UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea in Fisica

Tesi di Laurea

Riduzione della dimensionalità dell'attività cerebrale tramite autoencoder variazionale

Relatore

Dr. Michele Allegra

Laureando

Jacopo Frignani

Anno Accademico 2021/2022

Abstract

In questa tesi, si descrive la teoria degli autoencoder variazionali (variational autoencoders, VAE) e si descrive una sua applicazione a un insieme di dati complesso (serie temporali di attività cerebrale). Per prima cosa, si richiama brevemente il problema generale della riduzione di dimensionalità. Quindi, si introduce la teoria dei VAE, che costituiscono uno dei metodi più usati per la riduzione dimensionale. Successivamente, si testa il funzionamento di un semplice VAE su dati artificiali (database MNIST). Infine, partendo da dati reali che descrivono l'attività cerebrale provenienti dalla risonanza magnetica funzionale, si implementa un VAE per ottenere una riduzione della dimensionalità dell'attività e confrontare i risultati tra soggetti sani e pazienti che hanno avuto un ictus.

Indice

Capitolo 1

Introduzione

Nella tesi verrà trattata la riduzione della dimensionalità di un insieme di dati, che descrivono l'attività cerebrale di diversi soggetti, tramite l'impiego di una rete neurale, nello specifico di un autoencoder variazionale (VAE).
Nella prima parte, si introdurrà il concetto di riduzione della dimensionalità e si discuterà il metodo dell'analisi delle componenti principali, una delle tecniche più usate per attuare la riduzione della dimensionalità. Inoltre, dopo un breve accenno alle reti neurali, verrà trattato il metodo che si avvale di una rete, nello specifico di un autoencoder, per la riduzione delle caratteristiche e la ricostruzione dei dati. Successivamente, verranno illustrati i motivi che portano all'introduzione di un autoencoder variazionale per trattare il problema inerente alla ricostruzione dei dati e alla generazione di dati nuovi. Si affronterà, quindi, la trattazione matematica di un VAE, concentrandosi sulle assunzioni probabilistiche del modello e la formalizzazione del problema di ottimizzazione della ricostruzione dei dati. In aggiunta, verrà discussa una possibile architettura per implementare concretamente un autoencoder variazionale.
Nella seconda parte della tesi, si implementerà concretamente un VAE. Per prima cosa si testerà la rete sull'insieme di dati del database MNIST, osservando che il VAE riesce a comprimere e ricostruire le immagini del dataset. Oltre a ciò, avvalendosi della struttura variazionale della rete, si proverà a generare dati nuovi, osservando che questi si accordano ottimamente con i dati iniziali. A seguire, dopo aver variato alcuni parametri della rete, si è adoperata la rete per l'analisi di un set di dati che esprimono l'attività cerebrale di un gruppo di pazienti affetti da ictus cerebrale e un gruppo di pazienti sani. Dallo studio della ricostruzione di dati in ingresso alla rete per i due gruppi di pazienti, si noterà che i dati dei pazienti sani hanno dei pattern che li accomunano maggiormente tra loro, rispetto a quelli pazienti malati. Inoltre, dallo studio della funzione di costo al variare della dimensione dello spazio latente della rete si dedurrà che non è possibile una riduzione dimensionale dei dati dell'attività cerebrale in uno spazio con dimensione bassa.

Capitolo 2

Riduzione della dimensionalità

Nel seguente capitolo verrà definito il problema della riduzione della dimensionalità in ambito computazionale, introducendo anche una delle principali tecniche usate a questo scopo. Successivamente, si osserverà come poter usare una rete neurale, in particolare un autoencoder, per ridurre la dimensione di un dataset.

2.1 Riduzione della dimensionalità in un dataset

Come sottolineato in [1] da Van Der Maaten et al. "*I dati reali, come ad esempio i segnali vocali, le fotografie digitali o le scansioni provenienti da una risonanza magnetica funzionale (fMRI), di solito hanno un'elevata dimensionalità. Per gestire adeguatamente tali dati, la loro dimensionalità deve essere ridotta...*". Questo approccio trasforma lo spazio ad alta dimensione, che descrive i dati di partenza, in uno spazio di dimensione inferiore, andando a ridurre il numero di caratteristiche che descrivono i dati. Idealmente, la riduzione ottimale dovrebbe riuscire a mantenere solo le caratteristiche intrinseche dei dati, cioè quelle necessarie per la loro completa descrizione. Per attuare la riduzione dimensionale, occorre passare per un processo detto estrazione delle caratteristiche (*feature extraction*), che consiste nel creare k nuove caratteristiche, partendo dalle d iniziali.

2.2 Compressione e decompressione delle caratteristiche

Si introduce il termine *encoder* per riferirsi al processo di estrazione in cui vengono create le nuove caratteristiche, mentre il processo inverso è detto *decoder*. Formalizzando questi procedimenti, l'encoder può essere visto come una funzione $e\colon x \in \mathbb{R}^d \to e(x) \in \mathbb{R}^k$. L'encoder quindi comprime i dati dallo spazio iniziale allo spazio con dimensione ridotta, che viene chiamato *spazio latente*. Il decoder, invece, viene definito da una funzione $d\colon z \in \mathbb{R}^k \to e(z) \in \mathbb{R}^d$. Questo processo decomprime i dati, cercando di ricostruire le caratteristiche dello spazio iniziale da quelle dello spazio latente. Il seguente grafico aiuta a comprendere meglio le azioni dell'encoder e del decoder.

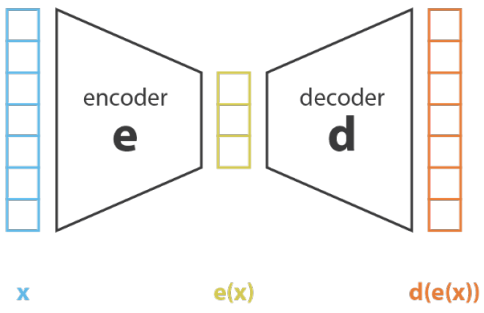


Figura 2.1: Illustrazione del processo di riduzione della dimensionalità e del suo inverso

A seconda del dataset, della dimensione dello spazio latente e dell'architettura dell'encoder, una parte dell'informazione sui dati può essere persa durante la compressione. Ovviamente, poiché l'informazione smarrita non può essere recuperata durante il processo di decoder, i dati ridotti saranno diversi da quelli iniziali, evidenziando un'irreversibilità nel processo. Si avrà quindi:

$$x \neq d(e(x)) \tag{2.1}$$

Può anche essere costruita una funzione $\epsilon(x, d(e(x)))$ che esprime l'errore associato alla ricostruzione dei dati. In particolare si può misurare l'errore con lo scarto quadratico:

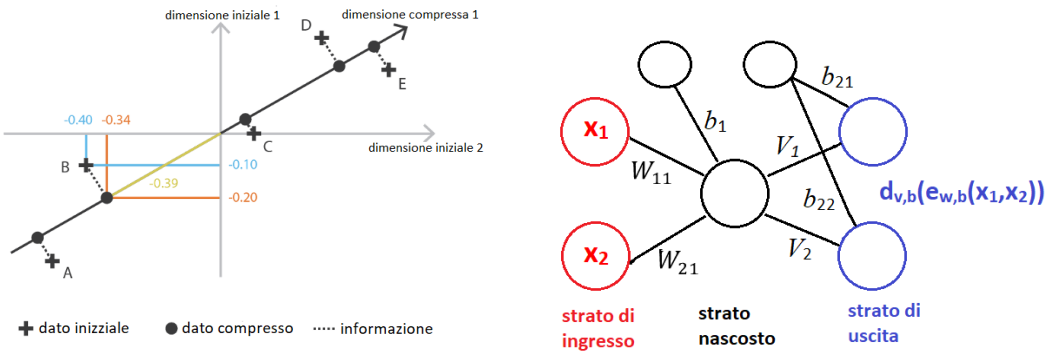
$$\epsilon(x, d(e(x))) = ||x - d(e(x))||^2 \tag{2.2}$$

Il principale scopo della riduzione della dimensionalità è trovare la miglior coppia encoder/decoder tra le diverse possibilità, tale da massimizzare l'informazione trattenuta, durante la compressione, e minimizzare $\epsilon(x, d(e(x)))$. Formalmente, date le famiglie di funzioni E e D che descrivono l'encoder e il decoder, vogliamo trovare la coppia (e^*, d^*) tale che:

$$(e^*, d^*) = \underset{(e,d)\in E\times D}{\operatorname{argmin}}\ \epsilon(x, d(e(x))) \tag{2.3}$$

2.3 Analisi delle componenti principali

Tra le tecniche più usate per la riduzione della dimensionalità ci sono quelle lineari, di cui l'esempio più comune è l'analisi delle componenti principali (PCA). Il metodo della PCA cerca il miglior sottospazio lineare, creato dai vettori di base ortogonali delle nuove caratteristiche, tale da minimizzare la somma degli errori di ricostruzione $\epsilon(x, d(e(x)))$. Qui e è una mappa lineare che proietta i vettori x in un sottospazio di dimensione ridotta, mentre d rappresenta semplicemente la mappa che immerge il sottospazio nello spazio originario. La PCA, quindi, trova il sottospazio che rende minima la distanza euclidea tra i dati iniziali e le loro proiezioni nel sottospazio lineare. In Fig. 2.2a si riporta un esempio grafico.



(a) Esempio grafico dell'analisi delle componenti principali

(b) Esempio dell'architettura neurale di un autoencoder

Figura 2.2

Inoltre, la riduzione della dimensionalità tramite la PCA può essere formulata come un problema matriciale. Infatti si può dimostrare che gli autovettori unitari corrispondenti ai k autovalori con norma maggiore della matrice di covarianza delle d caratteristiche iniziali sono ortogonali e definiscono il migliore sottospazio di dimensione k in cui proiettare i dati iniziali con il minimo errore $\epsilon(x, d(e(x)))$. Quindi il problema di trovare le k caratteristiche ottimali per la riduzione è stato ricondotto al calcolo degli autovalori e autovettori della matrice di covarianza dei dati iniziali. Dunque, in questa formulazione, l'encoder è espresso dalla matrice di dimensione $k \cdot d$, dove le k righe sono gli autovettori che descrivono gli autovalori con norma maggiore che descrivono le caratteristiche nel nuovo sottospazio. Mentre il decoder risulta essere la una matrice di dimensione $d \cdot k$, in cui le colonne rappresentano questi k autovettori.

2.4 Uso delle reti neurali per la riduzione della dimensionalità: gli autoencoders

Una rete neurale artificiale, che da ora in avanti verrà chiamata semplicemente rete neurale, è un modello matematico e informatico che si basa sulle reti neurali biologiche. Come si osserva in figura 3.3, questo modello è realizzato tramite una sequenza di diversi strati che contengono un insieme di neuroni artificiali (i nodi), raffigurati tramite cerchi. Ogni nodo è connesso ai neuroni artificiali dello strato (layer) precedente e successivo, se esistono, tramite dei legami. A ogni legame tra due nodi i e j si associa un peso W_{ij}, che descrive quanto i due neuroni artificiali sono legati. Per rendere più flessibile il modello, a ogni neurone viene associato anche un bias b_i, che definisce la soglia di segnale in input sotto la quale il neurone artificiale non si attiva durante il processo. Inoltre a ogni nodo è anche associata una funzione di attivazione, che descrive il segnale in uscita dal nodo dato l'insieme di input che arrivano dai nodi dello strato precedente.
Per la riduzione della dimensionalità può essere costruita una rete neurale, chiamata autoencoder, che consiste nell'accoppiare un encoder e un decoder. Questa è costituita da un'architettura neuronale in cui il numero di input e output sia lo stesso, e in cui si richiede che i dati in uscita siano il più simili possibile a quelli in ingresso. Per soddisfare questà richiesta bisogna addestrare l'intera rete tramite l'*algoritmo di retropropagazione*. Facendo riferimento alla rete nella figura 2.2b, l'algoritmo confronta l'output $d_{V,b}(e_{W,b}(x)) = d_{v,b}(e_{w,b}(x_1, x_2))$ con il dato iniziale $x = (x_1, x_2)$. Successivamente, l'algoritmo cambia i pesi $W_{i,1}, V_j$ e i biases $b_1, b_{2,j}$ della rete, cercando di minimizzare l'errore di ricostruzione $\epsilon(x, d_{V,b}(e_{W,b}(x)))$, calcolato tramite il metodo *steepest descent* [2].

Molto spesso vengono usate reti neurali profonde (molti strati) e non lineari, in cui si hanno diversi strati latenti e una funzione di attivazione non lineare. In questi casi, l'architettura risulta più complessa e meno intuitiva, ma la rete diventa più performante.

Capitolo 3

Autoencoder variazionali

Nel seguente capitolo verranno trattati i principali aspetti che caratterizzano gli autoencoder variazionali. Si partirà giustificando la loro introduzione e definendoli. Successivamente verra affrontata la loro trattazione matematica.

3.1 Limitazioni degli autoencoders per la generazione di nuovi dati

Si è osservato nel capitolo precedente che lo strato nascosto di un autoencoder, nel quale si ha un numero di neuroni minore rispetto allo strato di input, realizza una riduzione della dimensionalità. Le caratteristiche di questa riduzione e del nuovo spazio creato dipendono dallo scopo per cui si è allenata la rete [3]. Infatti un autoencoder può essere usato, come già visto, per comprimere dati già esistenti in modo che una volta decompressi siano il più simili possibile a quelli originali. Oltre a questo, un AE può anche essere realizzato per generare nuovi dati in output, andando ad addestrare la rete e successivamente decomprimendo dei punti campionati casualmente dallo spazio latente. Dovendo rispondere a due esigenze diverse, il processo di addestramento di un autoencoder che deve comprimere dei dati sarà diverso rispetto a quello di un AE che deve generare dati. Infatti per un autoencoder che generi nuovi dati, ad esempio, si vorrebbe che due punti vicini nello spazio latente non restituiscano due dati decodificati significativamente diversi e che ogni punto latente campionato fornisca un dato sensato, una volta decodificato. Quindi la qualità di questi nuovi dati dipende dalla regolarità delle distribuzioni che descrivono i punti iniziali nello spazio latente. Questo risulta essere un argomento molto delicato, in quanto garantire delle distribuzioni organizzate in modo da avere una generazione di dati sensata è molto difficile. Infatti, per un autoencoder semplice, la distribuzione nello spazio latente è descritta da un'unione di punti distanti tra loro che non riesce a ricoprire in maniera efficace lo spazio. Pertanto, il campionamento di punti dallo spazio latente potrebbe portare a ottenere degli output nuovi che non hanno senso se relazionati alle caratteristiche dell'insieme dei dati iniziale. Nella Fig. 3.1 è riportato uno spazio con delle distribuzioni regolarizzate e uno in cui questa condizione non è verificata.

