3.1 si può vedere come per un nodo $x_i$ servano 12 test, quindi corrisponderà tale profondità all'interno dell'albero, mentre per un punto anomalo $x_o$, come osservabile in figura 3.2, servano 4 test: da questo semplice esempio e partendo dalle ipotesi precedenti il nodo $x_o$ verrà considerato maggiormente una anomalia rispetto a $x_i$.
Tale risultato è ottenibile anche utilizzando 1000 ITrees: determinando la distanza di $x_i$ e $x_o$ dalla radice di ogni albero, calcoleremo la media delle distanze. In figura 3.3 notiamo che il percorso medio di $x_i$ e $x_o$ converge all'aumentare del numero degli alberi: in particolare $x_o$ assumerà un valore medio di 4.02 mentre $x_i$ di 12.82. Come già anticipato le anomalie hanno associate distanze minori rispetto ai punti positivi quindi di nuovo $x_o$ verrà considerato maggiormente una anomalia rispetto a $x_i$.

Considerando un albero con n foglie questi avrà $n - 1$ nodi interni: il numero totale di nodi sarà quindi $2n-1$. Di conseguenza lo spazio di memoria necessario sarà dipendente linearmente da $n$.

## 3.1 Punteggio di anomalia

Come già accennato per determinare le anomalie in un dataset servirà produrre una valutazione di ogni nodo di un ITree e valutarne il suo punteggio di anomalia. Determineremo una classifica ordinando i nodi in base alle loro profondità e grado di anomalia: i valori anomali risulteranno all'inizio della lista. Definiamo profondità $h(x)$ di una istanza $x$ del dataset il numero di bordi che devono essere percorsi dal nodo radice dell'albero fino a raggiungere il nodo $x$. Un punteggio di anomalia risulta necessario in metodi di rilevamento di anomalie. Avendo un ITree la stessa struttura di un albero binario

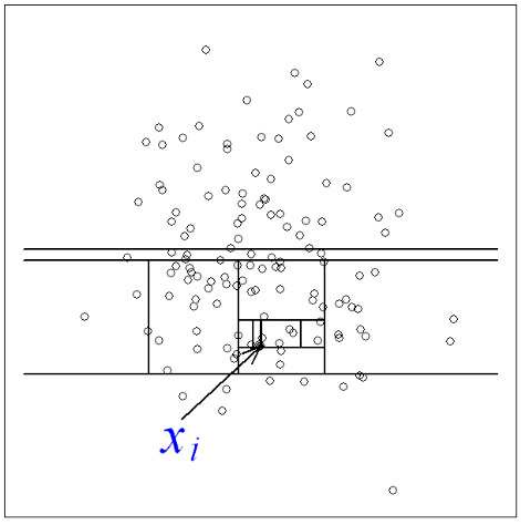


Figura 3.1: Punto Positivo

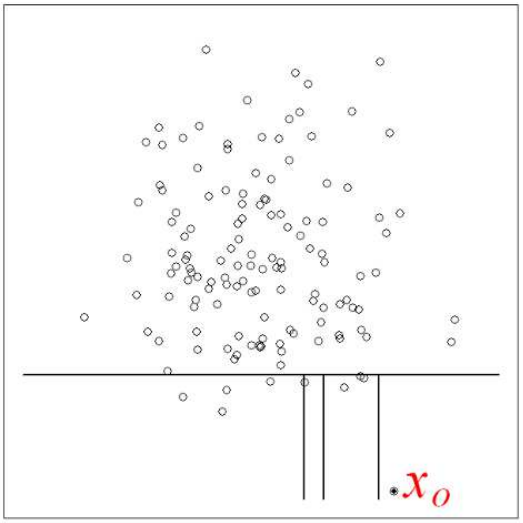


Figura 3.2: Punto anomalo

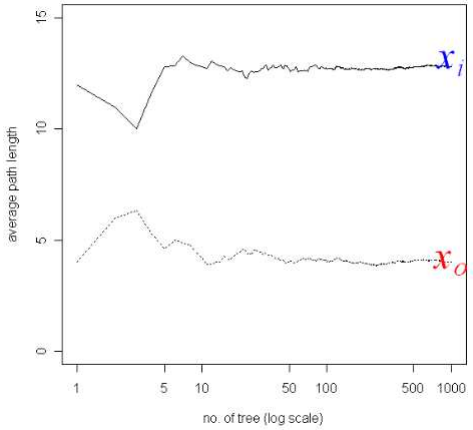


Figura 3.3: Media delle distanze che converge a un valore

(Binary Search Tree, BST) quindi la stima per una media $h(x)$ per le foglie di un ITree è la stessa di quelle per un BST. Possiamo quindi ricondurre l'analisi di un ITree a quella per un albero binario (BST): dato $H(i) = \sum_{k=0}^{i} 1/k$ numero armonico approssimabile a $H(i) \approx \ln i + 0.5772156649$ (costante di Eulero), $n$ il numero 'di istanze di un dataset, $c(n)$ media della altezza $h(x)$ per diversi ITree per ricerche senza successo che useremo successivamente per normalizzare $h(x)$.

$$c(n) = 2H(n-1) - (2(n-1)/n) \tag{3.1}$$

Il punteggio di anomalia $s$ per una istanza $x$ diventa:

$$s(x, n) = 2^{-E((h(x)))/c(n))} \tag{3.2}$$

dove $E((h(x))$ è la media di $h(x)$ per una collezione di ITrees. Data l'equazione (3.2):

1. quando $E((h(x)) = c(n)$, $s \leftarrow 0.5$;
2. quando $E((h(x)) = 0$, $s \leftarrow 1$;
3. quando $E((h(x)) = n - 1$, $s \leftarrow 0$.

