Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea in Fisica

APPLICAZIONI DEL MACHINE LEARNING ALLA FISICA DELLE ALTE ENERGIE

Relatore:
Prof. Alberto Cervelli

Co-relatore:
Dr. Roberto Morelli

Presentata da:
Schiazza Filippo Antonio

Anno accademico 2019/2020

Sommario

Negli ultimi anni sono stati sviluppati sistemi sempre più complessi di analisi di grandi quantità di dati: nel campo della fisica delle alte energie, il grande numero di eventi prodotti in collisionatori rendono questi sistemi molto utili per la ricerca di eventi rari. In questa tesi da prima verrà descritta l'evoluzione degli algoritmi utilizzati per l'analisi multivariata di campioni molto estesi di dati; ci si focalizzerà in particolare su sistemi di machine learning, con particolare attenzione sui Variational Autoencoders e la loro applicazione nel campo della fisica delle alte energie. Verranno quindi presentati i risultati dell'applicazione di un Variational Autoencoder per la ricerca di fisica oltre il Modello Standard (BSM). Verrà descritto il processo di addestramento di tale algoritmo, effettuato su campioni di eventi simulati secondo le predizioni del Modello Standard, e verrà valutata la sensibilità a processi di produzione elettrodebole di particelle supersimmetriche ($pp \to \tilde{\chi}_1^{\pm} + \tilde{\chi}_2^0, \tilde{\chi}_1^{\pm} \to W + \tilde{\chi}_1^0, \tilde{\chi}_2^0 \to h + \tilde{\chi}_1^0$), dove dalla collisione di due protoni si ottengono un chargino $\tilde{\chi}_1^{\pm}$ ed un neutralino pesante $\tilde{\chi}_2^0$; in seguito il chargino decade in un bosone W ed un neutralino leggero $\tilde{\chi}_1^0$,mentre il neutralino pesante decade in un Bosone di Higgs h ed in un altro neutralino leggero. La ricerca è volta all'osservazione dell'elettrone e del muone prodotti dal decadimento del W ed alla coppia di b-quarks, prodotti dal decadimento dell' h.
Con l'applicazione dell'algoritmo descritto è stato ottenuto il limite di 800 GeV per la massa del chargino per la sensibilità al segnale BSM, nell'ipotesi di massa nulla del neutralino leggero.

Indice

1 Introduzione **1**

2 Analisi multivariata e Machine Learning **2**

2.1 Sistema di tagli . . . 3

2.2 Processi multivariati e analisi discriminante lineare . . . 4

2.3 Machine Learning . . . 5

3 Machine Learning : metodi e caratteristiche **7**

3.1 Apprendimento supervisionato . . . 7

3.2 Discesa del gradiente . . . 9

3.3 Apprendimento non supervisionato . . . 10

3.4 Metodo di clustering basato sulla distanza euclidea . . . 11

3.5 Iperparametri e Grid Search . . . 13

3.6 Reti Neurali . . . 15

3.7 Alberi Decisionali . . . 20

3.8 Curse of dimensionality e riduzione della dimensionalità . . . 23

3.9 Autoencoders . . . 26

3.10 Variational Autoencoders (VAEs) . . . 28

3.10.1 Formulazione matematica dei VAEs . . . 30

4 Ricerca di fisica Behind Standard Model con i VAEs **34**

4.1 Dataset . . . 35

4.2 Architettura del modello . . . 36

4.3 Addestramento del VAE . . . 36

4.4 Risultati . . . 38

4.4.1 Processo di rigenerazione degli eventi . . . 38

4.4.2 Distribuzione della loss di ricostruzione . . . 39

4.4.3 Esperimento di conteggio e regione di esclusione . . . 41

4.4.4 Regione di esclusione ottimizzata . . . 45

4.4.5 Effetti della variazione dei pesi sulle variabili fisiche nel processo di apprendimento 46

5 Conclusioni **48**

Riferimenti bibliografici **49**

1 Introduzione

Il modello Standard (SM) e' una teoria che ha avuto grande successo, ed e' riuscita a spiegare e prevedere molti dei fenomeni osservati a livello subnucleare. D'altra parte, sebbene la scoperta del *Bosone di Higgs* [5] abbia confermato la rottura di simmetria elettrodebole, ha portato in primo piano lo *hierarchy problem* [15, 13, 9, 14], cioè la grande differenza tra l'accoppiamento elettrodebole e quello gravitazionale. Inoltre, nonostante l'evidenza di materia oscura nell'universo, lo SM non prevede alcuna particella che possa giustificare la presenza della materia oscura osservata, ad esempio negli aloni galattici.
Da queste considerazioni sembrerebbe essere ormai arrivati ad un punto di stallo per quanto riguarda il MS, tuttavia è evidente da ciò che è stato accennato precedentemente che rimangono aperte molte domande; una ipotesi è che il MS rappresenti il limite a basse energie di una teoria più complessa, quindi una serie di fenomeni o non avvengono alle attuali energie raggiungibili al *Large Hadron Collider* oppure sono estremamente rari.
Per esempio, la teoria della *Supersimmetria* (SUSY), che è una estensione del MS, risolve il Problema della Gerarchia introducendo un nuovo fermione/bosone per ogni fermione/bosone del MS; inoltre tali particelle sarebbero stabili e poco interagenti e quindi costituirebbero delle ottime candidate per la spiegazione della materia oscura.
Per ottenere una buona reiezione del fondo accompagnata da una buona efficienza sul segnale si possono utilizzare sistemi di analisi statistica molto complessi. Il machine learning (ML) rappresenta una serie di metodologie di natura statistica-computazionale che permettono di estrarre informazioni da enormi moli di dati senza la supervisione dell'analista. In fisica delle alte energie gli algoritmi di ML, attraverso l'apprendimento delle correlazioni tra le proprietà cinematiche delle particelle presenti in un evento, consentono di catalogare ciascun evento come affine al segnale o al fondo. In particolare i Variational Autoencoders (VAEs) [12], una cui applicazione e' descritta nel capitolo 4, si basano sulla riduzione della dimensione delle variabili che descrivono gli eventi, seguita da una fase di ricostruzione del campione. Nello specifico il VAE comprime ogni singolo evento di input che gli viene presentato non come un punto in uno spazio di dimensione minore (detto spazio latente), bensì come una distribuzione; si campiona quindi un punto nello spazio latente a partire da tale distribuzione, che viene ricostruito a seguito di un processo di decompressione.
Una volta addestrato il VAE a riconoscere e riprodurre le distribuzioni delle variabili cinematiche che descrivono gli eventi SM e' possibile, confrontando questi risultati con i dati ottenuti dall'esperimento, osservare (o valutare il limite di esclusione) anomalie che possono essere dovute alla presenza di eventi non descritti dal Modello Standard.
Questa tesi è strutturata in tre capitoli: nel primo vengono spiegate le differenze fra gli algoritmi di machine learning ed i metodi di analisi multivariata e non, nel secondo vengono approfondite le metodologie di machine learning e si presentano i Variational Autoencoders, mentre nell'ultimo viene applicato quest'ultimo metodo nel campo della fisica delle alte energie.

per ottenere una buona reiezione...

2 Analisi multivariata e Machine Learning

Le ricerche di eventi rari nella fisica delle alte energie hanno come scopo quello di riuscire a selezionare un campione il più puro possibile di eventi di segnale. Il parametro di merito della selezione e' la sensibilità al processo a cui si è interessati, che e' funzione del rapporto tra il numero di eventi di segnale attesi ed il numero di eventi di background:

$$\sigma = N_s/\sqrt{N_s + N_b} \tag{1}$$

In generale un evento può essere pensato come una collezione di dati e quindi lo si può rappresentare come un vettore in uno spazio n-dimensionale:

$$\vec{x} = (x_1, ..., x_n) \tag{2}$$

dove x_i sono le informazioni ottenute dalle particelle rivelate dal detector: ad esempio l'impulso di ciascuna particella, le masse invarianti ottenute sommando i quadrivettori di due o più particelle, il numero di hit in un detector, o l'energia trasversa mancante dovuta a particelle non rivelate.
Una volta individuate le quantità che si vogliono utilizzare per separare il campione di segnale dal fondo, si possono utilizzare varie tecniche per ottimizzare la selezione, e quindi ottenere la migliore sensibilità al processo di interesse. Si possono individuare tre classi metodologiche:

1. Sistema di tagli sulle variabili (*cut and count*). Con questo metodo si selezionano sottoinsiemi dei valori di ciascuna variabile, in modo indipendente fra loro (Cut) per poi fare un esperimento di conteggio sulle regioni selezionate;

2. Analisi multi-variata come, ad esempio, l'analisi discriminante lineare;

3. Machine Learning, cioè sistemi che permettono l'apprendimento automatico e quindi l'algoritmo è in grado di imparare in maniera autonoma direttamente dai dati che gli vengono forniti.

Bisogna porre un confine arbitrario fra analisi multivariata e ML, per la trattazione fatta è stato deciso

2.1 Sistema di tagli

Con questo sistema si applicano delle selezioni sulle varie componenti x_i che definiscono un evento, in modo da ricavare un ipercubo nello spazio n-dimensionale degli eventi stessi. Per capire meglio questa metodologia si consideri un caso semplificato nel quale lo spazio in questione è bi-dimensionale e quindi i vettori di input sono del tipo $\vec{x} = (x_1, x_2)$; per raggiungere l'obiettivo di separazione bisognerà dunque applicare due tagli, uno sulla variabile x_1 ed uno su x_2, come riportato in figura 1.

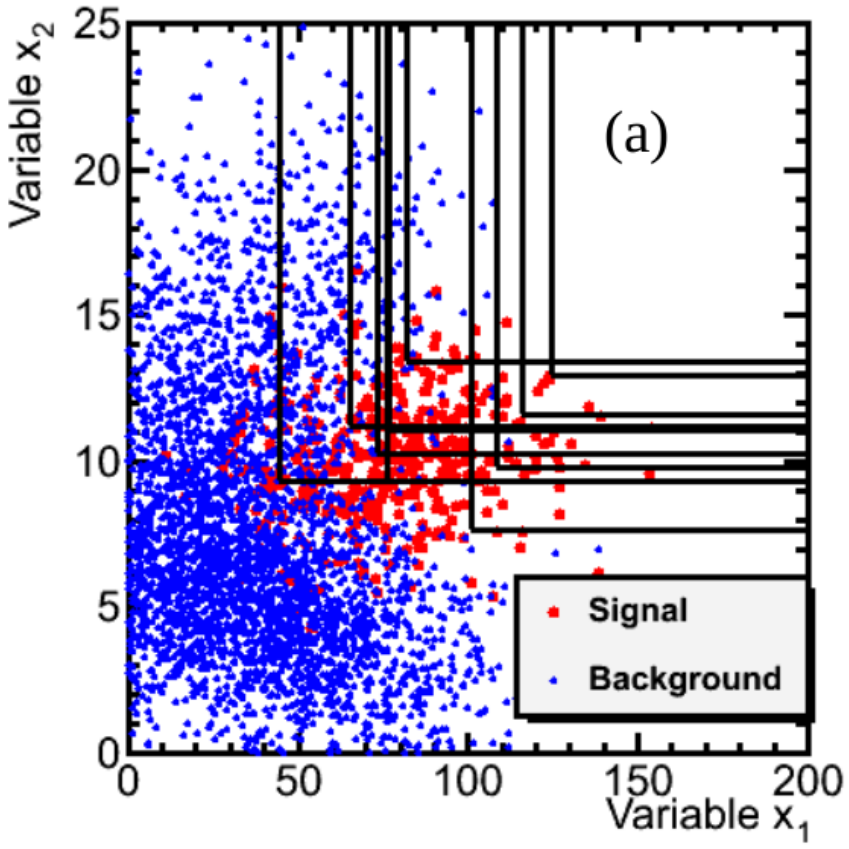

Figura 1: risultato grafico di un processo di taglio sulle due variabili x_1 e x_2 per la separazione del segnale dal background [2].

La scelta della regione in cui fare il conteggio e' ottenuta grazie ad una ottimizzazione del rapporto segnale rumore al variare dei tagli. Questa ottimizzazione può essere ottenuta con molti metodi, di cui un esempio e' il Random Grid Search (RGS) che verrà presentato nella sezione 3.5.
In questo caso le selezioni sulle variabili x_i sono fatte in modo indipendente, di conseguenza non si tiene conto di eventuali correlazioni tra le variabili e, se le distribuzioni di tali variabili per segnale e fondo sono molto simili, questo sistema può non essere molto efficiente.

2.2 Processi multivariati e analisi discriminante lineare

Nella sezione precedente è stato messo in evidenza il fatto che i sistemi di tagli non permettono di tener conto di eventuali correlazioni fra i dati.
Per poter tener conto delle correlazioni tra le variabili x_i in esame, e poter sfruttare al meglio le variabili poco discriminanti e' possibile utilizzare delle selezioni ottenute grazie a selezioni su delle funzioni di un sottoinsieme delle variabili prese in esame.
Ogni volta che si considera una selezione fatta su una funzione di più di una variabile è possibile parlare di analisi multivariata e, per la ragione appena presentata, è necessario considerare i processi multivariati.
Il metodo di analisi multivariata più semplice e' quello del discriminante lineare, o discriminante di Fisher: si immagini di avere a disposizione un determinato set di eventi in input $\vec{x}_i$, ciascuno caratterizzato da un numero n di variabili (spazio n-dimensionale) e di volerli ripartire fra segnale e background.
Si definisce la funzione discriminante lineare nel seguente modo:

$$D(x_1, x_2, ..., x_n) = c_0 + c_1 x_1 + ... + c_n x_n = c_0 + \sum_{i=0}^{n} c_i x_i \tag{3}$$

cioè una combinazione lineare delle componenti del vettore che rappresenta l'evento; il valore assunto dalla funzione per ogni singolo evento ne permette la separazione nelle due classi (nel presente caso segnale e background), utilizzando un valore di riferimento D_0.
A questo punto l'obiettivo è quello di massimizzare la distanza fra le due classi, ossia rendere massima la differenza dei valori assunti dalla funzione $D(\mathbf{x})$ fra gli eventi appartenenti al background e quelli relativi al segnale.
Per ottenere una ottimizzazione del valore di selezione uno dei metodi più comuni è quello proposto da Fisher: si consideri un campione di eventi appartenenti al segnale e se ne definisca la media $\boldsymbol{\mu}_\mathbf{s}$ e la deviazione standard σ_s ed un campione appartenente al background, definendo anche qui la media $\boldsymbol{\mu}_\mathbf{b}$ e la deviazione standard σ_b. A questo punto la migliore configurazione dei parametri è quella che massimizza la seguente funzione:

$$F(\mathbf{c}) = \frac{(\boldsymbol{\mu}_\mathbf{s} - \boldsymbol{\mu}_\mathbf{b})^2}{\sigma_s^2 + \sigma_b^2} \tag{4}$$

Il discriminante lineare e' il sistema multivariato con la forma funzionale più semplice. Si possono costruire discriminanti sempre più complessi nella forma funzionale.

likelihood analysis ??
commenti vuoti

2.3 Machine Learning

Perché è utile il ML per l'obiettivo che è stato prefissato? Bisogna considerare il fatto che i sistemi multivariati descritti in sezione 2.2 dipendono fortemente dalla scelta dell'analista della funzione, o del metodo, da utilizzare per applicare la selezione del proprio campione. Il Machine Learning invece sfrutta algoritmi in grado di apprendere in maniera semi-autonoma la struttura dei campioni di background e segnale, ed è quindi in grado di stabilire quale sia il modo migliore per separare tali campioni.
L'approccio classico all'analisi dei dati prevede la disponibilità di un modello matematico, che dipende da una serie di parametri incogniti. Questi parametri vengono ricavati a partire dai dati sperimentali attraverso processi che possono essere sia analitici che numerici. A differenza dei sistemi descritti fin ora, i sistemi di selezione autonoma non necessitano di un modello fisico-matematico su cui basare la propria selezione.
Bisogna distinguere tre macro-tipologie di approccio all'analisi dati nel machine learning:

- APPRENDIMENTO SUPERVISIONATO
  In questa tipologia di apprendimento vengono presentati all'algoritmo degli input di esempio ed i relativi output desiderati, con lo scopo di apprendere una relazione generale che lega gli uni con gli altri; quindi per prima cosa si utilizza un campione di addestramento (*training data set*), in cui l'algoritmo ottimizza la selezione per legare gli input agli output forniti in fase di addestramento. Una volta addestrato, l'algoritmo viene validato utilizzando un campione di test (*test data set*) dove non vengono forniti gli output e se ne valuta l'efficienza di selezione;

- APPRENDIMENTO NON SUPERVISIONATO
  A differenza del caso supervisionato, nel training data set non sono presenti gli output attesi, quindi l'algoritmo deve essere in grado di apprendere autonomamente sia la struttura degli output desiderati, sia la miglior selezione per dividere i due campioni. Nel capitolo 4 verrà presentato un esempio di algoritmo non supervisionato per applicazioni nel campo della fisica delle particelle.

- APPRENDIMENTO PER RINFORZO
  Il *Reinforcement Learning* è basato sul concetto di ricompensa, cioè si permette all'algoritmo di esplorare un così detto ambiente e, in base all'azione compiuta, gli si fornisce un feedback positivo, negativo o indifferente. Un esempio classico prevede di voler addestrare un algoritmo per un particolare gioco: si farà in modo di fargli compiere una serie di partite in maniera iterativa e gli si assegnerà una ricompensa in caso di vittoria o una penalità in caso di sconfitta.

Oltre al modo in cui gli algoritmi ottimizzano la selezione a partire dai loro campioni di addestramento, si deve distinguere anche il modo in cui vengono presentati i dati in uscita. In quest'ottica si possono individuare tre differenti tipologie di algoritmi:

- CLASSIFICAZIONE
  Gli algoritmi di classificazione sono caratterizzati da un output discreto, cioè una serie di classi alle quali l'input può appartenere. Di solito questo metodo e' utilizzato da sistemi con apprendimento supervisionato. Un esempio di algoritmo di classificazione è quello che permette di distinguere se un particolare oggetto è presente o meno in un'immagine;

- REGRESSIONE
  La regressione è simile alla classificazione con la differenza che, in questo caso, l'output è continuo. Anche gli algoritmi di regressione sono adatti ad essere trattati con metodologie di apprendimento supervisionato;

- CLUSTERING
  Nel clustering l'obiettivo è sempre quello di dividere gli input in delle classi, tuttavia in questo caso tali classi non sono stabilite a priori. La natura di algoritmi di questo tipo li rende adatti ad essere trattati tramite metodi di apprendimento non supervisionato, proprio perché nel training data set gli eventi di input non sono etichettati (non è noto il relativo output) e quindi si richiede all'algoritmo di ricavare autonomamente le classi.