Figura 3.1: Differenza tra uno spazio latente con distribuzioni regolarizzate e uno in cui non sono regolarizzate

Queste problematiche riguardanti la creazione di nuovi dati portano all'introduzione degli autoencoder variazionali.

3.2 Definizione degli autoencoders variazionali

Come afferma Doersch in [4], un autoencoder variazionale (VAE) può essere visto come la combinazione di due reti accoppiate, ma parametrizzate in modo indipendente, che descrivono un encoder e un decoder, proprio come un AE, in cui però ci si pone l'obbiettivo di generare nuovi dati. Quindi, un VAE può essere pensato come un autoencoder in cui l'addestramento è realizzato in modo da assicurarsi che le distribuzioni nello spazio latente abbiano le giuste proprietà per consentire la generazione di nuovi dati $d(z)$. Un VAE, invece che codificare un dato in input x con un singolo punto z nello spazio latente come fanno gli autoencoders, associa ad ogni input una distribuzione di probabilità $p(z|x)$ nello spazio latente. In seguito, viene campionato casualmente un punto z dalla distribuzione $p(z|x)$, che verrà poi decodificato. Adesso è possibile calcolare $\epsilon(x, d(e(x)))$ e retropropagarlo attraverso la rete per aggiornarne i parametri. Inoltre, la condizione che $p(z|x)$ sia forzata ad essere vicina a una gaussiana standard rende facilmente possibile avere delle distribuzioni di probabilità nello spazio latente regolarizzate, caratterizzate da gaussiane con media μ_x e matrice di covarianza σ_x calcolate dall'encoder per ogni input.

Nella figura 3.2 si riassume la differenza tra un autoencoder e un autoencoder variazionale, in cui notiamo che il primo risulta essere un processo deterministico mentre il secondo probabilistico. Nella figura 3.3 Viene inoltre riportato un esempio di architettura che implementa un VAE.

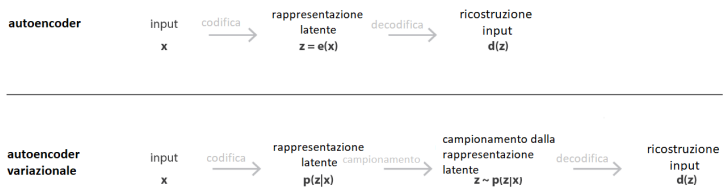


Figura 3.2: Differenza tra un autoencoder e un VAE

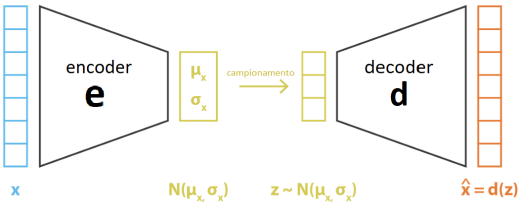


Figura 3.3: Schema di un VAE

3.3 Trattazione matematica di un VAE

Nella seguente sezione tratteremo gli aspetti matematici degli autoencoders variazionali. Come accennato precedentemente, la formulazione sarà probabilistica.

3.3.1 Assunzioni probabilistiche del modello

Si indichi con x la variabile che rappresenta i dati. Per poter generare nuovi dati, occorre conoscere la distribuzione $p(x)$. Assumiamo un modello generativo dei dati, dove la variabile x è generata a partire dalla variabile latente z, attraverso una funzione di verosimiglianza $p(x|z)$:

$$p(x) = \int p(x|z)p(z)dz \tag{3.1}$$

dove $p(z)$ è la distribuzione a priori (prior) nello spazio latente. Dato questo modello, è possibile associare ad ogni x una distribuzione $p(z|x)$ nello spazio latente. Ricordando il teorema di Bayes, infatti, è possibile mettere in relazione $p(z|x)$ con $p(z)$ e $p(x|z)$:

$$p(z|x) = \frac{p(x|z)p(z)}{\int p(x|z')p(z')dz'} \tag{3.2}$$

La distribuzione a posteriori $p(z|x)$ rappresenta un *encoder probabilistico*, mentre la verosimiglianza $p(x|z)$ rappresenta un *decoder probabilistico*.

La definizione del modello dovrebbe essere svolta con molta attenzione: ad esempio, sarebbe sempre meglio definire un prior robusto con delle larghe code per non limitare eccessivamente lo spazio dei parametri [3]. Nel nostro caso il prior sarà preso con lo scopo di semplificare il modello e rendere più facile il futuro processo di inferenza. Si ipotizza che $p(z)$ sia una Gaussiana standard e che $p(x|z)$ sia una distribuzione normale con media definita da una funzione f e una matrice di covarianza che sia l'identità a meno di una costante moltiplicativa c. Per il momento si assuma che f sia nota e appartenga a una generica famiglia di funzioni F. Dunque:

$$\begin{cases} p(z) \equiv \mathcal{N}(0, \mathbb{I}) \\ p(x|z) \equiv \mathcal{N}(f(z), c\mathbb{I}) \quad f \in F \quad c > 0 \end{cases} . \tag{3.3}$$

Nonostante la semplicità del modello, nel caso generale risulta comunque molto complicato trovare una forma analitica della $p(z|x)$ tramite la formula 3.2. Per questo motivo, bisogna usare delle tecniche per approssimare la distribuzione di probabilità $p(z|x)$, come ad esempio l'inferenza variazionale.

3.3.2 Inferenza variazionale

In questa sezione sarà descritta la tecnica dell'inferenza variazionale (VI), che viene usata in diverse applicazioni in Machine Learning per la sua velocità di convergenza a un risultato sensato rispetto ad altri metodi per l'approssimazione di distribuzioni di probabilità. L'idea alla base di questo metodo è definire una famiglia di distribuzioni parametrizzate da certe variabili, che approssimino la funzione da computare, e trovare l'elemento di questa famiglia che minimizzi la divergenza Kullback-Leibler D_{KL} tra la funzione di partenza e l'elemento stesso. Dove la D_{KL} può essere pensata, solo intuitivamente, come una funzione che quantifica la distanza tra le due distribuzioni (non può essere una distanza perché non è simmetrica).
Inoltre si introduce un'ipotesi spesso presa in considerazione, come ad esempio nell'articolo [5] di Kingma e Welling, in cui la distribuzione $p(z|x)$ viene approssimata da una famiglia di distribuzioni gaussiane $q_x(z)$, parametrizzate da due funzioni appartenenti alle famiglie G e H. Questa approssimazione della posterior è fatta per facilitare l'ottimizzazione della distribuzione di verosimiglianza. In particolare:

$$q_x(z) \equiv \mathcal{N}(g(x), h(x)) \quad g \in G \quad h \in H \tag{3.4}$$

L'approssimazione risulta esatta esclusivamente nel caso in cui $f(z) = z$, mentre nei i casi in cui $f(z)$ è lineare in z la distribuzione non è normale, ma si può ben approssimare con $q_x(z)$. Ora dobbiamo stimare le due funzioni che rendono minima la divergenza Kullback-Leibler. In particolare, sfruttando il teorema di Bayes:

$$\begin{aligned}
(g^*, h^*) &= \underset{(g,h)\in G\times H}{\operatorname{argmin}} D_{KL}(q_x(z)||p(z|x)) = \underset{(g,h)\in G\times H}{\operatorname{argmin}} \left(\mathbb{E}_{z\sim q_x}(\log q_x(z)) - \mathbb{E}_{z\sim q_x}\left(\log \frac{p(x|z)p(z)}{p(x)} \right) \right) \\
&= \underset{(g,h)\in G\times H}{\operatorname{argmin}} \left(\mathbb{E}_{z\sim q_x}(\log q_x(z)) - \mathbb{E}_{z\sim q_x}(\log p(z)) - \mathbb{E}_{z\sim q_x}(\log p(x|z)) + \mathbb{E}_{z\sim q_x}(\log p(x)) \right) \\
&\doteq \underset{(g,h)\in G\times H}{\operatorname{argmax}} \left(\mathbb{E}_{z\sim q_x}(\log p(x|z)) - D_{KL}(q_x(z)||p(z)) \right) \\
&= \underset{(g,h)\in G\times H}{\operatorname{argmax}} \left(\mathbb{E}_{z\sim q_x}\left(-\frac{||x - f(z)||^2}{2c} \right) - D_{KL}(q_x(z)||p(z)) \right)
\end{aligned} \tag{3.5}$$

Dove nel passaggio $\doteq$, si è raccolto un segno meno e si è scartato il termine $\mathbb{E}_{z\sim q_x}(\log p(x)))$ che, non dipendendo da z, risulta essere costante. Nell'ultima equazione si riesce a distinguere un primo termine che rappresenta l'errore di ricostruzione, che quantifica la confidenza che abbiamo nei nostri dati, e un secondo termine di regolarizzazione, che esprime la confidenza che abbiamo sul prior. Il bilanciamento

tra questi due termini è regolato da c, infatti l'aumento del valore di c accresce la varianza di $p(x|z)$, favorendo il termine di regolarizzazione[6].
Quando si sono assunte le ipotesi 3.3, si è considerata la funzione f come nota, ma questa è un parametro da stimare attraverso l'ottimizzazione del modello e risulta cruciale, le prestazioni dell'architettura dipendono fortemente da f. Perciò, la rete può essere ottimizzata tramite i parametri (g, h) e (c, f) con cui modellizzare le distribuzioni $p(z|x)$ e $p(x|z)$. Per trovare f, si cerca di massimizzare la probabilità di avere $\hat{x} = x$, dove $\hat{x}$ è campionato da $p(x|z)$ e z da ${q_x}^*(z)$, ottimizzando la distribuzione di verosimiglianza. Concretamente:

$$f^* = \underset{f \in F}{\operatorname{argmax}}(\mathbb{E}_{z \sim q_x^*}(\log p(x|z))) = \underset{f \in F}{\operatorname{argmax}}\left(\mathbb{E}_{z \sim q_x^*}\left(- \frac{||x - f(z)||^2}{2c}\right) - D_{KL}(q_x(z)||p(z))\right) \quad (3.6)$$

Considerando tutti i contributi, il problema di ottimizzazione può essere formalizzato nel seguente modo:

$$(f^*, g^*, h^*) = \underset{(f,g,h) \in F \times G \times H}{\operatorname{argmax}}\left(\mathbb{E}_{z \sim q_x}\left(- \frac{||x - f(z)||^2}{2c} - \right) - D_{KL}(q_x(z)||p(z))\right) \quad (3.7)$$

Anche in questo caso, si possono riconoscere i termini di ricostruzione e regolarizzazione, bilanciati dalla valore di c, che insieme costituiscono il limite minimo variazionale(VLB).

3.3.3 Ricostruzione dei dati e regolarizzazione dello spazio latente

Come già espresso, un VAE deve riuscire a ricostruire i dati in input con un errore $\epsilon(x, d(e(x)))$ ragionevole e, allo stesso tempo, avere un uno spazio latente abbastanza regolarizzato per poter generare dei dati sensati, rispetto alle caratteristiche di quelli iniziali. Queste due prerogative sono descritte dal termine di ricostruzione e di regolarizzazione dell'equazione 3.7. La massimizzazione del primo termine incoraggia la distribuzione $q_x(z)$ a essere un delta di Dirac $\delta_x(z)$, promuovendo le singolarità nello spazio latente tipiche degli autoencoder, che si vogliono evitare nei VAEs. Per quanto riguarda il secondo termine, la massimizzazione del VLB corrisponde a una minimizzazione del termine di regolarizzazione. Il decremento di questo spinge la distribuzione $q_x(z)$ verso il prior $p(z)$, prevenendo $q_x(z)$ dall'essere simile a $\delta_x(z)$, infatti $D_{KL}(\delta(z)||p(z))) \to \infty$. Questa condizione tende ad assicurare uno spazio latente regolarizzato, in cui poter generare dati sensati.
Quindi il processo di addestramento cerca di trovare le migliori distribuzioni $q_x(z)$ e $p(x|z)$, agendo su (f, g, h), per avere un bilanciamento tra questi due termini.

3.4 Uso delle reti neurali nel modello

Finora, si è costruito un modello probabilistico, per poter generare nuovi dati, che dipende dalle tre funzioni (f, g, h) e si è ricondotto, tramite l'inferenza variazionale, il problema di trovare il miglior schema di codifica/decodifica a un processo di ottimizzazione per ottenere (f^*, g^*, h^*), come specificato nell'equazione 3.7. Poiché questa ottimizzazione sull'intero spazio delle funzioni risulta molto complicata, si ipotizza che le tre funzioni (f, g, h) siano delle reti neurali, dove le famiglie F, G, H rappresentano l'insieme delle architetture che (f, g, h) possono assumere. Il processo di ottimizzazione verrà fatto sui parametri della rete e porterà a trovare l'insieme dei pesi e dei biases dell'architettura che definiscono (f^*, g^*, h^*).

3.4.1 Architettura della rete

A questo punto, si può costruire l'architettura della rete, per implementare l'autoencoder variazionale. Per quanto riguarda la parte dell'encoder, si ipotizza che le reti g e h non siano indipendenti, ma condividano alcuni dei primi strati tra cui quello iniziale, cosi da avere il maggior numero possibile di parametri nei primi strati e sfruttare al massimo l'encoder. In particolare:

$$g(x) = g_2(g_1(x)) \quad h(x) = h_2(h_1(x)) \quad g_2(x) = h_1(x) \quad (3.8)$$

Inoltre si assume anche che la matrice quadrata di covarianza della distribuzione normale $q_x(z)$, definita da $h(x)$, sia in forma diagonale. Quindi si definisce $h(x)$ tramite il vettore contenente gli autovalori della matrice di correlazione, che avrà la stessa dimensione k del vettore che descrive $g(x)$. Questa ipotesi rende il problema computazionalmente meno complicato, ma offre una stima meno accurata di $q_x(z)$, e quindi di $p(z|x)$. Si è quindi creata una rete che si avvale di un singolo encoder per eseguire l'inferenza della distribuzione a posteriori su tutti i punti del nostro insieme di dati. Mentre nei metodi di inferenza variazionale più tradizionali, i parametri variazionali (in questo caso i pesi e i biases della rete) non sono condivisi, ma vengono ottimizzati separatamente e iterativamente per ogni dato. La strategia sfruttata per condividere i parametri variazionali è chiamata inferenza variazionale ammortizzata [7]. Mentre nell'architettura non si sono poste ulteriori ipotesi sull'architettura del decoder. Nel complesso la rete neurale è schematizzata dalla seguente figura.