Da questi possiamo derivare che se:

1. se le istanze ritornano $s \approx 1$ abbiamo a che fare con una anomalia
2. se le istanze ritornano $s << 0.5$ parliamo di punti positivi
3. se tutte le istanze ritornano $s \approx 0.5$ allora l'intero campione non presenta anomalie significative

## 3.2 Caratteristiche degli alberi di isolamento

La seguente sezione descrive nel dettaglio le caratteristiche degli ITrees i quali, essendo parte di una IForest, riescano a :

1. identificare anomalie avendo queste associate una profondità inferiore
2. agire come esperti dentro il modello tale da concentrarsi prevalentemente su valori anomali

IForest detiene una caratteristica particolare che va contro a diversi modelli anche già precedentemente analizzati: è in grado di avere performance migliori lavorando su modelli parziali senza isolare tutti i nodi positivi alla classe di interesse e costruisce modelli usando una piccola taglia per il campione di riferimento.

Ad esempio SVDD ha bisogno di grandi dataset per gestire al meglio eventuali anomalie: grandi dataset riducono la capacità della IForest nel isolare le anomalie essendo i nodi positivi in grado di interferire con il processo di isolamento e quindi alla rivelazione delle stesse.

IForest riesce a gestire gli effetti di "swamping" e di "masking". Con swamping intendiamo la capacità del me di rivelare valori positivi come anomali. Quando i valori positivi sono troppo vicini alle anomalie la differenza di partizioni tra i due risulta simile di conseguenza non si riescono a distinguere. Con masking intendiamo la presenza di troppe anomalie dentro un insieme di elementi del dataset chiamato cluster: maggiore è la presenza di anomalie dento un cluster maggiore è la difficoltà nel separare le anomalie dai dati positivi alla classe di interesse quindi il numero di partizioni ad esse associate sarà elevato. Notare che sia per swamping che per masking si hanno problemi causa grande taglia del dataset quindi problema di overfitting. La soluzione consiste nel usare un sottocampione in quanto:

1. controlla la dimensione del dataset aiutando IForest ad isolare anomalie

2. ogni ITree può essere specializzato essendo che ogni sottocampione può contenere un differenti set di anomalie oppure anche nessuna.

In figura 3.4 viene mostrato come nel dataset siano presenti valori positivi che influenzano i cluster di anomalie e questi ultimi siano più densi. In figura 3.5 con un sottocampione i cluster di anomalie sono chiaramente identificabili da quelli positivi dato che i primi sono più piccoli e per questo è più semplice la loro identificazione.

## 3.3 Rilevamento di Anomalie con Foresta di Isolamento

Il Rilevamento di Anomalie mediante Foresta di Isolamento è un processo a due fasi: il primo consiste in un allenamento atto a creare ITrees usando sottocampioni del dataset, il secondo consiste in una fase di testing dove verranno studiate delle istanze di prova mediante gli ITrees, precedentemente costruiti, i quali assoceranno a ciascuna istanza un punteggio di anomalia.

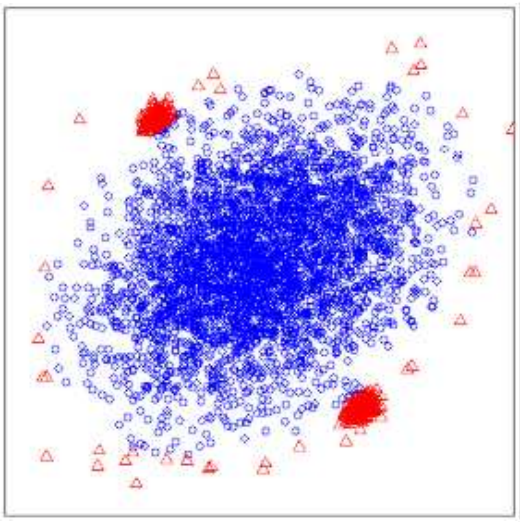

Figura 3.4: Overfitting

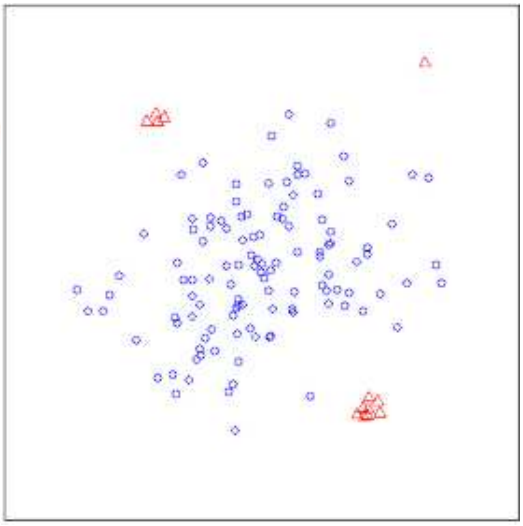

Figura 3.5: sottocampione

### 3.3.1 Allenamento

Come anticipato gli ITrees sono costruiti partizionando ricorsivamente il dataset finché il nodo raggiunge una altezza prestabilita o quando le istanze vengono tutte isolate. Notiamo che l'altezza limite $l$ è impostata automaticamente dalla taglia del sottocampione $\psi$ : $l = \lceil \log_2 n \rceil$ la quale è approssimativamente l'altezza media dell'albero. Ricordiamo che per cercare le anomalie ci focalizzeremo su tutti quei nodi che hanno una profondità minore di $l$. Eventuali dettagli possono venir analizzati negli algoritmi 1 e 2.

**Algorithm 1** iForest$(X, t, \phi)$

**Require:** $X$ dataset, $t$ numero di alberi, $\phi$ taglia del sottocampione
**Ensure:** un set t di ITrees
  Inizializza la IForest
  poni altezza limite $l = \lceil log_2(w) \rceil$
  **for** $i = 1$ to $t$ **do**
    $X' \leftarrow sample(X, \phi)$
    $Forest \leftarrow Forest \cup ITree(X', 0, l)$
  **end forreturn** Forest

La taglia del sottocampione controlla la taglia dell'allenamento del IForest: settando $\psi$ a $2^8$ otteniamo notevoli performance mediamente su una vasta gamma di data. Aumentare il valore di $\psi$ oltre tale soglia non porterebbe ad alcun vantaggio pratico.