3 Machine Learning : metodi e caratteristiche

Lo schema logico che verrà seguito in questo capitolo prevede di approfondire inizialmente i due approcci principali al ML, ovvero l'apprendimento supervisionato(3.1) e non supervisionato (3.3), per poi presentare due metodi di apprendimento supervisionato, ovvero le *Reti Neurali* (3.6) e gli *Alberi Decisionali* (3.7) ed un metodo di apprendimento non supervisionato, il *Variational Autoencoders* (3.10). Nel fare ciò verranno presentati due importanti concetti del ML, come quello di *Iperparametro* (3.5) ed il *Curse of dimensionality* (3.8).

3.1 Apprendimento supervisionato

In questa sezione viene portata avanti una descrizione più approfondita e formale dell'apprendimento supervisionato.
Come già accennato precedentemente, quando si parla di apprendimento supervisionato si hanno a disposizione sia gli input $\mathbf{x}$ che i corrispettivi target di output $\mathbf{y}$; esisterà quindi una funzione $\mathbf{y} = \mathrm{f}(\mathbf{x})$ che mette in relazione gli input con gli output. Tuttavia, come detto, tale funzione è incognita ed è quindi ciò che viene ricercato con l'algoritmo di apprendimento. Nella pratica si cerca di approssimare la funzione agendo su una serie di parametri $\boldsymbol{\theta}$, quindi si avrà un qualcosa del tipo: $\hat{\mathbf{y}} = f(\mathbf{x}, \boldsymbol{\theta})$.

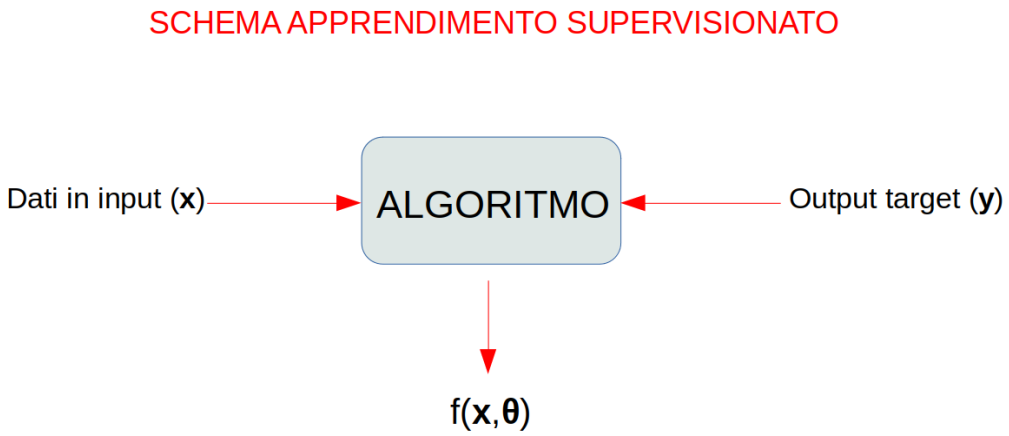

Figura 2: si riporta uno schema intuitivo del funzionamento di un algoritmo di apprendimento supervisionato

Per ogni vettore $\mathbf{x}$ del training data set è possibile definire una particolare funzione detta *Loss Function* $\mathrm{L}(\mathbf{y},\mathrm{f}(\mathbf{x},\boldsymbol{\theta}))$; a questo punto è possibile fare una media di tale funzione sull'intero set di dati a disposizione, ottenendo la funzione di rischio:

$$R(\boldsymbol{\theta}) = \frac{1}{N}\sum_{k=1}^{N} L(\mathbf{y}, f(\mathbf{x}, \boldsymbol{\theta})) \tag{5}$$

dove N è il numero di eventi del training data set.
Un esempio di funzione di rischio molto diffusa è l'errore quadratico medio:

$$R(\boldsymbol{\theta}) = \frac{1}{N}\sum_{k=1}^{N}(\mathbf{y}_k - f(\mathbf{x}_k, \boldsymbol{\theta}))^2 \tag{6}$$

Quando si addestra un modello si vuole inoltre evitare il così detto *overfitting*, che consiste in un eccessivo adattamento del modello ai dati di training e che di conseguenza porta al non raggiungimento di una sufficiente generalità. Un modo per verificare un eventuale overfitting è quello di verificare se il modello è nettamente migliore per il data set di allenamento rispetto al data set di test.
Per arginare questo problema è possibile modificare la funzione di rischio, aggiungendo una funzione $Q(\boldsymbol{\theta})$; in questo modo si ottiene la funzione di costo:

$$C(\boldsymbol{\theta}) = R(\boldsymbol{\theta}) + \lambda Q(\boldsymbol{\theta}) \tag{7}$$

con λ parametro che esprime la rigidità del vincolo.
A questo punto l'obiettivo è quello di minimizzare la funzione di rischio (o di costo in caso di overfitting) e per fare ciò esistono diversi metodi, fra i quali il più comune è il metodo di discesa del gradiente.

3.2 Discesa del gradiente

La discesa del gradiente è una tecnica di ottimizzazione utilizzata per minimizzare l'errore che si introduce stimando la $\hat{\mathbf{y}} = f(\mathbf{x}, \boldsymbol{\theta})$ rispetto alla funzione "vera" $\mathbf{y} = f(\mathbf{x})$; quindi si avranno una Loss function $L(\mathbf{y}, f(\mathbf{x}, \boldsymbol{\theta}))$ ed un vettore dei parametri $\boldsymbol{\theta}$.
Esistono tre varianti del metodo di discesa del gradiente:

- *Batch Gradient Descent.*
  L'aggiornamento del vettore dei pesi $\boldsymbol{\theta}$ avviene solo dopo che sono stati presentati tutti i pattern all'algoritmo. Si calcola

$$\mathbf{G} = \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{\nabla}_{\theta} L(\mathbf{y}_k, f(\mathbf{x}_k, \boldsymbol{\theta})) \tag{8}$$

  e con tale risultato viene aggiornato il vettore dei parametri nel seguente modo:

$$\boldsymbol{\theta} - \epsilon \mathbf{G} \rightarrow \boldsymbol{\theta} \tag{9}$$

  Qui ϵ prende il nome di *learning rate* e regola l'aggiornamento del vettore dei pesi nella direzione opposta a quella del gradiente $\mathbf{G}$.
  Quindi con questa tecnica si calcola la discesa del gradiente una sola volta, tuttavia si impiega molto tempo per arrivare ad una convergenza ed è quindi poco adatta quando si hanno grandi moli di dati a disposizione.

- *Stochastic Gradient Descent.*
  Viene calcolata la discesa del gradiente per ogni pattern fornito all'algoritmo:

$$\mathbf{G}_i = \boldsymbol{\nabla}_{\theta} L(\mathbf{y}_i, f(\mathbf{x}_i, \boldsymbol{\theta})) \tag{10}$$

  e quindi anche l'aggiornamento dei pesi avviene tante volte quanti sono i pattern iniziali:

$$\boldsymbol{\theta} - \epsilon \mathbf{G}_i \rightarrow \boldsymbol{\theta} \tag{11}$$

  Questa tecnica è, all'opposto della precedente, utile quando il numero di pattern di input è molto elevato.

- *Mini Batch Gradient Descent.*
  Si tratta di una via di mezzo fra i due metodi appena presentati perché l'aggiornamento dei pesi avviene più volte dopo che sono stati presentati dei sottogruppi dell'intero data set di addestramento.

3.3 Apprendimento non supervisionato

Come si è visto, gli algoritmi di apprendimento supervisionato risultano essere molto utili nel caso in cui si abbiano a disposizione sia i vettori di input che i corrispettivi output target, perché si riesce ad ottenere un'approssimazione della relazione esistente input-output. Tuttavia non è sempre possibile avere a disposizione gli output target e bisogna capire se si riesce comunque ad estrarre informazione utile dai dati.
Come già accennato nelle prime pagine di questa trattazione quando non si hanno a disposizione gli output target si possono applicare tecniche di apprendimento non supervisionato, dove l'obiettivo è quello di trovare eventuali partizioni degli input (Clustering). Si consideri la figura 3 dove sono riportate tre diverse configurazioni possibili nel caso di input bidimensionali: nel caso a) è possibile la separazione in due sotto-gruppi e nel caso b) in un unico sotto-gruppo, mentre nel caso c) sembrerebbe non si possano stabilire graficamente eventuali separazioni.

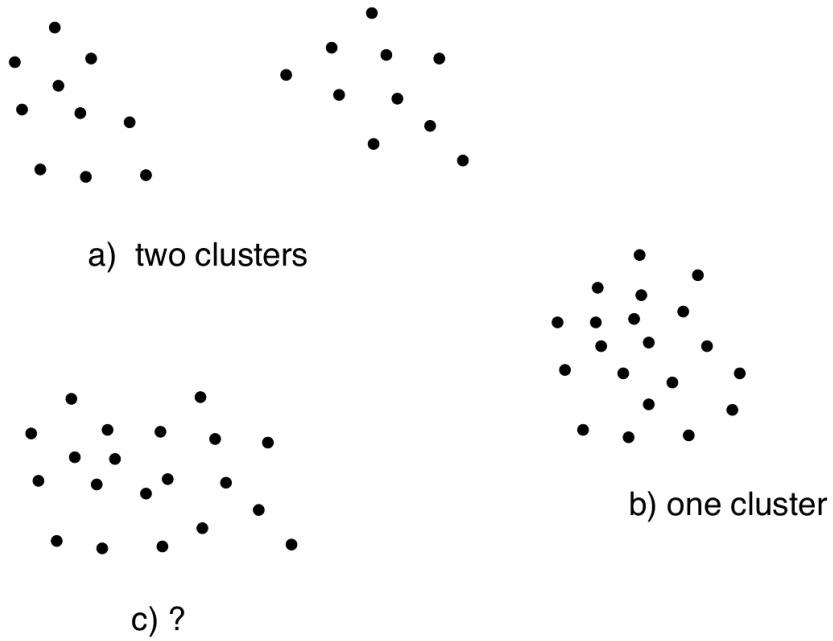

Figura 3: vettori di input in uno spazio bidimensionale in tre situazioni differenti. L'immagine è presa da [10]

Quindi un algoritmo di clustering si occupa della suddivisione del set di input Σ in un numero N di sottogruppi Σ_1,...,Σ_n, detti appunto cluster; si noti che lo stesso numero N non viene stabilito a priori e fornito all'algoritmo, ma viene anch'esso ricavato a partire dai dati. Una volta fatto sarà possibile implementare un classificatore per collegare nuovi vettori di input con i cluster precedentemente individuati.
Inoltre, aumentando il livello di complessità, è possibile trovare eventuali gerarchie di partizionamento, ovvero cluster di cluster.
Nella prossima sezione verrà presentato un metodo di clustering geometrico basato sulla distanza euclidea.

3.4 Metodo di clustering basato sulla distanza euclidea

Gli algoritmi di apprendimento non supervisionato sfruttano una qualche misura di similarità per separare i pattern di input nei vari cluster. Una possibilità è quella di utilizzare la semplice distanza euclidea per poter separare lo spazio n-dimensionale dei pattern in delle sotto-aree, che sono appunto i cluster.
Per fare ciò viene implementato un metodo iterativo, basato sulla definizione di alcuni punti particolari nello spazio dei pattern, detti *cluster seekers* (letteralmente "cercatori di cluster").
Si definiscono M punti nello spazio n-dimensionale $\mathbf{C_1}, ..., \mathbf{C_M}$ e l'obiettivo è quello di fare in modo che ogni punto si muova verso il centro di ogni singolo cluster, in modo che ogni cluster abbia al suo centro uno di questi cluster seekers.
Come è già stato spiegato precedentemente, l'algoritmo non conosce a prescindere il numero di cluster ma riesce a ricavarlo dai pattern stessi; per questa ragione il numero di cluster seekers M è inizialmente casuale ed esiste un procedura per ottimizzarlo, che verrà illustrata in seguito.
I pattern del training data set Σ vengono presentati all'algoritmo uno alla volta: per ognuno di essi ($\mathbf{x_i}$) si cerca il cluster seekers più vicino ($\mathbf{C_k}$) e lo si sposta verso $\mathbf{x_i}$ nel seguente modo:

$$\mathbf{C_k} + \alpha_k(\mathbf{x_i} - \mathbf{C_k}) \rightarrow \mathbf{C_k} \tag{12}$$

dove α_k è un parametro di apprendimento che determina di quanto il cluster seeker k-esimo si muove verso il punto $\mathbf{x_i}$.
A questo punto è utile fare in modo che più il cluster seeker è soggetto a spostamenti minore diventa l'entità dello spostamento. Per fare ciò si definisce una massa m_k e le si assegna un valore pari al numero di volte in cui $\mathbf{C_k}$ è stato soggetto a spostamenti (quindi anche il valore della massa verrà aggiornato di volta in volta); dopodiché si assegna ad α_k il seguente valore

$$\alpha_k = \frac{1}{1 + m_k} \tag{13}$$

e, dato che ad ogni iterazione che coinvolge $\mathbf{C_k}$ il valore di m_k aumenta di una unità, il parametro di apprendimento α_k diminuisce di volta in volta.
Il risultato di questo aggiustamento è che il cluster seeker si trova sempre nel punto che rappresenta la media dei punti del cluster.
Una volta che sono stati presentati tutti i pattern del training data set all'algoritmo, i vari cluster seeker andranno a convergere ai "centri di massa" dei cluster e la classificazione (cioè la delimitazione dei cluster nello spazio n-dimensionale) può essere fatta con una partizione dello spazio di Voronoi, di cui si riporta la seguente definizione:

> *In ogni insieme (topologicamente) discreto S di punti in uno spazio euclideo e per quasi ogni punto x, c'è un punto in S che è il più vicino a x. Il "quasi" è una precisazione necessaria dato che alcuni punti x possono essere equidistanti da 2 o più punti di S. Se S contiene solo due punti, a e b, allora il luogo geometrico dei punti equidistanti da a e b è un iperpiano, ovvero un sottospazio affine di codimensione 1. Tale iperpiano sarà il confine tra l'insieme di tutti punti più vicini ad a che a b e l'insieme di tutti i punti più vicini a b che ad a. È l'asse del segmento ab. In generale, l'insieme dei punti più vicini a un punto c ∈ S che ad ogni altro punto di S è la parte interna di un politopo (eventualmente privo di bordi) detto dominio di Dirichlet o cella di Voronoi di c. L'insieme di tali politopi è una tassellatura dell'intero spazio e viene detta tassellatura di Voronoi corrispondente all'insieme*

S. Se la dimensione dello spazio è solo 2, è facile rappresentare graficamente le tassellazioni di Voronoi; è a questo caso che si riferisce solitamente l'accezione diagramma di Voronoi.
(Wikipedia, Diagramma di Voronoi.)

Un esempio didattico del risultato di questa partizione è riportato in figura 4.

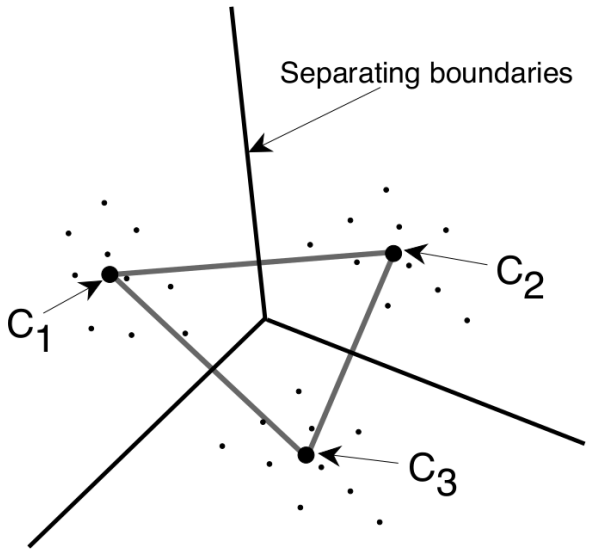

Figura 4: Si riporta un esempio di partizione dello spazio (bi-dimensionale) di Voronoi in tre sotto regioni . L'immagine è presa da [10]

Si è accennato qualche riga fa che il numero di cluster seeker è inizialmente scelto in maniera casuale, per poi essere ottimizzato; per il processo di ottimizzazione si utilizza la varianza dei pattern $\{\mathbf{x_i}\}$ per ogni cluster:

$$\sigma^2 = \frac{1}{L}\sum_{i=1}^{L}(\mathbf{x_i} - \boldsymbol{\mu})^2 \tag{14}$$

dove L è il numero di pattern nel cluster e $\boldsymbol{\mu}$ ne è la media:

$$\boldsymbol{\mu} = \frac{1}{L}\sum_{i=1}^{L}\mathbf{x_i} \tag{15}$$

A questo punto, se la distanza d_{ij} fra due cluster seeker $\mathbf{C_i}$ e $\mathbf{C_j}$ è minore di un determinato valore ϵ, allora si sostituiscono i due cluster seeker con uno nuovo posto nel loro centro di massa (tenendo conto delle due masse m_i e m_j); dall'altro lato, se vi è un cluster per il quale la varianza σ^2 è più grande di un valore δ, si aggiunge un nuovo cluster seeker vicino a quello già esistente e si eguagliano entrambe le loro masse a zero.
Come osservazione finale bisogna dire che nei metodi che si basano sul concetto di distanza è importante ri-scalare i valori delle componenti dei pattern (in linea di principio si possono avere componenti diverse con ordini di grandezza di molto differenti) in modo da evitare che alcune componenti pesino più di altre.

