

Università degli Studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Magistrale in
Scienze Statistiche



**CLASSIFICAZIONE ROBUSTA AGLI ERRORI SISTEMATICI IN
APPLICAZIONI ALLA FISICA DELLE PARTICELLE**

Relatore Prof. Tommaso Dorigo
Dipartimento di Scienze Statistiche
INFN - Sezione di Padova

Correlatore Dott. Lukas Layer
Dipartimento di Fisica dell'Università degli Studi di Napoli Federico II
INFN - Sezione di Padova

Laureanda: Dal Sasso Benedetta
Matricola N. 2018899

Anno Accademico 2021/2022

Indice

Introduzione	5
1 La classificazione per l'estrazione di un segnale	7
1.1 Modelli a più componenti	8
1.2 Il problema di classificazione come riduzione della dimensionalità	10
1.3 Il problema degli errori sistematici	14
1.3.1 Approccio di INFERNO	16
1.4 Altri approcci per ridurre l'impatto delle sistematiche	20
2 Applicazione di INFERNO a dati sintetici	23
2.1 Il modello sintetico	23
2.2 Studi sugli effetti sistematici sul background	26
2.3 Performance sulla frazione di segnale	34
3 Ricerca di segnali in fisica delle particelle	37
3.1 Il modello standard	37
3.2 L'apparato sperimentale	42
3.2.1 LHC	42
3.2.2 CMS	46
3.2.3 Acquisizione dei dati	50
3.2.4 La sezione d'urto del top	52

4 Applicazione di INFERNO ai dati di CMS	55
4.1 Dati sperimentali	56
4.2 L'analisi pubblicata	57
4.3 I parametri di disturbo	60
4.4 Analisi sui dati di Asimov	62
4.4.1 Analisi sulla porzione di dati	65
4.4.2 Analisi sui parametri della rete neurale	70
Conclusioni	79
Riferimenti	81
A Codice dati sintetici	85
B Codice dati di CMS	111

Introduzione

Il lavoro presentato in questa tesi costituisce un contributo al progetto di ricerca condotto da Tommaso Dorigo, Giles Strong e Lukas Layer in merito alla procedura INFERNO (*Inference-Aware Neural Optimization*) proposta da De Castro e Dorigo 2019.

La ricerca di nuova fisica tramite l'acceleratore di particelle LHC si serve di reti neurali per classificare eventi di segnale e di fondo e costruire delle statistiche sommario che riducono la dimensione delle caratteristiche degli eventi osservati per utilizzarle come input per l'inferenza. L'aspetto innovativo di INFERNO sta nel fatto che l'imperfetta conoscenza delle proprietà del segnale e del fondo, rappresentata dai parametri di disturbo, viene considerata, diversamente dall'approccio tradizionale, già durante la fase di addestramento del classificatore, aumentando così la potenza statistica della statistica sommario usata nell'inferenza.

L'obiettivo cardine del lavoro qui presentato è quello di approfondire le proprietà di INFERNO attraverso applicazioni a problemi di fisica delle particelle (HEP) con dati sintetici e dati veri, dando un apporto alle possibili future applicazioni dell'algoritmo in quest'ambito.

Vengono condotti studi con diversi setup per capire in quali condizioni INFERNO può ridurre considerevolmente l'impatto delle incertezze sistematiche. Tutto il lavoro è condotto tramite sviluppo di codice appropriato per i fini preposti, a partire dall'implementazione di INFERNO in PYTORCH di Strong 2021.

Nel Capitolo 1 si introduce il problema in HEP della classificazione e dell'inferenza statistica sulla misura della frazione di segnale. Si espone il problema degli errori sistematici e di come la procedura INFERNO lo risolve.

Il lavoro sui dati sintetici, presentato nel Capitolo 2, è prezioso in quanto permette di confrontare le performance delle procedure sotto diverse condizioni controllate. Queste analisi permettono di mostrare che INFERNO supera l'approccio classico basato sulla *binary cross entropy* (BCE) nel problema della stima dell'intervallo di confidenza del numero di eventi di segnale in presenza di parametri di disturbo. Le analisi che si conducono in questo lavoro evidenziano che all'aumentare della larghezza delle distribuzioni delle sistematiche corrisponde un aumento dell'errore sul parametro di interesse, ma INFERNO risulta essere molto più robusto rispetto a BCE ai cambiamenti delle incertezze delle sistematiche. Inoltre i test condotti permettono di affermare che il training di INFERNO, al fine di ottenere la più alta precisione possibile sulla misura finale, decorrela maggiormente i parametri di disturbo con quello di interesse rispetto a quanto faccia BCE e, al contempo, incoraggia, anche se meno, ad essere precise le misure dei disturbi e indipendenti tra loro.

Nel Capitolo 3 si presenta il Modello Standard della fisica delle particelle e l'apparato sperimentale LHC situato al CERN di Ginevra per la ricerca di nuova fisica tramite collisioni protone-protone. Gli aspetti chiave della fenomenologia delle collisioni ad LHC sono ritenuti utili per comprendere l'analisi sui dati veri presentata successivamente.

Infine il Capitolo 4 riporta un'estensione dell'algoritmo INFERNO a dati veri di CMS Open Data per la misura della sezione d'urto del quark top (CMS Collaboration 2013). In quest'ambito ci si concentra sullo studio dell'effetto della porzione di dati tra quelli disponibili sulle performance delle procedure e l'impatto dei parametri di tuning della rete neurale sull'efficienza delle stime. Gli esiti di queste analisi hanno dato un apporto significativo all'analisi completa su questi dati svolta da Lukas Layer (Layer 2022).

Capitolo 1

La classificazione per l'estrazione di un segnale

Le tecniche di apprendimento automatico, in particolare l'apprendimento supervisionato, sono sempre più utilizzate nell'analisi sperimentale della fisica delle particelle al Large Hadron Collider (LHC). La maggior parte degli algoritmi di apprendimento supervisionato applicati agli esperimenti di collisione possono essere visti come un modo per approssimare le variabili latenti del modello generativo dei dati, basato su osservazioni simulate. Queste approssimazioni di variabili latenti sono spesso molto informative sui parametri di interesse e possono essere utilizzate per costruire statistiche riassuntive delle osservazioni che permettono di effettuare l'inferenza in modo efficiente.

Le analisi dei dati ad LHC hanno l'obiettivo di estrarre informazioni dalle grandi quantità di dati ad alta dimensionalità sulle particelle subatomiche prodotte dalla collisione energetica di protoni, acquisiti da rivelatori altamente complessi costruiti attorno al punto di collisione.

La modellazione accurata dei dati è disponibile solo tramite la simulazione Monte Carlo di una complicata catena di processi fisici che va dalla riproduzione dell'interazione fondamentale sottostante, alla definizione delle interazioni delle particelle con gli elementi del rivelatore, fino alla lettura e il salvataggio dei dati. Nella maggior parte dei casi si è interessati all'inferenza su un sottoinsieme dei parametri del modello generativo dei dati che per essere efficiente deve tener conto dell'incertezza del modello e della possibile errata

specificazione dei restanti parametri, chiamati di disturbo.

L'obiettivo principale dell'elaborato è quello di esporre ed applicare INFERNO, un metodo che definisce delle statistiche riassuntive non lineari minimizzando una funzione di perdita, tramite discesa del gradiente stocastico, per fornire la più piccola incertezza inferenziale per i parametri di interesse. Come caso d'uso, viene considerato il problema della stima dell'intervallo di confidenza per il coefficiente di mistura in un modello multidimensionale bicomponente: segnale e background.

1.1 Modelli a più componenti

Le osservazioni che provengono dagli esperimenti di collisione fra particelle elementari si possono pensare come delle variabili casuali X , con una certa distribuzione caratterizzata dai parametri di interesse. Si vuole dunque modellare la densità di probabilità che genera una data osservazione \mathbf{x}_i condizionata ai parametri, ossia $p(\mathbf{x}_i|\theta)$, dove θ denota tutti i parametri di interesse e che influenzano il risultato del rilevatore.

Il processo sottostante che genera x può essere trattato come un modello di mistura, ossia come la composizione probabilistica di campioni da più distribuzioni, corrispondenti a diversi tipi di processi di interazione che avvengono nella collisione.

Conoscendo la funzione di distribuzione probabilistica di ogni componente della mistura $p_j(x|\theta)$, $p(x|\theta)$ può essere espressa come:

$$p(x|\theta) = \sum_{j=0}^{K-1} \phi_j p_j(x|\theta)$$

dove K è il numero di componenti della mistura e ϕ_j sono i pesi, ossia le probabilità, per ogni osservazione, di essere originata dalla componente j della mistura.

Tale modello si può descrivere anche tramite la sua definizione generativa:

$$z_i \sim \text{Categorical}(\phi) \rightarrow x_i \sim p_{z_i}(x|\theta)$$

che descrive un campionamento casuale di una $z_i \in \{0, \dots, K - 1\}$ da una distribuzione categoriale con probabilità $\phi = \{\phi_0, \dots, \phi_{K-1}\}$ e un successivo campionamento di una x_i dalla corrispondente distribuzione della mistura.

Spesso, si è interessati a studiare un sottoinsieme S , di tutti gli H processi di interazione prodotti dai collider, chiamato set di segnale (S). Questo può essere un singolo tipo di processo fisico, o diversi, che possono essere visti dunque, come una componente della mistura. Possiamo di conseguenza definire il sottoinsieme di background o fondo, il sottoinsieme $B = H - S$, come il risultato di tutti gli altri processi generati che non sono di interesse. Tale distinzione tra i processi generatori di interesse S e di fondo B è alla base di ogni analisi a LHC ed è motivata dal fatto che piccoli cambiamenti dei parametri dello *Standard Model* (SM) o delle sue estensioni teoriche, influenzano solo un sottoinsieme dei processi prodotti, quelli che sono governati dalle interazioni legate al parametro.

Di fatto, normalmente l'inferenza statistica a LHC non viene effettuata direttamente sui parametri dello SM o dell'estensione studiata, ma sulla frequenza relativa dell'insieme dei processi di interesse ϕ_S o sulle proprietà della sua distribuzione $p_S(x|\theta)$.

Data la struttura del modello di mistura prevista per $p(x|\theta)$ e il fatto che si è interessati solo a una piccola parte dei processi generati e rilevati per ogni collisione che in generale ha $\phi_S \ll \phi_B$, deve essere considerato l'effetto dei trigger o di qualsiasi altro strumento di selezione degli eventi che viene utilizzato in tutti gli apparati di collisione. Il ruolo della selezione degli eventi, svolto dal trigger, è quello di ridurre la frazione di eventi che non contengono informazioni utili per il compito di inferenza di interesse. La selezione del trigger può essere pensata come un requisito tecnico: riducendo il tasso totale di quantità osservabili (reads) registrate dal rivelatore si soddisfano dei requisiti per l'acquisizione dei dati. Lo scopo della selezione è quello di ridurre il contributo atteso dei processi di fondo che non sono ben modellati dalla simulazione, così come quello di aumentare la frazione attesa di eventi di segnale.

1.2 Il problema di classificazione come riduzione della dimensionalità

Lo spazio delle misure del rilevatore x di cui sopra abbiamo assunto una distribuzione mistura, ha un'alta dimensionalità che complica il suo diretto utilizzo in qualsiasi tipo di inferenza statistica. L'alta dimensionalità dello spazio delle misure grezze del rivelatore $x \in X$ rende molto difficile specificare un'efficace selezione di eventi che sia in grado di ridurre i contributi dei processi di fondo non interessanti o non ben modellati. Questo motiva la necessità di costruire una statistica riassuntiva a bassa dimensionalità per eseguire l'inferenza, che si traduce nell'uso di una funzione di riduzione della dimensionalità $f(x) : X \rightarrow Y$, dallo spazio delle misure grezze del rivelatore $X \subseteq R^d$ a uno spazio di dimensione inferiore $Y \subseteq R^b$.

Denotiamo con $\mathbf{y} \in Y$ il risultato della trasformazione $f(x)$ applicata alla misurazione del rilevatore; se la funzione f è differenziabile e biettiva, la funzione di densità di y si può scrivere così:

$$p(y|\theta) \left| \det \frac{dx}{dy} \right|$$

dove il secondo termine corrisponde al determinante dello Jacobiano dell'inversa di f .

I metodi di ricostruzione degli eventi forniscono un modo molto efficiente per trasformare le misurazioni ad alta dimensionalità del rivelatore in uno spazio che può essere più facilmente interpretato da un punto di vista fisico. Infatti, la ricostruzione può essere vista come una complessa tecnica procedurale di inferenza su un sottoinsieme delle variabili latenti data la misura x di un evento. Questi metodi tentano di ripercorrere la catena generativa dell'osservazione per recuperare il sottoinsieme delle sue caratteristiche interessanti, per fornire un riassunto compresso delle informazioni portate dall'evento sui parametri di interesse θ .

A causa del rumore e delle caratteristiche del rivelatore, la funzione di ricostruzione non può recuperare completamente le caratteristiche dell'evento. Questo per esempio è il caso dei neutrini che passano dal rilevatore senza lasciare alcun segnale rilevabile. Gli eventi simulati possono quindi essere usati

per fare affermazioni probabilistiche calibrate degli oggetti fisici ricostruiti e la loro relazione con le particelle non osservate che passano attraverso il rivelatore.

L'identificazione delle particelle può essere pensata come un'inferenza di variabili latenti, il che equivale ad usare le informazioni aggiuntive del detector intorno a un oggetto per misurarne le proprietà più precisamente.

Alcune delle più recenti tecniche di apprendimento automatico, come le reti neurali, potendo trattare input di dimensioni variabili, forniscono un nuovo modo per affrontare il menzionato problema della riduzione della dimensionalità. La combinazione dei suoi output in una statistica riassuntiva di dimensione inferiore, che può essere una funzione di ogni singola osservazione o un insieme di osservazioni multiple, permette di costruire modelli statistici più semplici che mettono in relazione i parametri di interesse con le osservazioni.

Nella fisica delle alte energie un riassunto unidimensionale, come l'output di un classificatore, viene spesso utilizzato per l'inferenza statistica sui parametri di interesse attraverso la costruzione di una funzione di verosimiglianza *binned*. Intuitivamente, l'obiettivo di classificazione potrebbe essere dichiarato come la separazione dei risultati del rivelatore provenienti da processi che contengono informazioni sui parametri di interesse da quelli che non lo fanno, che corrispondono al segnale e al background.

Le due classi sono spesso non separabili, cioè un dato risultato del rivelatore x (o qualsiasi funzione di esso) potrebbe essere stato prodotto sia da processi di segnale che di fondo, e si possono fare solo affermazioni probabilistiche dell'appartenenza ad una certa classe.

Al fine di utilizzare tecniche di apprendimento automatico supervisionato per classificare i risultati del rivelatore, sono necessari campioni etichettati, ma, per i dati raccolti, solo la misura x del rivelatore è nota. Possono invece essere utilizzati come dati di training osservazioni realistiche simulate, generate specificamente per modellare gli eventi di un dato set di processi (segnale e fondo), dove la variabile latente categorica z_i che rappresenta un dato insieme di processi, può effettivamente essere usata come etichetta di classificazione.

In aggiunta c'è da considerare che se il modello del simulatore è specificato male, ad esempio a causa dell'effetto di incognite, i classificatori risultanti verrebbero addestrati per ottimizzare l'obiettivo di classificazione per distribuzioni diverse da quelle vere.

La struttura mista del modello statistico appena discusso per il risultato delle collisioni facilita il suo inquadramento come un problema di classificazione con obiettivo di stimare efficacemente un'approssimazione di $t_b(\mathbf{x}) = p_s(\mathbf{x})/(p_s(\mathbf{x}) + p_b(\mathbf{x}))$, che varierà in modo monotono con il rapporto di verosimiglianza. Per definire con più precisione il problema nel più ampio obiettivo dell'inferenza statistica di un sottoinsieme di parametri di interesse in un modello mistura, consideriamo il modello a due componenti. La prima componente è il segnale $p_s(x|\theta)$, la seconda il background $p_b(x|\theta)$, mentre θ identifica tutti i parametri da cui le distribuzioni potrebbero dipendere. La funzione di distribuzione di probabilità della mistura può essere espressa come:

$$p(\mathbf{x}|\mu, \boldsymbol{\theta}) = (1 - \mu)p_b(\mathbf{x}|\boldsymbol{\theta}) + \mu p_s(\mathbf{x}|\boldsymbol{\theta})$$

dove μ è il parametro corrispondente alla frazione di segnale nella mistura e sarà l'unico parametro di interesse. Infatti, in misure di sezione d'urto di produzione di specifici processi fisici, l'osservabile di interesse è proporzionale al coefficiente di mistura del segnale del modello statistico appena definito. Si può usare anche la parametrizzazione che prevede come coefficienti il numero atteso di eventi per il segnale e il background rispettivamente, purché b sia noto e fisso ed s sia l'unico parametro di interesse.