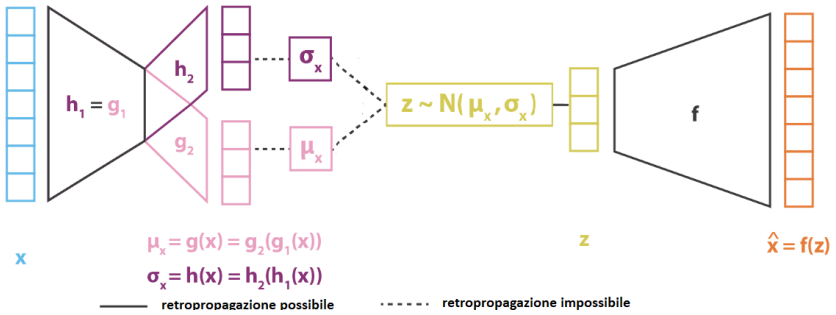


Figura 3.4: Raffigurazione intuitiva dell'architettura neurale del VAE

In questa architettura si è introdotto un metodo generativo in cui si campiona z da $q_x(z)$. Tuttavia, se esprimiamo il processo di campionamento in questo modo, risulta impossibile retropropagare l'errore attraverso la rete neurale. Bisogna esprimere l'operazione di campionamento di z, in modo da poter attuare l'algoritmo di backpropagation per addestrare la rete. Per ovviare a questa problematica si ricorre al trucco di riparametrizzazione.

3.4.2 Trucco di riparametrizzazione

L'essenza del trucco di riparametrizzazione, introdotto da Kingma e Welling in [8], è semplice. Data una variabile continua casuale $z \sim q_x(z)$, dove z è campionata da una certa distribuzione condizionata $q_x(z)$, risulta possibile esprimere la variabile casuale z come una variabile deterministica $z = t_\phi(\zeta, x)$, dove ζ è una variabile ausiliaria con una probabilità marginale $r(\zeta)$. In questo caso assumiamo che $r(\zeta)$ sia una distribuzione normale standard. Dunque:

$$z = h(x)\zeta + g(x) = \sigma_x\zeta + \mu_x \qquad \zeta \sim \mathcal{N}(0, \mathbb{I}) \tag{3.9}$$

Ora risulta possibile applicare l'algoritmo di backpropagation, in quanto la variabile z risulta essere deterministica
L'architettura finale dell'autoencoder variazionale, illustrata nella figura 3.4, è data dall'equazione 3.7.

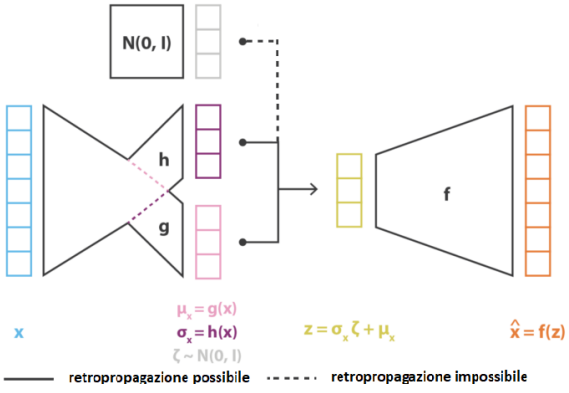


Figura 3.5: Raffigurazione dell'architettura neurale del VAE

Capitolo 4

Analisi databas MNIST tramite VAE

Nel seguente capitolo verrà affrontata l'analisi di dati provenienti dal database MNIST [1], tramite una rete neurale che descrive un autoencoder variazionale. Dapprima, la rete è stata addestrata per ottimizzare la ricostruzione dei dati iniziali, tramite encode e decode. Successivamente si sono campionati casualmente punti dello spazio latente, decomprimendoli per generare dati nuovi.

4.1 Dataset del MNIST e struttura della rete

Il database MNIST è costruito partendo dai due set di dati del database NIST chiamati *Special Dataset 1 (SD1)*, creato per l'addestramento della rete, e *Special Dataset 3 (SD3)*, creato per la fase di test. Questi gruppi di dati contengono delle immagini che raffigurano un numero da zero a nove, dove ogni figura è descritta da un insieme di 28×28 pixel, in cui ognuno di essi ha un'intensità specificata da un numero reale da 0 (pixel nero) a 1 (pixel bianco), in cui i valori intermedi descrivono diverse tonalità di grigio. I due set di dati del NIST combinano immagini di numeri disegnati da due gruppi di soggetti diversi, il che non risulta efficiente per degli esperimenti di machine learning. Infatti è richiesto che il risultato dell'esperimento non dipenda dalla scelta di quale gruppo usare come set di addestramento e di test tra i due disponibili. Per ovviare a questo problema, il set di addestramento del MNIST è composto da 30000 dati appartenenti al *SD1* e 30000 appartenenti al *SD3*, mentre quello di test è composto da 5000 dati contenuti nel *SD1* e 5000 dati contenuti nel *SD3*. Si può osservare un'immagine del set di addestramento del MNIST nella figura 4.1. La scelta di questo database è legata alla sua efficienza per esperimenti di machine learning. Infatti nel corso degli anni, si è arrivati a costruire degli esperimenti in cui l'error-rate associato al riconoscimento dei numeri risulta essere 0.18%, paragonabile con le performance del cervello umano [9].
La rete neurale che attua la riduzione della dimensionalità nello spazio latente prende in input un'immagine del dataset che viene decritta da un tensore di dimensioni 28×28, in cui ogni entrata è un numero reale compreso tra 0 e 1. Quindi, lo strato iniziale della rete avrà una dimensione pari a 784. Lo strato successivo è un layer intermedio di dimensione 400 che risulta utile per aumentare la non linearità della rete e per facilitare la ricerca dei minimi della funzione di costo, durante l'algoritmo di retropropagazione. Questo layer intermedio viene collegato con lo strato nascosto di dimensione 20. Quindi la rete attua una riduzione della dimensionalità che porta a un uno spazio latente di dimensione 20, partendo da uno spazio di dimensione 784. Per la parte di ricostruzione dei dati, la rete è composta da un layer intermedio di dimensione 400, successivo a quello dello spazio latente, e allo strato di uscita della stessa dimensione di quello di ingresso, ossia 784. Inoltre, la funzione di attivazione dei due layer intermedi è la funzione rettificatore (*relu*). Mentre, per lo strato di uscita la funzione di attivazione è la funzione sigmoidea (*sigmoid*), per avere un valore in output che possa descrivere un pixel (ossia compreso tra 0 e 1).