3.5 Iperparametri e Grid Search

Prima di parlare del Grid Search è necessario introdurre il concetto di **iperparametro**. Come detto nelle sezioni precedenti, un modello di apprendimento è caratterizzato da una serie di parametri che vengono modificati in maniera iterativa in modo da minimizzare la Loss function e, come noto, tale processo avviene attraverso un continuo confronto con il training data set. Quando si parla di iperparametri si intende invece una serie di parametri che caratterizzano il modello implementato che non sono modificati nel processo di addestramento con il training data set ma vengono prestabiliti dall'utente.
Chiaramente al variare degli iperparametri cambia anche la qualità del processo di apprendimento del modello e quindi anch'essi devono essere ottimizzati. A questo punto entra in gioco il metodo del Grid Search che è appunto un metodo di ottimizzazione degli iperparametri.
Il Grid Search è piuttosto semplice sia da comprendere concettualmente sia da implementare nella pratica; fa parte dei così detti *Brute-Force Search*, cioè di quei metodi che si basano sulla sistematica verifica di tutte le possibili soluzioni ad un problema per poi considerare la migliore. Per esempio si consideri il problema di dover cercare i divisori di un numero n: un approccio "Brute-Force" prevedrebbe di considerare tutti i numeri minori di n e verificare quelli per i quali la divisione non dà resto. Questo esempio permette anche di mettere in evidenza il limite principale di tale tipologia di approccio: il numero di possibilità da esplorare può aumentare molto velocemente, soprattutto se si considera un processo multivariato.
Tornando ora nello specifico al Grid Search, si consideri un modello caratterizzato da un numero k di iperparametri. Si può definire, in analogia a ciò che è stato fatto con i parametri, un vettore le cui componenti sono appunto gli iperparametri:

$$\boldsymbol{\mu} = (\mu_1, ..., \mu_k) \tag{16}$$

Tale vettore apparterrà ovviamente ad uno spazio k-dimensionale, sul quale può essere costruita una griglia i cui nodi corrispondono a particolari combinazioni degli iperparametri.
A questo punto si può avviare l'apprendimento del modello per ogni particolare configurazione degli iperparametri ed ottenere un valore per la Loss function. Si arriva allora ad avere una valore della Loss per ogni nodo della griglia e quindi basta considerare quello per il quale tale valore è minore, ottenendo la miglior configurazione degli iperparametri.
In Figura 5 nella pagina seguente è riportato per chiarezza un esempio visivo dell'esito di un processo di ottimizzazione degli iperparametri attraverso il metodo Grid Search.

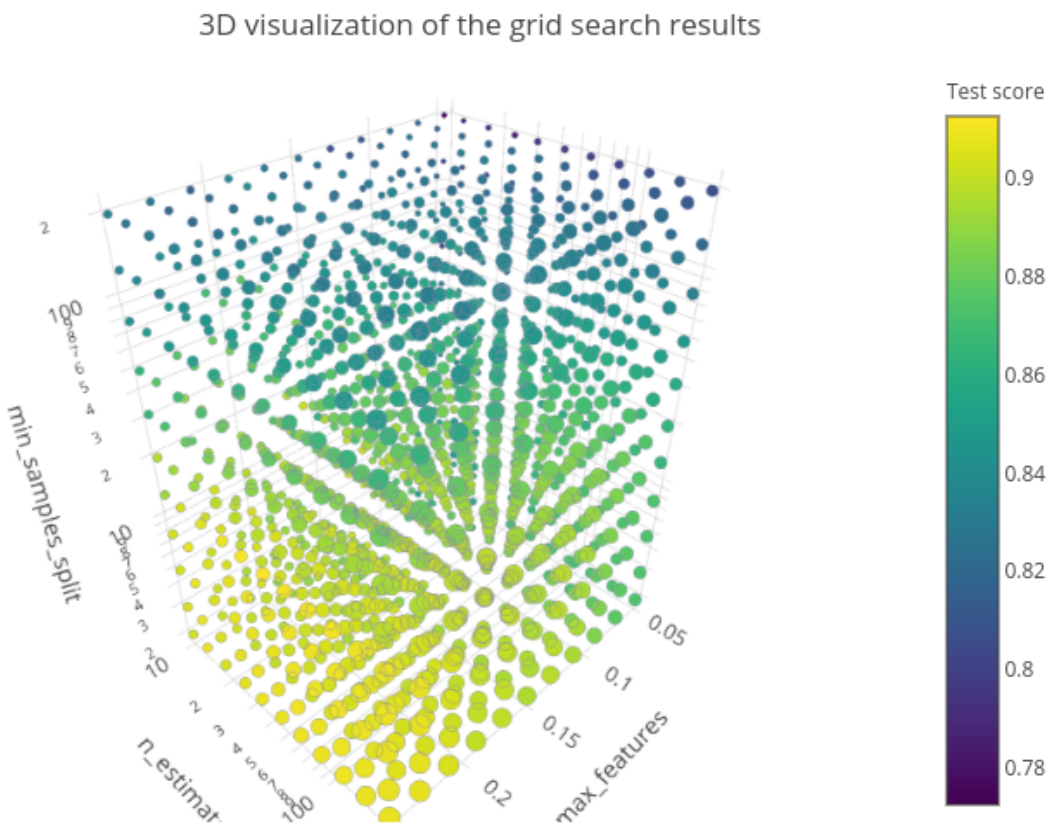

Figura 5: la figura illustra visivamente l'esito di un processo di ottimizzazione degli iperparametri attraverso il metodo Grid Search [8]

Come accennato precedentemente, man mano che aumenta la complessità del modello è molto probabile che aumenti il numero degli iperparametri e quindi la dimensionalità dello spazio introdotto precedentemente; ciò implica l'aumento considerevole del numero di configurazioni degli iperparametri da esplorare attraverso il Grid Search e quindi il tempo necessario per concludere l'ottimizzazione.
E' possibile ovviare parzialmente a questo problema attraverso il Random Grid Search (RGS), dove non sono considerati tutti i nodi della griglia, ma solo una loro parte selezionata in maniera casuale secondo una particolare distribuzione (ciò permette anche di tener conto di conoscenze pregresse).
Un ulteriore accorgimento può essere quello di arrestare le configurazioni meno promettenti prima di portarle a termine, risparmiando tempo e risorse computazionali. Per far ciò, basta impelmentare degli *scheduler* che tengano conto dell'andamento dei diversi training relativi alle diverse configurazioni. Un altra possibilità riguarda l'esplorazione di nuove varianti a partire dalle configurazioni iniziali. In quest'ultimo caso non è nemmeno necessario definire in modo rigido lo spazio degli iperparamentri da esplorare dato che le nuove configurazioni vengono ricercate a partire dall'andamento delle precedenti (*popultion based training*).

3.6 Reti Neurali

Le reti neurali sono probabilmente il metodo di apprendimento supervisionato più conosciuto ed utilizzato nel campo dell'analisi dati.
La struttura di una rete neurale prevede la presenza di unità fondamentali, dette neuroni, che sono organizzate in strati e legate fra di loro mediante delle connessioni (sinapsi), ciascuna delle quali è caratterizzata da un peso. Sono proprio questi pesi a giocare un ruolo fondamentale nel processo di apprendimento della rete perché sono loro i parametri soggetti a modifica.
Il nome rete neurale (artificiale) deriva dal fatto che la loro struttura è inspirata dalle corrispondenti strutture biologiche (seppur di molto semplificata).
In una rete neurale è sempre presente uno strato di input ed uno di output, mentre il numero di livelli nascosti può variare a seconda della complessità della rete; In figura 6 è riportato un esempio di rete neurale con un singolo strato interno nascosto.

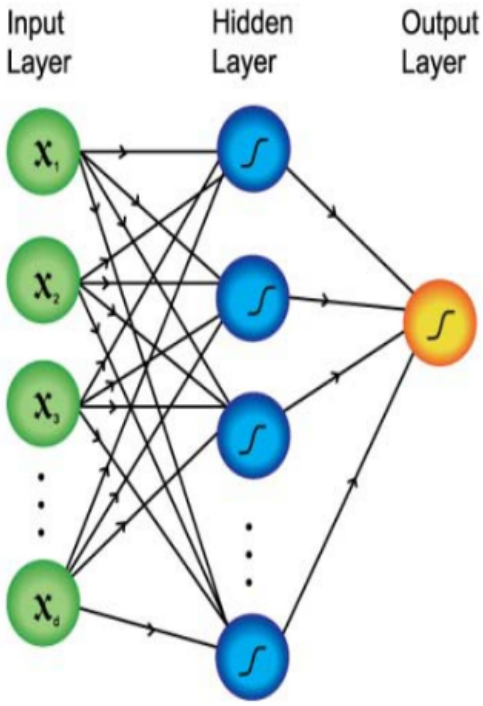

Figura 6: si riporta un esempio grafico di rete neurale formata da un unico strato nascosto. L'immagine è presa da [2].

Si può passare ora a presentare il modello del singolo neurone per capire com'è strutturato e quale compito svolge. Gli elementi che caratterizzano il singolo neurone sono:

1. Una serie di connessioni in ingresso (ciascuna caratterizzata da un proprio peso);
2. Un sommatore che ha il compito di svolgere la somma pesata degli input, utilizzando i pesi caratteristici delle connessioni;
3. Un output e la relativa funzione di attivazione, che viene usata per limitarne l'ampiezza (tipicamente ad intervalli [0,1] o [-1,1]);

4. Un valore di soglia che viene usato per aumentare o diminuire il valore ottenuto dalla somma pesata.

Si riporta in figura 7 lo schema grafico di un singolo neurone (k), dove $\mathbf{x} = (x_1, ..., x_m)$ è il vettore degli input input, $\mathbf{w}_k = (w_{k1}, ..., w_{km})$ è il vettore dei pesi, ϕ(x) è la funzione di attivazione, b_k è il valore di soglia e y_k è l'output.

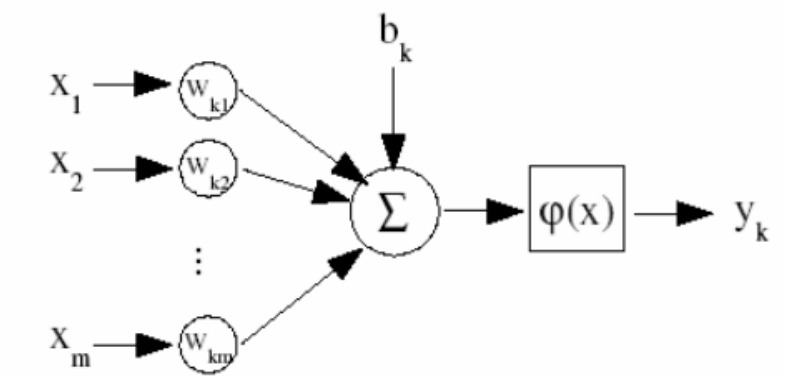

Figura 7: Illustrazione della struttura di un neurone [3].

Quindi il neurone opera la seguente somma pesata:

$$s_k = \mathbf{x} \bullet \mathbf{w_k} = \sum_{i=1}^{m} x_i w_{ki} \tag{17}$$

e si ottiene l'output attraverso la funzione di attivazione:

$$y_k = \phi(s_k + b_k) \tag{18}$$

Risulta utile spendere qualche parola in più sul tipo di funzione di attivazione più utilizzata, ovvero la funzione sigmoide:

$$sig(x) = \frac{1}{1 + e^{-\alpha x}} \tag{19}$$

dove α è un parametro che permette di regolare la pendenza della curva, come si evince dalla figura 8

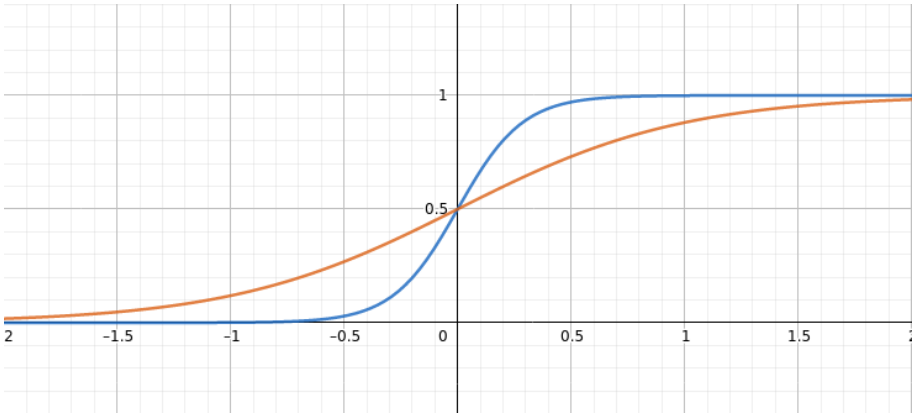

Figura 8: Si riportano due sigmoidi, dove per quella in rosso si ha α=2 e per quella in blu α=7.

Un singolo neurone è l'ingrediente fondamentale di una rete neurale e nella gran parte dei casi non è in grado di svolgere da solo nessun compito interessante ma deve sempre essere inserito in una rete. Tuttavia esiste una caso particolare nel quale un singolo neurone può portare a termine un compito di classificazione; affinché ciò sia possibile è necessario che i vettori evento siano riconducibili a due sole categorie e che il loro spazio possa essere separato (in relazione alle due categorie) da un singolo iper-piano. In questo caso esiste un teorema di convergenza che garantisce appunto la convergenza dei pesi nel processo di addestramento.

A questo punto è possibile passare ad un livello di complessità superiore, osservando in che modo possono essere organizzati i neuroni per formare la rete neurale; si distinguono due tipologie di reti:

1. Reti feedforward con uno o più strati: in questo caso il segnale si propaga dai nodi di input verso quelli di output, senza connessioni fra i neuroni di uno stesso strato;
2. Reti feedback: sono reti cicliche dove il segnale si propaga anche fra i neuroni di uno stesso strato.

Bisogna chiedersi ora in che modo apprende il singolo neurone. Una delle possibilità (nel caso in cui sia noto l'output target) è l'apprendimento con correzione di errore, che viene presentato velocemente nel seguito. Si consideri un singolo neurone che ha in ingresso una serie di input $(x_1, ..., x_n)$ e quindi produce un valore di output y attraverso la somma pesata già introdotta precedentemente; tale valore può quindi essere confrontata con il risultato atteso R, ottenendo così un errore $err = R - y$; si può quindi definire la funzione di costo

$$E = \frac{1}{2} err^2 \tag{20}$$

sulla quale si applicherà il metodo di discesa del gradiente già discusso nel paragrafo 3.2 per ottimizzare i parametri.
Una rete neurale può svolgere diverse funzioni, tuttavia quella principale che viene utilizzata in questa tesi prende il nome di riconoscimento. Il riconoscimento consiste nell'associazione da parte della rete di un vettore evento ad una delle varie categorie possibili. Tale obiettivo può essere ottenuto a seguito di una fase di addestramento dove vengono forniti alla rete sia i vettori in input che le categorie alle quali questi appartengono (si

tratta chiaramente di un processo di apprendimento supervisionato). Si ipotizzi di avere a disposizione dei vettori evento con un numero n di componenti (i dati) e, chiaramente, possono essere pensati come dei punti in uno spazio n-dimensionale; questo spazio potrà essere allora diviso in delle regioni che corrispondono alle varie categorie di cui si è parlato precedentemente ed i confini di queste zone si ottengono a seguito del processo di addestramento.
Arrivati a questo punto è possibile esporre la trattazione su come una rete neurale viene addestrata. Precedentemente si è introdotta la struttura di una rete neurale, specificando le differenze fra lo strato di input, quello di output e gli strati nascosti. Ogni singolo neurone nel suo processo di addestramento deve aggiornare i suoi pesi, in modo che l'output della rete neurale sia simile a quello atteso.
Uno dei metodi migliori per addestrare la rete neurale è l'algoritmo di back-propagation. Una rete neurale è caratterizzata da due tipologie di segnale: da un lato vi è un segnale di funzione che si propaga dallo strato di input verso quello di output e, dall'altro, vi è un segnale di errore che ha origine nello strato di output e si propaga verso quello di input. E' il segnale di errore a giocare un ruolo fondamentale nel processo di apprendimento tramite ottimizzazione dei pesi che caratterizzano la rete neurale.
Addentrandosi nell'algoritmo di back-propagation bisogna fare una distinzione fra il modo in cui esso viene applicato allo strato di output ed il modo in cui viene applicato agli strati nascosti:

1. Neurone nello strato di output.
   Si consideri uno strato di output con un numero n di neuroni e ci si focalizzi sul k-esimo. In un certo momento del processo di apprendimento, alla rete neurale si starà presentando il j-esimo elemento del training data set, quindi per il neurone k si otterrà il seguente segnale di errore:

$$err_k^{(j)} = R_k^{(j)} - y_k^{(j)} \tag{21}$$

   dove con la lettera y si intende il valore ottenuto in output dal neurone e con R il valore atteso.
   L'errore totale dello strato di output per il vettore evento j-esimo viene definito nel seguente modo:

$$E^{(j)} = \frac{1}{2}\sum_{k=1}^{n}(err_k^{(j)})^2 \tag{22}$$

   Se poi N è il numero totale di elementi del training data set, allora la funzione di costo può essere definita nel seguente modo:

$$E_{tot} = \frac{1}{N}\sum_{j=1}^{N} E^{(j)} \tag{23}$$

   e l'obiettivo è quello di minimizzare tale funzione di costo. Per fare ciò si procede aggiustando i pesi a seguito della presentazione di ogni singolo vettore evento. Si utilizza il metodo di discesa del gradiente, procedendo nel seguente modo:
   il gradiente è dato da

$$\frac{\partial E^{(j)}}{\partial w_{ki}^{(j)}} \tag{24}$$

e gli aggiornamenti del peso vengono applicati nel verso opposto del gradiente, ovvero

$$\Delta w_{ki}^{(j)} = -\mu \frac{\partial E^{(j)}}{\partial w_{ki}^{(j)}} \tag{25}$$

con μ fattore di apprendimento, definito nella sezione precedente come *learning rate*. Manca a questo punto il calcolo esplicito del gradiente, che può essere eseguito con la regola della catena

$$\frac{\partial E^{(j)}}{\partial w_{ki}^{(j)}} = \frac{\partial E^{(j)}}{\partial err_k^{(j)}} \frac{\partial err_k^{(j)}}{\partial y_k^{(j)}} \frac{\partial y_k^{(j)}}{\partial S_k^{(j)}} \frac{\partial S_k^{(j)}}{\partial w_{ki}^{(j)}} \tag{26}$$

dove $S_k^{(j)} = s_k^{(j)} + b_k^{(j)}$ (si faccia riferimento all'equazione (17)).
Una volta calcolate le quattro derivate si ottiene:

$$\frac{\partial E^{(j)}}{\partial w_{ki}^{(j)}} = -err_k^{(j)} \phi'(S_k^{(j)}) y_i^{(j)} \tag{27}$$

e quindi:

$$\Delta w_{ki}^{(j)} = err_k^{(j)} \phi'(S_k^{(j)}) y_i^{(j)} \mu \tag{28}$$

2. Neurone in uno strato nascosto
   In questo caso l'output del neurone non ha un diretto valore con il quale può essere confrontato, quindi il segnale di errore deve essere determinato a partire dai segnali di errore di tutti i neuroni dello strato successivo, da cui il nome di back-propagation proprio perché il segnale di errore prosegue all'indietro dall'output verso l'input.