Un modo per approssimare il rapporto di densità $r(x)$ tra due distribuzioni arbitrarie $p(x)$ e $q(x)$ è quello di addestrare un classificatore, ad esempio una rete neurale che ottimizza l'entropia incrociata. Se i campioni da $p(x)$ sono etichettati come $y = 1$, mentre $y = 0$ è usato per osservazioni da $q(x)$, il rapporto di densità può essere approssimato come:

$$\frac{s(x)}{1 - s(x)} \approx \frac{p(y = 1|x)}{p(y = 0|x)} = \frac{p(y = 1|x)p(y = 1)}{p(y = 0|x)p(y = 0)} = r(x) \frac{p(y = 1)}{p(y = 0)}$$

così $r(x)$ può essere approssimato da una semplice funzione del classificatore addestrato direttamente dal campione osservato. Il fattore $p(y = 1)/p(y = 0)$ è indipendente da x e può essere semplicemente stimato come il rapporto tra il numero totale di osservazioni di ogni categoria nel set di dati di addestramento, cioè uguale a 1 se quest'ultimo è bilanciato.

Un noto risultato della statistica classica, il lemma di Neyman-Pearson, stabilisce che il rapporto di verosimiglianza $\Delta(\mathbf{x}) = p(\mathbf{x}|H_0)/p(\mathbf{x}|H_1)$ è il test

più potente quando sono considerate due ipotesi semplici: H_0 , quella nulla e H_1 , quella alternativa.

Nel modello mistura a due componenti si pone sotto H_0 il caso in cui ci sia solo il contributo del background $p(x|\mu = 0, \theta)$ mentre sotto H_1 si pone una certa mistura di segnale e fondo $p(x|\mu = \mu_0, \theta)$, dove μ_0 è fissato. Per il momento, gli altri parametri di distribuzione θ saranno assunti noti e fissati agli stessi valori per entrambe le ipotesi. Il rapporto di verosimiglianza in questo caso può essere espresso come:

$$\Delta(D; H_0, H_1) = \prod_{x \in D} \frac{p(x|H_0)}{p(x|H_1)} = \prod_{x \in D} \frac{p(x|\mu = 0, \theta)}{p(x|\mu = \mu_0, \theta)}$$

dove il fattore $p(x|\mu = 0, \theta)/p(x|\mu_0, \theta)$ potrebbe essere approssimato dall'output di un classificatore probabilistico addestrato a distinguere le osservazioni provenienti da $p(x|\mu = 0, \theta)$ da quelle provenienti da $p(x|\mu = \mu_0, \theta)$.

Se si considera invece l'inverso del rapporto di verosimiglianza Δ , ogni termine del fattore è proporzionale al seguente rapporto:

$$\Delta^{-1} \sim \frac{p(x|H_1)}{p(x|H_0)} = \frac{(1 - \mu_0)p_b(x|\theta) + \mu_0 p_s(x|\theta)}{p_b(x|\theta)}$$

che a sua volta può essere espresso come

$$\Delta^{-1} \sim (1 - \mu) \left(\frac{p_s(x|\theta)}{p_b(x|\theta)} - 1 \right)$$

quindi ogni fattore del rapporto di verosimiglianza è una funzione biettiva del rapporto $p_s(x|\theta)/p_b(x|\theta)$.

Dato che le quantità $p(x|H_0)$ e $p(x|H_1)$ non sono disponibili, nella pratica, dopo aver specificato un certo μ_0 vengono utilizzati campioni simulati di $p(x|\mu = \mu_0, \theta)$ per ottenere un'approssimazione del rapporto di verosimiglianza, attraverso il classificatore. Da un punto di vista di inferenza statistica dunque, l'apprendimento automatico supervisionato definito come la classificazione del segnale rispetto al fondo, può essere visto come un modo per approssimare il rapporto di verosimiglianza direttamente dai campioni simulati.

Si deve inoltre tener conto, come già detto, dei parametri di disturbo: il vero rapporto di densità tra segnale e fondo dipende da qualsiasi parametro θ

che modifica la densità di probabilità del segnale $p_s(x|\theta)$ e del fondo $p_b(x|\theta)$. Quindi l'approssimazione usando l'apprendimento automatico può diventare complicata. In pratica, il classificatore può essere addestrato per il valore più probabile dei parametri di disturbo e il loro effetto può essere adeguatamente inserito durante la calibrazione, ma l'inferenza risultante sarà degradata. La fase di calibrazione viene di solito effettuata utilizzando un istogramma e un set di dati di osservazioni simulate, costruendo effettivamente una probabilità sintetica dell'intero gamma di output del classificatore oppure viene definito a priori il numero di eventi osservati dopo il taglio nel classificatore.

Il metodo di apprendimento automatico INFERNO costruisce statistiche riassuntive che ottimizzano direttamente la quantità di informazioni prevista sul sottoinsieme di parametri di interesse utilizzando campioni simulati e tenendo conto in modo esplicito e diretto dell'effetto dei parametri di disturbo. INFERNO consente di permettere così di riallineare la riduzione dimensionale all'obiettivo di misura precisa del parametro di interesse (sezione d'urto o parametro della mistura).

1.3 Il problema degli errori sistematici

Mentre l'uso dei classificatori per definire una statistica può essere efficace per aumentare la sensibilità del segnale, le simulazioni utilizzate per generare i campioni necessari per addestrare il classificatore spesso dipendono da ulteriori parametri incerti, quelli che abbiamo definito come parametri di disturbo.

Questi parametri non sono di interesse immediato, ma devono essere presi in considerazione per poter formulare dichiarazioni quantitative sui parametri del modello sulla base dei dati disponibili. Le statistiche riassuntive basate sulla classificazione non possono tenere conto facilmente di tali effetti, quindi quando vengono presi in considerazione i parametri di disturbo la potenza inferenziale della procedura viene degradata.

Tali parametri devono essere visti come strumenti necessari per fare inferenze statistiche imparziali e calibrate quando non si dispone di un modello perfettamente noto dei dati. Questo aspetto è strettamente legato al disallineamento tra l'obiettivo delle analisi di fisica delle particelle, ossia quello di dedur-

re informazioni sui modelli generatori dei dati, e gli obiettivi di classificazione degli approcci di apprendimento supervisionato.

Un modo diretto per tenere conto dell'effetto dei parametri di disturbo nella costruzione di una statistica riassuntiva è di includerli nel modello fisico attraverso una parametrizzazione del loro effetto sulle caratteristiche osservabili. Questo richiede l'inserimento nel modello della loro funzione di densità (PDF) nota a priori, o aggiunta tramite una misura accessoria; ciò può essere o meno pratico da implementare a seconda dal problema.

Nei casi in cui i dati sperimentali siano informativi sul valore dei parametri di disturbo, si può cercare di sfruttare questa dipendenza nella costruzione di stimatori per i parametri di interesse. In particolare nell'estrazione di una statistica sufficiente per la frazione di segnale per un classificatore binario, si deve tener conto del fatto che essa è influenzata da parametri incogniti α tra i parametri θ che possono modificare la distribuzioni di probabilità (PDF) del segnale e/o degli eventi di fondo. La verosimiglianza per N osservazioni dipenderà dunque anche da α e non considerarlo causerebbe una perdita di informazione, dato che gli eventi di fondo da soli hanno un potere vincolante sul valore di α . Il solito compito del classificatore di stimare il rapporto tra le PDF di segnale e fondo non è più sufficiente a risolvere il problema come sarebbe se non fossero presenti disturbi (Neal et al. 2008).

La soluzione implica la costruzione di sommatorie bidimensionali sia per i parametri di disturbo α che per le caratteristiche osservabili dell'evento x , utilizzando, ad esempio, una rete neurale. Se si riecono a costruire buoni modelli parametrici delle statistiche sommario costruite, si possono usare queste ultime per fare inferenza, sfruttando il potere informativo dei dati stessi per vincolare i parametri di disturbo.

Nella letteratura ci sono diverse metodologie per superare situazioni in cui il processo di generazione dei dati non è perfettamente noto e quindi le prestazioni del compito di apprendimento supervisionato considerato (tipicamente la classificazione) potrebbero essere degradate una volta applicato su dati reali. Tuttavia, lavori recenti hanno dimostrato che alcune delle innovazioni nel campo dell'apprendimento automatico sono abbastanza flessibili da essere riproposte per affrontare più da vicino l'inferenza statistica delle analisi HEP. Se ne presentano alcune nel Capitolo 1.4

Le soluzioni proposte di recente si allontanano dall'obiettivo generale di ottimizzare i modelli per diventare performanti in compiti di apprendimento supervisionato, come la classificazione, e tentano di inquadrare il problema direttamente come un problema di inferenza statistica.

Questo cambio di paradigma è spesso indicato come *likelihood-free* o verosimiglianza *simulation-based*, ed è una linea di ricerca in rapida evoluzione, con applicazioni nella fisica delle particelle così come in altri domini scientifici che si basano solidamente su modelli generativi complessi, come l'epidemiologia o la cosmologia. Alcuni di questi approcci di *inferenza-aware* possono essere utili per trattare i parametri di disturbo nel contesto della fisica delle particelle.

1.3.1 Approccio di INFERNO

Negli ultimi anni è stata proposta una famiglia di metodi il cui obiettivo è la costruzione di statistiche riassuntive basate sull'apprendimento automatico che sono meglio allineate con l'obiettivo di inferenza statistica dell'analisi HEP, compresi i parametri di disturbo.

Una volta costruite, queste statistiche riassuntive possono essere utilizzate al posto di sintesi fisiche semplificate o dell'output di classificazione segnale vs background.

Una tecnica in questa categoria, che ha un'applicabilità diretta alle analisi HEP, è INFERNO, acronimo che sta per *Inference-Aware Neural Optimization*. Nel lavoro in cui la procedura viene proposta, De Castro e Dorigo 2019 dimostrano come si possa costruire una statistica riassuntiva non lineare minimizzando perdite motivate dall'inferenza attraverso la discesa stocastica del gradiente specifica per l'obiettivo dell'analisi.

Per esempio, per un'analisi incentrata sulla misurazione di una quantità fisica, l'approccio proposto può essere utilizzato per minimizzare direttamente, come perdita, un'approssimazione dell'incertezza attesa sul parametro di interesse, tenendo pienamente conto dell'effetto dei parametri di disturbo rilevanti.

In INFERNO dunque i parametri di una rete neurale sono ottimizzati tramite gradiente stocastico all'interno di un quadro di differenziazione automatica.

ca, dove la funzione di perdita considerata tiene conto dei dettagli del modello statistico così come l'effetto atteso dei parametri di disturbo.

Una rappresentazione grafica di questa tecnica è fornita in Figura 1.1.

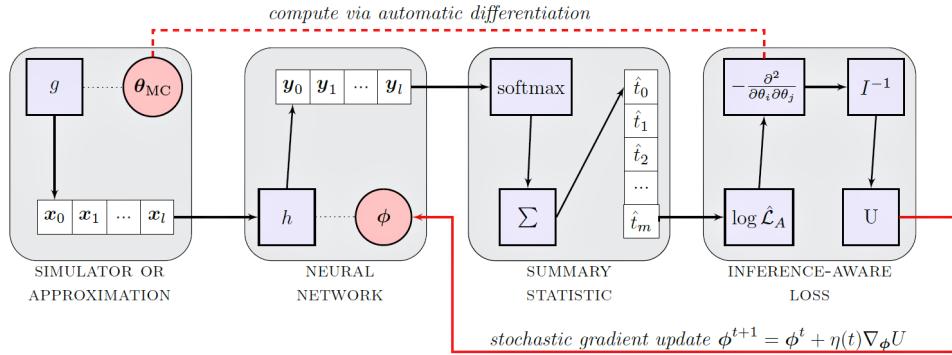


Figura 1.1: Apprendimento di statistiche riassuntive inference-aware: INFERNO. Figura realizzata da De Castro e Dorigo 2019.

Il blocco più a sinistra nel grafico si riferisce al campionamento di un simulatore differenziabile o all'approssimazione dell'effetto dei parametri θ sulle osservazioni simulate esistenti, compresi i parametri rilevanti di disturbo.

Queste osservazioni passano attraverso una rete neurale che dipende da una serie di parametri ϕ (secondo blocco da sinistra) e produce come output una statistica riassuntiva che è un vettore rappresentabile come un istogramma (terzo blocco da sinistra). Sempre nell'ambito della differenziazione automatica, viene costruita una likelihood sintetica, ad esempio, un prodotto di conteggi di Poisson per una statistica sommario simile a un istogramma.

Una funzione di perdita finale *inference-aware*, per esempio un'approssimazione dell'incertezza attesa per i parametri di interesse tenendo conto dei parametri di disturbo, può quindi essere costruita sulla base dell'inversa della matrice Hessiana e utilizzata per ottimizzare i parametri della rete neurale.

Si noti che le approssimazioni utilizzate per rendere una funzione di perdita differenziabile (ad esempio il rilassamento continuo di un istogramma) non influenzano il rigore dell'inferenza statistica risultante. Una volta che è stata appresa la trasformazione statistica sommario con la procedura descritta sopra, essa può essere usata, per esempio attraverso un operatore argmax, per calco-

lare la quantità utile per eseguire l'inferenza statistica con le solite procedure, come si farebbe per qualsiasi altra statistica sommaria basata sull'istogramma.

La sfida principale dell'uso di questo approccio nelle analisi HEP è che l'effetto dei parametri di disturbo deve essere incluso nella struttura di auto-differenziazione, per esempio trasformando le caratteristiche di input, oppure interpolando i pesi delle osservazioni simulate o ancora, considerando l'interpolazione tra i conteggi degli istogrammi. Se questi problemi possono essere superati (anche per solo per una parte dei parametri di disturbo, quelli più significativi), questo metodo fornisce un'alternativa per eseguire la riduzione della dimensionalità usando direttamente un'approssimazione dell'obiettivo di inferenza di una data analisi, in contrasto con una trasformazione basata sulla classificazione probabilistica o su una caratteristica fisica.

Gli autori dimostrano l'efficacia di questa tecnica in un esempio sintetico multidimensionale con fino a tre parametri di disturbo che viene replicato e analizzato approfonditamente nel Capitolo 2. Nell'esempio l'efficacia delle statistiche riassuntive *inference-aware* superano anche quelle ottimali di classificazione.

Si presentano ora nello specifico i passaggi della procedura INFERNO.

L'algoritmo INFERNO mira a minimizzare la varianza attesa del parametro di interesse (POI) ottenuto tramite una verosimiglianza sintetica non parametrica basata sulla simulazione.

I parametri della rete neurale sono ottimizzati dalla discesa del gradiente stocastico tramite differenziazione automatica, dove la funzione di perdita considerata tiene conto dei dettagli del modello statistico così come dell'effetto atteso dei parametri di disturbo. Viene calcolata una statistica riassuntiva *inference-aware* ottimizzando i parametri ϕ di una rete neurale f al fine di ridurre la dimensionalità dei dati di input x :

$$f(\mathbf{x}; \phi) : \mathbb{R}^d \rightarrow \mathbb{R}^b$$

La rete viene addestrata con batch di campioni simulati G_s ottenuti da un simulatore g con parametri θ_s . Il numero di nodi nell'ultimo strato della rete determina la dimensione b della statistica riassuntiva.