**Algorithm 2** ITree$(x, e, l)$

**Require:** $X$ dataset, $e$ altezza albero corrente, $l$ altezza limite
**Ensure:** un ITree
**if** $e \geq l$ || $|X| \leq 1$ **then return** $exNodeSize \leftarrow |X|$
**else**
dato Q una lista di attributi di X
selezionando randomicamente un attributo $q \in Q$
selezionando randomicamente un valore di split $p$ tra due valori, massimo e minimo, di un attributo $q \in X$
$X_l \leftarrow filter(X, q < p)$
$X_r \leftarrow filter(X, q \geq p)$
**return** $inNodeLeft \leftarrow ITree(X_l, e + 1, l), Right \leftarrow ITree(X_r, e + 1, l), SplitAt \leftarrow q, SplitValue \leftarrow p$
**end if**

**Algorithm 3** PathLength$(X, t, e)$

**Require:** $x$ una istanza del dataset $X$, $T$ un ITree, $e$ profondità corrente inizializzata a zero
**Ensure:** profondità di $x$
**if** $T$ è una foglia **then return** $e + c(T.size)$ ▷ c(.) definito nella equazione 3.1
**end if**
$a \leftarrow T.splitAtt$
**if** $x_a < T.splitValue$ **then return** $PathLength(x, T.left, e + 1)$
**else** $x_a \geq T.splitValue$ **return** $PathLength(x, T.right, e + 1)$
**end if**

Alla fine dell'allenamento una collezione di alberi viene restituita ed è pronta per il processo di valutazione. La complessità del metodo IForest è $O(t\psi log(\psi))$

## 3.4 Valutazione della anomalia

Viene assegnato un punteggio di anomalia in base alla profondità media di un nodo $E(h(x))$ per ogni istanza di test. Usiamo l'algoritmo 3 $PathLength$ dove $e$ serve per tenere conto della profondità del nodo corrente, $x$ rappresenta il nodo corrente e $T$ è l'albero del test. Se $x$ è una foglia ritorneremo $e$ più un fattore $c(Size)$ la quale tiene conto per un sottoalbero non costruibile perché andrebbe oltre l'altezza limite dell'albero. Il calcolo del punteggio di anomalia si ottiene mediante l'equazione 3.2. Per determinare le prime $m$ anomalie basta porre i dati in ordine decrescente di s.

# Capitolo 4

# Autocodificatore

Come anticipato nell'introduzione un autocodificatore (in inglese autoencoder, AE) consiste in una rete neurale, ovvero un modello artificiale ispiratosi alle cellule dei neuroni, sottoposta ad allenamento di tipo non supervisionato formata da due componenti (anch'esse realizzate mediante reti neurali): codificatore (in inglese encoder) e decodificatore (in inglese decoder). Il primo prende i dati di ingresso e li trasforma in una rappresentazione di dimensione inferiore rispetto all'originale, il secondo prende il risultato del codificatore e tenta di ricostruire il dato originario. In campo OCC un AE, venendo allenato unicamente per campioni appartenenti alla classe positiva, non riuscirà a ricostruire elementi anomali a differenza di quelli positivi. Il compito principale del AE consiste nel minimizzare l'errore di ricostruzione tra il dato originale in input e l'output del decodificatore: maggiore è la compressione del dato iniziale da parte del codificatore minore la probabilità che l'output del decodificatore sia uguale al dato originale, a meno che il modello riesca a catturare le caratteristiche principali del dataset in modo efficace. Tale errore viene misurato generalmente mediante funzioni di perdita come l'errore quadratico medio (MSE) che studieremo più avanti. In campo OCC un AE, venendo allenato unicamente per campioni appartenenti alla classe positiva, non ricostruirà gli elementi anomali a differenza di quelli positivi.

Secondo lo studio [4] conviene utilizzare la tecnica "Ponderazione delle Caratteristiche per Batch" (Batch-wise Feature Weighting, BFW): questa consiste nel aggiungere un blocco dentro l'AE, posto tra il codificatore e il decodificatore, la quale selezionerà le caratteristiche più importanti per classificare i dati positivi assegnando loro pesi di rilevanza maggiore mentre assegnerà un peso di importanza inferiore a tutte quelle caratteristiche per rilevare dati anomali. Un AE con BFW diventerà più selettivo e otterrà performance migliori rispetto ad un classico AE per un problema di OCC.

In generale un AE si baserà su due fasi principali: la prima consiste in

una fase di allenamento dove gli verranno passate istanze positive alla classe di interesse tali da fargli imparare le caratteristiche necessarie ad un elemento per appartenere alla classe positiva, successivamente una fase di test dove lo stesso AE dovrà capire se un valore del dataset, non appartenente alla fase di allenamento, è anomalo o meno. Andremo quindi a studiare prima un AE classico e successivamente uno basato su BFW.

## 4.1 Autoencoder classico

Useremo la seguente notazione: consideriamo un dataset, con $N$ campioni, $X_{all} = \{x_1, ..., x_N\}^T \in R^{NxD}$ e ogni campione $x_i$ come un vettore $D$-dimensionale. La funzione di codifica (in inglese encoding) $f_e(\cdot, \theta_e)$ parametrizzata da $\theta_e$ costituita dai parametri $W_e, b_e$, con $W_e$ matrice di pesi e $b_e$ vettore di bias parametri allenabili del codificatore, consente di mappare un vettore di ingresso $x$ in una rappresentazione a dimensione inferiore $z$ come

$$z = f_e(x; \theta_e) \tag{4.1}$$

dato $D$ dimensionalità del valore di input originale $x$ e $E$ dimensionalità della rappresentazione latente $z$, $z \in R^L$, dove, generalmente, $E << D$. La funzione di decodifica $f_d(; \theta_d)$ parametrizzata da $\theta_d$ costituita dai parametri $W_d, b_d$, con $W_d$ matrice di pesi e $b_d$ vettore di bias allenabili del decodificatore, utilizza come argomento $z$ tentando di ricostruire l'input iniziale $x$. Fornisce il risultato $\hat{x}$ in uscita:

$$\hat{x} = f_d(z; \theta_d) \tag{4.2}$$

Un Autoencoder classico (vedi figura 4.1) si baserà, per capire se un punto è anomalo o meno, sulla formula dell'errore quadratico medio (in inglese Mean Squared Error, MSE). Tale tecnica consiste nel minimizzare l'errore ottenuto dal confronto tra dato originale e dato ricostruito: dato $x_i$ valore originale del nostro dataset $X_{all}$, contenente $N$ campioni di riferimento, e data $\hat{x_i}$ rappresentazione di $x_i$ ottenuta come risultato del decodificatore. Definiamo l'MSE come:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} \|x_t - \hat{x}_t\|_2^2 \tag{4.3}$$

dove $\|\cdot\|$ indica la norma euclidea. In campo OCC un AE viene allenato con campioni appartenenti alla classe positiva affinché riesca a catturare le caratteristiche principali degli elementi di tale classe: le anomalie non verranno incluse nell'allenamento. Dato un campione di test $x_t$ calcoliamo il suo errore di ricostruzione come:

$$\epsilon_t = \|x_t - \hat{x}_t\|_2^2 \tag{4.4}$$

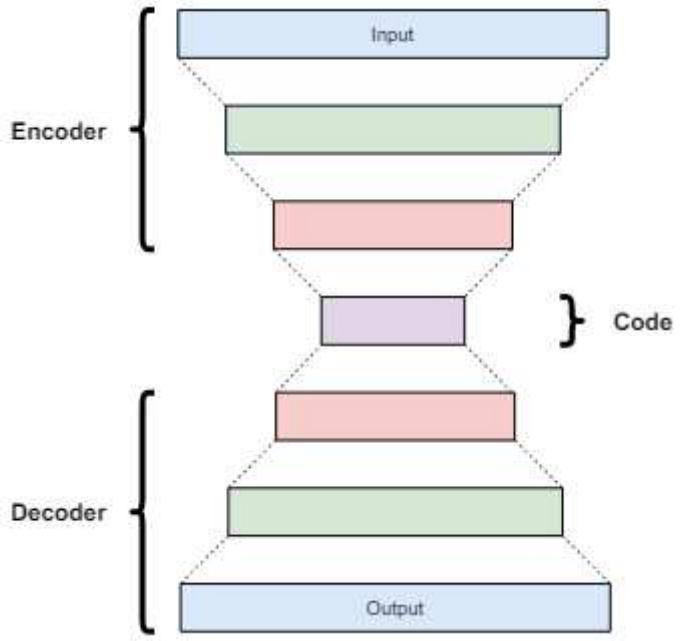


Figura 4.1: In figura classica struttura di un AE. Con "code" intendiamo la rappresentazione latente $z$.

Definiamo $\epsilon$ come valore di soglia predefinito, la quale viene definito in base al tasso di falsi negativi tollerabile (False Negative Rate, FNR) ovvero un valore che stabilisce, dentro un dataset, quanti valori anomali non vengono rilevati dal AE che li classificherà erronamente come valori positivi. Possiamo determinare la natura di $x_t$, ovvero se appartiene o meno alla classe di interesse, come

$$x_t \in \begin{cases} \text{classe conosciuta, se } \epsilon_t < \epsilon \\ \text{classe sconosciuta, altrimenti} \end{cases} \tag{4.5}$$

## 4.2 Comportamento indesiderato del AE classico in OCC

Nonostante il successo della vasta scala di applicazioni di AE per problemi di OCC si riscontrano problemi di ricostruzione non indifferenti.

Dato il dataset MNIST e considerando la cifra "2" come classe di allenamento per un AE con tre differenti capacità per il codificatore 32,64,128 e il resto delle cifre come classi sconosciute notiamo come l'AE riesca a ottenere

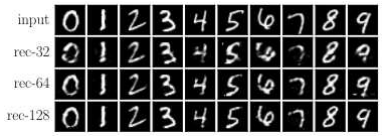


Figura 4.2: Ricostruzioni di tutte le cifre da 0 a 9 data come classe di allenamento la cifra 2 per diverse capacità del AE.


Figura 4.3: Ricostruzioni di tutte le cifre girate (prima riga) e cifre incrinate (terza riga) data come classe di allenamento la cifra 2.

buone abilità di ricostruzione anche avendo una piccola capacità. Come risultato i dati da classi sconosciute vengono ben ricostruiti ciò vuol dire che campioni della classe positiva e quelli delle anomalie hanno piccoli e indistinguibili errori di ricostruzione come mostrato in figura 4.2. Notiamo quindi che AE ottiene un povero rendimento in campo OCC. In figura 4.3 possiamo osservare che in un dataset di cifre "girate", nonostante l'AE venga allenato unicamente per la cifra 2 girata al contrario, tutte le cifre vengono ricostruite con buon margine di approssimazione: stesso ragionamento applicabile a cifre con aggiunta di un'ombra nel mezzo del numero (riga 3 della foto 4.2): anche per questo dataset tutti i campioni sono stati ricostruiti con alta qualità. Simili considerazioni possono essere fatte per altri dataset come Fashion-MNIST e CIFAR-10 per le classi di allenamento Shirt e Frog per l'AE (figura 4.4).

Tutto ciò non è dovuto al fatto che l'AE impara immagini specifiche per le cifre o, in generale, per una immagine di un dataset: quello che l'AE


Figura 4.4: Nonostante gli AE fossero stati allenati sulla colonna 7 tutti gli altri campioni del dataset sono stati ricostruiti con buon margine di qualità

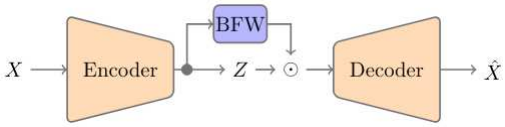


Figura 4.5: Struttura completa di un AE basato su BFW

impara sono le caratteristiche più importanti del dataset come lo stile (ad esempio la forma del numero), regole di composizione ecc per ricostruire il valore originale. La cifra 2 sembra contenere quindi tutte le informazioni per classificare correttamente tutte le cifre portando così l'AE ad avere problemi di generalizzazione indesiderati. Ci serve quindi un metodo che riesca a considerare effettivamente le informazioni di maggiore importanza per un dato appartenente alla classe positiva. Introduciamo il metodo di Ponderazione delle Caratteristiche per Batch (BFW).