Come ultima considerazione sulle reti neurali bisogna sottolineare che il coefficiente di apprendimento deve essere scelto in maniera accurata, infatti se fosse troppo piccolo si avrebbe una convergenza estremamente lenta e, viceversa, un valore troppo grande porterebbe ad una instabilità con comportamento oscillatorio. Per gestire meglio questo aspetto, nella pratica viene definito un learning rate variabile. Generalmente si fa in modo che questo fattore parta da un certo valore per decrescere mano mano che si ci avvicina ad una soluzione ottimale. In prima istanza infatti è utile avere un valore abbastanza grande da poter seguire in modo efficace la discesa del gradiente, rendendo più rapida la ricerca della soluzione ottimale. Successivamente, però, risulta utile diminuire questo stesso parametro per evitare oscillazioni attorno al punto di minimo.

3.7 Alberi Decisionali

Gli alberi decisionali sono, al pari delle reti neurali, un metodo ML di apprendimento supervisionato e la loro caratteristica fondamentale è il presentarsi in maniera particolarmente intuitiva perché è possibile avere una semplice rappresentazione grafica del meccanismo del loro funzionamento.
Gli alberi decisionali rappresentano un mezzo estremamente interessante per le operazioni di classificazione (sia per output continui che discreti) ed operano attraverso una serie di test sugli attributi degli input (con il termine attributo o *features* si intende una componente del vettore di input, come già specificato nella sezione introduttiva di questa tesi [2]).
Come primo passo è utile discutere come è strutturato un albero decisionale, introducendo alcune notazioni (come ausilio alla trattazione si riporta in figura 9 un esempio di albero decisionale molto semplice).

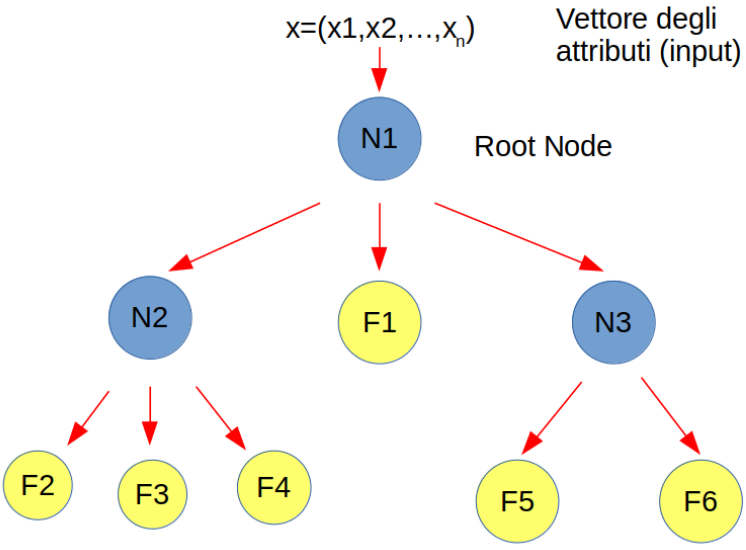

Figura 9: esempio di come è strutturato un albero decisionale

Gli elementi che caratterizzano un albero decisionale sono:

- Nodo.
  I nodi sono riportati in figura 9 come dei cerchi colorati in azzurro e contrassegnati dalla lettera N. Ogni nodo si occupa di eseguire un test su di un singolo attributo (il nodo iniziale dove avviene il primo test e quindi la prima differenziazione degli input è detto ”root node”);

- Ramo.
  I rami (detti anche archi) determinano le regole di ”splitting”, ovvero le regole attraverso le quali vengono separati gli esempi nelle rispettive categorie a seconda dei loro attributi; tali rami determinano quindi i percorsi all'interno degli alberi decisionali e quindi, in ultima analisi, la classificazione finale;

- Foglie.
  Le foglie sono una classe particolare di nodi, ovvero i nodi finali che, in quanto tali,

non generano nuove diramazioni ma rappresentano il risultato finale del processo di classificazione.

Bisogna tenere a mente che si sta parlando di una metodologia di ML, che è quindi volta ad estrarre dai dati di input (training data set) e dai corrispettivi output target l'informazione generale, per poter poi applicare l'algoritmo a casi per i quali non si è in possesso degli output di riferimento.
Detto ciò è necessario capire in che modo possa essere costruito un albero decisionale e l'idea di base è quella di stabilire, di volta in volta, il criterio sugli attributi che più discrimina gli input.
Nella costruzione degli alberi decisionali bisogna distinguere due fasi successive:

- "Building", ovvero costruzione.
  In questo prima fase l'obiettivo è quello di far crescere l'albero in dimensione, quindi in termini di rami e nodi per avere un numero adeguato di regole di splitting così da ottenere una classificazione in classi omogenee nella fase finale. In questa prima fase si ottiene un albero particolarmente folto e quindi probabilmente soggetto all'overfitting (come primo argine a ciò è possibile introdurre un criterio d'arresto capace di fermare la crescita dell'albero al realizzarsi di particolari condizioni);

- "Pruning", ovvero potatura.
  Questa fase è quella che permette di ridurre l'overfitting perché vengono eliminati i rami che non contribuiscono in maniera significativa al processo di classificazione.

Come già detto la prima fase è quella di "Building", durante la quale viene costruito l'albero decisionale aggiungendo nodi e rami, con il fine di ottenere una classificazione finale il più possibile omogenea; è altresì noto che ad ogni nodo corrisponde un attributo sul quale è effettuato un test, quindi è evidente che, dato un particolare numero di attributi, il numero di alberi possibili è molto elevato. Bisogna trovare un modo per disporre nella maniera più efficace i nodi all'interno dell'albero: l'idea è quella di scegliere per primo l'attributo attraverso il quale si ha una maggiore discriminazione dei dati in input. Per fare ciò si possono percorrere due strade distinte, utilizzando:

- Coefficiente di impurità di Gini, ovvero un indicatore della frequenza con cui un elemento casuale, appartenente al data set, sia identificato in modo non corretto.

- Guadagno informativo, ovvero un indicatore, preso in prestito dalla teoria dell'informazione e basato sul concetto di entropia, di quanto la scelta di uno specifico attributo consenta di ridurre l'entropia informativa dell'insieme (semplificando, la variabilità dei valori dell'attributo presenti all'interno di un nodo foglia).

E' chiaro che entrambi questi indici vengono utilizzati con la stessa finalità, tuttavia ogni algoritmo ha una differente logica di costruzione e quindi adotterà uno solo dei due indici, con la possibilità di ottenere risultati differenti.
A questo punto l'albero decisionale è stato costruito e quindi si deve passare alla fase di "pruning", con l'obiettivo di ridurre le dimensioni dell'albero per evitare l'ormai noto overfitting. Per fare ciò le strade sono due, infatti da un lato si può seguire un approccio "top-down", partendo dalla radice e suddividendo l'intera struttura in sotto alberi e dall'altro un approccio "bottom-up", partendo dalle foglie ed analizzando l'impatto di ogni

singola potatura; è altresì possibile introdurre nella fase di costruzione stessa un criterio di *early-stopping* o *pre-pruning* richiedendo un valore minimo di miglioramento dell'algoritmo fra un'iterazione e l'altra: ad ogni passaggio di separazione dell'albero decisionale viene fatto un controllo sulla Loss function e, se tale errore non diminuisce significativamente tra un passaggio e l'altro, si interrompe il processo di costruzione dell'albero. Il problema dell'*early stopping* è che potrebbe portare ad una classificazione non ottimale, infatti non è detto che nell passaggio successivo di separazione non ci sarebbe potuta essere una riduzione significativa dell'errore; per questa ragione in genere si utilizzano entrambi i metodi di *pruning* e di *early stopping* parallelamente, per poi confrontare i risultati.

In conclusione bisogna sottolineare che gli alberi decisionali sono particolarmente utilizzati per i seguenti motivi:

- semplicità nell'interpretazione e nella visualizzazione;
- tolleranza ad eventuali attributi mancanti per alcuni input nel training data set o nel test data set;
- insensibilità ad eventuali attributi irrilevanti nella classificazione;
- invarianza per trasformazioni monotone effettuate sugli attributi, che rende la fase di pre- processamento dei dati non necessaria.

Gli alberi decisionali hanno tuttavia una serie di limiti elencati nelle righe che seguono:

- instabilità rispetto al variare del training data set, cioè data set di allenamento di poco differenti fra loro producono risultati molto diversi;
- frequente problema dell'overfitting.

Tuttavia è possibile combinare vari alberi decisionali insieme per ottenere migliori prestazioni predittive rispetto all'utilizzo di un solo albero; infatti, come è stato appena illustrato, gli alberi decisionali singoli hanno alcuni problemi non di poco conto, che possono però essere parzialmente arginati considerando più alberi e prendendo come decisione finale una media delle decisioni di ciascun albero.
Arrivati a questo punto ci si chiede in che modo possano lavorare insieme vari alberi decisionali per ottenere una decisione finale migliore rispetto a quella del singolo albero; una nota tecnica prende il nome di *Bagging*, nella quale si generano in maniera casuale dei sottogruppi del training data set ed ognuno di questi viene utilizzato per l'addestramento di un albero decisionale. Il risultato sarà una collezione di alberi decisionali e, come decisione finale, si utilizzerà la media delle decisioni dei singoli alberi.
Esiste un'estensione di questo metodo conosciuta con il nome di *Random Forest*; in questo caso viene aggiunto un passaggio ulteriore al processo appena illustrato perché viene scelto casualmente anche un sottogruppo degli attributi dei pattern, ottenendo così un metodo capace di agire anche su data set ad alta dimensionalità e che permette di ridurre sensibilmente il problema dell'overfitting.

3.8 Curse of dimensionality e riduzione della dimensionalità

A questo punto si è giunti finalmente al cuore di questa trattazione, dove vengono illustrate le basi teoriche di un metodo di apprendimento non supervisionato, il Variational Autoencoders (VAEs), del quale si studierà nel prossimo capitolo un'applicazione al campo della fisica delle alte energie.
Gli argomenti trattati nelle prossime pagine per presentare le basi teoriche del VAEs seguono la seguente struttura logica:

- Presentazione del problema della dimensionalità;
- Una delle possibili soluzioni: Autoencoders;
- Evoluzione dell'autoencoders: il Variational Autoencoders.

Come è stato più volte detto in questa trattazione, quando si parla di input (o pattern) ci si riferisce a dei vettori, le cui componenti sono i dati veri e propri; questi vettori, in quanto tali, possono essere pensati all'interno di un opportuno spazio n-dimensionale (con n=numero di componenti del vettore).
Quando si parla di "Curse of dimensionality" (letteralmente "la maledizione della dimensionalità") ci si riferisce ad una serie di problemi che ci si trova ad affrontare quando bisogna trattare spazi con un'alta dimensionalità, che altrimenti non comparirebbero in spazi a bassa dimensionalità.
Dato che all'aumentare della dimensionalità i volumi nello spazio aumentano in maniera significativa, ci si troverà nella situazione per cui i pattern risultano sparsi nello spazio e questo è chiaramente un problema per ogni analisi che ne si vuole fare basata sulla statistica; infatti, per ottenere dei risultati significativi a livello statistico, la quantità di dati necessari aumenta in maniera esponenziale e questo risulta essere un problema a livello pratico.
Quando si parla di riduzione della dimensionalità ci si riferisce ad una serie di tecniche, attraverso le quali viene ridotto il numero delle variabili che caratterizzano i vettori di input; l'obiettivo di base è quello di proiettare gli elementi dello spazio n-dimensionale (i vettori di input) su di uno spazio a dimensione inferiore, cogliendo l'essenza stessa dei dati.
Ciò che è necessario notare è che avere degli input con più bassa dimensionalità permette di avere anche meno parametri (gradi di libertà) e quindi una struttura più semplice del modello. Il prediligere la semplicità alla complessità, oltre che per gli ovvi motivi, deriva dal fatto che la seconda è molto soggetta al fenomeno dell'overfitting.
Il processo di riduzione della dimensionalità è una metodologia di preparazione dei dati, per poi essere presentati all'algoritmo di apprendimento, che si troverà di fronte delle informazioni più compatte e quindi più facilmente processabili.
Inoltre bisogna notare che, se il processo di riduzione della dimensionalità viene svolto sul training data set, allora deve essere attuato anche sul test data set, per garantire un processo di verifica valido.

Il processo di riduzione della dimensionalità può essere portato avanti attraverso due metodologie differenti:

- Selezione, dove solo alcune componenti dei vettori di input vengono conservate;
- Estrazione, dove viene creato un numero ridotto di nuove componenti a partire da quelle originali.

A prescindere da questa distinzione, bisogna sottolineare che tutti i processi di riduzione della dimensionalità hanno una struttura comune, ovvero sono caratterizzati da una fase di *encoding* (che rappresenta il vero e proprio processo di riduzione della dimensionalità) e da una fase di *decoding*, nella quale si verifica quanta informazione è stata persa nel processo.
Si riporta in figura 10 l'illustrazione grafica del processo *encoding-decoding*

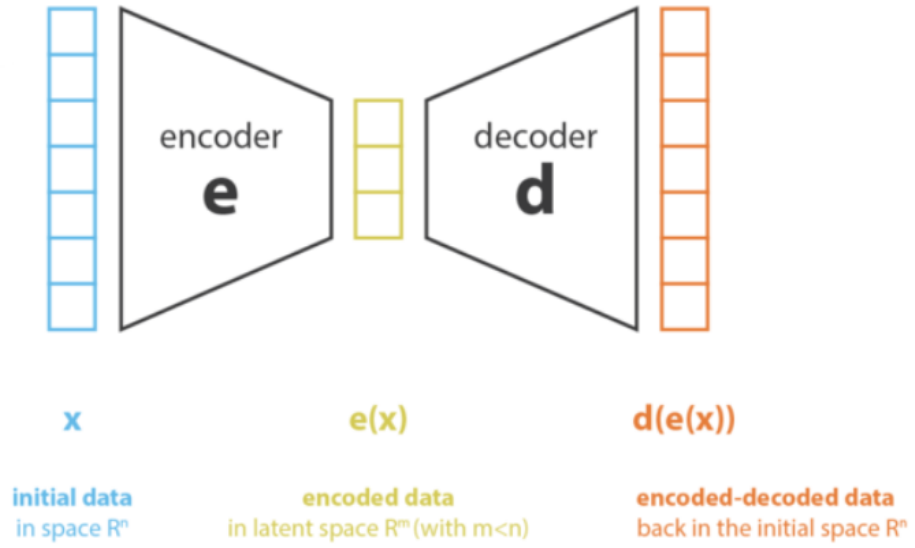

Figura 10: Strutture generale di un processo di riduzione della dimensionalità . L'immagine è presa da [12]

Il vettore di input $\mathbf{x}$ (n-dimensionale) viene compresso dall'encoder $\mathbf{e}$ in un vettore $\mathbf{e}(\mathbf{x})$ di uno spazio m-dimensionale (con m<n), detto *spazio latente*; l'encoder, come detto, può agire per selezione o per estrazione.
Il decoder $\mathbf{d}$ svolge la funzione opposta, ovvero decomprime il vettore $\mathbf{e}(\mathbf{x})$ in $\mathbf{d}(\mathbf{e}(\mathbf{x}))$ per tornare allo spazio originario n-dimensionale.
Nel caso in cui $\mathbf{x} = \mathbf{d}(\mathbf{e}(\mathbf{x}))$ (caso ideale) si dice che il processo è un *lossless encoding*, ovvero non c'è stata perdita di informazioni nella riduzione della dimensionalità; viceversa, se $\mathbf{x} \neq \mathbf{d}(\mathbf{e}(\mathbf{x}))$, si parla di un *lossy encoding*, cioè un processo nel quale parte dell'informazione viene persa e non può essere recuperata con la fase di decoding.

Come conseguenza di ciò che è stato appena illustrato, l'obiettivo di un processo di riduzione della dimensionalità è quello di trovare la coppia encoder-decoder (e,d) fra una famiglia di encoder E e di decoder D, che minimizzi l'informazione persa:

$$(e, d) = \min_{E \times D} \epsilon(\mathbf{x}, \mathbf{d}(\mathbf{e}(\mathbf{x}))) \tag{29}$$

dove $\epsilon(\mathbf{x}, \mathbf{d}(\mathbf{e}(\mathbf{x})))$ è la grandezza attraverso la quale viene quantificata la quantità di informazione persa nel processo di riduzione.

A questo punto è possibile illustrare le varie metodologie di riduzione della dimensionalità, secondo la distinzione già incontrata fra selezione ed estrazione.
I metodi di selezione ("*Feature Selection Methods* (FSM)") sono metodi attraverso i quali vengono selezionate le componenti dei vettori di input da tenere e quelle da eliminare perché irrilevanti per le analisi successive. I FSM si includo i *wrapper methods* ed i *filter methods*: i primi valutano il modello con varie combinazioni di subset delle variabili originali e selezionano quella per la quale si ha la maggiore accuratezza del modello, mentre i secondi si basano maggiormente sulle caratteristiche intrinseche dei dati (correlazioni, contenuto informativo, etc...).
I metodi di estrazione, invece, si basano fortemente sull'algebra lineare; in particolare vengono utilizzati spesso per la riduzione della dimensionalità i metodi di fattorizzazione delle matrici per cogliere la parte più importante dei dati.
Il più comune di questi metodi prende il nome di *Principal Component Analysis* (PCA), del quale verrà presentata brevemente l'idea di base, evitando di addentrarsi troppo nella trattazione matematica che è essenzialmente riconducibile ad un calcolo di autovalori ed autovettori.
L'idea del PCA è quella di costruire un numero n_e di nuove variabili indipendenti che siano combinazione lineare delle n variabili di partenza; tale costruzione viene fatta in modo tale che la proiezione delle vecchie variabili sul nuovo sottospazio generato da quelle nuove sia il più possibile vicina ai dati iniziali, dove la vicinanza è da intendere in termini della distanza euclidea. In altre parole con il PCA si ricerca il sottospazio dello spazio dei pattern di partenza per il quale l'errore che viene compiuto nell'approssimazione dei dati tramite proiezioni sia il più piccolo possibile. Si riporta in figura 11 un'illustrazione di ciò che è stato appena detto.