Poiché gli istogrammi non sono differenziabili, l'algoritmo usa una funzione softmax come approssimazione differenziabile per l'output della rete neurale y :

$$\hat{s}_i(\mathbf{x}; \boldsymbol{\phi}) = \sum_x \frac{e^{f_i(\mathbf{x}; \boldsymbol{\phi})/\tau}}{\sum_{j=0}^b e^{f_j(\mathbf{x}; \boldsymbol{\phi})/\tau}}$$

dove l'iperparametro di temperatura τ regola la *softness* dell'operatore. Per $\tau \rightarrow 0^+$, la probabilità della componente più grande tenderà a 1, mentre le altre a 0, mettendo in evidenza i valori più grandi e nascondendo quelli che sono molto più piccoli del massimo. Con questa approssimazione è possibile costruire una statistica riassuntiva per ogni batch calcolando la verosimiglianza Asimov dei conteggi di Poisson: $\hat{\mathcal{L}}_A$:

$$\hat{\mathcal{L}}_A(\boldsymbol{\theta}; \boldsymbol{\phi}) = \prod_{i=0}^b \text{Pois}(\hat{s}_i(G_s; \boldsymbol{\phi}) \mid \hat{s}_i(G_s; \boldsymbol{\phi}))$$

dove Asimov significa che il valore di $\hat{\mathcal{L}}_A$ si è calcolato con i parametri posti ai valori attesi basati sui campioni simulati G_s , in modo tale che la stima di massima verosimiglianza per la verosimiglianza di Asimov è il vettore di parametri $\boldsymbol{\theta}_s$ usato per generare il set di dati simulati G_s . Dalla verosimiglianza di Asimov viene poi calcolata la matrice di informazione di Fisher tramite differenziazione automatica secondo:

$$\mathbf{I}(\boldsymbol{\theta})_{ij} = \frac{\partial^2}{\partial \theta_i \partial \theta_j} (-\log \hat{\mathcal{L}}_A(\boldsymbol{\theta}; \boldsymbol{\phi}))$$

La matrice di covarianza può essere stimata dall'inverso della matrice di informazione di Fisher se $\hat{\boldsymbol{\theta}}$ è uno stimatore non distorto dei valori di $\boldsymbol{\theta}$:

$$\text{cov}_{\boldsymbol{\theta}}(\hat{\boldsymbol{\theta}}) \geq I(\boldsymbol{\theta})^{-1}$$

È anche possibile includere misure ausiliarie che impongono dei vincoli sui parametri di disturbo, caratterizzati dalle probabilità $\{\mathcal{L}_C^0(\boldsymbol{\theta}), \dots, \mathcal{L}_C^c(\boldsymbol{\theta})\}$, considerando la verosimiglianza aumentata $\hat{\mathcal{L}}'_A$:

$$\hat{\mathcal{L}}'_A(\boldsymbol{\theta}; \boldsymbol{\phi}) = \hat{\mathcal{L}}_A(\boldsymbol{\theta}; \boldsymbol{\phi}) \prod_{i=0}^c \mathcal{L}_C^i(\boldsymbol{\theta})$$

La funzione di perdita utilizzata per l'ottimizzazione dei parametri della rete neurale ϕ può essere una qualsiasi funzione dell'inverso della matrice di informazione di Fisher, a seconda del problema di inferenza di cui ci si sta occupando.

Gli elementi diagonali $I_{ii}^{-1}(\boldsymbol{\theta}_s)$ corrispondono alla varianza attesa per il parametro θ_i . Così, se lo scopo è un'inferenza efficiente su uno dei parametri $\omega_0 = \theta_k$ una possibile funzione di perdita è:

$$U = I_{kk}^{-1}(\boldsymbol{\theta}_s)$$

che corrisponde all'ampiezza attesa dell'intervallo di confidenza per ω_0 tenendo conto anche dell'effetto degli altri parametri di disturbo in $\boldsymbol{\theta}$.

1.4 Altri approcci per ridurre l'impatto delle sistematiche

Nella letteratura ci sono altri approcci implementati per ridurre l'impatto delle sistematiche, ossia dei parametri di disturbo, in ambito di fisica delle particelle.

Una di queste tecniche è stata presentata da Charnock, Lavaux e Wandelt 2018. Nel loro lavoro, gli autori propongono delle reti neurali massimizzanti l'informazione (*information-maximizing neural networks*, IMNN), una tecnica di apprendimento automatico per trovare funzioni non lineari dei dati che massimizzano l'informazione di Fisher. L'informazione di Fisher durante la fase di training è calcolata dal determinante della matrice di Fisher, che è a sua volta calcolata dalle derivate degli output della rete rispetto ai parametri di inferenza a valori di riferimento per differenziazione numerica o direttamente dal gradiente di un gran numero di simulazioni.

Gli autori inoltre propongono l'inclusione nella funzione di perdita del determinante della matrice di covarianza dell'output della rete neurale per controllare l'ampiezza delle statistiche di sintesi. Per costruzione, questo approccio non considera specificamente il problema dei parametri di disturbo, ma ha l'obiettivo di trovare le trasformazioni che sono il minor possibile influenzate

dai parametri di disturbo e contemporaneamente il più sensibili possibile ai parametri di interesse.

Più recentemente, ci sono diversi lavori che si basano sulle idee che stanno alla base di INFERNO tentando di semplificare la sua applicazione all'analisi della fisica delle alte energie o di estendere le sue funzionalità.

Per esempio, Wunsch et al. 2021 suggeriscono di usare una trasformazione differenziabile di una rete neurale con un singolo nodo, per costruire, come base per la funzione di perdita *inference-aware*, una probabilità di conteggio di Poisson, invece che una softmax. Analogamente a quanto osservato per INFERNO, gli autori dimostrano l'utilità di una costruzione *inference-aware* in un esempio sintetico che include i parametri di disturbo.

Seguendo un percorso diverso, Heinrich e Simpson 2020, autori di NEOS, usano una tecnica chiamata differenziazione a punto fisso per calcolare i gradienti della verosimiglianza profilo per evitare l'approssimazione dell'inversa dell'Hessiana, e per minimizzare direttamente i limiti superiori attesi CLs. Sia Wunsch et al. 2021 che gli autori di NEOS limitano la modellazione dell'effetto dei parametri di disturbo all'interpolazione dell'istogramma.

Oltre agli approcci menzionati, vale la pena citare altre alternative che sebbene abbiano una gamma più limitata di applicabilità potrebbero risultare utili per alcuni casi d'uso.

Elwood e Krücker 2018 propongono di utilizzare la formula di approssimazione della significatività attesa per un esperimento di conteggio a singolo bin, includendo opzionalmente l'effetto di una singola fonte di incertezza sistematica direttamente come perdita di una rete neurale.

Per un diverso tipo di modello, Xia 2019 sviluppa una variazione del training di un *boosted decision tree*, chiamato QBDT, che mira direttamente alla significatività statistica e che può anche includere nella sua approssimazione l'effetto dei parametri di disturbo.

In entrambi i casi, gli autori dimostrano con esempi pratici che gli algoritmi di ottimizzazione della significatività superano le loro controparti di classificazione.

In conclusione la riduzione dell'effetto delle incertezze sistematiche nella stima dei parametri di interesse è un problema cruciale nella fisica delle particelle.

Nell'era dell'apprendimento automatico, sono diventati disponibili metodi automatizzati che possono ridurre significativamente l'impatto che la conoscenza imprecisa delle caratteristiche latenti dei dati hanno sulle misurazioni fisiche. Mentre è già stato accumulato un arsenale significativo di tecniche, non è ancora emersa alcuna procedura universale, per cui è ancora necessaria un'intuizione per discernere le caratteristiche salienti del problema da risolvere e il metodo appropriato da impiegare.

Le strade più promettenti per una procedura di gestione dei parametri di disturbo sono quelle nelle quali gli obiettivi di ottimizzazione sono più direttamente collegati all'obiettivo di inferenza, come la procedura INFERNO.

Capitolo 2

Applicazione di INFERNO a dati sintetici

2.1 Il modello sintetico

Per studiare l'efficienza dell'ottimizzazione *inference-aware* si utilizzano, in primo luogo, dati sintetici descritti da un modello mistura, di cui è nota la verosimiglianza. La procedura INFERNO applicata al problema, viene confrontata con la procedura standard BCE basata sulla minimizzazione della *binary cross-entropy*.

I due metodi confrontati hanno un duplice obiettivo: discriminare in un set di dati il segnale dal fondo e misurare la porzione di segnale nel campione utilizzando l'esito della classificazione. In generale nei problemi di classificazione l'obiettivo è ottenere la massima separazione tra segnale e fondo, ma in fisica delle particelle si utilizza questa separazione per condurre un'analisi inferenziale ed estrarre una misura della frazione di segnale nel campione con la massima precisione, dunque i due obiettivi non sono perfettamente allineati.

L'approccio tradizionale ai casi di inferenza su un parametro di interesse con disturbi, rappresentato dalla procedura BCE, prevede di implementare un classificatore binario, in questo caso una deep neural network, che riceve in ingresso un certo modello del segnale e uno del fondo e conduce la fase di training ignorando gli errori sistematici. L'output della rete neurale poi

viene raggruppato, al fine di costruire una statistica sommario da usare per l'inferenza sulla potenza del segnale, ossia porzione di segnale del campione; solo in questa fase si considerano i parametri di disturbo e l'incertezza che essi determinano sulla misura finale, proponendo diversi modelli per il segnale e per il fondo e confrontando i diversi esiti inferenziali a valle. In questo modo si ottiene una misura della variabilità del rate di segnale, chiamato errore sistematico relativo, che però, nel caso siano presenti parametri di disturbo consistenti, risulta sub-ottimale. L'approccio tradizionale assegna dunque un errore sistematico alla misura in modo ex-post.

L'approccio di INFERNO invece prevede di includere gli errori sistematici, dovuti al fatto che non si conosce alla perfezione il modello del background, direttamente nella fase di training della deep neural network. Questo è fatto, come già spiegato nel Capitolo 1, definendo come funzione di perdita l'elemento corrispondente al parametro di interesse dell'inversa dell'Hessiana della log-verosimiglianza del modello saturato calcolata con la statistica sommario definita dal classificatore.

Per questa analisi si considera un modello mistura tridimensionale a due componenti. Una componente sarà indicata come *background* e ha distribuzione:

$$f_b(\mathbf{x}|r, \lambda) = N\left((x_0, x_1) \middle| (2 + r, 0), \begin{bmatrix} 5 & 0 \\ 0 & 9 \end{bmatrix}\right) Exp(x_2|\lambda)$$

mentre la seconda componente sarà indicata come *segnale* e ha distribuzione:

$$f_s(\mathbf{x}) = N\left((x_0, x_1) \middle| (1, 1), \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) Exp(x_2|2)$$

Nel modello appena descritto (x_0, x_1) sono distribuite come normali multivariate, mentre x_2 segue una distribuzione esponenziale indipendente da (x_0, x_1) sia per il background che per il segnale. Le distribuzioni delle tre variabili per un campione di 50'000 osservazioni sono riprese da De Castro e Dorigo (2019) riportate in Figura 2.1.

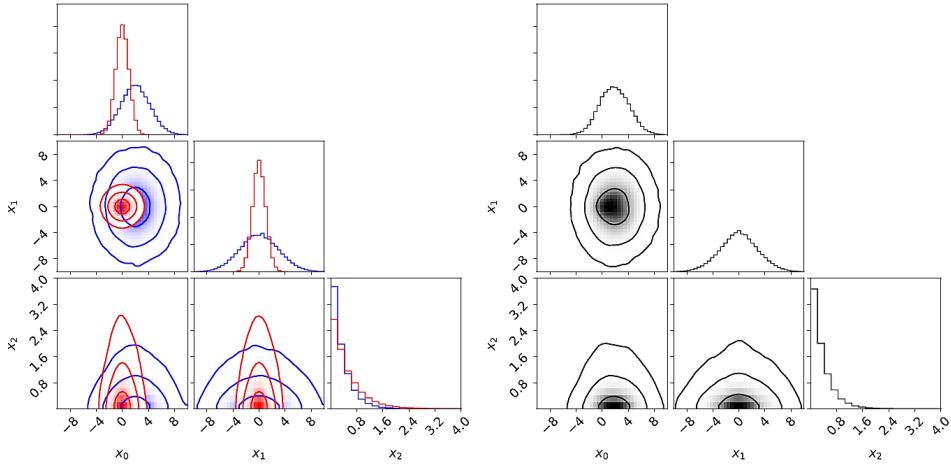


Figura 2.1: In rosso la distribuzione del segnale, in blu quella del backgroud di un campione di 50 000 osservazioni dal modello considerato con $r = 0$, $\lambda = 3$ e $\mu = 50/1050$. In nero le distribuzioni sotto il modello mistura. Figura realizzata da De Castro e Dorigo 2019.

La distribuzione del segnale è completamente nota, mentre quella del background è caratterizzata dai parametri incogniti r e λ , chiamati parametri di disturbo. In particolare r incide nella media, mentre λ nel parametro della distribuzione esponenziale che caratterizza x_2 .

Si sceglie un modello mistura che ha la seguente struttura:

$$p(\mathbf{x}|\mu, r, \lambda) = (1 - \mu)f_b(\mathbf{x}|r, \lambda) + \mu f_s(\mathbf{x})$$

dove μ è il peso del segnale nella mistura mentre $(1 - \mu)$ è il peso della componente di background.

Si assume per il numero totale di eventi osservati una distribuzione di Poisson di media $s + b$ che corrispondono al valore atteso di eventi di segnale e di background corrispettivamente. Da questo si può dunque giungere ad una riparametrizzazione più conveniente per la distribuzione di probabilità delle osservazioni:

$$p(\mathbf{x}|s, r, \lambda, b) = \frac{b}{s+b}f_b(\mathbf{x}|r, \lambda) + \frac{s}{s+b}f_s(\mathbf{x})$$

Per le analisi fisiche ad LHC è comune questa riparametrizzazione perché la teoria fornisce informazioni proprio sul numero atteso di osservazioni.

Avendo la distribuzione degli eventi nota, Poisson, e il numero atteso di eventi che dipende dai parametri del modello, si può definire la verosimiglianza aggiungendo un termine di conteggio di Poisson come segue:

$$L(s, r, \lambda, b) = Poi(n|s + b) \prod_{i=1}^n p(\mathbf{x}|s, r, \lambda, b)$$

Un'altra quantità rilevante è il rapporto di densità condizionato, che corrisponde, nel caso di un campione bilanciando, al miglior classificatore di segnale e background:

$$t_B(\mathbf{x}|r, \lambda) = \frac{f_s(\mathbf{x})}{f_s(\mathbf{x}) + f_b(\mathbf{x}|r, \lambda)}$$

La natura sintetica di questo modello permette di generare facilmente il dataset di training. Viene simulato un campione di training di 200 000 osservazioni, metà di segnale e metà di background. Quest'ultimo è generato usando $r = 0.01$ e $\lambda = 3.0$. Una parte del training set compone il validation set, usato per calcolare metriche rilevanti durante il training e per controllare l'over-fitting.

I valori conclusivi che permettono di confrontare i diversi approcci sono calcolati con un dataset più ampio, di un milione di osservazioni. Per semplicità, in entrambe le procedure confrontate, INFERNO e BCE, i mini-batch di osservazioni per ogni step di training, contengono un numero uguale di eventi di segnale e di background.

Per discriminare gli eventi come di segnale o di background si utilizza una rete neurale, un metodo di apprendimento supervisionato di machine learning, che considera i parametri r e λ fissati. L'output di questo metodo è comunemente un vettore di probabilità (c_s, c_b) per ogni osservazione data \mathbf{x} . Dato che questo tipo di output ignora l'effetto dei parametri di disturbo r e λ , è necessario utilizzare una stima non parametrica della densità, come l'istogramma.

2.2 Studi sugli effetti sistematici sul background

Il modello descritto nella Sezione 2.1 contiene quattro parametri ignoti: s il numero atteso di eventi di segnale, lo shift della media del background

r , λ il rate dell'esponenziale della terza dimensione x_2 e il numero atteso di eventi di background b . L'effetto di s e b può essere facilmente incluso nell'analisi riscalando opportunamente il vettore di conteggi di Poisson. Per quanto riguarda i parametri di disturbo r e λ risulta più semplice modellare i loro effetti trasformando i dati di input \mathbf{x} .