## 4.3 Ponderazione delle Caratteristiche per Batch (BFW)

La tecnica consiste nell'introdurre un componente che consenta di lavorare sullo spazio della rappresentazione inferiore dimensionalmente $z$, ovvero tra il codificatore e il decodificatore, per far imparare al AE le caratteristiche principali e così migliorarne la performance per problemi OCC. BFW assegnerà peso maggiore alle caratteristiche più significative mentre alle restanti peso inferiore.

Lo schema in figura 4.5 riassume appieno il concetto: ponendo un blocco dedicato alla tecnica BFW tra codificatore e decodificatore questi aiuterà la struttura principale del AE nel considerare e identificare solo gli aspetti critici della classe conosciuta e a questi assegnerà un peso maggiore rispetto agli altri. Successivamente tali caratteristiche verranno date al decodificatore per ricostruire il dato originale a differenza del AE classico dove viene calcolata solamente la rappresentazione $z$. Otterremo quindi, dato $w \in R^E$ vettore dei pesi assegnati da BFW, la rappresentazinone latente come

$$\hat{z} = w \odot z \tag{4.6}$$

dove $\odot$ denota il prodotto Hadamard, utile per operazioni di tipo elemento per elemento: così facendo ogni elemento della rappresentazione latente $z$ verrà moltiplicato per il suo peso di importanza. Simile al AE con MSE, con BFW otterremo

$$\hat{x} = f_d(\hat{z}, \theta_d) = f_d(w \odot z; \theta_d) \tag{4.7}$$

### 4.3.1 Trasformazione non lineare

Si considerino le rappresentazioni latenti $Z = [z_1, ...., z_N]^T$ dei valori del dataset $X_{all} = [x_1, ...., x_N]^T \in R^{NxD}$ con ogni campione $x_i$ avente $D$ attributi $x_i = [x_{i,1}, x_{i,2}, ..., x_{i,D}]^T$: per determinare il vettore di pesi $w$ da moltiplicare con le rappresentazioni latenti $z$ consideriamo un minibatch (ovvero una piccola collezione di dati di $X_{all}$) $X \in R^{BxD}$ di taglia $B$. Il minibatch verrà mappato ad una rappresentazione, di dimensione $E << D$, $Q = [q_1, q_2, ..., q_B]^T \in R^{BxE}$ mediante una trasformazione non lineare $\sigma(\cdot)$ (ReLU, tanh o sigmoide).

Dati $b_1, W_1, b_2, W_2$ parametri necessari alla seguente trasformazione non lineare con matrici di peso $W_1, W_2$ e il loro vettore di bias associato $b_1, b_2$:

$$q_i = W_2\sigma(W_1 \cdot z_i + b_1) + b_2 \tag{4.8}$$

Tali passaggi risultano necessari per riuscire a trovare relazioni complesse tra diverse caratteristiche durante la fase di allenamento. In particolare $W_1 \in R^{ExD}$, $W_2 \in R^{DxE}$, $b_1 \in R^E$, $b_2 \in R^D$.

Ad ogni campione di $Z$,$z_i$ , corrisponde la sua rappresentazione $q_i$: questo per la selezione di caratteristiche non è conveniente in quanto campioni diversi del minibatch $X$ potrebbero selezionare caratteristiche diverse come importanti. La soluzione consiste nel mediare le diverse rappresentazioni $q_i$ all'intero minibatch $X$ tale da così creare un unico vettore $\overline{q}$ che contenga le caratteristiche più importanti del minibatch per ogni iterazione di allenamento del AE. Segue

$$\overline{q} = \frac{1}{B}\sum_{i=1}^{B} q_i \tag{4.9}$$

### 4.3.2 Normalizzazione della Maschera di caratteristiche

Ottenuto il vettore $\overline{q} = [\overline{q_1}, \overline{q_2}, ..., \overline{q_D}]^T$ e basandoci sullo studio [2] otteniamo il vettore di pesi $w$ come:

$$w = \text{softmax}\,(\overline{q}) \tag{4.10}$$

così facendo il primo parametro della funzione (4.7), $w \odot z$ con $z = \{z_1, z_2, ..., z_N\}$, consisterà in un vettore che terrà conto unicamente delle caratteristiche più importanti nel identificare dati positivi alla classe di interesse consentendo la rivelazione di eventuali anomalie.

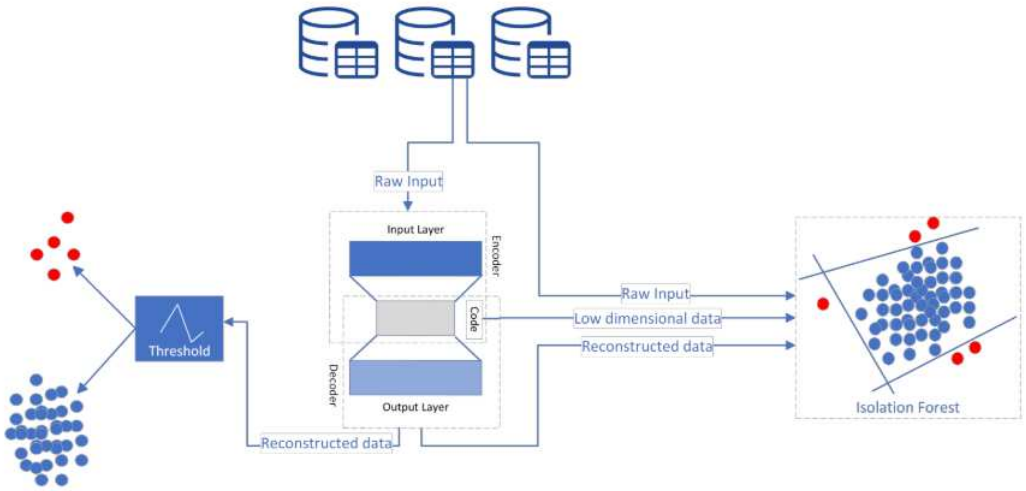


Figura 4.6: Rilevamento di anomalie tramite AE e IF

### 4.3.3 Funzione Di Perdita

Ricordiamo che le caratteristiche ignorabili non vengono eliminate ma verrà associato loro un peso inferiore.