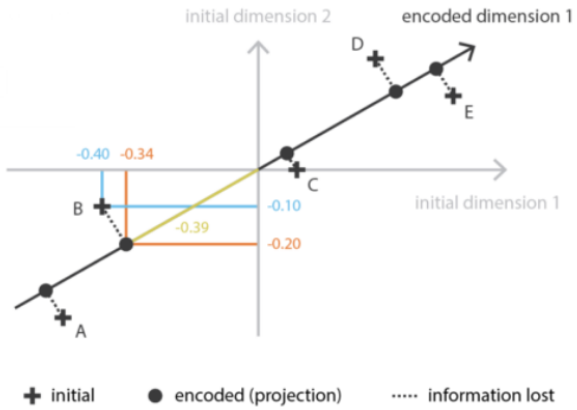

Figura 11: Illustrazione del processo di PCA nel caso di uno spazio dei pattern iniziali bi-dimensionale [12].

3.9 Autoencoders

Gli autoencoders, come ogni altro metodo di riduzione della dimensionalità, sono costituiti da un encoder e da un decoder; tuttavia in questo caso la peculiarità è che sia l'encoder che il decoder sono delle reti neurali, come è possibile vedere in figura 12.

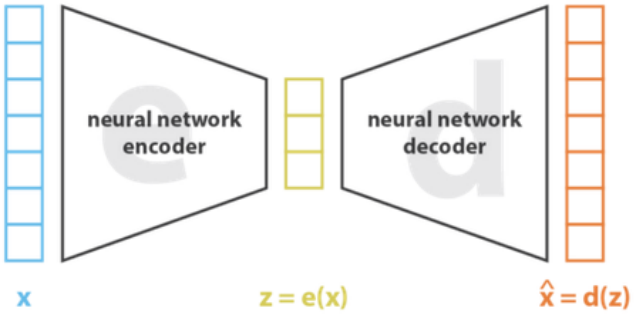

Figura 12: Struttura di un generico autoencoder ([12]).

L'obiettivo è chiaramente quello di individuare la coppia encoder-decoder che ottimizza il processo di ricostruzione degli input e ciò viene fatto attraverso il seguente processo iterativo: si presentano all'encoder i pattern di partenza uno alla volta, subiscono un processo di riduzione della dimensionalità e poi vengono ricostruiti (tornando alla dimensionalità di partenza), viene calcolato l'errore dal confronto fra l'input iniziale e quello ricostruito ed avviene l'aggiornamento dei pesi della rete neurale mediante il meccanismo di back-propagation, già incontrato nella sezione 3.6.
Intuitivamente l'autoencoder può essere pensato come un collo di bottiglia, attraverso il quale solo una parte dell'informazione riesce a passare oltre e a formare i vettori dello spazio latente. Facendo riferimento alla figura 12 si osserva che, a partire dai pattern in input $\mathbf{x}$, come primo passo si costruisce lo spazio latente degli $\mathbf{z} = e(\mathbf{x})$ per poi procedere alla fase di decodifica nella quale si ottengono i pattern ricostruiti $\hat{\mathbf{x}} = d(\mathbf{z})$; si procede successivamente al calcolo degli errori nel seguente modo:

$$L = \|\mathbf{x} - \hat{\mathbf{x}}\| \tag{30}$$

dove L è l'errore di ricostruzione.
Una considerazione necessaria circa l'errore, che potrebbe sembrare in contraddizione con quanto detto fino ad ora sul concetto di ottimizzazione, è che si vuole di norma evitare che $\mathbf{x} = \hat{\mathbf{x}}$, perché questo vuol dire che l'autoencoder ha imparato la funzione identità e, come conseguenza, la struttura dello spazio latente, che è quella interessante per il processo di riduzione della dimensionalità, non porta alcuna informazione interessante; ciò è dovuto al fatto che l'encoder non impara se vi siano variabili più o meno importanti di altre o se esse possano essere compattate in nuove variabili di dimensionalità minore.
Per fornire un esempio pratico di ciò che è stato appena affermato, si consideri un insieme di vettori di input N dimensionali; una possibilità è quella di prendere una per una le componenti dei pattern e disporle lungo una retta (spazio latente 1-dimensionale) nella fase di encoding, per poi procedere in maniera inversa nella fase di decodifica. L'errore con questo procedimento sarà nullo ma non si può essere soddisfati essenzialmente per due motivi, ovvero perché lo spazio latente non è interpretabile e sfruttabile e perché in

un processo di riduzione della dimensionalità si vuole fare in modo che i dati continuino a conservare una qualche struttura.
Una possibilità per evitare il risultato appena illustrato, che è in fin dei conti una sfaccettatura del concetto di overfitting, è di aggiungere alla funzione L un fattore di regolarizzazione che penalizza i risultati per i quali $\mathbf{x} = \hat{\mathbf{x}}$.
Quindi bisogna sempre porre particolare attenzione alla scelta della profondità dell'encoder, ovvero alla sua capacità di riduzione della dimensionalità.
Per completezza nella trattazione si osserva che gli autoencoder possono essere sia lineari che non; il primo caso si ottiene quando non si inserisce una funzione di attivazione non lineare e si utilizzano solo due strati, quindi le trasformazioni possono essere rappresentate come matrici e si ottiene un risultato simile a quello del PCA (3.8).
Il caso di autoencoder non lineari (*deep autoencoder*) può essere pensato come un passo successivo per quanto riguarda la riduzione della dimensionalità. Infatti, come è stato già detto, il PCA ricerca il miglior iperpiano nello spazio dei pattern originali sul quale questi possano essere proiettati in modo da ridurre la perdita di informazione; dall'altro lato gli autoencoder non lineari non si limitano alla ricerca di iperpiani, ma possono esplorare anche superfici più complesse, come si evince chiaramente dalla figura 13.

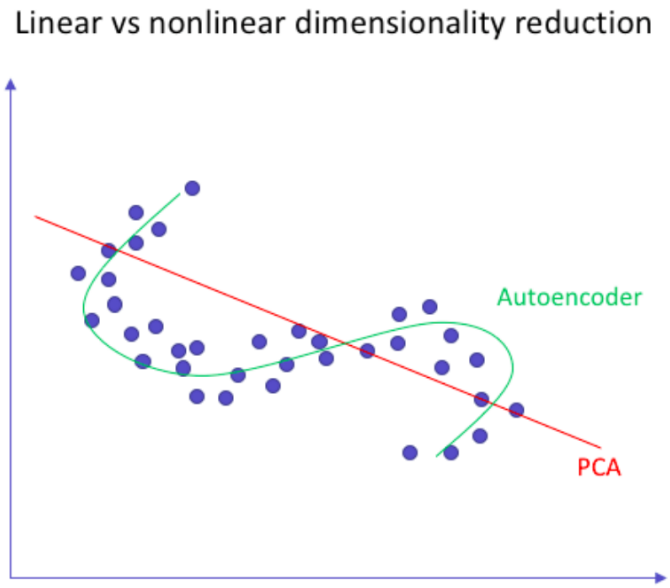

Figura 13: Differenza fra i metodi lineari (PCA) e gli autoencoder non lineari [7].

3.10 Variational Autoencoders (VAEs)

Nella sezione precedente è stato presentato l'autoencoder come un metodo di riduzione della dimensionalità; tuttavia, apportando una sostanziale modifica, tale metodo può essere utilizzato per la generazione di pattern ed in questo caso si parla di Variational Autoencoders (VAEs).
Bisogna domandarsi a questo punto in che modo si possa passare dalla riduzione della dimensionalità al generare nuovi pattern e perché il semplice autoencoder, così come è stato incontrato, non permette di raggiungere tale obiettivo e debba essere modificato opportunamente.
Come primo passo è necessario capire perché gli autoencoders non siano adatti al processo di generazione e la risposta può essere intuita già da ciò che è stato detto nella sezione precedente ma si cercherà di argomentarla meglio. L'autoencoder, come noto, agisce su di un pattern originale $\mathbf{x}$ trasformandolo in un $\mathbf{z} = e(\mathbf{x})$ nello spazio latente (fase di encoding) e poi trasforma nuovamente $\mathbf{z}$ per tornare allo spazio originale; si potrebbe pensare che, per far svolgere la funzione di generazione, si potrebbe scegliere un punto a caso dello spazio latente (che è stato costruito nella fase di encoding) e darlo in pasto al decoder, ottenendo così un nuovo pattern per nulla collegato a quelli originali. Il problema nel ragionamento appena esplicato sta nel non tenere in conto la regolarità dello spazio latente.
L'autoencoder codifica in modo puntuale ogni esempio associando ad ognuno di essi un punto specifico e diverso dagli altri all'interno dello spazio latente. In questo modo, se viene dato in input un esempio mai incontrato durante la fase di training, si presentano delle difficoltà nella sua successiva rigenerazione. In questo caso infatti il modello comincia la fase di rigenerazione a partire da un punto nello spazio latente a cui non è associato nessun significato, come se in quel punto lo spazio latente fosse discontinuo (un esempio didattico è riportato in figura 14). Per ovviare a questo problema la fase di encoding deve assumere un significato probabilistico, associando ai vari pattern non più un solo punto ma un'intera distribuzione. In questo modo lo spazio latente sarà meno soggetto ad avere discontinuità al suo interno, facilitando così il processo di generazione.

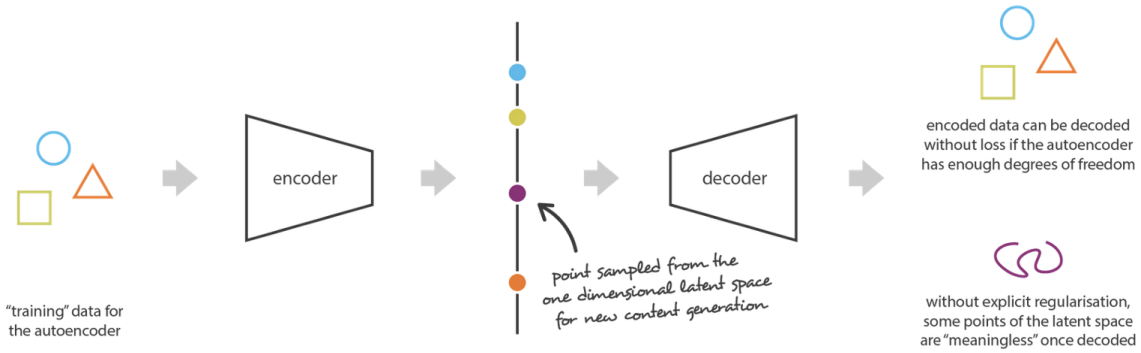

Figura 14: Illustrazione grafica e semplificata dei limiti nell'applicazione degli autoencoder per fini di generazione di nuovi pattern [12].

Si può quindi pensare al VAEs come un autoencoder per il quale si attua una regolarizzazione dello spazio latente durante il processo di addestramento ed è quindi adatto alla generazione di nuovi pattern.
I VAEs, al pari degli autoencoders, sono costituiti da una coppia di encoder e decoder con

una sostanziale differenza, nel senso che per gli autoencoders il pattern di partenza viene codificato come un punto nello spazio latente ($\mathbf{z}$), mentre per i VAEs la codifica avviene tramite una distribuzione nello spazio latente ($p(\mathbf{z}|\mathbf{x})$). Il processo seguito nella fase di addestramento è il seguente:

1. Il pattern iniziale viene codificato nello spazio latente come una distribuzione di probabilità;
2. Si campiona un punto dello spazio latente a partire dalla distribuzione del punto precedente;
3. Tale punto viene decodificato dal decoder, ottenendo il pattern ricostruito;
4. Avviene il confronto fra il pattern iniziale e quello ricostruito da cui si calcola l'errore, che viene poi propagato mediante il meccanismo della backpropagation.

Nella pratica si cerca di fare in modo che la distribuzione ottenuta alla fine del processo di codifica sia il più possibile vicina ad una distribuzione Normale, perché come si vedrà in seguito si riesce a garantire una certa regolarità dello spazio latente; in particolare si farà in modo che l'encoder restituisca la media e la matrice di covarianza della distribuzione Normale.
Seguendo questa linea, si ottiene che il processo di addestramento è regolato da una funzione di perdita, composta da un termine relativo alla ricostruzione ed uno relativo alla regolarizzazione dello spazio latente; tale termine di regolarizzazione viene espresso mediante la *Divergenza di Kullback-Leibler*, che rappresenta una misura della differenza ta due distribuzioni di probabilità.
Nelle righe precedenti è stata richiesta la regolarità dello spazio latente ed è giusto chiarire cosa si intenda effettivamente. Lo spazio latente è regolare se:

- è continuo, ovvero due punti vicini portano ad un risultato simile una volta decodificati;
- è completo, nel senso che un qualunque punto porta ad un risultato sensato una volta decodificato.

Il solo fatto di codificare i pattern come delle distribuzioni non garantisce la continuità e la completezza, perché se non si inserisce il termine di regolarizzazione nella funzione di perdita il VAE continuerà a comportarsi come un semplice autoencoder, tendendo semplicemente a minimizzare l'errore di ricostruzione. Questo può avvenire in due modi, ovvero codificando i pattern come distribuzioni o con varianze molto piccole (quasi come singoli punti) o con medie molto diverse fra loro (punti molto lontani nello spazio latente); nel primo caso non viene garantita la continuità e nel secondo la completezza.
Per ottenere la regolarità dello spazio latente si richiede allora che le distribuzioni con cui vengono codificati i pattern siano il più possibile vicine a distribuzioni Normali con media zero e matrice di covarianza uguale all'identità; le medie saranno allora vicine con conseguente sovrapposizione delle distribuzioni, anche perché la matrice di covarianza così fatta impedisce la codifica come punti nello spazio latente. Il prezzo da pagare sarà chiaramente un più alto errore nella fase di ricostruzione.
Prima di passare alla formulazione matematica dei VAEs, bisogna notare che la regolarità

dello spazio latente implica la presenza di un gradiente, il quale permette di mischiare le caratteristiche dei pattern in input e quindi di dare un significato al campionamento nello spazio latente.

3.10.1 Formulazione matematica dei VAEs

Per formulare in maniera rigorosa i VAEs è necessario utilizzare l' inferenza variazionale e verranno dati alcuni concetti fondamentali della teoria dell'informazione.
Per prima cosa bisogna introdurre una grandezza, che è in grado di quantificare la quantità di informazione di una proposizione ed è appunto detta *Informazione*:

$$I = \log p(x) \tag{31}$$

dove x è l'evento.
Questo concetto di informazione coincide con quello che si possiede intuitivamente, ovvero che ad eventi certi o molto probabili corrisponde una quantità di informazione nulla o molto bassa, mentre ad eventi poco probabili corrisponde una quantità di informazione più alta.
Un'altra quantità fondamentale nella teoria dell'informazione è l'*entropia*, ovvero l'informazione media, ed è definita nel seguente modo:

$$H = \sum p(x) \log p(x) \tag{32}$$

A questo punto si introduce la *KL divergency* (già accennata precedentemente) che è di fondamentale importanza nel processo di addestramento dei VAEs; si tratta di una misura della dissimilarità di due distribuzioni (p(x) e q(x)):

$$KL(p(x)||q(x)) = -\sum p(x) \log q(x) + \sum p(x) \log p(x) = -\sum p(x) \log \frac{q(x)}{p(x)} \tag{33}$$

si nota come essa sia molto simile alla differenza delle entropie delle due distribuzioni e ciò è in linea con l'intuizione perché due distribuzioni che hanno un'informazione media uguale saranno pressoché uguali. Le due proprietà fondamentale della *KL divergency* sono le seguenti:

1. $$KL(p(x)||q(x)) \geq 0 \tag{34}$$

2. $$KL(p(x)||q(x)) \neq KL(q(x)||p(x)) \tag{35}$$

Ora è possibile ricollegarsi al discorso sui VAEs e si definisca con $\mathbf{x}$ la grandezza osservabile e con $\mathbf{z}$ quella nascosta dello spazio latente. Il teorema di Bayes [6] permette di scrivere:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})} \tag{36}$$

dove

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})dz \tag{37}$$

tuttavia tale integrale è difficile da calcolare ed è in questo punto che entra in gioco l'inferenza variazionale per approssimare $p(\mathbf{z}|\mathbf{x})$ con una qualche funzione $q(\mathbf{z}|\mathbf{x})$. Si assume inoltre che quest'ultima debba essere scelta dalla famiglia delle distribuzioni Normali, andando a variare i parametri in modo che risulti il più possibile simile a $p(\mathbf{z}|\mathbf{x})$, che viene detta *prior* e rappresenta la scelta a priori per la modellizzazione dello spazio latente; generalmente viene scelta una distribuzione normale multivariata standard e la $q(\mathbf{z}|\mathbf{x})$ deve approssimare tale distribuzione durante la fase di addestramento. Per vincolare la forma di $q(\mathbf{z}|\mathbf{x})$ a quella di $p(\mathbf{z}|\mathbf{x})$ viene utilizzata la *KL divergency*:

$$KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = -\sum q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} \tag{38}$$

e sostituendo $p(z|x)$ con la 36 si ottiene:

$$\begin{aligned} KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) &= -\sum q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})q(\mathbf{z}|\mathbf{x})} \\ &= -\sum q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} + \sum q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x}) \end{aligned}$$

ma in entrambi i termini della somma la sommatoria è estesa sulle z, quindi:

$$\sum q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x}) = \log(\mathbf{x}) \sum q(\mathbf{z}|\mathbf{x}) = \log p(\mathbf{x})$$

perché $\sum q(\mathbf{z}|\mathbf{x}) = 1$.
Si arriva quindi al seguente risultato:

$$\log p(\mathbf{x}) = KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) + \sum q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \tag{39}$$

La sommatoria nell'equazione 39 prende il nome di *Variational lower bound* e viene indicata con la lettera $\mathcal{L}$.
A questo punto si deve osservare che $\mathbf{x}$ è fissato e quindi il termine sinistro dell'equazione 39 è una costante; di conseguenza minimizzare la *KL divergency* equivale a massimizzare la $\mathcal{L}$, che può essere riscritta in maniera semplificata:

$$\begin{aligned} \mathcal{L} &= \sum q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \\ &= \sum q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{x}|\mathbf{z}) + \sum q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \\ &= E_q \log p(\mathbf{x}|\mathbf{z}) - KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \end{aligned} \tag{40}$$

Quindi, ricapitolando, l'obiettivo iniziale è quello di trovare la distribuzione $p(\mathbf{z}|\mathbf{x})$ che però è molto complessa per essere calcolata; allora si cerca di approssimarla con una $q(\mathbf{z}|\mathbf{x})$ scelta tra un'opportuna famiglia e, per scegliere quella più vicina, si deve minimizzare $KL(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$, che come visto equivale a massimizzare la $\mathcal{L}$.
In figura 15 è possibile osservare in che modo si passa da $\mathbf{x}$ a $\mathbf{z}$ e viceversa.