[1] il dataset è reperibile tramite il seguente link http://yann.lecun.com/exdb/mnist.

4.2 Ricostruzione dei dati iniziali

Dopo aver costruito la rete neurale descritta nella sezione precedente, si è addestrata la rete, usando il dataset di addestramento del MNIST, in modo da avere una ricostruzione dei dati iniziali, tramite il processo di encode e decode, più efficiente possibile. Per fare ciò, si è usato l'algoritmo di retropropagazione dove la funzione di costo è stata definita tenendo conto del termine di ricostruzione e di regolarizzazione come nella formula 3.7. In questo caso, si è deciso di sostituire lo scarto quadratico del termine di ricostruzione con la funzione di entropia binaria incrociata $H(x_i, \hat{x_i})$. Questa scelta è motivata dal fatto che si vuole avere una classificazione binaria di ogni numero che descrive un pixel, in cui il valore 0 corrisponde ad avere un pixel nero, mentre a 1 è associato un pixel bianco. Dati il valore dell'i-esimo pixel x_i dell'immagine iniziale x e il suo valore ricostruito $\hat{x}_i$, la funzione di costo è stata definita nel seguente modo:

$$\begin{aligned} loss_i &= H(x_i, \hat{x}_i) + D_{KL}(q_x(z)||p(z))) \\ &= -\frac{1}{N}\sum_{n=1}^{N}\left[x_i \cdot log(\hat{x}_i) + (1 - x_i) \cdot log(1 - \hat{x_i})\right] - \frac{1}{2} \cdot k \cdot \sum_{n=1}^{N}\left(1 + log(var_x(z) - \mu_x(z)^2 - var_x(z)\right) \end{aligned} \tag{4.1}$$

In cui $\mu(z_i)$ e $var(z_i)$ si riferiscono alla media e alla varianza della distribuzione di probabilità gaussiana $q_x(z)$, calcolata dalla rete nella fase di encode. Si osserva che agendo sulla costante arbitraria k si possono pesare in modo differente i due contributi di ricostruzione e regolarizzazione. Questa costante assume quindi lo stesso ruolo della costante c nell'equazione 3.7. Nel nostro caso, per avere sia una ricostruzione accurata dei dati iniziali sia una generazione di nuovi dati sensata, il valore di k risulta essere 25. Inoltre per la ricerca dei minimi si è usato il metodo di Adam. Come affermano D.Kingma e J.Ba nell'articolo [10] in cui è stato introdotto, questo è un'estensione del metodo di discesa stocastica del gradiente che richiede solo il computo di gradienti del primo ordine. Questo metodo risulta semplice da implementare, è computazionalmente efficiente, ha pochi requisti di memoria e, soprattutto, è adatto per problemi di grandi dimensioni in termini di dati e/o parametri.
Durante l'addestramento è necessario eseguire più cicli di elaborazione del set di addestramento per il computo dei parametri della rete. Si definisce numero di epoche la quantità di cicli in cui si allena la rete con tutti i dati del set di addestramento. Inoltre, l'insieme di dati potrebbe essere troppo grande per essere elaborato tutto in una volta, come ad esempio nel nostro caso in cui abbiamo 60000 immagini. Quindi si può suddividere il set di addestramento in sottogruppi uniformi, chiamati batch. La rete aggiorna i suoi parametri solo dopo l'addestramento di un intero batch. Il numero di dati contenuti in ogni batch è chiamato dimensione del batch. Nel caso specifico si è usato un numero di epoche pari a 15 e una dimensione del batch uguale a 100.
Dopo aver addestrato la rete, sono stati scelti casualmente alcuni dati del set di addestramento e si sono ricostruiti, tramite encode e decode, per osservare l'efficacia del VAE. Nelle seguenti immagini sono riportate le immagini di un dato del set di addestramento e la sua ricostruzione, prima e dopo l'addestramento della rete.

Figura 4.1: Immagine di un dato iniziale del set di dati di addestramento

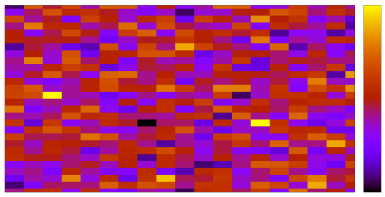


Figura 4.2: Immagine del dato iniziale ricostruito prima dell'addestramento

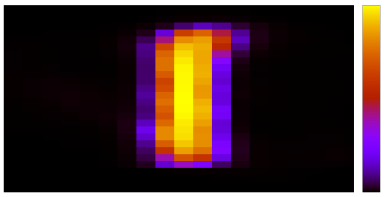


Figura 4.3: Immagine del dato iniziale ricostruito dopo l'addestramento

Prima che la rete sia addestrata i pesi e i biases della rete sono casuali: conseguentemente, la ricostruzione del dato risulta completamente scorretta. Invece, dopo aver ottimizzato i parametri dell'ar-

chitettura, tramite l'addestramento, l'immagine del dato ricostruito risulta compatibile con quella del dato iniziale.

4.3 Generazione di dati nuovi

Dopo aver addestrato la rete e verificato la sua efficacia nel ricostruire i dati iniziali, si sono generati dati nuovi, campionando casualmente dei punti dallo spazio latente e decomprimendoli. In questo processo, come già descritto nel capitolo 3, è fondamentale avere un'architettura variazionale dell'autoencoder e un termine di regolarizzazione nella funzione di costo. Infatti, come si può evincere dalle due figure seguenti, se non si considera il termine di regolarizzazione e quindi si implementa un semplice autoencoder, si osserva che l'immagine generata non ha alcun senso. Mentre tenendo conto del termine di regolarizzazione e pesandolo opportunamente tramite la costante arbitraria k, si riesce a generare un dato nuovo sensato. Nelle seguenti figure sono rappresentate le due immagini dei nuovi dati generati, nel caso in cui sia presente o meno il termine di regolarizzazione.

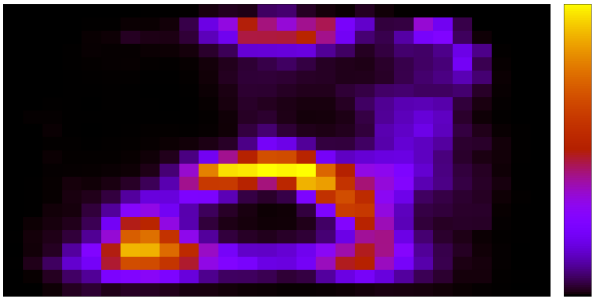


Figura 4.4: Immagine di un dato nuovo generato senza il termine regolarizzazione

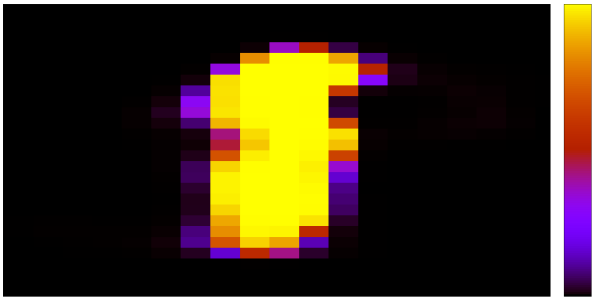


Figura 4.5: Immagine di un dato nuovo generato con il termine regolarizzazione

Si specifica che per una migliore visualizzazione dei risultati è stata cambiata la scala cromatica delle immagini. Infatti al valore 0 corrisponde un pixel nero, mentre al valore 1 è associato un pixel giallo. Dall'immagine 4.5 si evince che la rete è in grado di generare nuove immagini di numeri coerenti con le immagini del set di dati iniziale.