La funzione di perdita rimane MSE (4.3) così come il rilevamento della classe conosciuta e sconosciuta che segue quello definito nella sezione precedente del AE classico (4.5).

## 4.4 Autoencoder e Foresta di Isolamento

Secondo lo studio [1] possiamo unire un Autoencoder con una Foresta di Isolamento (Isolation Forest, IF) e ottenere una tecnica di rivelazione di anomalie come mostrato in figura 4.6. Per integrare l'AE con una IF devono essere aggiungenti diversi passaggi:

1. La prima consiste nella divisione del dataset in due sottoset: uno verrà dedicato all'allenamento del AE mentre il secondo verrà usato come fase di test. Ognuno dei due sottoset viene diviso in due dataset: uno conterrà istanze positive alla classe di interesse mentre il secondo conterrà anomalie.

2. Successivamente l'AE verrà allenato usando due configurazioni: la prima userà un dataset misto allenando l'AE in modalità non supervisionata, la seconda allenerà l'AE usando dati positivi, implementando così una strategia ad allenamento semi supervisionato.

3. Rilevamento di anomalia: procedimento dell'AE classico dove viene calcolata la funzione MSE per ogni configurazione di allenamento e, ordinando in ordine crescente i punteggi MSE ottenuti, la selezione di un insieme di possibili valori utili alla determinazione di un ottimo valore di soglia. Seguendo la (4.5) se un elemento assume errore di ricostruzione maggiore del valore di soglia allora verrà identificato come anomalia.

4. Per finire l'ultimo passaggio consiste nel rivelamento di anomalie mediante IF. Passiamo come parametri a IF:

   (a) il dataset originale: qui non vi è sinergia tra le due tecniche in quanto IF si baserà unicamente sui suoi risultati ottenuti

   (b) rappresentazione latente ottenuta dal AE allenato in maniera non supervisionata

   (c) rappresentazione latente ottenuta dal AE allenato in maniera semi supervisionata

   (d) rappresentazione decodificata ottenuta dal AE non supervisionato

   (e) rappresentazione decodificata ottenuta dal AE semi supervisionato

Tale tecnica migliora l'accuratezza nella rilevazione di anomalie comparandolo ai risultati dovuti alla applicazione dei metodi presi singolarmente.

# Capitolo 5

# PCA

Abbiamo già visto per il caso dell'Autoencoder (vedi capitolo 5) come riducendo la dimensionalità del dataset di input sia più facile ed efficiente riconoscere la classe di interesse. La tecnica Analisi delle componenti principali (Principal Component Analysis, PCA) è un metodo di apprendimento non supervisionato: consiste in una trasformazione lineare che trasforma il dataset di input in un sottospazio (chiamato spazio delle caratteristiche) di dimensione inferiore. Tale tecnica per un dataset consente di determinare le caratteristiche principali degli elementi della classe positiva ignorando le restanti, permettendo così l'isolamento di eventuali anomalie.

Tale sottospazio secondo [6] è rappresentato dagli autovettori della matrice di covarianza del dataset $C$: dato $N$ numero di campioni del dataset, $x_i$ campione del dataset $X_{all}$ e $\mu$ valore medio del dataset $X_{all}$ calcolato come $\mu = \sum_{i=1}^{N} x_i$, $C$ viene definita come:

$$C = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T \tag{5.1}$$

Mediante PCA vengono trovati un set di autovettori $\{v_1, v_2, ..., v_N\}$ corrispondenti agli autovettori di $C$. Data una immagine di test $x_{test}$ questa può essere proiettata nello spazio definito dai vettori $\{v_1, v_2, ..., v_N\}$ dove la proiezione della ampiezza sull'autovettore $v_i^{t-simo}$, $||v_i||_2$, è calcolata come $||v_i||_2 = v_i^T(x_{test} - \mu)$. Una caratteristica viene definita considerando la collezione di ampiezze $v_{test} = ||v_1||_2, ||v_2||_2, ..., ||v_N||_2$.

## 5.1 Kernel PCA

Kernel PCA permette di rendere più efficiente la tecnica PCA nel gestire la non-linearità dei dati. Supponiamo di avere una funzione $\phi : R^n \rightarrow R^d$

che consente di mappare un valore $x_i$ di un dataset $X_{all}$ in uno spazio delle caratteristiche $\phi(x_i)$ di dimensione $d$ con, generalmente, $d < n$.

Per semplicità supponiamo che i valori mappati siano centrati nello spazio delle caratteristiche, questo significa togliere ad ogni valore del dataset la media dei valori, quindi $\frac{1}{N}\sum_{i=1}^{N}\phi(x_i) = 0$: questo comporta vantaggi per eventuali bias dovuti alla posizione dei punti all'interno del dataset $X_{all}$. Otteniamo la matrice di covarianza nello spazio delle caratteristiche come:

$$\overline{C} = \frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\phi(x_i)^T \tag{5.2}$$

Per eseguire la PCA nello spazio delle caratteristiche dobbiamo trovare un set di autovettori e autovalori non nulli. Denotiamo $\lambda_j$ e $v_j$ autovalore e autovettore corrispondenti a $\overline{C}$. In particolare si può rappresentare $v_j$ come $v_j = \sum_{i=1}^{N}\alpha_i^j\phi(x_i)$ con $\alpha_1^j, \alpha_2^j, ..., \alpha_N^j$ set di coefficienti pesati. L'autovalore e l'autovettore devono soddisfare $\lambda_j v_j = \overline{C}v_j$, per calcolare la PCA nello spazio delle caratteristiche possiamo considerare un sistema equivalente come:

$$\lambda_j\langle\phi(x_i), v_j\rangle = \langle\phi(x_i), \overline{C}v_j\rangle, \text{ for } i = 1, ..., N \tag{5.3}$$

sostituendo $v_j = \sum_{i=1}^{N}\alpha_i^j\phi(x_i)$ in (5.3) otteniamo:

$$\lambda_j\langle\phi(x_i), \sum_{l=1}^{N}\alpha_i^j\phi(x_l)\rangle = \langle\phi(x_i), \overline{C}\sum_{l=1}^{N}\alpha_i^j\phi(x_l)\rangle, \tag{5.4}$$

dove $\langle\cdot,\cdot\rangle$ indica il prodotto interno tra due vettori.