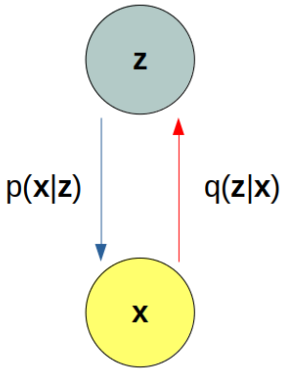

Figura 15: Illustrazione grafica della modalità attraverso la quale si ottiene il passaggio dalla variabile **x** osservabile alla **z** nello spazio latente e viceversa.

Il significato del termine di *KL divergency* che compare in $\mathcal{L}$ suggerisce che la distribuzione $q(\mathbf{z}|\mathbf{x})$ debba essere il più possibile simile ad una distribuzione $p(\mathbf{z})$ che può essere scelta e quindi si assume una distribuzione Normale multivariata standard, per la quale la KL divergency è più facile da calcolare.
L'altro termine in $\mathcal{L}$ è riconducibile ad un errore di ricostruzione, infatti il processo di decodifica, una volta campionato **z** è deterministico e quindi si ottiene che:

$$p(\mathbf{x}|\mathbf{z}) = p(\mathbf{x}|\hat{\mathbf{x}}) \tag{41}$$

dove $\hat{\mathbf{x}}$ è il pattern ricostruito. Inoltre se si considerano distribuzioni gaussiane si troverà che:

$$p(\mathbf{x}|\hat{\mathbf{x}}) \propto e^{-|\mathbf{x}-\hat{\mathbf{x}}|^2} \tag{42}$$

e quindi

$$\log p(\mathbf{x}|\hat{\mathbf{x}}) \propto -|\mathbf{x}-\hat{\mathbf{x}}|^2 \tag{43}$$

Quindi si osserva che l'autoencoder tende a minimizzare semplicemente $|\mathbf{x}-\hat{\mathbf{x}}|^2$, mentre il VAE tende a minimizzare la seguente quantità:

$$|\mathbf{x}-\hat{\mathbf{x}}|^2 + KL(q(\mathbf{z}|\mathbf{x})||N(\boldsymbol{\mu}, \boldsymbol{\Sigma})) \tag{44}$$

Nella pratica si costruisce la rete neurale che si occupa della fase di codifica in modo che restituisca i parametri della distribuzione Normale e quindi la media e la matrice di covarianza, che si impone essere diagonale per semplicità; da qui viene campionato un punto dello spazio latente a partire da tale distribuzione e avviato verso il decoder per ottenere il pattern ricostruito da confrontare con quello iniziale (nella fase di addestramento). Lo schema di questo processo è riportato in figura 16.

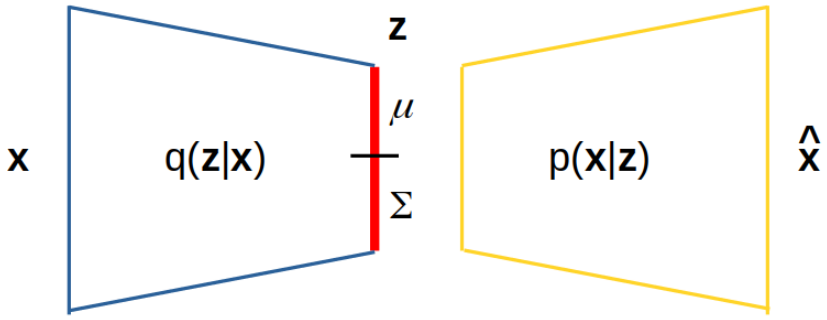

Figura 16: Schema di funzionamento del VAE, dove viene messo in evidenza che l'encoder è forzato a restituire come output i parametri della distribuzione Normale, ovvero la media e la matrice di covarianza.

Infine, per la fase di generazione di nuovi pattern, sarà sufficiente eliminare l'encoder e campionare dalla distribuzione che si è ottenuta alla fine dell'addestramento e fornire tale punto al decoder per costruire un nuovo pattern.

4 Ricerca di fisica Behind Standard Model con i VAEs

In quest'ultimo capitolo verrà presentata una possibile applicazione dei Variational Autoencoders nel campo della fisica delle alte energie, con lo scopo di ricercare segnali di nuova fisica BSM, cioè oltre il Modello Standard.
Come noto, gli esperimenti portati avanti al *Large Hadron Collider* hanno l'obiettivo di esplorare la fisica spingendosi sempre a più alte energie; attualmente, dopo la scoperta del *Bosone di Higgs*, il Modello Standard sembrerebbe essere completo, anche se rimango aperti alcuni problemi, come lo *Hierarchy Problem* e la spiegazione dell'origine della *Dark Matter*.
Nella ricerca di nuova fisica possono essere portati avanti due approcci, detti *model dependent* e *model independent*. Nel primo caso la ricerca di nuova fisica avviene con un particolare modello in mente ed i risultati sono ottimi nel caso in cui il modello utilizzato è corretto, come per la scoperta del Bosone di Higgs; il limite di una ricerca di questo tipo è chiaramente dovuto al fatto che i risultati sono strettamente legati alla bontà della teoria stessa. Dall'altro lato una ricerca model independent ha il pregio di non essere legata ad una particolare teoria fisica e quindi è capace di ricercare eventuali segnali di nuova fisica a prescindere da un modello teorizzato in anticipo.
Nelle pagine seguenti si cercherà di capire se è possibile addestrare un Variational Autoencoder sugli eventi di background (ovvero sulla fisica prevista dal Modello Standard) in modo che sia capace di rilevare eventuali segnali di nuova fisica come delle anomalie. L'approccio seguito è da un lato model dependent, nel senso che i dati utilizzati sono prodotti attraverso simulazioni Montecarlo in base alla SUSY (*Supersimmetry theory*) per la ricerca della coppia di particelle fermione/bosone (chargino e gluino), e dall'altro model independent, perché le masse di queste due particelle non sono stabilite e quindi la ricerca deve essere sensibile a tutte le varie combinazioni possibili. Successivamente, nel caso in cui l'algoritmo si dimostri efficace nella discriminazione del segnale usando le simulazioni MC, è ragionevole pensare di estendere l'applicazione di questo metodo direttamente sui dati sperimentali prodotti al Large Hadron Collider. Questo ulteriore passaggio è possibile solo in virtù dell'approccio *Unsupervised* per cui non è necessario dare all'algoritmo le etichette fondo/segnale durante la fase di addestramento (sarebbe infatti impensabile avere in anticipo questa informazione per i dati sperimentali). Per lo stesso motivo si capisce perché un algoritmo *Supervised* non possa essere in generale direttamente applicato ai dati sperimentali; infatti questa seconda categoria di modelli richiede nella fase di addestramento le etichette fondo/segnale per ciascun evento fisico al fine di impararne la distinzione. Si deve perciò ricorrere necessariamente alle simulazioni MC con tutte le incertezze modellistiche annesse.

4.1 Dataset

Per l'addestramento del modello e per la successiva fase di verifica sono stati utilizzati i dati prodotti attraverso simulazioni Montecarlo (MC), in base alla teoria di riferimento (SUSY). Le variabili fisiche che definiscono ogni evento sono otto (*met*, *mt*, *mbb*, *mct2*, *mlb1*, *lep1Pt*, *njet30*, *nBjet30-MV2c10*) e sono le stesse utilizzate nell'analisi fisica relativa allo studio [4]. Di conseguenza lo spazio iniziale, che dovrà essere compresso dal VAE, sarà 8-dimensionale.
Attraverso la simulazione MC vengono prodotti eventi sia di background che di segnale e, nel caso in cui l'algoritmo sia capace di discriminare tra eventi di fondo e di segnale, potrà essere applicato ai dataset reali, nei quali chiaramente non vi è questo tipo di differenziazione.
Prima di passare alla fase di codifica, gli eventi (sia di segnale che di background) sono stati sottoposti ad una serie di tagli di preselezione sulle variabili, come riportato nella tabella 1.

	Preselezione
Esattamente un leptone di segnale	Vero
met trigger	Vero
$2-3$ jets con $p_T > 30GeV$	Vero
b-tagged jet	[1-3]
met	> 220 GeV
mt	> 50 GeV
mbb	$[100-140]$GeV
mct	> 100GeV

Tabella 1: Sono riportati i tagli di preselezione applicati sia agli eventi di segnale che di background, prodotti attraverso una simulazione MC.

Gli eventi prodotti con il metodo MC sono stati divisi, come si richiede in un processo di apprendimento automatico, in una training data set, un validation data set ed un test data set. Successivamente, per rendere coerente la selezione del segnale simulando una raccolta dati con le attuali capacità e specifiche del LHC, gli eventi di validation e test sono stati ripesati per ottenere la giusta luminosità.
Per ottenere i nuovi pesi $w_j^{'}$ a partire da quelli validi prima dello split w_j, è stata usata la seguente formula [1]:

$$w_j^{'} = w_j \frac{\sum_i w_i \mathbf{1}\{y_i = fondo\}}{\sum_{i \in S'} w_i \mathbf{1}\{y_i = fondo\}} \tag{45}$$

Dove y_i è l'etichetta (segnale o fondo) dello i-esimo evento e $\mathbf{1}$ è la funzione indicatrice ($\mathbf{1}$ y_i = il fondo è zero per un evento di segnale e pari ad uno per un evento di fondo).

4.2 Architettura del modello

In questa sezione vengono fornite alcune specifiche tecniche sull'architettura del modello e sul processo di apprendimento.
Per quanto riguarda la struttura del VAE si specifica che il numero di neuroni negli strati nascosti è pari a cinquanta e che la dimensione dello spazio latente è pari a tre; inoltre la funzione di attivazione dei neuroni è la ReLU, cioè una funzione definita nel seguente modo:

$$f(x) = \max(0, x) \tag{46}$$

Parlando invece del processo di apprendimento, sono state impostate 2000 epoche, cioè tutti i dati vengono riproposti all'algoritmo per duemila volte; inoltre la *Loss Function* utilizzate è quella standard, composta da un termine legato all'errore di ricostruzione ed uno relativo alla *KL divergency* (moltiplicato per un fattore $\beta = 0.6$). Per il processo di discesa del gradiente è stato impostato inizialmente un *learning rate* pari a 0.003, con una diminuzione del 20% ogni qualvolta il modello non migliora per venti epoche consecutive; inoltre è stata impostata una *batch size* pari a 200, quindi l'aggiornamento dei pesi della rete avviene dopo aver cumulato l'errore su 200 eventi di training. Infine si sottolinea che il processo di addestramento termina o perché si è arrivati alle 2000 epoche o perché la *Validation Loss* non migliora per 50 epoche consecutive.
Il modello è stato sviluppato in Python usando le librerie Keras (tensorflow backend) ed il codice necessario all'implementazione è contenuto nel repository git al seguente link: https://github.com/robomorelli/vae_bachelor_thesis_code [11].

4.3 Addestramento del VAE

Come è stato detto nelle sezioni precedenti, il VAE deve essere addestrato in modo tale da rilevare eventuali indizi di fisica BSM come delle anomalie, cercando di dimostrarsi sensibile ad una ampia gamma di possibili segnali. Ma perché un tale compito non può essere svolto da un algoritmo di apprendimento supervisionato?
Un classificatore binario, ovvero un modello capace di discriminare fra due sole categorie (segnale e background), viene addestrato su un training data set i cui eventi di segnale sono generati facendo riferimento ad un determinato modello; tuttavia quando ne verranno presentati altri prodotti con diversi modelli, allora la classificazione risulterà totalmente arbitraria ed è qui che si evince il limite principale di una ricerca model dependent. Un ulteriore vantaggio del VAE ed in generale degli approcci model independent, rispetto a quelli model dependent, è quello di poter essere applicato direttamente sui dati come anticipato nella sezione introduttiva di questo capito (4). In questo modo si evitano quei problemi di incertezze modellistiche legate alle simulazioni montecarlo.
In linea con ciò che è già stato illustrato nel capitolo 3.10, gli eventi di segnale e di background, che sono stati prodotti attraverso simulazioni MC, sono rappresentabili in uno spazio 8-dimensionale. Durante il processo di addestramento del VAE gli eventi di background vengono compressi nello spazio latente (tridimensionale), decompressi per essere ricostruiti e poi confrontati con quelli iniziali per il calcolo dell'errore e quindi per dare il via al processo di backpropagation (3.6).
Dopo la fase di addestramento si verifica che il VAE abbia imparato come ricostruire gli eventi fisici di background usati per l'addestramento, dopo averli compressi nello spazio latente. A tal proposito vengono confrontate le distribuzioni originali date in input con

quelle rigenerate dalla rete. Allo stesso tempo, si ci aspetta che il modello commetta un errore di ricostruzione maggiore quando, invece che eventi di background, vengono dati in input eventi di segnale. La limitata capacità di generalizzare su questa nuova categoria di eventi mai visti durante la procedura di addestramento dovrebbe indurre il VAE ad una ricostruzione meno accurata. Come si vedrà è possibile allora usare la distribuzione dell'errore di ricostruzione per eventi di background e segnale per discriminare tra queste due tipologie di eventi.
Tra i contributi originali di questa tesi vi è anche la possibilità di pesare in maniera diversa il contributo che le diverse variabili fisiche che definiscono un evento possono apportare al processo di discriminazione e, per questo motivo, verranno presentati due casi: nel primo le otto variabili avranno tutte lo stesso peso, mentre nel secondo si proverà a capire se, dando maggiore importanza ad alcune di esse, si otterrà un processo di discriminazione più efficiente. E' infatti possibile che il VAE, concentrandosi su di una variabile più significativa per la ricostruzione del background ma non altrettanto per il segnale, possa sfruttare questa differenza per aumentare la forbice di errore di ricostruzione tra background e segnale. Rimane a questo punto da capire quali delle variabili possano aiutare in questo compito. Le possibilità sono almeno due: da un lato si possono sfruttare conoscenze o intuizioni teoriche riguardo il potere discriminante di alcune delle variabili, mentre dall'altro si può procedere in maniera brute-force(3.5) sulle possibili configurazioni dei pesi da dare alle diverse variabili.
In ogni caso, verranno illustrati i risultati ottenuti pesando tutte le variabili allo stesso modo.

4.4 Risultati

4.4.1 Processo di rigenerazione degli eventi

Come primo passo bisogna capire se, a seguito del processo di addestramento, il VAE è in grado di ricostruire gli eventi di background in maniera ottimale. Il risultato del processo di ricostruzione è riportato in figura 17, dove vengono confrontate le distribuzioni dei dati in input (punti blu) con quelle ricostruite dal VAE (punti rossi).

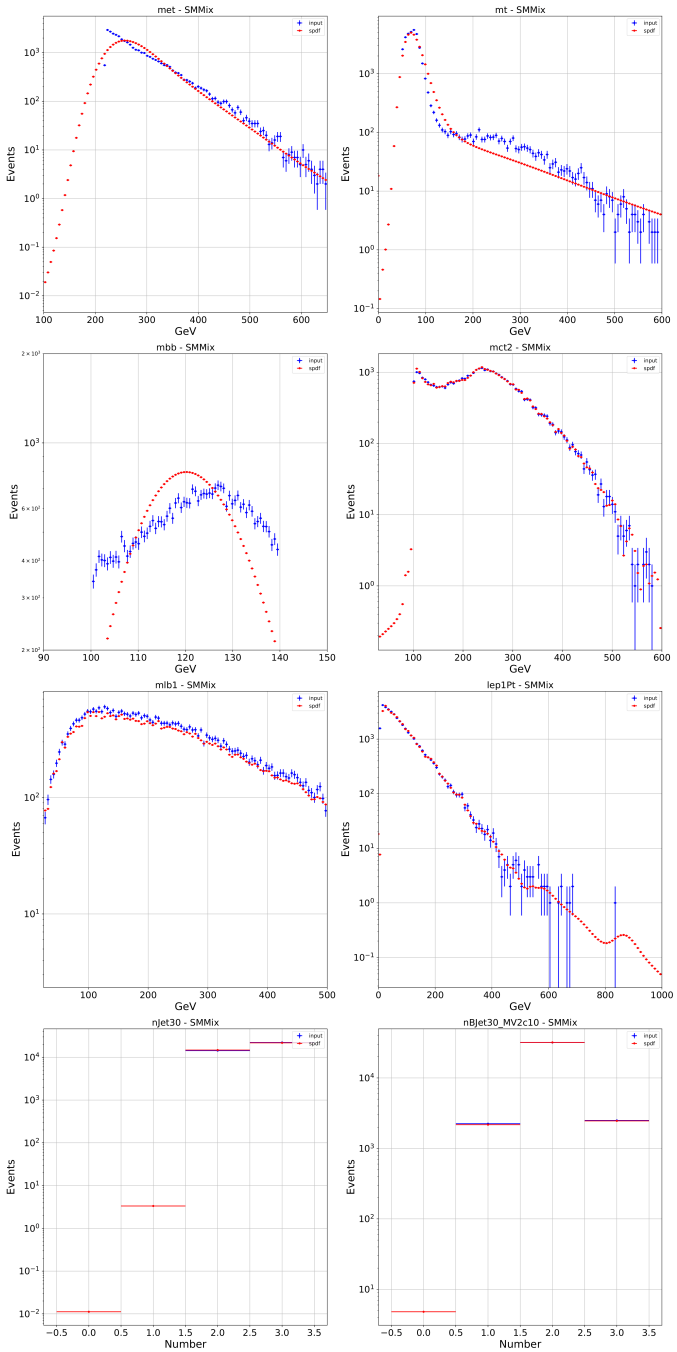

Figura 17: Confronto tra gli input in ingresso del VAE (in blu) e quelli ricostruiti (in rosso) per le otto componenti degli eventi di input relativi al background.