Capitolo 5

Analisi dell'attività cerebrale

In questo capitolo verrà trattata l'analisi di dati che descrivono l'attività cerebrale. Come osservato da Van Essen et. al nell'articolo [11], il cervello risulta essere un sistema altamente integrato, in cui le diverse componenti non agiscono in maniera indipendente tra loro, ma collaborano influenzandosi le une con le altre. Questo porta a pensare che i dati che descrivono l'attività cerebrale di aree distinte del cervello siano correlati e possano essere descritti, tramite una riduzione dimensionale, da un numero minore di caratteristiche. In particolare, si userà un VAE per attuare un riduzione della dimensionalità dei dati, variando la dimensione dello spazio latente. Un simile approccio è già stato usato da Sanz Perl et al. [12].

5.1 Dataset dell'attività cerebrale

È stata analizzata la dinamica cerebrale macroscopica a riposo di un gruppo di individui sani e di un gruppo pazienti affetti da ictus cerebrale, registrata tramite risonanza magnetica funzionale (fMRI) [13, 14]. Ciascuna registrazione è composta da serie temporali di lunghezza variabile, provenienti da 90 diverse regioni cerebrali. In ciascuna serie temporale, il valore corrisponde all'intensità del segnale BOLD (blood-oxygen-level dependent), campionato con frequenza 0.5 Hz. Le serie temporali all'interno di ciascun gruppo sono state concatenate, formando così 2 dataset *'estesi'*.

5.2 Riduzione della dimensionalità dell'attività cerebrale

Per analizzare i dati si è usata la stessa architettura adoperata nel capitolo 4, in cui si hanno quindi cinque strati. Lo strato di ingresso e di uscita hanno una dimensione pari a 90, infatti la rete riceve in input vettori appartenenti a $\mathbb{R}^{90}$. I due layers intermedi sono composti da 40 nodi, mentre lo spazio nascosto ha una dimensione pari a 20. Rispetto alla rete del capitolo precedente, oltre alle dimensioni degli strati, vi sono altre due differenze. Infatti la funzione di attivazione dei due layer intermedi e dello strato d'uscita è la tangente iperbolica (*tanh*), invece della funzione rettificatore (*relu*). Inoltre il termine di ricostruzione della funzione di costo è descritto dallo scarto quadratico tra il dato iniziale il suo dato in uscita, come già visto nella formula 2.2.

5.2.1 Ricostruzione dei dati iniziali

In questa sezione la rete è stata addestrata tenendo conto esclusivamente del termine di ricostruzione, andando a porre il parametro k uguale a zero (4.1), per cercare di ricostruire nel miglior modo possibile i dati in ingresso. L'addestramento è stato effettuato sia per il campione di dati proveniente da soggetti sani sia per quello proveniente da pazienti che hanno avuto un ictus cerebrale.
In primo luogo si è verificato che la rete riesca a ricostruire i vettore iniziali dati in input alla rete. In Fig. 5.1 e 5.2 sono rappresentati due grafici che confrontano le entrate di un vettore iniziale e le entrate del suo relativo vettore ricostruito dalla rete, per un vettore appartenente al campione dei pazienti sani e uno relativo ai soggetti malati.

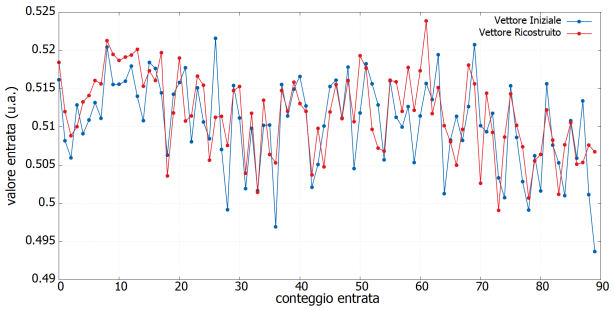


Figura 5.1: Confronto di un vettore iniziale e della sua ricostruzione per pazienti sani

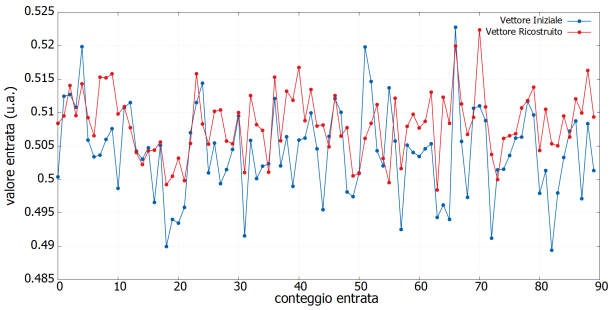


Figura 5.2: Confronto di un vettore iniziale e della sua ricostruzione per pazienti malati

Inoltre, per osservare la qualità della ricostruzione della rete vengono riportati in seguito due grafici di dispersione (in ascissa sono riportate le entrate del vettore iniziale, mentre in ordinata le entrate del vettore ricostruito) che mostrano la qualità della ricostruzione per due vettori, uno scelto casualmente dall'insieme dei pazienti sani e uno dai soggetti malati.

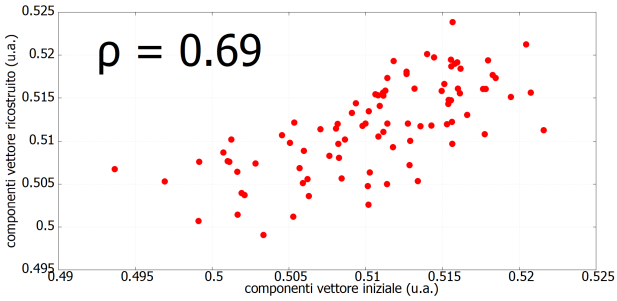


Figura 5.3: Grafico di dispersione per un vettore dall'insieme di dati campionati tra pazienti sani

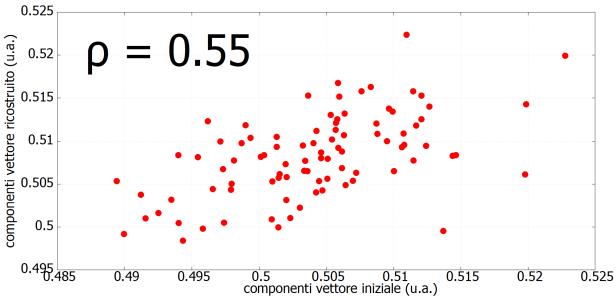


Figura 5.4: Grafico di dispersione per un vettore dall'insieme di dati campionati tra pazienti malati