In particolare per introdurre l'effetto di λ si aggiunge ad ogni osservazione del mini-batch generato dal processo di background, un vettore $(r, 0.0, 0.0)$. Similmente, si tiene conto dell'effetto di λ moltiplicando la componente x_2 per il rapporto tra λ_0 usata per generare e quella che viene modellata.

Questa applicazione ha l'obiettivo di fare inferenza sul parametro di interesse s . Per fare ciò si confronta il metodo di ottimizzazione *inference-aware* con *BCE* per una serie di vincoli sui parametri di disturbo.

La quantità principale utilizzata per i confronti è l'incertezza attesa del parametro di interesse s condizionato ai veri valori dei parametri $s = 50, r = 0.0, \lambda = 3.0$ e $b = 1000$. Quest'ultima, per entrambi i metodi, può essere stimata tramite la semi-ampiezza della log-verosimiglianza profilo normalizzata ad altezza 0.5, grazie al teorema del limite centrale che definisce per la log-verosimiglianza un'approssimazione parabolica quando si ha una quantità sufficientemente grande di dati. Questa stima, rispetto alla stima che si basa sull'Hessiana, è più precisa, in quanto tiene conto della possibile non parabolicità della log-verosimiglianza. In tutti i casi descritti, le incertezza ottenute usando l'Hessiana della log-verosimiglianza sono molto simili a quelle ottenute calcolando la semi-ampiezza della verosimiglianza profilo normalizzata.

Si definisce la stessa rete neurale per entrambe le procedure con 2 strati latenti, ciascuno di 100 nodi con funzione di attivazione ReLU e 10 nodi di output. L'ultimo livello è seguito da una funzione di attivazione *softmax* con parametro di *temperature* $\tau = 0.1$ per garantire che le approssimazioni siano differenziabili e vicine ai valori veri, come descritto nel Capitolo 1. I valori iniziali della rete sono invece inizializzati secondo un algoritmo pseudo-casuale.

Tutti i modelli stimati con INFERNO sono allenati attraverso un certo numero di epoche con discesa del gradiente stocastica (SGD) usando dei mini-batch di 2000 osservazioni e un learning rate $\gamma = 10^{-5}$; mentre quelli basati sulla cross-entropy sono caratterizzati da mini-batch di 64 osservazioni con learning rate $\gamma = 0.001$.

Nei vari modelli, le incertezze attese sono calcolate attraverso un binning della verosimiglianza, esito dell’interpolazione delle distribuzioni ad istogramma di segnale e background per i valori dei parametri di disturbo variabili.

Volendo specificare una certa variabilità ai parametri di disturbo, si inseriscono nei due modelli delle misure ausiliarie che determinano la forma e la variabilità delle sistematiche. In questo modo si definiscono dei vincoli sui parametri di disturbo. Nel caso specifico si prevede sia per λ che per r una distribuzione Gaussiana.

Ci si pone come primo obiettivo quello di analizzare la varianza stimata del parametro di interesse s dai due approcci, al variare della standard deviation dei parametri di disturbo. Ci si aspetta che all’aumentare dell’incertezza delle sistematiche gli algoritmi peggiorino nella stima di s , ossia riportino varianze più grandi. Questo comportamento ci si aspetta sia più evidente per BCE, rispetto ad INFERNO il quale è costruito per essere robusto ai parametri di disturbo.

Per quanto riguarda INFERNO aumentare l’incertezza sulle sistematiche consiste nel fornire, durante la fase di addestramento, una larghezza maggiore per il vincolo ausiliario e durante la fase di inferenza interpolare i modelli valutati a $\pm 1\sigma$ dei valori dei disturbi. Ciò significa usare un metodo di interpolazione dei modelli valutati nei casi di shift di $\pm 1\sigma$ dei parametri di disturbo. L’interpolazione consiste nella generazione dei dati con i valori shiftati di r e λ dai loro valori nominali, seguita dal fit dei modelli e infine dall’interpolazione delle tre stime ottenute.

Al fine di valutare la significativa differenza tra l’incertezza stimata di s con i due modelli, si simula ciascun setup 10 volte, definito dai vari valori di varianza attribuiti alle distribuzioni di λ ed r . Si calcola poi la media delle semiampiezze stimate ad altezza 0.5 della verosimiglianza profilo relativa e come indice di variabilità di quest’ultima la deviazione standard campionaria (SD), ossia la radice della varianza campionaria corretta. Queste quantità vengono riportate in Figura 2.3.

Per ogni run, implementato con un nuovo set di dati generato, viene verificato che la varianza stimata in funzione del numero di epoche di training converga. E’ da notare che per INFERNO la loss è esattamente la varianza del parametro di interesse s , mentre non lo è per la BCE. Nel caso in

cui non si verifichi la convergenza, episodio raro di cui dunque non ci si occupa, il run viene scartato. E' necessario verificare che il training sia stato eseguito un numero sufficiente di volte per consentire all'algoritmo di arrivare almeno vicino all'ottimo, per poter considerare i valori restituiti consistenti. Come si vede in Figura 2.2, INFERNO necessita di più epoche per giungere ad un'accettabile convergenza; questo è giustificato dal fatto che lavora su una funzione di verosimiglianza molto più complessa rispetto a quella su cui lavora BCE. L'andamento è comunque esponenziale, come ci si aspetta, anche se con decadimento più lento rispetto a BCE.

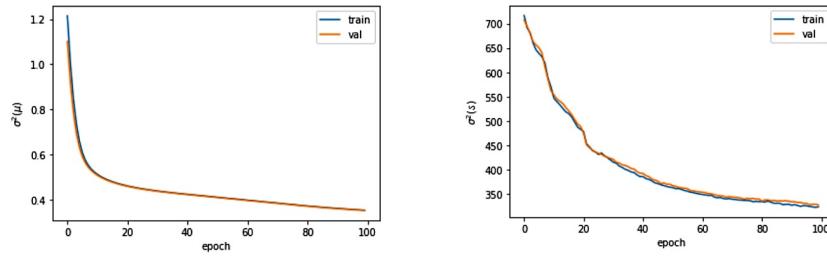


Figura 2.2: A sinistra: radice quadrata della varianza stimata di μ con BCE. A destra: varianza stimata di s con INFERNO al variare del numero di epoche per l'addestramento.

In Figura 2.3 si vede che gli errori di s calcolati da INFERNO sono, in tutti i casi inferiori rispetto a quelli calcolati con BCE. L'aumento della larghezza della sistematica r , analogamente per λ , genera un aumento dell'errore su s . Nei risultati di BCE si nota che il parametro di disturbo λ ha un impatto più forte sulla varianza di s rispetto a quanto ne abbia r . L'aspetto più significativo, che si riesce ben ad identificare nella Figura 2.4, è che INFERNO è molto robusto, rispetto a BCE ai cambiamenti delle incertezze delle sistematiche.

BCE (auxiliary constraints)	Run1	Run2	Run3	Run4	Run5	Run6	Run7	Run8	Run9	Run10	Mean	SD
r~N(0,0.2) $\lambda \sim N(3,0.5)$	20.41	20.57	20.65	21.58	19.82	20.78	20.50	20.91	19.71	19.81	20.48	0.58
r~N(0,0.5) $\lambda \sim N(3,0.5)$	20.26	21.67	20.98	21.27	21.87	21.19	20.61	20.67	21.14	20.61	21.03	0.50
r~N(0,1) $\lambda \sim N(3,0.5)$	21.21	21.26	20.83	20.85	21.58	21.09	21.15	22.01	20.87	21.13	21.20	0.36
r~N(0,0.2) $\lambda \sim N(3,0.8)$	21.52	22.87	22.17	21.51	21.77	22.26	22.48	21.77	22.96	21.98	22.13	0.52
r~N(0,0.2) $\lambda \sim N(3,1.4)$	23.60	24.41	24.20	24.81	23.51	23.73	25.83	25.11	24.56	23.18	24.29	0.82
INFERNO	Run1	Run2	Run3	Run4	Run5	Run6	Run7	Run8	Run9	Run10	Mean	SD
r~N(0,0.2) $\lambda \sim N(3,0.5)$	17.79	17.50	17.98	17.26	17.90	17.10	17.67	17.63	17.43	17.85	17.61	0.29
r~N(0,0.5) $\lambda \sim N(3,0.5)$	17.40	17.92	18.00	17.42	17.83	18.15	18.62	18.15	18.01	18.05	17.95	0.36
r~N(0,1) $\lambda \sim N(3,0.5)$	17.59	18.06	17.62	18.76	17.47	18.92	17.36	18.46	17.83	17.44	18.00	0.56
r~N(0,0.2) $\lambda \sim N(3,0.8)$	17.74	17.76	18.31	17.31	18.00	18.40	17.49	18.00	17.06	18.54	17.86	0.48
r~N(0,0.2) $\lambda \sim N(3,1.4)$	17.47	18.28	17.51	20.14	18.28	17.48	17.11	17.52	17.36	18.11	17.93	0.88

Figura 2.3: Stima dell'incertezza del parametro di interesse s , di BCE sopra e INFERNO sotto, per 5 diversi setup delle distribuzioni dei due parametri di disturbo, per 10 iterazioni; le ultime due colonne corrispondono alla media e alla deviazione standard delle 10 misure.

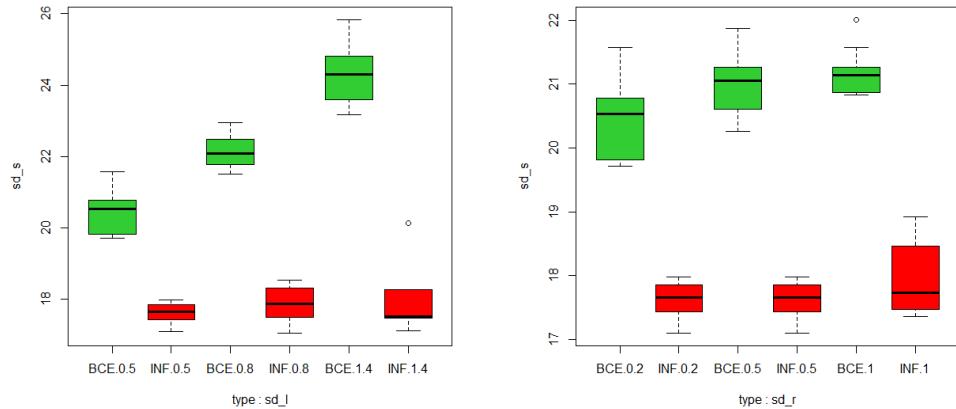


Figura 2.4: Distribuzione della standard deviation stimata di s , da INFERO e BCE, a sinistra per i 3 valori di standard deviation di r impostati con λ fissata a 0.5, a sinistra per i 3 valori di standard deviation di λ con r fissata a 0.2, per 5 fit con uno stesso dataset.

Oltre ad analizzare la varianza stimata del parametro di disturbo è interessante visualizzare il comportamento dell'intera matrice di varianza e covarianza dei parametri del modello: s , r e λ .

Per fare ciò si esegue un fit del modello per 5 campioni diversi, per ciascuno dei 5 casi presi in esame in Figura 2.3. Si ottiene una misura media e di standard deviation della matrice di varianza e covarianza. Per avere una lettura più immediata si estraggono le varianze e le correlazioni, piuttosto che le covarianze. Ogni run è eseguito su un campione diverso (c1, c2, c3, c4, c5), ma viene usato lo stesso campione per lo stesso run di ogni vincolo; per esempio il campione “c1” viene usato per il primo run del caso1 (ossia $r \sim N(0, 0.2)$ e $\lambda \sim N(3, 0.5)$), per il primo run del caso2, e per ciascun primo run di ogni altro caso. Sono stati considerati buoni solo quei run per cui le loss di INFERO e di BCE fossero giunte a convergenza, per i motivi detti sopra. La media e la standard deviation per ogni caso, sono riportati nelle Figura 2.5. Si noti che la matrice di varianza e covarianza è simmetrica, perciò la correlazione tra r e λ è uguale alla correlazione tra λ ed r . Inoltre, se si valutano i parametri di disturbo in corrispondenza di $s = 50$ e il vero valore di s è 50, il gradiente dei disturbi è nullo, non c'è quindi alcuna ottimizzazione, siamo nella stima di

massima verosimiglianza.

BCE	var(s): mean sd	var(r): mean sd	var(l): mean sd	cor(r,l): mean sd	cor(s,r): mean sd	cor(s,l): mean sd
r~N(0,0.2) $\lambda \sim N(3,0.5)$	354.37 4.96	0.40 0.10	0.59 0.26	0.02 0.12	0.44 0.10	-0.33 0.12
r~N(0,0.5) $\lambda \sim N(3,0.5)$	402.72 5.01	0.11 0.01	0.71 0.03	-0.03 0.004	0.57 0.004	-0.36 0.01
r~N(0,1) $\lambda \sim N(3,0.5)$	412.20 6.28	0.03 0.00	0.71 0.02	-0.04 0.02	0.58 0.01	-0.36 0.01
r~N(0,0.2) $\lambda \sim N(3,0.8)$	395.04 2.53	0.44 0.004	0.48 0.03	-0.05 0.01	0.47 0.01	-0.48 0.01
r~N(0,0.2) $\lambda \sim N(3,1.4)$	434.88 8.67	0.44 0.004	0.26 0.05	-0.04 0.02	0.45 0.02	-0.54 0.02

INFERNO	var(s): mean sd	var(r): mean sd	var(l): mean sd	cor(r,l): mean sd	cor(s,r): mean sd	cor(s,l): mean sd
r~N(0,0.2) $\lambda \sim N(3,0.5)$	273.94 3.12	0.28 0.08	0.17 0.12	0.25 0.21	0.30 0.02	-0.10 0.05
r~N(0,0.5) $\lambda \sim N(3,0.5)$	295.00 22.55	0.05 0.01	0.22 0.14	0.16 0.13	0.36 0.04	-0.13 0.05
r~N(0,1) $\lambda \sim N(3,0.5)$	276.97 6.64	0.01 0.00	0.15 0.07	0.18 0.11	0.33 0.02	-0.11 0.03
r~N(0,0.2) $\lambda \sim N(3,0.8)$	280.64 15.41	0.35 0.18	0.08 0.04	0.31 0.17	0.31 0.05	-0.09 0.04
r~N(0,0.2) $\lambda \sim N(3,1.4)$	275.85 4.00	0.26 0.03	0.02 0.004	0.27 0.10	0.30 0.02	-0.11 0.01

Figura 2.5: Per ciascun vincolo sulle sistematiche i valori medi di varianza e correlazioni con relativi standard error dei 5 modelli stimati tramite BCE sopra e INFERNO.

Dalla Figura 2.5 si nota che aumentando l'incertezza delle sistematiche, la varianza delle misure di s aumenta. Infatti, rilassando i vincoli sulle sistematiche, permettiamo alle misure dei parametri di disturbo di essere più precise, a discapito della misura su s . Dunque, rilassare i vincoli significa sia aumentare la loro incertezza, in modo tale che i modelli a $\pm 1\sigma$ corrispondano a maggiori variazioni di forma, che consentire al fit di stimare in modo più preciso le stesse. Questo è causato principalmente dalla derivata seconda che aumenta

all'aumentare della larghezza delle varianze dei vincoli; se aumentano gli elementi diagonali dei parametri di disturbo, l'inversa dell'Hessiana presenta una diminuzione netta delle varianze, e dunque la varianza delle sistematiche diminuisce all'aumentare dell'incertezza dei vincoli. In conclusione, se rilassiamo i vincoli sui disturbi permettiamo alle misure di essi di essere più precise a discapito della precisione sull'incertezza di s ; invece, se sulle sistematiche diamo un vincolo più ristretto, il parametro di interesse può essere misurato più precisamente.

Il training di INFERNO incoraggia la precisione maggiore possibile sulla misurazione finale di interesse e la sua indipendenza dai disturbi, che infatti, come mostrato, vengono maggiormente decorrelati in INFERNO rispetto a quanto faccia BCE; al contempo incoraggia, anche se meno, ad essere precise le misure dei disturbi e indipendenti tra loro.