Dall'osservazione qualitativa della figura 17 emerge che il processo di ricostruzione del VAE risulta piuttosto accurato per tutte le variabili ad eccezione della *mbb*.
Tuttavia, come si vedrà più avanti, è possibile correggere questo risultato impostando un peso maggiore per tale variabile (ad esempio un peso pari a due o tre rispetto agli altri tutti pari ad uno). Inoltre, dopo aver constatato che è possibile indirizzare particolare attenzione sulla ricostruzione di specifiche variabili, si capirà come sfruttare questa situazione per ottenere un modello più sensibile ai vari tipi di segnale.

4.4.2 Distribuzione della loss di ricostruzione

Per rendere un segnale riconoscibile è opportuno che il suo indice di anomalia lo contraddistingua rispetto a quello degli eventi di fondo. In questa tesi la misura adottata per discriminare tra fondo e segnale è l'errore di ricostruzione che il VAE commette durante la ricostruzione degli eventi. Si chiarisce che questa non era l'unica scelta possibile in quanto si sarebbe potuto utilizzare anche la loss totale data dalla somma della divergenza di Kullback-Lieber e della loss di ricostruzione oppure, al contrario, solo la divergenza di Kullback-Lieber. Ad ogni modo, per ottenere l'errore di ricostruzione per ogni evento è sufficiente sommare quello sulle otto variabili. Da questi valori si ottiene quindi la distribuzione della *Loss*, come quella riportata in figura 18.

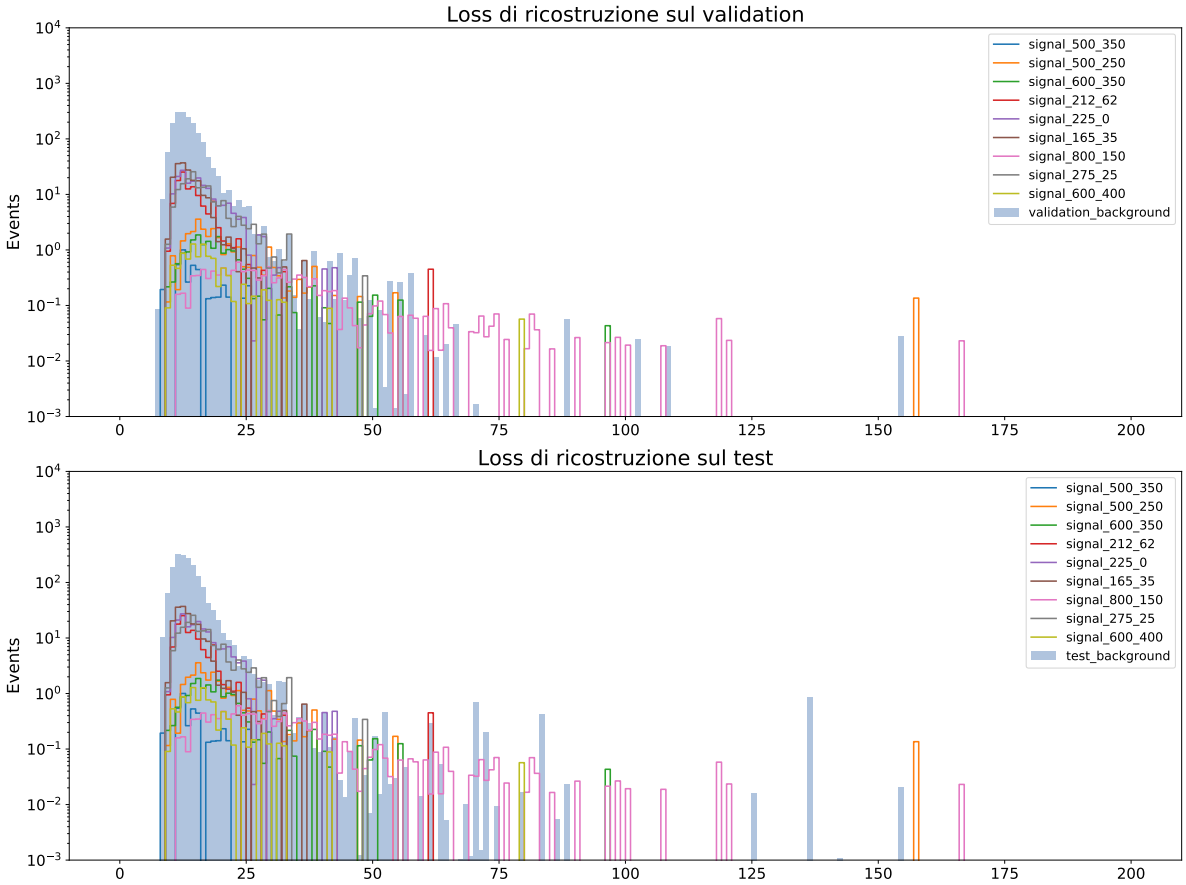

Figura 18: Distribuzione della *Loss* per gli eventi di background e per quelli di segnale relativi ad alcune combinazioni delle masse di Chargino-Gluino. La prima immagine è relativa al Vaidation data set, mentre la seconda al test data set.

In questa immagine, oltre alla distribuzione relativa agli eventi di background (istogramma blu) sia per il dataset di validation (sopra) che per quello di test (sotto), sono presenti anche le distribuzione relative ad alcune delle ipotesi di segnale (curve colorate) contraddistinte dalla diverse ipotesi sulle masse delle due particelle (chargino/neutralino). E' importante ricordare che le simulazioni MC degli eventi di segnale non sono mai state usate fino ad ora e solo in questa fase di test vengono date in input al modello.
Una situazione ideale prevede una distribuzione degli errori per gli eventi di fondo concentrata su valori della Loss inferiori rispetto alle analoghe distribuzioni relative agli eventi di segnale. In questo modo, infatti, la selezione del segnale risulta facilitata e il rapporto segnale/background aumenta.
Osservando le figure, una prima considerazione riguarda proprio questo aspetto. Infatti, la distribuzione della Loss per gli eventi di background presenta un picco spostato verso sinistra rispetto alle distribuzione relative agli eventi di segnale. Allo stesso tempo però si può già intuire che questa separazione non risulta molto netta, in particolare per alcuni tipi di segnale. Per questo motivo si cercherà, a partire dalle prossime sezioni, di trovare un modello che risulti più efficace in questo compito, sfruttando, ad esempio, una diversa pesatura delle variabili che caratterizzano un evento fisico. Con questo procedimento infatti si spera di dare la giusta importanza a quelle variabili che possono aumentare la distanza tra gli eventi di background e di segnale durante la ricostruzione.
Un altro aspetto non trascurabile e verificabile confrontando i due grafici della stessa figura 18, riguarda un fenomeno introdotto durante la parte compilativa di questa tesi: l'overfitting. Infatti, giudicando la distribuzione degli eventi di background nei due grafici, si può vedere come la forma sia pressoché la stessa, indicando come il modello non abbia imparato in modo particolare un solo dataset di dati (validation) senza poi generalizzare con altrettanta capacità sul campione di test mai visto durante la fase di addestramento.
Un'ultima osservazione può essere fatta pensando ad una possibile applicazione di questo algoritmo sui dati sperimentali, laddove la distinzione fra background e segnale non è nota a priori; infatti, osservando come le distribuzioni dei segnali si pongono rispetto a quella degli eventi di fondo, non è difficile credere che andando a selezionare gli eventi sulla coda destra della distribuzione della Loss relativa ai dati, si possano filtrare una serie di eventi interessanti dal punto di vista fisico.

4.4.3 Esperimento di conteggio e regione di esclusione

A questa prima analisi qualitativa ne viene fatta seguire una quantitativa, con la quale si vogliono definire per quali combinazioni delle masse delle due particelle il VAE riesce a discriminare, con una certa confidenza statistica, gli eventi di segnale da quelli di background. In questo modo sarà possibile delineare una regione di esclusione per i diversi segnali caratterizzati da diverse ipotesi circa le masse delle due particelle candidate.
Per portare a termine questo obiettivo si designa un esperimento di conteggio per confrontare l'ipotesi nulla di presenza di solo fondo nei dati con quella alternativa che prevede invece anche la presenza del segnale. Nello specifico si procede in questo modo: dopo aver determinato il numero di eventi di fondo N_b da selezionare, si ricava il valore di soglia della Loss relativa a questo numero. Successivamente, utilizzando lo stesso valore di soglia, si selezionano anche tutti gli eventi di segnale N_s alla destra di tale valore per poi procedere all'esperimento di conteggio. In figura 19 si riporta il caso in cui si vogliano selezionare 10 eventi SM. La soglia di selezione è quindi indicata dalla linea verticale della stessa immagine.

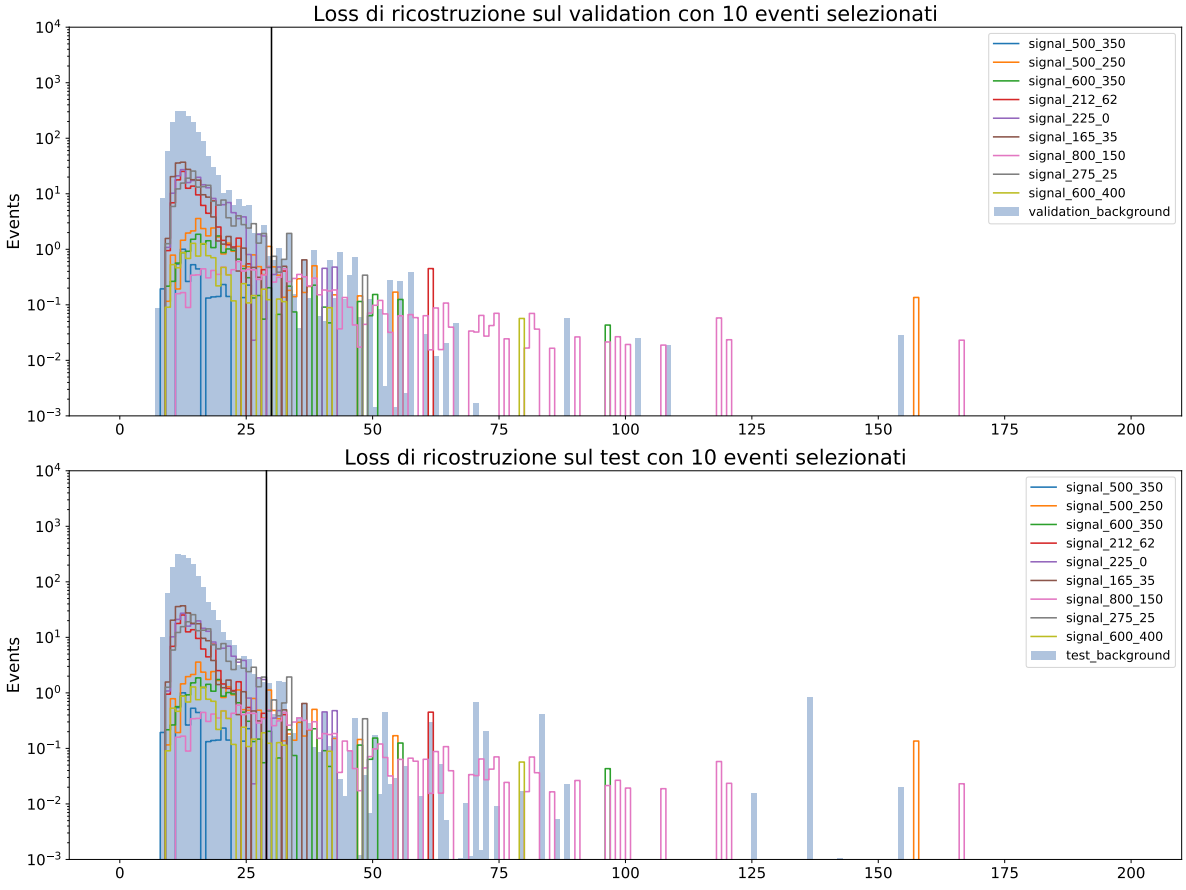

Figura 19: Figura analoga all 18, dove però la linea in nero mette in evidenza il processo di selezione degli eventi di background nella parte destra della distribuzione.

Quindi, dopo aver ricavato N_b ed N_s, si procede alla verifica dell'ipotesi nulla, ovvero la presenza di solo fondo nei dati; nello specifico si verifica che la probabilità di avere la somma $N_s + N_b$ di eventi sia compatibile con una distribuzione di Poisson centrata su di un valore pari a N_b (in altre parole si calcola $p(N_b + N_s|N_b)$). Nel caso in cui tale probabilità sia inferiore al 5%, si procede all'esclusione dell'ipotesi nulla in favore di quella

alternativa di presenza di segnale nei dati.
Nelle figure 20 e 21 sono riportati i risultati dell'esperimento di conteggio e rappresentano la cosidetta regione di esclusione dove, lungo l'asse delle ascisse sono riportate da chiedere mentre lungo quello delle ordinate le da chiedere . Questi risultati sono riportati al variare del numero N_b di eventi di fondo da selezionare per vedere se è in qualche modo possibile ottimizzare questo parametro in funzione della selezione del segnale. In altre parole si opera uno scan sui possibili valori N_b che tornerà utile nella prossima sezione.
Tornando alla descrizione di queste immagini, si osservano sia punti rossi che verdi. I primi rappresentano le particolari combinazioni delle masse delle due particelle per le quali il VAE è in grado di discriminare fra segnale e background con sufficiente confidenza statistica, mentre i punti verdi indicano quei segnali per cui tale discriminazione non può essere compiuta.
E' importante osservare come, aumentando il numero di eventi da selezionare nella zona di ipotesi con basse masse ($x < 300$), si evidenzia un incremento della sensibilità al segnale; l'opposto avviene invece per le zone con $x > 600$, mentre per la zona intermedia un buon risultato sembra emergere nel range $50 - 80$ eventi di fondo da selezionare.
In ogni caso scegliendo tre selezioni diverse, una per regione, è possibile designare un'unica regione di esclusione avendo ottimizzato la procedura nelle diverse regioni di massa. E' giusto rimarcare anche come questo scan sia stato condotto su dati di validation in quanto relativo ad un processo di ottimizzazione degli iperparametri e quindi riconducibile ad un fine-tuning del modello. Come tale non può essere condotto sugli stessi dati di test su cui i risultati finali verrano estratti. Infatti, solo i risultati finali della prossima sezione prendono in considerazione il test set per evitare possibili bias sugli esiti finali.

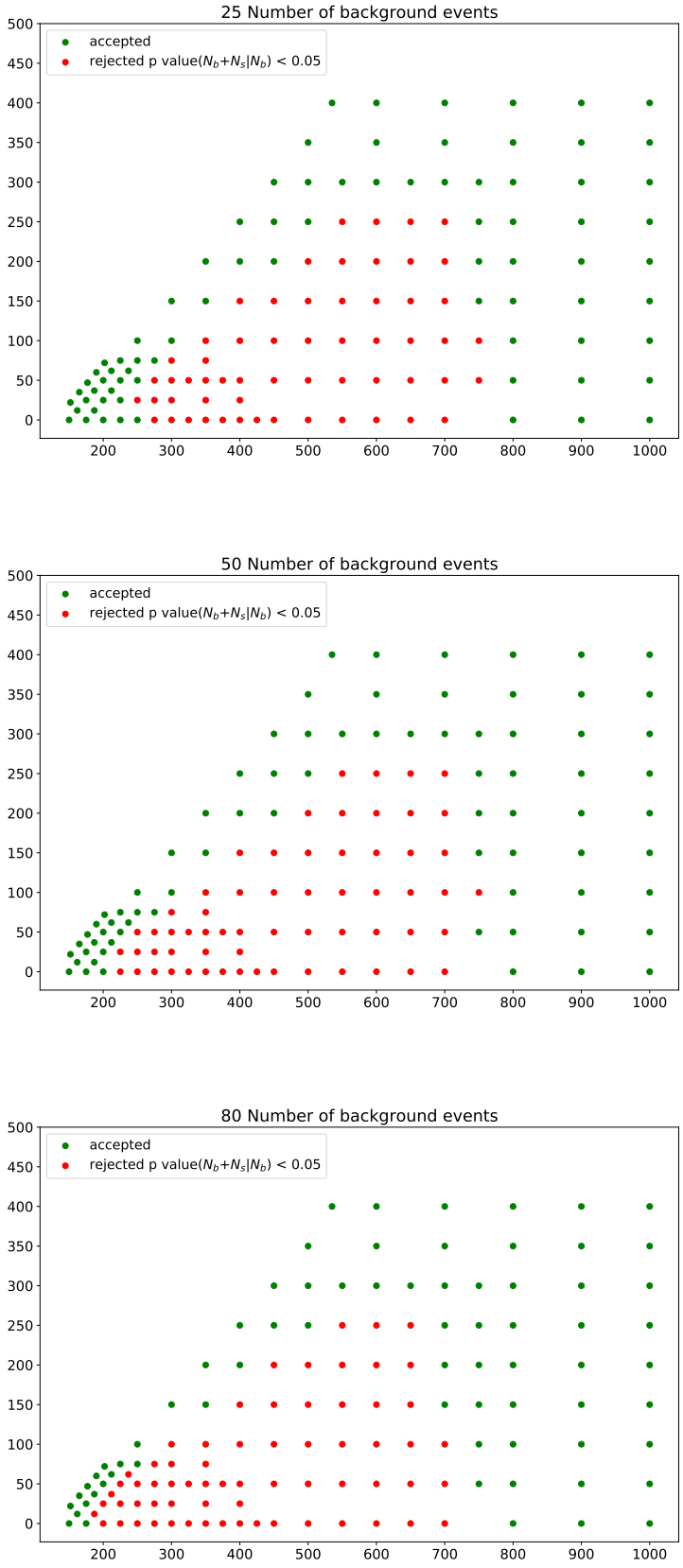

Figura 20: Risultati degli esperimenti di conteggio per, rispettivamente, 25, 50 e 80 eventi di background selezionati nella parte destra della distribuzione della Loss.