Definiamo vettore colonna come $\alpha^j = [\alpha_1^j, \alpha_2^j, ..., \alpha_N^j]^T$ e matrice $K$ di dimensione $NxN$ dove ogni valore viene definito come $K_{rt} := \langle\phi(x_r), \phi(x_t)\rangle$. Possiamo semplificare (5.4) come:

$$N\lambda_j\alpha^j = K\alpha^j \tag{5.5}$$

Troviamo la soluzione $\alpha_j$, per autovalori diversi da zero, quando i vettori corrispondenti in $R^d$ sono normalizzati ovvero $\langle v_j, v_j\rangle = 1$: determiniamo così anche gli autovettori $v_j$. I primi autovettori avranno maggiore importanza, in ordine decrescente, nel determinare le caratteristiche principali utili a identificare punti appartenenti alla classe di interesse. Per una immagine vettorizzata $x_{test}$ possiamo determinare la sua proiezione nello spazio definito dagli autovettori calcolando la proiezione di $x_{test}$ rispetto al $j-$simo autovettore nello spazio delle caratteristiche come

$$\langle v_j, \phi(x_{test})\rangle = \sum_{i=1}^{N}\alpha_i^j\langle\phi(x_i), \phi(x_{test})\rangle \tag{5.6}$$

Notare che in (5.6) non necessitiamo la definizione della funzione mappante $\phi$ ma ne abbiamo bisogno solo nella forma del prodotto scalare. Chiamiamo la funzione $\phi$ funzione di kernel. Un esempio di funzione di kernel potrebbe essere la funzione di base radiale (radial basis function, RBF), ovvero una funzione con argomenti due dati del dataset, il cui valore dipende dalla distanza di $x_i$ da $x_j$ con $x_j$ parametro fisso. RBF viene definita come:

$$K(x_i, x_j) = e^{\frac{-||x_i - x_j||^2}{2\sigma^2}} \tag{5.7}$$

con $\sigma$ parametro del kernel: maggiore il valore assunto da $\sigma$ maggiore è l'influenza di due punti distanti sul valore di $K(x_i, x_j)$.

## 5.2 Analisi delle Componenti Rilevanti

La tecnica analisi delle componenti rilevanti (Relevant Component Analysis, RCA), secondo [11], ottiene risultati migliori rispetto alla classica PCA: è una tecnica basata su allenamento di tipo non supervisionato, come PCA, la quale si basa sul calcolo della distanza di Mahalanobis mediante l'impiego di relazioni di equivalenza positiva, realizzate basate sul calcolo di matrici di covarianza di punti.

Dato un dataset $X$ e un suo chunklet, un set di dati di $X$, i dati di un chunklet appartengono alla stessa classe ma senza saperne la loro natura, ovvero se sono punti anomali o positivi. ogni chunklet viene determinata una matrice di covarianza, fondamentale nella determinazione di un adeguato spazio delle caratteristiche. Tale tecnica permette l'assegnamento di pesi maggiori alle caratteristiche rilevanti, pesi inferiori alle caratteristiche restanti.

### 5.2.1 Descrizione Analisi delle Componenti Rilevanti

Parleremo della realizzazione pratica di RCA: Descrizione Analisi delle Componenti Rilevanti (Relevant Component Analysis Data Description, RCADD). Innanzitutto dividiamo il dataset $\{x_i\}_{i=1}^{N} \in R^n$ in almeno $D > 1$ datasets ognuno chiamato chunklet. Dato $D$ numero dei chunklets, $n_d$ numero di campioni dentro $d$-simo chunklet, $\overline{x_d}$ media del $d$-simo chunklet la RCA calcola la matrice di covarianza dei chunklet come:

$$M = \frac{1}{N} \sum_{d=1}^{D} \sum_{j=1}^{n_d} (x_j^d - \overline{x_d})(x_j^d - \overline{x_d})^T \tag{5.8}$$

RCA applica una trasformazione di sbiancamento, ovvero prende la matrice di covarianza $M$ e ne rimuove eventuali correlazioni normalizzando le varianze:

$$z = M^{-\frac{1}{2}}x \tag{5.9}$$

Il dataset $\{x_i\}_{i=1}^N \in R^n$ viene trasformato in $\{z_i\}_{i=1}^N \in R^n$. Questo processo viene identificato come l'estrazione delle caratteristiche: assegniamo pesi maggiori a caratteristiche rilevanti e pesi inferiori a caratteristiche irrilevanti.

Ora applichiamo lo stesso ragionamento al vettore $z$: la tecnica consente di definire un nuovo sottospazio definito da autovettori della matrice di covarianza $Q \in R^{nxn}$. Dato $\overline{z}$ media di $\{z_i\}_{i=1}^N \in R^n$, $Q$ viene calcolata come:

$$Q = \sum_{i=1}^{N}(z_i - \overline{z})(z_i - \overline{z})^T \tag{5.10}$$

Supponendo di usare i primi $p$ autovettori $e^i \in R^n, i = 1, ..., p$ di Q, avendo questi maggiore rilievo nella identificazione di punti positivi, RCADD può fornire una nuova matrice di trasformazione $A = [e_1, ..., e_p] \in R^{nxp}$. Per classificare un nuovo oggetto $y \in R^n$, RCADD deve calcolare l'errore di ricostruzione come:

$$g(x_i) = ||z_i - z_i'|| \tag{5.11}$$

dove $z_i$ è la proiezione di un oggetto $x_i$ secondo RCA mentre $z_i'$ è la ricostruzione di $x_i$ dato dall'equazione

$$z_i' = A(A^TA)^{-1}A^Tz \tag{5.12}$$

l'errore medio di ricostruzione $\theta$ è dato da

$$\theta = \frac{1}{N}\sum_{i=1}^{N} g(x_i) \tag{5.13}$$

Così un oggetto di test $y$ prima diventa $y_{new}$ grazie all'equazione 5.9 e poi rende $y_{new}$ parte integrante del sottospazio composto da autovettori $e_i \in R^n, i = 1, ..., p$. Poi se il test oggetto $y$ assume :

$$g(y) \leq \theta \tag{5.14}$$

allora $y$ appartiene alla classe di interesse, altrimenti verrà considerato valore anomalo.