Più precisamente si nota che in Figura 2.5 in media non c'è un regolare aumento della varianza di s all'aumentare della varianza delle sistematiche, per ogni caso specifico, questo perchè le quantità riportate presentano una variabilità intrinseca dell'ottimizzazione, dovuta all'inizializzazione della rete neurale, che dunque è interessante analizzare. Per un unico dataset generato, si esegue 5 volte il fit del modello, per ciascun caso, ottenendo una misura del range di variazione della matrice di varianza e covarianza dei parametri del modello intrinsecamente dovuto alla minimizzazione. Si riportano nei grafici in Figura 2.6 una visualizzazione della distribuzione MC della varianza del parametro di interesse. Tutti i 25 fit (5 per ogni caso) sono stati eseguiti sullo stesso campione di dati.

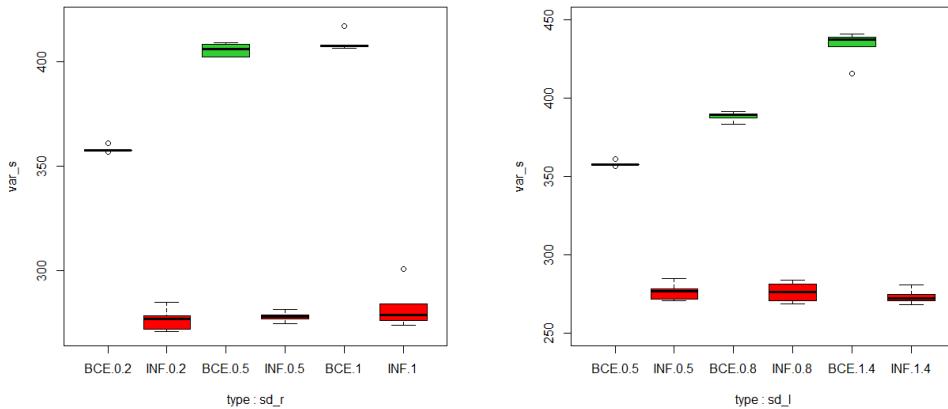


Figura 2.6: Distribuzione della varianza stimata di s , da INFERNO e BCE: a sx per i 3 valori di standard deviation di r con quella di λ fissata a 0.5; a dx per i 3 valori della standard deviation di λ con quella di r fissata a 0.2

La Figura 2.6 dimostra che c’è una certa variabilità dovuta all’inizializzazione della rete neurale che rispecchia nuovamente l’andamento descritto finora all’aumentare della larghezza delle sistematiche. Questa variabilità, intrinseca del processo di ottimizzazione, è da tenere in considerazione e giustifica le considerazioni fatte in precedenza.

2.3 Performance sulla frazione di segnale

Tutte le analisi riportate finora sono state svolte con un fissato rapporto segnale-rumore pari a 50/1000. Per accertarsi che le performance esposte siano coerenti e non distorte a causa di questo valore fissato, si conduce un’analisi ulteriore. Si svolgono 5 fit per diversi rapporti di segnale e rumore S/B per il caso con vincoli $r \sim N(0, 0.2)$ e $\lambda \sim N(3, 0.5)$ e se ne riportano i risultati. In questo caso le reti sono implementate con 130 epoche.

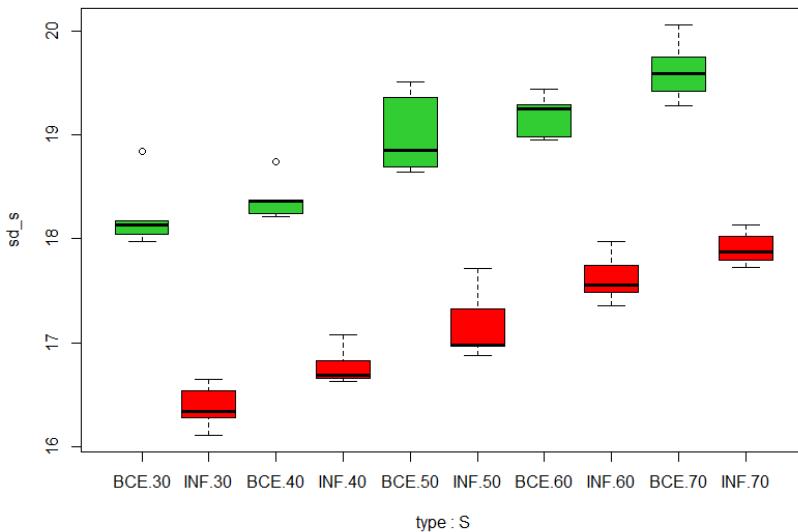


Figura 2.7: Distribuzione della deviazione standard stimata da INFERNO e BCE del parametro di interesse s per 5 fit, al variare del numero di eventi di segnale simulati su 1000 di background con vincoli $r \sim N(0, 0.2)$ e $\lambda \sim N(3, 0.5)$.

La Figura 2.7 mostra che all'aumentare del numero di eventi di segnale nel campione simulato, la stima della standard deviation di s aumenta, sia per BCE che per INFERNO, coerentemente con il fatto che stiamo considerando processi di Poisson: in un esperimento con $B = 0$ la varianza varia come la media. I due andamenti per BCE e INFERNO, non si differenziano a meno del fatto che INFERNO ha valori uniformemente inferiori a BCE.

Se si analizza invece l'incertezza relativa in Figura 2.8, ossia il rapporto tra standard deviation medio di s stimato e il numero di eventi di segnale considerati S , questa è strettamente decrescente all'aumentare del numero di eventi di segnale nel campione, questo ci permette di sostenere che i risultati ottenuti in precedenza sono consistenti e non dipendono dalla frazione di segnale scelta per condurre le analisi.

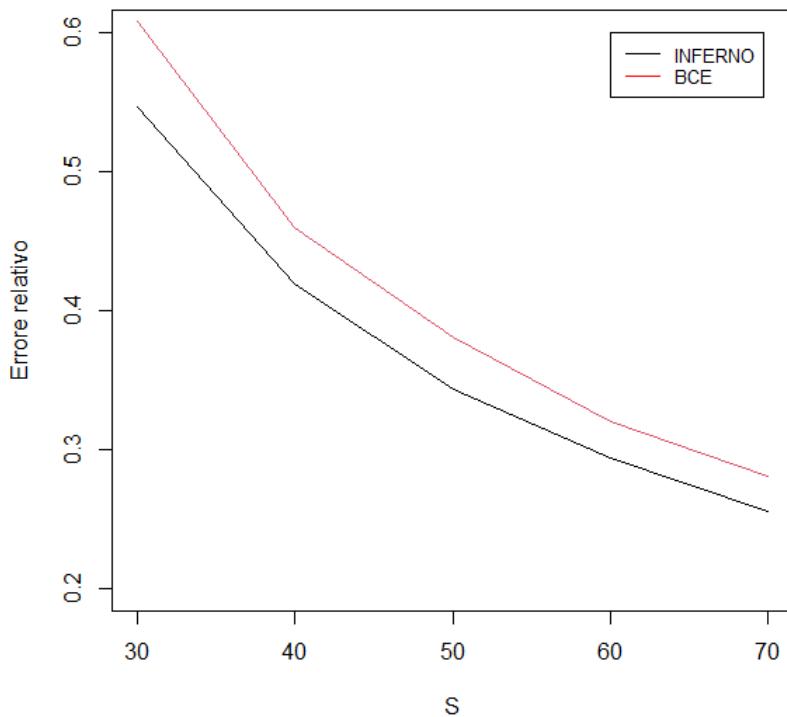


Figura 2.8: Errore relativo, ossia il rapporto tra standard deviation medio di s stimato e numero di eventi di segnale considerati S , per 5 fit con $r \sim N(0, 0.2)$ e $\lambda \sim N(3, 0.5)$, al variare di S

In conclusione questo esempio sintetico dimostra che l’approccio *inference-aware* è efficace, in quanto l’ottimizzazione della funzione di perdita tiene conto, già nella fase di training, degli effetti dei parametri di disturbo. La metodologia INFERNO, che minimizza direttamente la varianza attesa del parametro di interesse, risulta essere considerevolmente più efficiente di BCE, quando si hanno parametri di disturbo. Naturalmente sono necessari più esperimenti per valutare l’utilità di questa procedura in problemi inferenziali come quelli delle alte energie ad LHC, di cui se ne vedranno alcuni nel Capitolo 4.

L’intero codice utilizzato per sviluppare queste analisi è riportato in Appendice A.

Capitolo 3

Ricerca di segnali in fisica delle particelle

3.1 Il modello standard

Evidenze sperimentali indicano che l'universo è regolato da quattro tipi di forze: forte, debole, elettromagnetica e gravitazionale. Mentre le forze gravitazionale e elettromagnetica hanno un raggio d'azione infinito, le forze deboli e forti si manifestano principalmente a corto raggio d'azione, ovvero su scale atomiche e subatomiche. La forza debole è più intensa della forza di gravità ma è meno intensa delle altre due; la forza forte, invece, è la più intensa tra le interazioni fondamentali. Il Modello Standard descrive tutte le forze eccetto quella gravitazionale, esso consiste in una teoria che descrive i componenti primi della materia e le loro interazioni: l'interazione elettromagnetica, l'interazione debole (unificate nella cosiddetta interazione elettrodebole) e l'interazione forte. L'impossibilità di includere l'interazione gravitazionale e il suo mediatore (il gravitone) rappresenta uno dei punti deboli del MS, il problema, però, risulta trascurabile negli attuali esperimenti di fisica delle particelle.

Lo sviluppo del Modello Standard ha origine negli anni '60 con l'idea di rottura spontanea di simmetrie di gauge, e il modello di Glashow, Salam e Weinberg. La teoria è stata ampiamente accettata dalla scoperta di correnti deboli neutre al CERN nel 1973. Varie precise misurazioni e scoperte hanno

validato fermamente il Modello Standard ad energie fino alla scala elettrodebole. La scoperta del bosone di Higgs, da parte delle collaborazioni ATLAS e CMS, nel 2012 ha completato lo spettro delle particelle del Modello Standard.

Nel MS, le particelle fondamentali si possono dividere in due tipi: i cosiddetti campi di materia (leptoni, che subiscono solo interazioni elettrodeboli e quark) e i bosoni mediatori (o di gauge) delle forze. Ciascuna interazione tra i campi di materia è regolata da un'opportuna simmetria locale (di gauge); l'interazione tra campi di materia può interpretarsi in termini di scambio di bosoni che, proprio per il loro ruolo, vengono detti bosoni mediatori (o di gauge).

Ogni gruppo di fermioni, leptoni e quark, consta di sei particelle che sono suddivise in tre generazioni. Leptoni e quark si differenziano per la massa, per il tipo di interazione alla quale sono soggette e per la loro carica elettrica avendo i primi, elettroni, muoni, tau e neutrini, carica elettrica di valore intero rispetto a quella dell'elettrone, +1, -1 e 0 (per i neutrini), mentre i secondi hanno tutti carica frazionaria rispetto a quella dell'elettrone $+2/3$ o $-1/3$. Ciò che differenzia leptoni e quark di generazioni diverse è la massa: le particelle più leggere e stabili fanno parte della prima generazione, quelle più pesanti e meno stabili appartengono alla seconda e alla terza generazione. Essi interagiscono per mezzo dello scambio dei quanti di campo: i bosoni mediatori.

Ogni forza fondamentale ha il suo corrispondente bosone: gluoni per la forza forte, fotone per la forza elettromagnetica e i bosoni W e Z per la forza debole.

Una rappresentazione della struttura del Modello Standard è fornita in Figura 3.1.

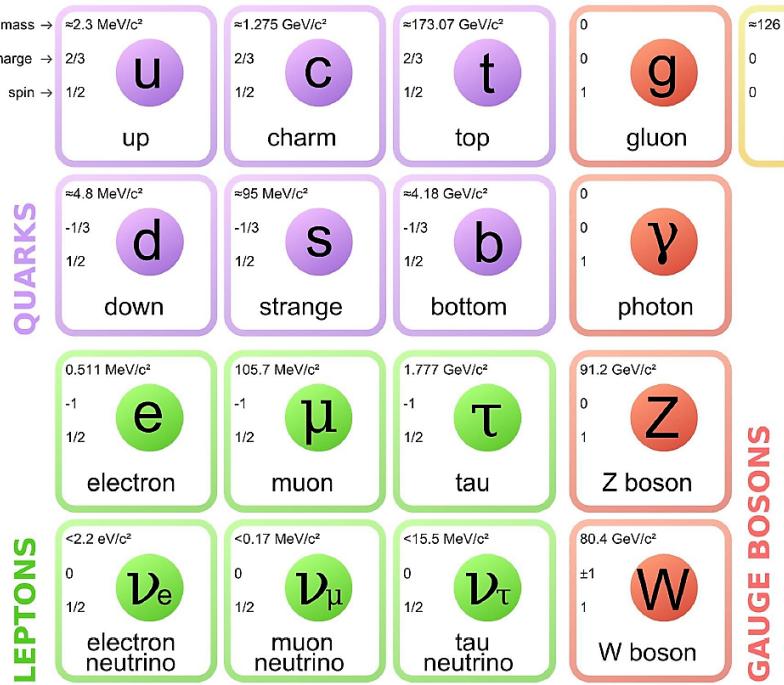


Figura 3.1: Struttura del Modello Standard.

Del Modello Standard è parte integrante il meccanismo di Higgs-Brout-Englert grazie al quale le particelle fondamentali e quelle mediatiche, tutte inizialmente poste a massa nulla, acquistano massa tramite un meccanismo definito di rottura spontanea della simmetria che regola le interazioni elettromagnetiche e deboli, che prevede l'introduzione di una nuova particella, il bosone di Higgs. La possibilità di mantenere la struttura fondamentale del modello, salvaguardandone predittività e consistenza teorica, è offerta dunque dal meccanismo di Higgs, che, a fronte dell'introduzione di un ulteriore campo scalare (un bosone di spin 0), consente di assegnare massa non soltanto ai bosoni W e Z, ma anche a tutti i fermioni del modello, rompendo in modo spontaneo la simmetria di gauge.

Benché il bosone di Higgs non sia ancora stato osservato in modo diretto, diverse speculazioni indirette basate sulla consistenza interna del Modello Standard e sulle correzioni quantistiche a quantità misurate sperimentalmente (come la massa del quark top), sembrano preferire una massa del bosone di Higgs dell'ordine della scala elettrodebole (ovvero dell'ordine di $200 \text{ GeV}/c^2$).

L’interazione elettromagnetica nel MS avviene tramite scambio di fotoni, ed interessa solo le particelle fondamentali dotate di carica elettrica. L’interazione debole, invece, ha luogo tra tutti i fermioni fondamentali (quark e leptoni). È descritta dallo scambio di bosoni mediatori di massa non nulla, da cui consegue la natura a corto raggio d’azione dell’interazione. Se i bosoni mediatori di forza sono carichi (W^+ , W^-) si parla di interazione di corrente carica, se invece essi sono neutri (Z) allora si avrà un’interazione di corrente neutra.

Nel confronto tra le due forze, si nota, inoltre, che la costante di accoppiamento debole risulta dello stesso ordine di grandezza di quella elettromagnetica, nonostante le osservazioni sperimentalistiche indichino una minore intensità delle interazioni deboli rispetto a quelle elettromagnetiche su scale di energia minori rispetto alla massa dei mediatori deboli. Pertanto, nel MS, a grandi energie, le due forze sono unificate seguendo il modello di Glashow-Weinberg-Salam (GWS). L’interazione debole di corrente carica trasforma il leptone nel suo corrispondente neutrino e viceversa. Per i quark si osserva il mescolamento tra le tre generazioni e quindi il passaggio da un quark con carica ($Q = e$) = +2/3, ad uno con carica ($Q = e$) = -1/3. Esse avvengono tramite lo scambio del bosone carico W e permettono di collegare generazioni diverse.

Tale fenomeno venne ben descritto da Cabibbo prima, e da Kobayashi e Maskawa poi, tramite la matrice CKM (Cabibbo 1963 e Kobayashi e Maskawa 1973). Il mescolamento è un processo fortemente gerarchico, sono infatti favorite le transizioni tra quark della stessa generazione. Anche per i neutrini esiste una matrice analoga a quella CKM detta PMNS (Pontecorvo, Maki, Nakagawa e Sakata).