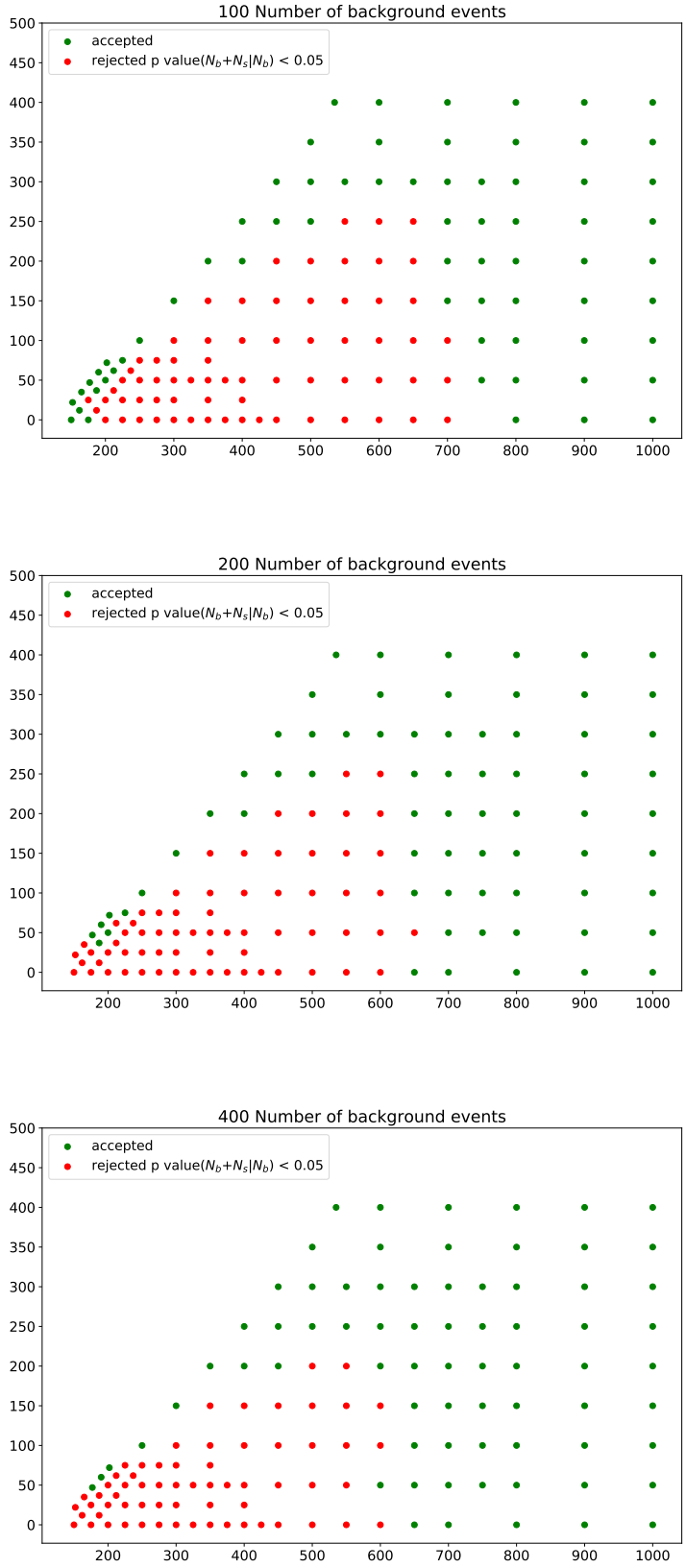

Figura 21: Risultati degli esperimenti di conteggio per, rispettivamente, 100, 200 e 400 eventi di background selezionati nella parte destra della distribuzione della Loss.

4.4.4 Regione di esclusione ottimizzata

A questo punto dopo aver individuato le tre zone lungo l'asse delle ascisse, rispettivamente per valori $x \leq 300$, $300 < x \leq 600$ e $x > 600$, è necessario ricavare il numero ottimale di eventi di background da selezionare in modo da ottimizzare la sensibilità agli eventi di segnale. Da un semplice conteggio dei punti evidenziati in rosso si ottiene che la combinazione ottimale prevede di selezionare 400 eventi di background per la prima zona, 100 per la seconda e 25 per la terza. In figura 22 viene riportato l'esito finale dell'esperimento, selezionando per ognuna delle tre zone il valore ottimale di eventi di background da selezionare.

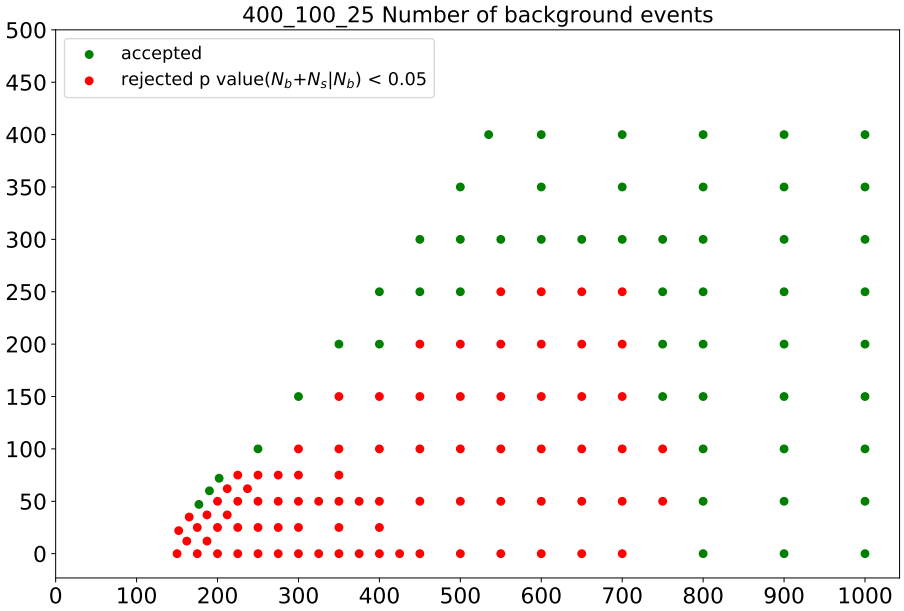

Figura 22: Risultato del processo di ottimizzazione nella distinzione fra background e segnale. Sono stati utilizzati i risultati ottimali in ciascuna delle tre zone individuate.

Quindi il lavoro può considerarsi concluso ma rimangono aperte un paio di domande:

1. La ricostruzione della variabile *mbb* non è ottimale; esiste un modo per renderla migliore?
2. E' possibile che nel processo di classificazione alcune variabili abbiano più importanza (siano più discriminanti) di altre? Se la risposta è affermativa, come si può mettere in evidenza questo fatto?

A queste domande si cercherà di rispondere nella prossima sezione.

4.4.5 Effetti della variazione dei pesi sulle variabili fisiche nel processo di apprendimento

Come visto nella sezione precedente, sono rimaste aperte un paio di domande alle quali si cercherà di dare una risposta. Per fare ciò bisogna provare a variare alcuni degli iperparametri del modello (già incontrati nella sezione 3.5). In particolare si vuole vedere in che modo cambia la sensibilità del modello agli eventi di segnale cambiando i pesi relativi alle diverse variabili che costituiscono un evento fisico. Nella sezione precedente si è fatta la scelta più ovvia, ovvero quella di impostare tutti i pesi uguali fra loro e pari ad uno, ma tale configurazione degli iperparametri ha dimostrato di non essere particolarmente soddisfacente.
In primo luogo è emerso che, nel processo di ricostruzione dei pattern da parte del VAE, il risultato per una delle otto variabili (*mbb*) non è stato soddisfacente. Per questo motivo si è provato ad impostare un peso maggiore per questa variabile ed il risultato è riportato in figura 23 (nello specifico è stato scelto un peso pari a tre per la variabile *mbb* e ad uno per le altre).

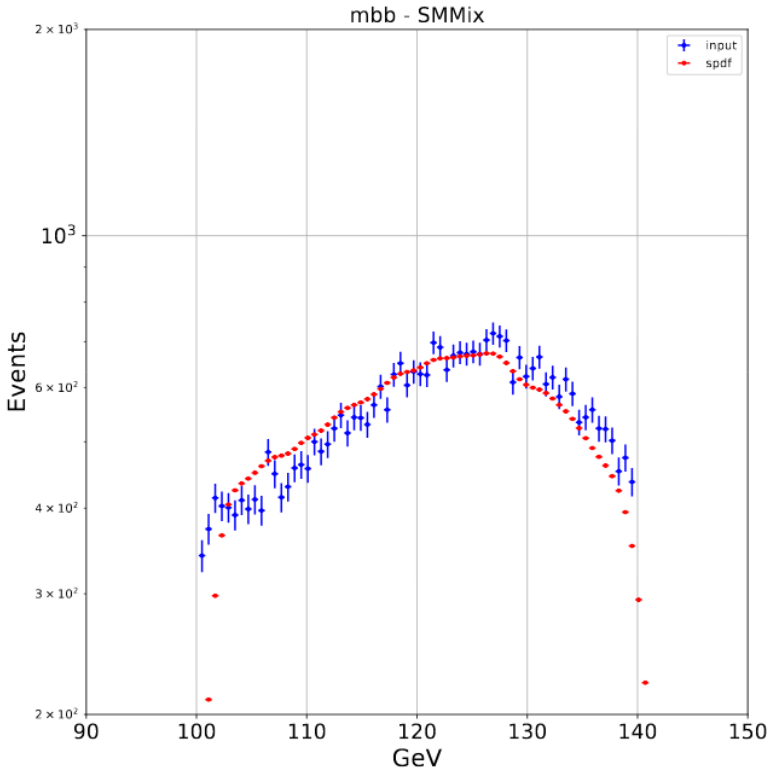

Figura 23: Esito del processo di ricostruzione della variabile *mbb* dopo aver impostato un peso pari a tre per tale variabile e mantenendo quelli delle altre variabili pari ad uno. In blu sono riportati i dati originali ed in rosso quelli ricostruiti.

Risulta evidente che questo piccolo accorgimento in fase di simulazione ha permesso di ottenere un'ottima ricostruzione della variabile *mbb*, mantenendo inalterata la qualità delle altre. Questo spunto suggerisce che è effettivamente possibile condizionare il modello sulla ricostruzione delle diverse variabili, di conseguenza ci si chiede se vincolando l'algoritmo su alcune grandezze fisiche particolari sia possibile ottenere un processo di discriminazione migliore.

Si passa quindi a verificare se nel processo di classificazione in segnale e background vi siano alcune variabili più discriminanti di altre; per far emergere ciò bisogna assegnare pesi diversi alle diverse variabili e osservare il conseguente risultato: emerge che ci sono effettivamente tre variabili più discriminanti delle altre, cioè *met*, *mt* e *mct2*). Nello specifico sono stati assegnati i pesi rispettivamente pari a 5,10,10 a queste tre variabili ed il risultato finale ottenuto è stato riportato in figura 24, per poter essere confrontato con il risultato in figura 22 dove i pesi erano tutti pari ad uno.

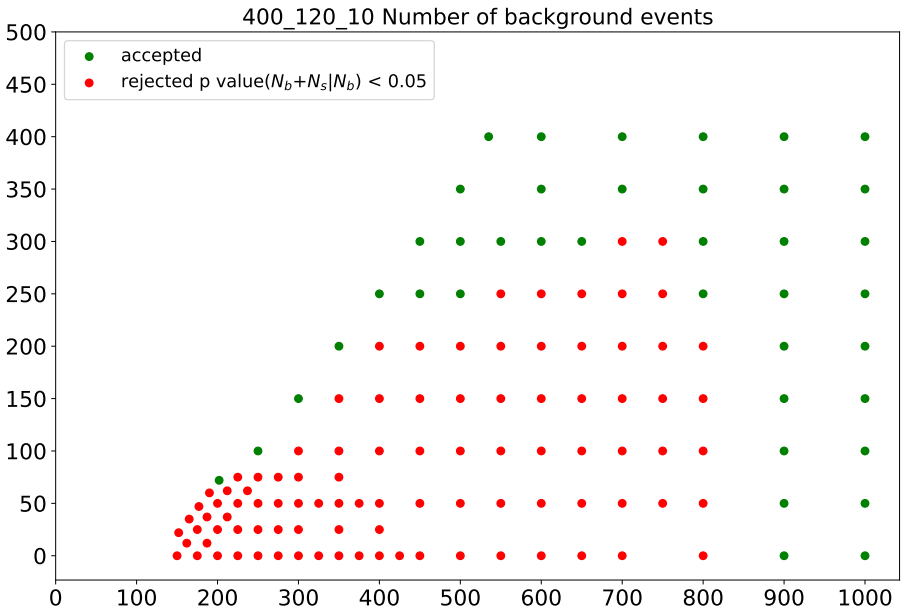

Figura 24: Risultato analogo a quello riportato in figura 22, ma utilizzando pesi differenti per le variabili più discriminanti.

Emerge dal confronto fra le due figure che quest'ultima configurazione di iperparametri permette la distinzione del segnale di background per un numero maggiore di possibili combinazioni delle masse delle due particelle. Nel primo caso con pesi tutti pari ad uno il numero totale di combinazioni delle masse delle particelle e quindi di modelli è 82, mentre in questo secondo caso si arriva a quota 96.
Quindi si giunge alla conclusione che, in questo caso specifico, pesare in maniere differente le variabili che costituiscono gli eventi permette di ottenere una sensibilità maggiore, ovvero il VAE riesce ad essere discriminante per un maggior numero di possibili eventi di segnale.

5 Conclusioni

In questa tesi è stato presentato un possibile approccio alla ricerca BSM, ovvero alla ricerca di nuova fisica oltre il Modello Standard. Sono state presentate le varie possibilità che si hanno a disposizione per la discriminazione degli eventi di segnale, ovvero quelli riconducibili a nuova fisica, e gli eventi di background, ovvero quelli riconducibili al Modello Standard già noto. Il più avanzato di questi approcci prevede l'utilizzo di algoritmi di apprendimento automatico (Machine Learning), dei quali è stata fatta un'ampia panoramica delle caratteristiche e delle metodologie più note ed utilizzate. In particolare ci si è focalizzati su un metodo di apprendimento non supervisionato, il Variational Autoencoder (VAE), per verificare se possa essere utilizzato nel processo di discriminazione fra segnale e background; per fare ciò il VAE è stato addestrato su pattern generati attraverso una simulazione Montecarlo ed i risultati finali sono stati assolutamente positivi. Nello specifico si è osservato che, a seguito del processo di addestramento sui dati di background, il VAE può essere utilizzato per la ricerca di eventi di segnale grazie al fatto che tali eventi, una volta ricostruiti, hanno un errore di ricostruzione tendenzialmente maggiore rispetto agli eventi di background. Di quest'ultimo aspetto è stata compiuta una dimostrazione qualitativa utilizzando la distribuzione della Loss ed una quantitativa, andando a verificare per quali combinazioni delle masse delle due particelle ricercate (Chargino e Gluino) il VAE fosse in grado di attuare la discriminazione.
In ultima analisi è stato effettuato un tentativo di ottimizzazione del processo tramite una variazione degli iperparametri, ovvero pesando in maniera diversa le variabili che compongono i pattern ed il risultato è stato incoraggiante perché è emerso che tale variazione degli iperparametri permette di rendere il VAE discriminante per delle combinazioni di masse per le quali precedentemente non lo era.
Quindi, per ciò che è stato appena detto, è stata dimostrata l'utilità del VAE per una ricerca model independent, basata però sulla SUSY. In conclusione, come possibili sviluppi futuri, si potrebbe verificare se il VAE così addestrato risulti discriminante anche per eventi di segnale riconducibili a teorie diverse dalla SUSY.

Riferimenti bibliografici

[1] Claire Adam-Bourdarios et al. "The Higgs boson machine learning challenge". In: *HEPML* (2014).

[2] Pushpa Bhat. "Advanced Analysis Methods in Particle Physics". In: *Annual Review of Nuclear and Particle Science* (2011).

[3] Samuel Rota Bulò. *Appunti di reti neurali*. URL: https://www.dsi.unive.it/~srotabul/files/AppuntiRetiNeurali.pdf.

[4] Alberto Cervelli et al. "Search for squarks, gluinos and electroweakinos in events with an isolated lepton, jets and missing transverse momentum at $\sqrt{s} = 13$ TeV with the ATLAS detector". In: (2018). URL: https://cds.cern.ch/record/2648719.

[5] ATLAS Collaboration. "Observation of a new particle in the search for the Standard Model Higgs bosonwith the ATLAS detector at the LHC". In: *Physics Letters B 716 (2012) 1–29* (2012).

[6] T. Del-Prete. *Methods of Statistical Data Analysis in High Energy Physics*. Istituto Nazionale di Fisica Nucleare, Sezione di Pisa, Italia, 2010.

[7] Jeremy Jordan. *Introduction to autoencoders*. 2018. URL: https://www.jeremyjordan.me/autoencoders/.

[8] Donald Knuth. *Knuth: Computers and Typesetting*. URL: https://towardsdatascience.com/using-3d-visualizations-to-tune-hyperparameters-of-ml-models-with-python-ba2885eab2e9.

[9] G. Ross L. Ibanez. "Low-energy predictions in supersymmetric grand unified theories". In: *Physics Letters B* (1981). URL: https://www.sciencedirect.com/science/article/abs/pii/0370269381912004?via%3Dihub.

[10] Nils J. Nilsson. *Introduction to Machine Learning*. Department of Computer Science, Stanford University, 1998.

[11] Filippo Schiazza Roberto Morelli. *Vae code git repository*. 2020. URL: https://github.com/robomorelli/vae_bachelor_thesis_code.

[12] Joseph Rocca. *Understanding Variational Autoencoders (VAEs)*. 2019. URL: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73.

[13] F. Wilczek S. Dimopoulos S. Raby. "Supersymmetry and the scale of unification". In: *Physical Review D* (1981). URL: https://journals.aps.org/prd/abstract/10.1103/PhysRevD.24.1681.

[14] H. Georgi S. Dimopoulos. "Softly broken supersymmetry and SU(5)". In: *Nuclear Physics B* (1981). URL: https://www.sciencedirect.com/science/article/pii/0550321381905228?via%3Dihub.

[15] N. Sakai. "Naturalness in supersymmetric GUTS". In: *Zeitschrift für Physik C Particles and Fields* (1981). URL: https://link.springer.com/article/10.1007%2FBF01573998.

[16] Emily Smith. *The Hierarchy Problem*. 2019. URL: http://theory.uchicago.edu/~sethi/Teaching/P445-S2019/Emily_Smith_QFT_III_Final_Paper.pdf.