# Capitolo 6

# Conclusioni

OCC rappresenta una soluzione nella identificazione di anomalie: questa tesi ha preso in analisi lo studio di diverse tecniche di OCC quali Descrizione Dati con Vettori di Supporto (Supported Vector Data Description, SVDD), foresta di isolamento (Isolation Forest, IForest), Autocodificatori (Autoencoder, AE), Analisi Componenti Principali (Principal Component Analysis, PCA).

**Riassunto delle tecniche utilizzate**

1. **SVDD**

   (a) Delinea una sfera tale da contenere tutte le istanze positive alla classe di interesse. Eventuali anomalie risulteranno fuori dalla sfera stessa.

2. **Isolation Forest**

   (a) Determina una struttura ad albero chiamata ITree: minore è la profondità associata di un nodo all'interno del ITree maggiore la sua probabilità di venir considerato anomalia. Algoritmo meno complesso rispetto alle altre soluzioni proposte.

   (b) In applicazioni pratiche consente per problemi a grandi dimensioni con elevato numero di attributi irrilevanti un efficiente rilevamento di eventuali anomalie [3].

3. **Autocodificatore**

   (a) Rete neurale che permette di gestire dati con relazioni non lineari e di alta dimensione.

(b) Una introduzione del rumore secondo [10] potrebbe migliorare ulteriormente la qualità di generalizzazione della rete, riducendo problemi di overfitting ovvero quando la rete risulta troppo specializzata e non riesce a cogliere le caratteristiche generali e principali del dataset.

4. **PCA**

(a) Calcolo di una rappresentazione chiamata spazio delle caratteristiche di dimensione inferiore rispetto l'originale. Mediante il calcolo di un opportuno valore di soglia vengono rilevate eventuali anomalie.

OCC risulta in continuo sviluppo con applicazione in svariati ambiti di ricerca: applicazione in sistemi meccanici nella rilevazione di anomalie in sistemi di diagnostica, sicurezza informatica, identificazioni di frodi, controlli statistici, big data, prognostica delle apparecchiature e gestione sanitaria [7] [6].

Nonostante il vasto utilizzo presenta dei difetti i quali necessitano ulteriori approfondimenti: in questo capitolo conclusivo ne tratteremo alcuni.

Innanzitutto tutti i metodi visti si basano su una fase di allenamento su elementi tutti positivi alla classe di interesse: questo comporta enormi limitazioni in fase pratica. OCC mediante allenamento di tipo semi supervisionato e dati annotati, non solo quelli positivi alla classe di interesse ma anche quelli provenienti da un dataset fuori distribuzione (Out Of Distribution,OOD), riuscirebbe a riconoscere più facilmente i confini della classe positiva quindi eventuali anomalie, portando così a una migliore efficienza rispetto a un metodo di allenamento non supervisionato.

Un secondo problema è dato dalla complessità del dataset in analisi [6]: i metodi sopracitati non presentano problemi per dataset di semplice complessità (vedi MNIST) ma per dataset complessi (esempio CIFAR10) OCC fallisce nel generalizzare il modello generale. Per un problema di OCC, applicato a dataset complessi, risulta quindi necessario trovare o migliorare la generalizzazione data dalle soluzioni di ogni metodo.

Dati avversari, ovvero dati manipolati appositamente per far fallire le predizioni a un modello, costituiscono un grande fattore limite per OCC. Studi specifici aiuterebbero nel creare modelli più sicuri.

Un ulteriore problema è l'assunzione, da parte del classificatore di OCC, che i valori in fase di test abbiano la stessa distribuzione dei valori in ingresso [8]. Sviluppare quindi tecniche atte a produrre classificatori generalizzabili attraverso differenti distribuzioni di dati rimane ancora un argomento da approfondire.

# Bibliografia


[1] Mahmood K. M. Almansoori e Telek Miklos. "Anomaly Detection using combination of Autoencoder and Isolation Forest". In: *2ELKH-BME Information Systems Research Group, Budapest, Hungary* (2023).

[2] Vaswani Ashish et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems* (2017), pp. 5998–6008.

[3] Alberto Bressan e Gian Antonio Susto. "Anomaly Detection: analisi teorica e indagine sul metodo Isolation Forest". In: *IEEE International Conference on Data Mining* (2023).

[4] Yiwen Liao e Bin Yang. "To Generalize or Not to Generalize: Towards Autoencoders in One-Class Classification". In: *Institute of Signal Processing and System Theory* (2022).

[5] Fei Tony Liu, Kai Ming Ting e Zhi-Hua Zhou. "Isolation Forest". In: *IEEE International Conference on Data Mining* (2008).

[6] Pramuditha Perera, Poojan Oza e Vishal M. Patel. "One-Class Classification: A Survey". In: *arXiv:2101.03064* (2021).

[7] Naeem Seliya, Azadeh Abdollah Zadeh e Taghi M. Khoshgoftaar2. "A literature review on one-class classification and its potential applications in big data". In: *Journal of Big Data* (2021).

[8] V. A. Sindagi e S. Srivastava. "Domain adaptation for automatic oled panel defect detection using adaptive support vector data description." In: *Int. J. Comput. Vision, 122(2):193–211* (2017).

[9] David M.J. Tax e Robert P.W. Duin. "Support Vector Data Description". In: *Pattern Recognition Group, Faculty of Applied Sciences, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands* (2004).

[10] Pascal Vincent, Hugo Larochelle e Yoshua Bengio. "Extracting and Composing Robust Features with Denoising Autoencoders". In: *ACM International Conference Proceeding Series* (2008).

[11] Zhe Wang e Daqi Gao. “A New One-class Classifier: Relevant Component Analysis Data Description”. In: *Journal of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China* (2012).