L’interazione forte interessa i quark e i gluoni. Le due specie a cui appartengono, fermioni e bosoni, possiedono una carica di colore in analogia alla carica elettrica per la forza elettromagnetica. A differenza dell’interazione elettromagnetica, esistono tre tipi diversi di carica di colore: Red, Green, Blue; per questo la teoria che descrive le interazioni forti tra i quark è detta cromodinamica quantistica (QCD). I gluoni sono bicolorati e la composizione di colore-anticolore rende possibile l’esistenza di otto differenti gluoni. Un esempio di interazione forte è riportata in Figura 3.2.

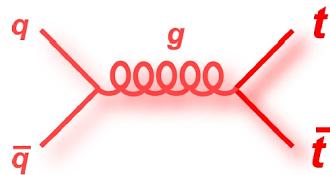


Figura 3.2: Diagramma di Feynman di un esempio di interazione forte mediata da un gluone a partire da una coppia di quark-antiquark.

L'interazione gluone-gluone definisce due proprietà della QCD: confinamento del colore e libertà asintotica. La prima è sostanzialmente una regola per la composizione dei quark nella formazione di particelle più pesanti e rende conto del fatto che i quark non possono mai essere isolati da altre cariche di colore, ma si trovano permanentemente confinati all'interno di particelle composite a carica di colore neutra. Inoltre, la forza di colore tra due quark non diminuisce con la distanza che li separa, il che è all'origine del loro confinamento all'interno degli adroni, che possono essere composti da quark e antiquark, mesoni, o da tre quark, barioni. L'altra proprietà invece afferma che ad altissima energia, il che ad altissima energia quark e gluoni interagiscano abbastanza debolmente da poter essere trattati come particelle libere. Nel corso degli ultimi vent'anni la QCD nel regime delle alte energie ha portato a predizioni sempre più precise, che sono state verificate sperimentalmente con grande accuratezza, fino alla scoperta del bosone di Higgs nelle collisioni tra protoni nell'acceleratore LHC del CERN.

Ad oggi, essenzialmente tutte le verifiche sperimentali del Modello Standard si sono dimostrate in accordo con le previsioni; nonostante ciò, il MS non può considerarsi una teoria completa delle interazioni fondamentali, dal momento che non include una descrizione della gravità e non è compatibile con la relatività generale. Ecco allora la necessità di esplorare oltre la scala elettrodebole, alla ricerca di simmetrie o dimensioni più estese di quelle che oggi caratterizzano il Modello Standard.

Una delle strade più promettenti per scoprire nuova fisica e dunque sviluppare una teoria completa delle interazioni fondamentali è lo studio della fisica nei collisori di particelle, come il Large Hadron Collider (LHC) di Ginevra.

3.2 L'apparato sperimentale

Nel Capitolo 3.1, è stato presentato il Modello Standard, la più nota teoria sulle interazioni fondamentali; le sue previsioni, così come i modelli teorici alternativi, possono essere testati confrontando i risultati teorici con i dati ottenuti dall'analisi delle collisioni protone-protone ad energie molto elevate. Il Large Hadron Collider (LHC) del CERN e gli esperimenti inseriti in esso, sono stati costruiti per registrare collisioni protone-protone in un ambiente controllato.

La fase iniziale delle collisioni protone-protone è dominata dalle funzioni di distribuzione dei partoni (PDF), che sono icostituenti del protone. I quark all'interno del protone interagiscono tra loro attraverso lo scambio di gluoni. Ad alte energie il continuo scambio di gluoni tra i tre quark genera un mare di coppie virtuali quark-antiquark da cui possono disperdersi altri partoni. Le distribuzioni dei momenti dei quark all'interno del protone sono espresse in termini di PDF. In pratica, le forme funzionali delle PDF dipendono dalla dinamica dettagliata del protone e devono essere dedotte dagli esperimenti.

3.2.1 LHC

Il Large Hadron Collider è il più grande e potente acceleratore di particelle del mondo. È progettato per raggiungere energie di centro di massa fino a 14 TeV con luminosità istantanee nominali che raggiungono $1 \times 10^{34} \text{ cm}^{-2} \text{ s}^{-1}$ per collisioni protone-protone e può anche far collidere ioni pesanti (Pb) con un'energia di 2.8 TeV per nucleone e una luminosità di $10^{27} \text{ cm}^{-2} \text{ s}^{-1}$. L'LHC è installato, in media, 100 metri sotto la superficie, in un tunnel lungo 26.7 km che è stato inizialmente costruito per l'acceleratore LEP.

LHC è progettato per far collidere due fasci di protoni o ioni che viaggiano in direzione opposta quasi alla velocità della luce in due camere a vuoto indipendenti. I fasci sono guidati intorno all'anello dell'acceleratore da un forte campo magnetico generato da elettromagneti superconduttori.

A LHC sono installati quattro esperimenti con diversi programmi di ricerca. Gli esperimenti sono gestiti da collaborazioni di scienziati provenienti da

istituti di tutto il mondo. I quattro grandi esperimenti di particelle installati nei punti di interazione (IP) di LHC sono:

- **ALICE**, "A Large Ion Collider Experiment", è un rivelatore dedicato alla fisica degli ioni pesanti. È progettato per studiare la fisica della materia per le interazioni forti a densità di energia estreme. Lo studio delle sue proprietà è un ingrediente chiave per una comprensione della cromodinamica quantistica.
- **ATLAS**, "A Toroidal LHC ApparatuS", è uno dei due rivelatori di uso generale. Indaga una vasta gamma di fisica, dalla ricerca del bosone di Higgs alle dimensioni extra e alle particelle che potrebbero costituire la materia oscura. È il più grande esperimento a LHC.
- **CMS**, "Compact Muon Solenoid", è l'altro rivelatore generico a LHC di cui se ne descriveranno alcuni dettagli. Ha un ampio programma di fisica e sebbene abbia obiettivi scientifici simili all'esperimento ATLAS, utilizza diverse soluzioni tecniche.
- **LHCb**, "Large Hadron Collider beauty", si concentra su misure precise delle proprietà e dei decadimenti dei quark bottom e charm, al fine di studiare le differenze tra materia ed antimateria che hanno portato all'attuale asimmetria tra le due.

Una vista schematica del complesso dell'acceleratore del CERN con i quattro esperimenti è mostrata nella Figura 3.3.

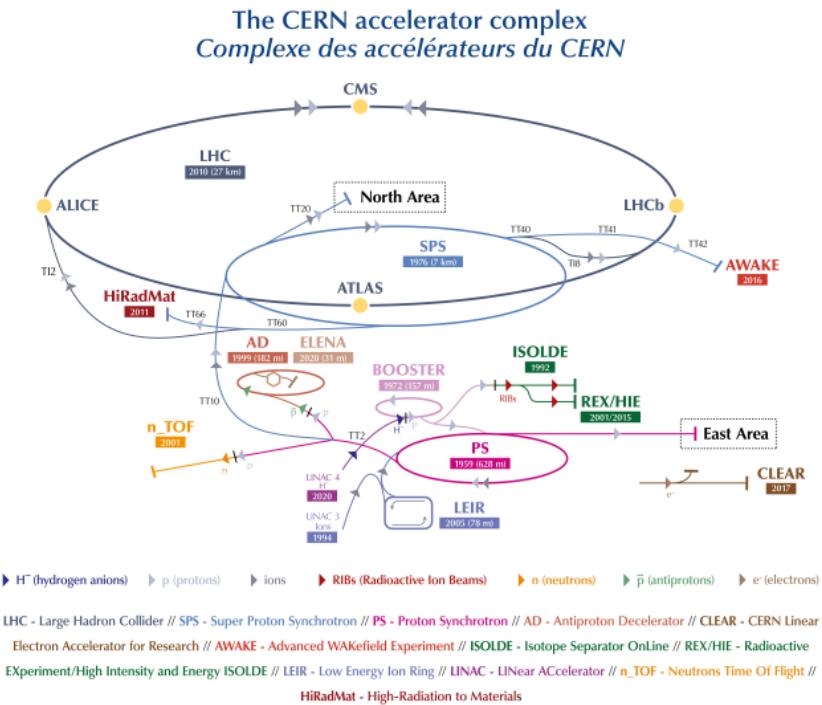


Figura 3.3: Visione schematica del complesso di accelerazione del CERN con i suoi principali esperimenti (Mobs 2019).

Per raggiungere energie di fascio dell'ordine dei TeV, i protoni devono passare attraverso diversi componenti del complesso acceleratore del CERN. Dopo l'estrazione di un fascio di protoni a bassa energia, i protoni vengono prima accelerati in un acceleratore lineare (LINAC). Come si vede in Figura 3.3 i fasci provenienti dal LINAC sono ulteriormente accelerati nei quattro anelli *Proton Synchrotron Booster* (PSB) fino a 1.4 GeV, e nella fase successiva dal *Proton Synchrotron* (PS) fino a 26 GeV. Infine, il *Super Proton Synchrotron* (SPS) fornisce protoni per l'LHC con un'energia di 450 GeV e li inietta in direzioni opposte nell'anello di LHC. Successivamente a questa prima fase i fasci vengono ulteriormente accelerati con un sistema a radiofrequenza superconduttore fino a un'energia di 7 TeV.

Una volta che l'acceleratore LHC è fornito dei fasci a sufficiente energia, questi ultimi possono essere usati per registrare le collisioni negli IP per diverse ore fino a quando la qualità dei fasci non si degrada.

Per piegare i protoni e condurli in una certa traiettoria, viene generato un campo magnetico di 8.3 T da magneti a dipolo superconduttori in Niobio-Titanio. In totale, per controllare i due fasci di protoni, vengono usati 8000 magneti superconduttori che sono raffreddati da un sistema di raffreddamento ad elio.

Più nello specifico, le particelle nei fasci sono immagazzinate in piccoli pacchetti chiamati bunches che si scontrano nei punti di interazione dove sono installati gli esperimenti principali. I prodotti risultanti dalle collisioni di particelle sono registrati con i diversi rivelatori. Poiché solo una piccola frazione delle collisioni è di interesse, è importante avere un tasso di eventi molto alto, che è caratterizzato dalla luminosità \mathcal{L} . Quest'ultima è definita come il numero di protoni collidenti per unità di tempo e di area (misurata in $cm^{-2}s^{-1}$) che è una misura di flusso di particelle, parametro caratteristico dell'acceleratore. Si tratta di una grandezza che dipende esclusivamente dalle caratteristiche della macchina acceleratrice e dalle particelle coinvolte nelle collisioni.

Con il termine sezione d'urto si indica, in modo del tutto generale, una grandezza che quantifica la probabilità che un dato evento si verifichi. Nello specifico la sezione d'urto di una certa particella, o più comunemente di una coppia di particelle, come top-antitop, è la probabilità che, fissata l'energia nel sistema del centro di massa dei fasci, dalla loro collisione si generi quella particella. Si può definire R_{tot} , il numero di eventi nell'unità di tempo, come:

$$R_{tot} = \frac{dN_{tot}}{dt} = \sigma_{tot}^{pp} \times \mathcal{L}$$

dove σ_{tot}^{pp} è la sezione d'urto totale protone-protone misurata in cm^2 e \mathcal{L} è la luminosità instantanea. Generalmente, la sezione d'urto di un determinato processo viene misurata fissando una serie di tagli di selezione che hanno lo scopo di ottimizzare il rapporto segnale-rumore. La sezione d'urto sperimentale si ottiene dall'equazione

$$\sigma = \frac{N_{sig}}{\epsilon \int \mathcal{L} dt} = \frac{N_{obs} - N_{bkg}}{\epsilon \int \mathcal{L} dt}$$

dove N_{sig} è il numero di eventi stimati del segnale, N_{obs} è il numero di eventi osservati dai dati dopo aver applicato i tagli di selezione, N_{bkg} è il numero di eventi stimati (via simulazione Monte Carlo o con i dati) di background,

ϵ è l'efficienza della selezione, misurata dalle simulazioni Monte Carlo per il particolare processo che stiamo considerando e infine $\int \mathcal{L} dt$ è la luminosità integrata nel tempo di durata della presa dati. La *luminosità integrata* è l'integrale della luminosità istantanea ed è strettamente legato alla misura della sezione d'urto. La conoscenza della luminosità integrata e della sezione d'urto permette di calcolare il numero di eventi atteso di quella particella, prodotto nell'esperimento: per farlo ci si serve della relazione fondamentale scritta sopra. La sezione d'urto è un valore che può essere previsto a partire dai modelli teorici, perciò lo scopo degli esperimenti è quello di verificare se c'è un accordo con i valori calcolati.

3.2.2 CMS

Il rivelatore *Compact Muon Solenoid* (CMS) al CERN LHC è un rivelatore generico progettato principalmente per cercare segnale di nuova fisica nelle collisioni protone-protone e ioni pesanti. L'apparato sperimentale CMS ha un diametro di 15m, è lungo 28.87m ed ha un peso totale di oltre 14000 tonnellate. È composto di sottorivelatori (*sub-detector*) capaci, ciascuno, di misurare un particolare tipo di particelle e le sue proprietà.

Il rivelatore CMS ha una geometria cilindrica che è azimutalmente (ϕ) simmetrica rispetto alla direzione dei fasci. La struttura è basata su un magnete soleinoidale costituito da una bobina superconduttrice che genera un campo magnetico; la parte centrale cilindrica è chiamata *barrel*, mentre le estremità sono chiamate *endcap*. In Figura 3.4 viene riportata una sua rappresentazione tridimensionale.

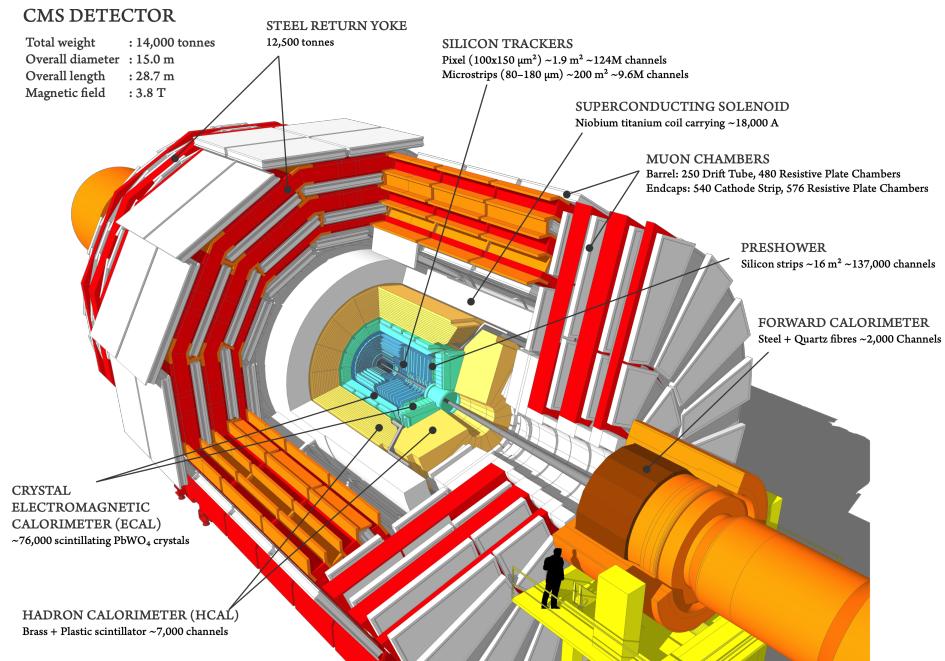


Figura 3.4: Rappresentazione tridimensionale di CMS (Sakuma e McCauley 2014).

Il sistema di coordinate usato per descrivere il rivelatore CMS ha il suo centro all'interno del rivelatore, nel punto di interazione nominale. L'asse x punta verso il centro dell'anello LHC e funge da coordinata orizzontale, mentre l'asse y punta verticalmente verso l'alto, rappresentando la coordinata verticale e l'asse z è diretto lungo la direzione del fascio. L'angolo azimutale, ϕ , è l'angolo di rotazione attorno all'asse z , con origine individuata sull'asse x e crescente in senso orario guardando nella direzione positiva dell'asse z . Infine l'angolo polare θ è definito come angolo di rotazione attorno all'asse x , con l'origine sull'asse z e crescente in senso orario.