Dai grafici e dal coefficiente di correlazione di Pearson ρ, riportato nelle Fig. 5.3 e 5.4, si osserva che le due ricostruzioni, nonostante vi sia una perdita di informazioni ineliminabile, risulta soddisfacente. Si specifica che per questa analisi si è usata una dimensione dello spazio latente pari a 20, una dimensione del batch uguale a 20 e un numero di epoche pari a 20. Dove la dimensione del batch e il numero di epoche sono stati decisi cercando, allo stesso tempo, di addestrare la rete con il maggior numero di dati possibili e rendere il problema computazionalmente trattabile. Si deve anche notare che la dimensione dello spazio latente gioca un ruolo cruciale nella compressione dei dati, come si vedrà nel prossimo sottoparagrafo 5.2.2. In questa analisi la dimensione dello strato nascosto è stata scelta cercando un compromesso tra la migliore ricostruzione possibile (minore perdita di informazione) e il minimo numero di caratteristiche per descrivere i dati.
Inoltre le due figure 5.3 e 5.4 mostrano come la rete riesca a ricostruire con maggiore successo i dati dei pazienti sani. Questo potrebbe essere dovuto al fatto che per tutti i pazienti sani sono presenti una serie di pattern comuni che rendono più facile l'individuazione di una numero ridotto di caratteristiche comuni. Mentre, poiché ogni ictus cerebrale altera il cervello dei pazienti in modo differente, i dati di questi soggetti saranno propensi ad avere dei pattern comuni che rendono più difficile la costruzione di caratteristiche comuni per la descrizione dei dati.

5.2.2 Dipendenza della ricostruzione dalla dimensione dello spazio latente

Come già anticipato, la dimensione dello spazio latente è essenziale nell'implementazione di un VAE. Infatti, come mostreremo in questo paragrafo, c'è una dipendenza tra il numero di caratteristiche usate per descrivere il set di dati e la qualità della ricostruzione. Quest'ultima può essere quantificata,

in prima approssimazione, guardando la funzione di costo dei dati usati per l'addestramento. In particolare, si è addestrata la rete per 20 epoche, per entrambi i campioni di dati, e si calcolata la media della funzione di costo di ogni dato iniziale e della sua ricostruzione sull'ultima epoca. Questa analisi è stata fatta per diversi valori della dimensione dello spazio latente. Nelle seguenti figure si riportano, per entrambi i set di dati, la media della funzione di costo in funzione della dimensione dello spazio latente.

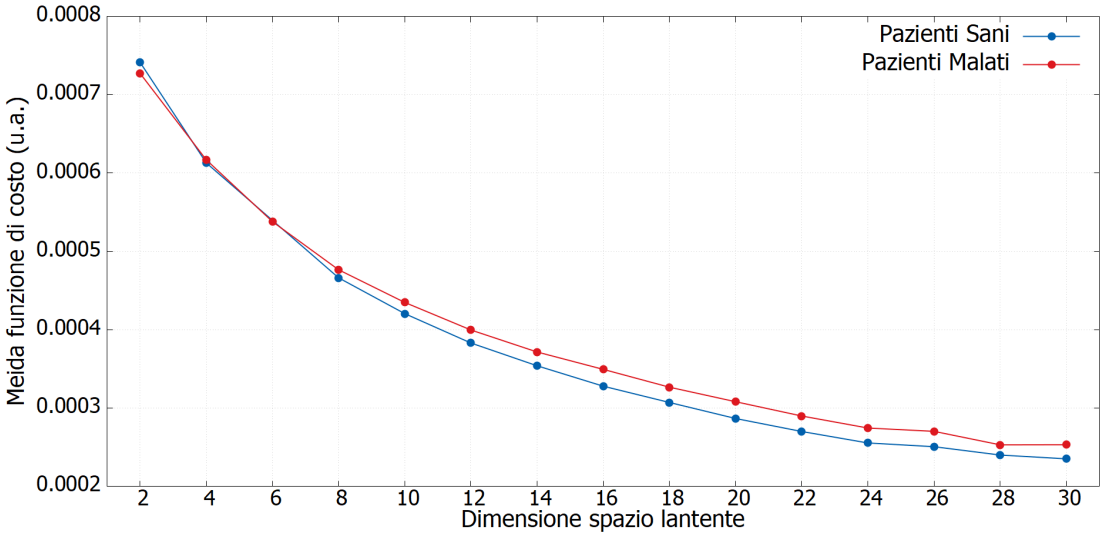


Figura 5.5: Confronto tra di due grafici della funzione di costo in funzione della dimensione dello spazio latente

Come ci si aspetta, le funzioni risultano monotone decrescenti, infatti un numero maggiore della dimensione dello spazio assicura un numero maggiore di caratteristiche per descrivere i dati, e quindi una minore perdita di informazione. Le curve non sembrano convergere a un asintoto per le dimensioni dello spazio latente prese in considerazione. Questo porta a pensare che le dimensioni dello spazio latente prese in considerazione siano troppo ridotte per avere una ricostruzione, successiva a una compressione, dei dati che restituisca dati simili a quelli iniziali. Inoltre eccetto che per il primo punto, come già riscontrato nella sezione 5.2.1, la funzione di costo dei dati provenienti da pazienti sani risulta essere sistematicamente minore rispetto quella dei pazienti che hanno avuto un ictus, a prescindere dalla dimensione dello spazio latente.

Bibliografia

[1] Laurens Van Der Maaten, Eric Postma e Jaap Van den Herik. ≪Dimensionality reduction: a comparative review≫. In: *J Mach Learn Res* 10 (2009), pp. 66–71.

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

[3] Ethem Alpaydin. *Introduction to Machine Learning*. 3ª ed. Cambridge, MA: MIT Press, 2014.

[4] Carl Doersch. *Tutorial on Variational Autoencoders*. 2016.

[5] Diederik P. Kingma e Max Welling. ≪An Introduction to Variational Autoencoders≫. In: *Foundations and Trends® in Machine Learning* 12.4 (), pp. 307–392.

[6] Joseph Rocca. *Understanding Variational Autoencoders (VAEs)*. 2019.

[7] Samuel Gershman e Noah Goodman. ≪Amortized inference in probabilistic reasoning≫. In: *Proceedings of the annual meeting of the cognitive science society*. Vol. 36. 36. 2014.

[8] Diederik P Kingma e Max Welling. *Auto-Encoding Variational Bayes*. 2013.

[9] Kamran Kowsari et al. ≪RMDL: Random Multimodel Deep Learning for Classification≫. In: *CoRR* abs/1805.01890 (2018).

[10] Diederik P Kingma e Jimmy Ba. ≪Adam: A method for stochastic optimization≫. In: *arXiv preprint arXiv:1412.6980* (2014).

[11] David C Van Essen, Charles H Anderson e Daniel J Felleman. ≪Information processing in the primate visual system: an integrated systems perspective≫. In: *Science* 255.5043 (1992), pp. 419–423.

[12] Yonatan Sanz Perl et al. ≪Generative embeddings of brain collective dynamics using variational autoencoders≫. In: *Physical Review Letters* 125.23 (2020), p. 238101.

[13] Joshua Sarfaty Siegel et al. ≪Disruptions of network connectivity predict impairment in multiple behavioral domains after stroke≫. In: *Proceedings of the National Academy of Sciences* 113.30 (2016), E4367–E4376.

[14] Chiara Favaretto et al. ≪Subcortical-cortical dynamical states of the human brain and their breakdown in stroke≫. In: *Nature Communications* 13.1 (2022), pp. 1–17.