Alcune grandezze utili per descrivere gli eventi all'interno dei rilevatori sono:

- η : pseudorapidità, grandezza spaziale utilizzata per descrivere l'angolo tra un vettore e l'asse di collisione dei fasci. È definita come $\eta = -\ln \tan(\frac{\theta}{2})$;
- p_T : quantità di moto trasversa. È definita come $p_T = p \sin \theta$ dove p è il modulo del vettore quantità di moto di una particella;

- E_T : energia trasversa. È definita in modo analogo a p_T dalla relazione $E_T = E \sin \theta$, dove E è l'energia di una particella o di un getto di particelle.

L'apparato sperimentale di CMS è studiato in maniera da garantire la maggiore copertura angolare possibile intorno al punto di interazione, in modo che si perda il meno possibile dell'informazione. L'energia delle particelle che interagiscono è nota ed è determinata dai parametri dell'acceleratore, mentre l'energia delle particelle prodotte dalla collisione è calcolabile a partire dai segnali che esse rilasciano nel rivelatore.

Come detto, CMS si serve di sub-detector specializzati a segnalare il passaggio di una specifica particella. A seguire si descrivono i sottorivelatori, dall'interno all'esterno rispetto al punto di collisione dei fasci. In Figura 3.5 si riporta la sezione trasversa dei rivelatori.

Le particelle che sono prodotte in una collisione nella regione di interazione passano prima attraverso un tracker, o tracciatore, in cui le traiettorie e i vertici delle particelle cariche sono ricostruite dai segnali negli strati sensibili. Ci sono tracker a pixel, fondamentali per la ricostruzione delle particelle con vita media molto piccola e tracker al silicio. Il silicio di cui sono costituiti questi ultimi, risulta eccitato quando attraversato dalle particelle cariche, formando elettroni carichi, i quali, per effetto del campo magnetico che devia la loro traiettoria vanno a contatto con dei sensori dando luogo ad un impulso elettrico. I segnali depositati e opportunamente amplificati, permettono di ricostruire la traiettoria delle particelle in modo molto preciso.

Dopo il tracker vi sono i calorimetri. Durante l'interazione con il calorimetro le particelle incidenti possono dare luogo a sciami di particelle secondarie che vengono assorbite nello strumento.

In particolare il più interno è il calorimetro elettromagnetico (ECAL) che è situato all'interno del solenoide superconduttore e permette di identificare e ricostruire fotoni ed elettroni. Per misurare l'energia con alta precisione, è necessaria un'accurata calibrazione del calorimetro.

Il secondo calorimetro, quello adronico (HCAL), ha lo scopo di misurare l'energia e la posizione degli adroni, come pioni, kaoni, protoni e neutroni. Al contempo fornisce la misura, indiretta, di particelle non interagenti, non cariche come i neutrini attraverso l'energia trasversa mancante.

A causa della non ermeticità completa del rivelatore e del fatto che in HCAL gli adroni decadono in altre particelle, è possibile che parte dell'energia sfugga attraverso particelle non interagenti o ci siano perdite laterali. Per ridurre tale problema i calorimetri sono costituiti da strati sfalsati tra loro, alternati di materiale assorbitore e scintillatore fluorescente, in modo da non avere zone in cui una particella potrebbe passare inosservata.

Il terzo strato è costituito dal magnete solenoidale, composto da spire di bobina superconduttrice che producono un campo magnetico uniforme quando vi scorre una corrente elettrica. Il solenoide è lungo 13 m, con diametro 5.9 m e pesa 12000 tons. Il raggio del magnete è sufficientemente grande per contenere il tracker e i calorimetri, mentre all'esterno di esso c'è il sistema di ritorno di flusso e il rivelatore di muoni. Un vantaggio di questa configurazione è l'assenza di perdita di energia delle particelle da interazioni con il materiale prima di giungere ai calorimetri. Il campo magnetico generato è cruciale per la misura della quantità di moto delle particelle cariche, poiché la misurazione si basa sulla flessione delle loro traiettorie. La combinazione di un alto campo magnetico e di un'alta precisione nella risoluzione spaziale del tracker assicura un'alta risoluzione della quantità di moto. La direzione della curvatura viene utilizzata anche per determinare la carica elettrica della particella.

All'esterno del solenoide, sono installati ulteriori strati di tracciamento che permettono di misurare i muoni che attraversano i calorimetri con poche o nessuna interazione. Molte delle quantità misurate da CMS includono i muoni, quindi esso è costruito con sub-detector per identificare e misurare i momenti e la carica dei muoni ad alta energia, che sono l'unico tipo di particelle cariche che possono attraversare tutti gli altri rivelatori senza una significativa perdita di energia. Il sistema è composto da rivelatori gassosi che inseriti opportunamente permettono ad un muone in transito di essere rilevato in più punti lungo il percorso. La posizione del muone che attraversa l'apparecchio è determinata combinando le informazioni che derivano da tre tipi diversi di rivelatori gassosi, ottenendo una misura accurata della coordinata alla quale il muone attraversa il volume del gas.

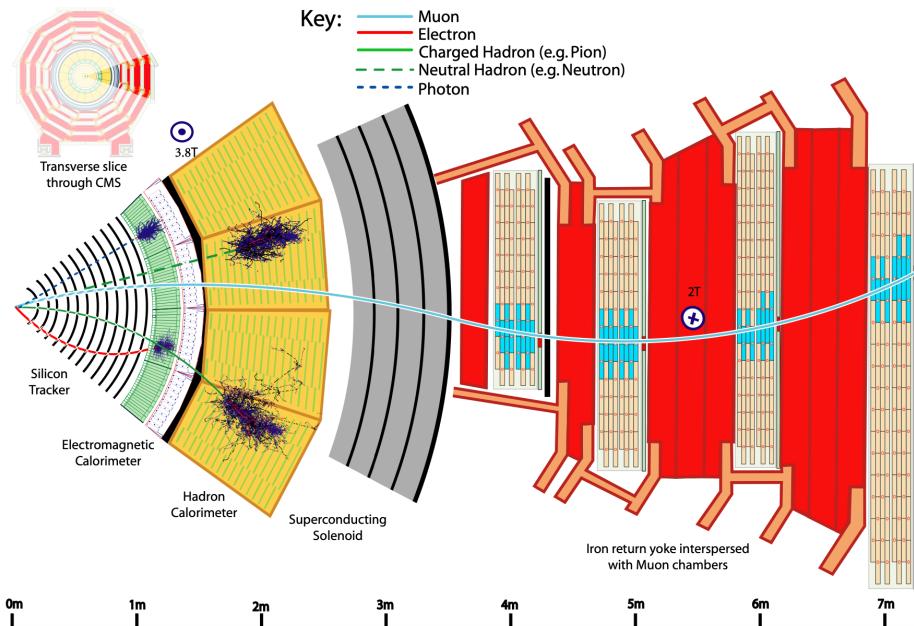


Figura 3.5: Sezione del rilevatore CMS con i principali sistemi di rilevazione (Sirunyan et al. 2017).

3.2.3 Acquisizione dei dati

Come già detto, LHC è progettato per far collidere protoni ad un'energia di centro massa di 14 TeV e una luminosità di $1 \times 10^{34} \text{ cm}^{-2} \text{ s}^{-1}$; solo una piccola frazione delle collisioni contiene eventi di interesse per il programma di fisica CMS, e a causa dell'alto tasso di eventi solo una piccola frazione di questi può essere memorizzata per l'analisi offline. E' dunque utilizzato un sistema di trigger e di acquisizione dati (*data-acquisition system*, DAQ), il primo che seleziona gli eventi interessanti dalla massa di particelle delle collisioni protone-protone e il secondo che acquisisce e memorizza le informazioni sotto forma digitale. Gli esiti di queste due prime fasi definiscono dei dati che sono poi processati offline utilizzando l'algoritmo *Particle Flow* (PF).

Per ottenere questo, CMS utilizza due livelli di selezione, che sfruttano algoritmi di crescente complessità per ridurre la frequenza degli eventi selezionati. Il primo livello è un trigger di carattere hardware ed effettua una prima selezione basandosi sui dati letti dal rilevatore di muoni e dai calorimetri per

selezionare gli eventi più interessanti in un intervallo di tempo fisso, inferiore a $4 \mu\text{s}$. Il secondo livello consiste nel trigger di alto livello (HLT), esso esegue ulteriori tagli sfruttando l'intera informazione sull'evento, comprese le informazioni dal tracker di silicio. Le soglie del primo livello di trigger sono regolate durante l'acquisizione dei dati, in risposta al valore della luminosità istantanea di LHC, al fine di limitare il tasso di uscita a 100 kHz, che è il limite superiore imposto dall'elettronica di lettura di CMS. L'HLT, implementato nel software, raffina ulteriormente la purezza degli oggetti fisici e seleziona un tasso medio di 400 Hz per la memorizzazione offline. Il tasso di uscita complessivo del trigger L1 e HLT può essere regolato preriscalando il numero di eventi che passano i criteri di selezione degli algoritmi specifici. Oltre a raccogliere i dati di collisione, i sistemi di trigger e quelli di acquisizione dati registrano informazioni per il monitoraggio del rivelatore.

La ricostruzione offline degli eventi per l'esperimento CMS, che avviene a seguito di quanto appena descritto ed è basata sull'algoritmo *Particle Flow* (Sirunyan et al. 2017). L'algoritmo si basa su una ricostruzione dell'evento globale correlando gli elementi di base (cioè tracce e cluster) ottenuti da tutti i sistemi di sub-detector, al fine di ricostruire tutte le particelle dell'evento e misurarne le proprietà. Le particelle sono ricostruite localmente nei diversi sottorivelatori che forniscono gli elementi costitutivi per la descrizione complessiva dell'evento. L'informazione completa del rivelatore viene sfruttata anche per fornire l'identificazione delle particelle. Gli oggetti finali che vengono resi disponibili per le analisi, sono oggetti del flusso di particelle come getti (jets), fotoni, elettroni, adroni carichi, neutri, muoni, tau, così come i vertici di interazione e l'energia trasversa mancante. Una grande sfida nella ricostruzione degli oggetti fisici è quella di mitigare gli effetti di *pileup*: interazioni protone-protone aggiuntive. Anche le particelle prodotte in queste interazioni sono registrate nel rivelatore, pertanto sono state implementate tecniche apposite di attenuazione degli effetti di *pileup*.

Le simulazioni di eventi sono cruciali sia per la progettazione e l'aggiornamento del rivelatore CMS, che per l'analisi e l'interpretazione dei dati registrati. Le simulazioni indicano quale segno una nuova particella lascerebbe nel rivelatore se esistesse. Tipicamente la simulazione del rivelatore consiste in tre passi: il primo passo è la simulazione delle interazioni di una particella

che attraversa il rivelatore e i depositi di energia risultanti nel rivelatore, il secondo passo è la digitalizzazione, dove i depositi di energia nel rivelatore sono convertiti in segnali digitali e il terzo passo è la ricostruzione, dove i segnali digitali sono successivamente ricostruiti in oggetti per l'analisi fisica (Elvira 2017). CMS ha sviluppato un quadro completo di simulazione del rivelatore che simula le interazioni tra le particelle che attraversano il rivelatore CMS in grande dettaglio grazie all'utilizzo di software che consentono la modellazione dettagliata della geometria del rilevatore e delle interazioni tra le particelle. In particolare viene utilizzato il pacchetto software chiamato CMSSW per produrre eventi di simulazione Monte Carlo. Questo aspetto sarà di particolare rilevanza nelle analisi presentante nel Capitolo 4.

3.2.4 La sezione d'urto del top

Il quark top è una delle particelle fondamentali del Modello Standard, osservato per la prima volta nel 1995 all'acceleratore Tevatron presso i laboratori del Fermilab negli Stati Uniti. È la particella più massiva prevista dal Modello Standard ed è caratterizzato da un tempo di decadimento molto basso, dell'ordine di $10^{-25} s$ che gli permette di decadere tramite un processo elettrodebole. Attraverso lo studio dei suoi diversi prodotti di decadimento è possibile risalire a molte delle sue proprietà. Effettuare misure accurate di queste ultime permette di consolidare la teoria del Modello Standard o di andare alla ricerca di indizi che potrebbero condurre a scenari di nuova fisica. Ci si aspetta infatti che il quark top giochi un ruolo speciale in varie estensioni del Modello Standard a causa della sua elevata massa.

Le collisioni protone-protone ad LHC creano tipicamente uno scontro fra gluoni dal quale si ottiene una coppia di quark top per interazione forte; si è interessati a questo processo perché potrebbe esserci una nuova particella supplementare che decade in $t\bar{t}$, prodotta dai due gluoni. Se consideriamo il caso standard di produzione $t\bar{t}$ il grafico della massa invariante dei due quark top è decrescente, perché più è elevata la loro massa meno probabilità c'è che la collisione produca quella coppia $t\bar{t}$, mentre se ci fosse una particella con una sua definita massa si vedrebbe un picco sulla massa invariante in corrispondenza

della massa di questa particella che quindi renderebbe più probabile la presenza della coppia $t\bar{t}$.

La massa, in particolare quella del quark top, è un parametro libero nel SM e deve essere dedotta da misure sperimentali. Può essere misurata attraverso misure dirette, che si basano sulla ricostruzione cinematica dello stato finale del top o indirette attraverso relazioni con altre cinematiche osservabili, come la sezione d'urto $t\bar{t}$ ad una data energia di centro di massa. Attualmente le misure di massa più precise provengono da misure dirette. Le particelle ricostruite allo stato finale sono confrontate con le previsioni dei generatori di eventi Monte Carlo per determinare il valore di massa del quark top che meglio descrive i dati. In questo approccio, la massa del top misurata corrisponde al parametro implementato nel generatore MC e le misure dirette producono una media di $m_{top}^{MC} = 173.34$ GeV con una precisione fino a 0.5 GeV.

Tutti gli studi diretti sul quark top sono effettuati grazie ad acceleratori in grado di raggiungere energie nel centro di massa sufficienti a produrre particelle così massive. Tali energie sono state raggiunte prima dal Tevatron e poi al CERN di Ginevra dall'acceleratore di protoni LHC.

Accanto ai processi di produzione attraverso interazione forte ci sono quelli di natura elettrodebole. I primi conducono alla formazione di coppie top-antitop, i secondi, principalmente, alla produzione di top singolo.

Come si è detto, i quark sono in grado di interagire sia mediante interazioni deboli che forti. I processi di produzione forti, nelle collisioni protone-protone ad LHC, sono quelli più frequenti data la maggiore sezione d'urto che li caratterizza. Una rappresentazione di produzione del top per interazione forte è rappresentato in Figura 3.6.

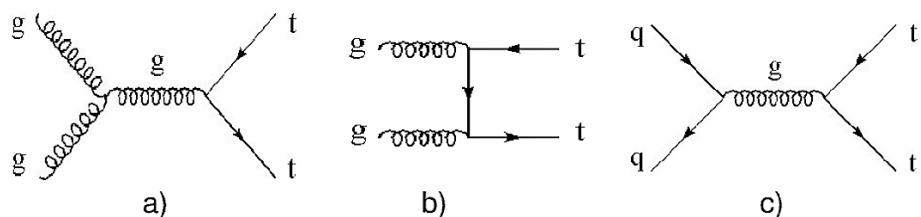


Figura 3.6: Diagrammi di Feynman per i processi di produzione forte di coppie $t\bar{t}$.

La frazione di decadimento del quark top in un certo canale si calcola considerando i branching ratio:

$$R = \frac{BR(t \rightarrow Wb)}{BR(t \rightarrow Wq)}$$

La coppia $t\bar{t}$ decade nella quasi totalità dei casi in Wb , il cui valore di R è pari al 99,9%. Il decadimento $t \rightarrow Wb$ ha tre possibili categoria di decadimento, classificabili a seconda del tipo di decadimento dei due bosoni W e quindi dal numero di getti e leptoni presenti nello stato finale:

- canale adronico ($R = 46,2\%$): i bosoni W decadono in coppie quark-antiquark;
- canale jet+leptone ($R = 43,5\%$): un bosone W decade in leptone e neutrino, l'altro in una coppia quark antiquark;
- canale di-leptonico ($R = 10,3\%$): entrambi i bosoni W decadono in leptoni e neutrini.

Dunque per misurare la sezione d'urto del top $\sigma_{t\bar{t}}$ sono disponibili diversi stati finali della coppia $t\bar{t}$. Il canale jet+leptoni è quello più conveniente in termini di purezza, ossia numero di eventi di segnale sul totale degli eventi, e di branching ratio. La richiesta di un leptone (elettrone o muone) nello stato finale è di estrema importanza per l'abbattimento del background dominante dovuto ad eventi con molti getti che hanno una sezione d'urto molto elevata. Il canale adronico è quello col branching ratio più elevato, ma con la più alta contaminazione. D'altra parte il canale di-leptonico è quello più pulito poiché richiede due leptoni, ma ha il branching ratio più basso. Il canale jet+leptoni consente una buona purezza combinata ad un elevato branching ratio.

Le misure più precise a LHC sono ottenute nel canale di-leptonico (Cristinziani e Mulders 2017), in particolare nello stato finale $e\mu$ poiché è essenzialmente privo di fondo.

CMS e ATLAS hanno anche misurato la sezione d'urto in eventi con un leptone τ decaduto adronicamente in entrambi i canali $\tau_{\text{had}} + \ell$ e $\tau_{\text{had}} + \text{jets}$.

L'analisi CMS del canale di decadimento $\tau_{\text{had}} + \text{jets}$ (CMS Collaboration 2013) è di particolare importanza per questa tesi, dal momento che la sua applicazione sarà presentata nel Capitolo 4.

Capitolo 4

Applicazione di INFERNO ai dati di CMS

Come discusso nel Capitolo 3, c'è una varietà di approcci innovativi che affronta il problema della costruzione di statistiche riassuntive con l'apprendimento automatico in presenza di parametri di disturbo; ciascun metodo innovativo è utile, per studiarne le performance, applicarlo ad un set di dati il più realistico possibile.

L'obiettivo di questa sezione è quello di applicare la procedura INFERNO ad un set di dati reali, raccolti dall'esperimento CMS di LHC al CERN per la ricerca del quark top (CMS Collaboration 2013), al fine di provare i vantaggi inferenziali che si ottengono utilizzando l'*inference-aware* che caratterizza INFERNO. In particolare l'applicazione avviene su dati semplificati e di Asimov per rispondere a due quesiti: quale effetto ha la porzione di dati tra quelli disponibili sulle performance e quale impatto hanno i parametri di tuning della rete neurale sull'efficienza delle stime. Il codice utilizzato per svolgere queste analisi è riportato in Appendice B. Uno studio più approfondito e ampio sui questi dati è condotto da Lukas Layer nel suo lavoro di dottorato (Layer 2022).

La procedura INFERNO viene confrontata con una procedura BCE entrambe basate su una rete neurale capace di discriminare il segnale dal background. Dagli output delle reti neurali si costruisce una stima del rapporto di verosimiglianza con cui si conduce un'analisi statistica estraendo una misura, con relativa incertezza, del numero di eventi di segnale nel campione. La quan-

tità stimata, usando la relazione fondamentale, consente di ottenere una stima della sezione d'urto del processo di interesse.

4.1 Dati sperimentali

I dati che si usano nell'applicazione sono pubblici e disponibili nel database Open Data del CERN. Essi sono stati utilizzati nell'analisi pubblicata da CMS Collaboration 2013 volta a misurare la sezione d'urto dell'evento $t\bar{t}$ nel canale $\tau+\text{jets}$ in collisioni protone-protone a $\sqrt{s} = 7\text{TeV}$ (TOP-11-004).

La misura CMS della sezione d'urto di produzione di $t\bar{t}$ nel canale $\tau+\text{jets}$ utilizza eventi che contengono un getto identificato come un leptone tau che decade adronicamente e almeno quattro getti, almeno uno dei quali è identificato come proveniente da un quark b . Il diagramma di Feynman di questo processo è mostrato in Figura 4.1.

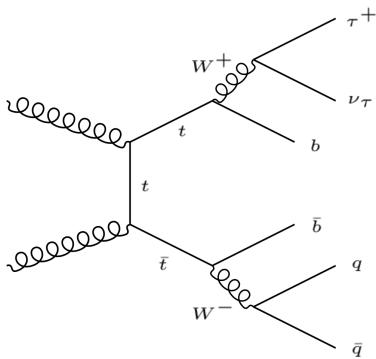


Figura 4.1: Diagramma di Feynman del processo di decadimento di una coppia $t\bar{t}$ nel canale $\tau+\text{jets}$.

Nel Modello Standard la frazione di branching di un decadimento del top in un bosone W e un quark b è vicina al 100%: ci si aspetta che il 98% delle coppie $t\bar{t}$ prodotte portino a questo stato finale.

La sezione d'urto della produzione di $t\bar{t}$ a $\sqrt{s} = 7\text{TeV}$ misurata in questa analisi è

$$\sigma_{tt} = 152 \pm 12 \text{ (stat.)} \pm 32 \text{ (syst.)} \pm 3 \text{ (lum.) pb}$$

che è coerente con la previsione del Modello Standard.

Questa analisi è di interesse per l'applicazione di INFERNO poiché, oltre al fatto che i dati sono pubblici, è dominata da errori sistematici molto ampi, tre volte più grandi dell'errore statistico, e utilizza una rete neurale per separare il segnale dal background che definisce una statistica riassuntiva utilizzabile come input per l'inferenza. Perciò essa costituisce un caso d'uso in cui la tecnica INFERNO può eventualmente migliorare la precisione della misurazione. Le due procedure utilizzano la stessa metodologia, rete neurale, per separare il segnale dal fondo, con l'obiettivo di misurare un particolare stato finale della sezione d'urto della coppia $t\bar{t}$ prodotta ad LHC per scontro di protoni.

Nel database Open Data del CERN sono disponibili quasi tutti i dati del primo run di LHC (2010-2012) con i corrispondenti campioni simulati e sono forniti in Lassila-Perini et al. 2021 gli strumenti per analizzarli. Per la presa dei dati, la rielaborazione degli eventi e l'analisi viene utilizzato il software CMSSW. I set di dati simulati sono generati da programmi generatori Monte Carlo, sottoposti a simulazione del rivelatore utilizzando, ancora una volta, CMSSW, e sono successivamente elaborati nello stesso formato dei dati di collisione.

In generale però, la riproduzione esatta di un'analisi e dei relativi dati è molto difficile a causa dei continui miglioramenti che vengono apportati nell'intero apparato oltre che negli algoritmi. Pertanto in questo lavoro si utilizzano i medesimi dati con un certo margine di differenza rispetto all'analisi TOP-11-004. L'intera procedura di riproduzione dell'analisi è stata condotta da Lukas Layer (Layer 2022) e si allontana dall'obiettivo di questa sezione della tesi che invece mira ad applicare INFERNO a dei dati veri e valutarne alcuni suoi aspetti. Lukas Layer è riuscito a replicare quasi esattamente gli stessi esiti dell'analisi di CMS consentendo il suo utilizzo come confronto.

4.2 L'analisi pubblicata

Nell'analisi condotta da CMS Collaboration 2013, per discriminare gli eventi di segnale $t\bar{t}$ da quelli di fondo di QCD viene addestrata un classificatore basato su una rete neurale che minimizza la perdita della *binary cross entropy* (BCE). Il suo output viene poi utilizzato come input dell'analisi statistica. Fu utilizzato l'algoritmo MultiLayerPerceptron (MLP) del pacchetto TMVA The-

rhaag 2010, che era all'avanguardia quando è stata effettuata questa analisi, mentre Layer, per addestrare la rete neurale, utilizza il pacchetto PYTORCH (Paszke et al. 2019). L'input del classificatore multivariato consiste in otto variabili di alto livello normalizzate, per avere media nulla e varianza unitaria, costruite tramite le variabili cinematiche dei getti selezionati e capaci di ben discriminare il segnale dal fondo. Nella riproduzione sono state utilizzate le stesse variabili di input dell'analisi originale, ad eccezione di una che non viene considerata da Layer poiché è difficile da riprodurre e nell'analisi originale ha dimostrato essere di scarsa importanza. Layer mostra che, servendosi di un'indice dell'impatto medio sulla grandezza dell'output del modello, la variabile cinematica più importante è l'energia trasversa mancante.

L'analisi originale stima il fondo e la frazione di segnale del processo $t\bar{t}$ tramite un fit di massima verosimiglianza della distribuzione dell'istogramma prodotto come statistica sommario dalla rete neurale. I background minori, ossia $W/Z+Jets$ e top singoli, vengono fissati e sottratti dai dati prima dell'addestramento della rete.

Le incertezze sistematiche sono calcolate ripetendo il fit con modelli variati di $\pm 1\sigma$ della rispettiva fonte sistematica e sommando le differenze rispetto al modello nominale. Questa procedura è chiamata anche *cut variation* ed è stato un approccio standard nelle prime analisi di LHC. Tuttavia, questa procedura è considerata problematica da un punto di vista statistico, poiché ignora correlazioni tra le incertezze sistematiche e il parametro di interesse e di conseguenza non possono essere definite in modo significativo quantità statistiche, come gli intervalli di confidenza. Per questo viene usato il metodo della verosimiglianza profilo, che incorpora tutti i parametri rilevanti nella verosimiglianza e permette di stimare intervalli di confidenza corretti. Il principio di INFERNO è quello di prendere in considerazione tutte le incertezze rilevanti durante l'addestramento della una rete neurale, rendendola sensibile all'analisi statistica successiva.

Nella riproduzione dell'analisi le incertezze e i coefficienti di correlazione sono calcolati con il pacchetto *minuit*. La stima dell'errore standard è condotta con HESSE che calcola l'intera matrice della derivata seconda, producendo gli errori dei parametri e le correlazioni tra essi. Tuttavia, poiché HESSE assume l'approssimazione parabolica attorno al minimo della verosimiglianza profilo,

per calcolare i corretti errori positivi e negativi del parametro di interesse viene utilizzato l'algoritmo MINOS (Murtagh e Saunders 1978). MINOS, segue la funzione di verosimiglianza fuori dal minimo per trovare i valori della funzione corrispondenti al 68% di confidenza (tipica confidenza usata in fisica delle particelle). Questo equivale a trovare i punti del parametro di interesse che corrispondono al valore 1 della log-verosimiglianza profilo relativa moltiplicata per 2 con segno negativo.

Seguendo l'analisi originale TOP-11-004, è stato aggiunto un parametro di scala per il modello QCD che permette a questo processo di fluttuare liberamente nel fit; pertanto è stata eliminata l'incertezza di normalizzazione corrispondente al mis-tagging del fondo QCD.

L'intensità del segnale, qui indicata con μ , con le relative incertezze sistematiche è calcolata essere:

$$\mu = 0.99 + 0.25 - 0.19 \text{ (syst.)} + 0.09 - 0.09 \text{ (stat.)}$$

dove l'incertezza statistica è valutata fissando le incertezze sistematiche al loro miglior valore di adattamento e ripetendo l'adattamento. La presenza di incertezze sistematiche causa una perdita di informazione sulla misura della quantità di interesse e questo si riflette in una log-verosimiglianza profilo molto larga, poco schiacciata attorno al minimo.

In Figura 4.2 è mostrato l'accordo tra i dati e le simulazioni per il classificatore a rete neurale usato in TOP-11-004. Mappando il modello del background disponibile sopra all'esito della classificazione si evidenzia un eccesso di eventi: è presente un piccolo accumulo di eventi per valori alti dell'output della rete neurale, dovuto al contributo del segnale, che è molto piccolo. Essendo il segnale molto piccolo è molto influenzato dagli errori di definizione del modello del background, si noti infatti che se il modello di background avesse una coda un po' più pesante non si riuscirebbe a distinguere significativamente il segnale. La ridotta conoscenza del modello di background si traduce nell'introduzione in esso degli errori sistematici, che riflettono l'effetto di tutte quelle quantità che non si conoscono.

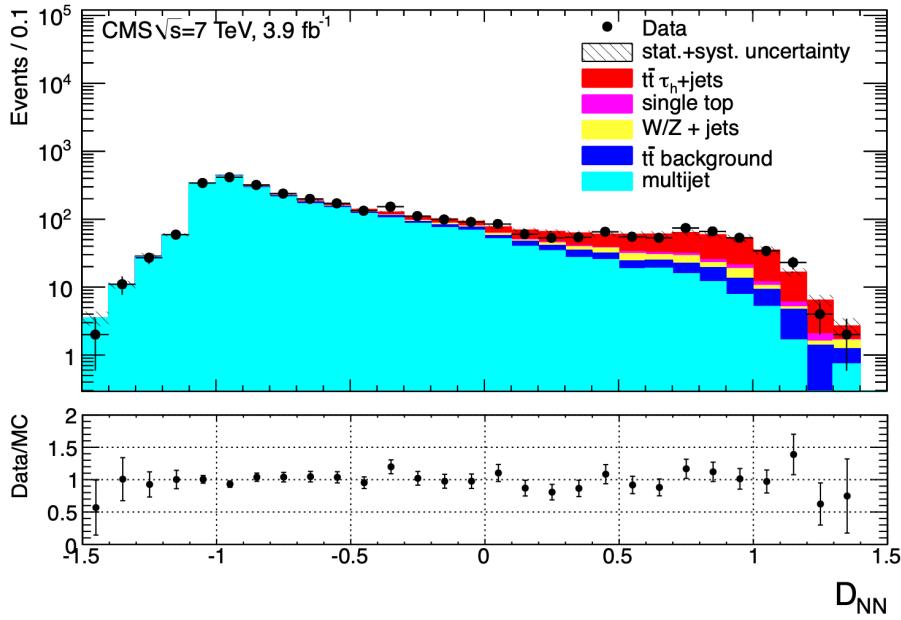


Figura 4.2: Accordo tra dati e simulazioni per il classificatore di rete neurale usato in TOP-11-004 (CMS Collaboration 2013).

4.3 I parametri di disturbo

In questa analisi sia il modello di segnale basato su osservazioni simulate che il modello di fondo guidato dai dati non sono perfettamente noti, quindi per tenere conto di tale mancanza devono essere considerati una serie di parametri di disturbo. Come già discusso ogni parametro di disturbo, che può influenzare il segnale, la componente di fondo o entrambi, porta ad un aumento dell'incertezza sul parametro di interesse, aumentando l'ampiezza totale dell'intervallo di confidenza. L'effetto di questi parametri nelle stime statistiche finali dei parametri di interesse è indicato come incertezza sistematica.

Ci sono principalmente due tipi di incertezze sistematiche che si incontrano tipicamente nell'analisi HEP. Il primo tipo sono semplici incertezze di normalizzazione, chiamate anche moltiplicative, che influenzano la normalizzazione di un processo; il secondo tipo di sistematica influenza sia la forma che la normalizzazione di un processo, a cui ci si riferisce anche come sistematica "ShapeNorm". Le incertezze di normalizzazione possono essere incluse nella

verosimiglianza profilo tramite parametri di disturbo (Conway 2011) come per esempio la luminosità integrata. Nelle applicazioni pratiche è spesso utile vincolare ogni parametro di disturbo ad una gaussiana di media nulla e deviazione standard unitaria e introdurre una funzione di normalizzazione che dipende dal parametro di disturbo e dalla sua incertezza. In termini Bayesiani, le funzioni di vincolo possono essere interpretate come densità a priori sui parametri di disturbo e quindi incluse in modo moltiplicativo nella verosimiglianza. Nel caso in cui il processo abbia incertezze asimmetriche, ossia una variazione di $+1\sigma$ è diversa da una di -1σ , si utilizza l'approccio di combinazione di CMS (ATLAS Collaboration e LHC Higgs Combination Group 2011) che prevede di definire una distribuzione log-normale asimmetrica liscia che si basa sull'interpolazione delle deviazioni.

Le sistematiche "ShapeNorm" causano invece anche una distorsione complessiva della forma delle caratteristiche degli eventi osservati (Conway 2011) oltre che ad una modifica della normalizzazione complessiva del processo. Un esempio è la *jet energy scale* (JES) che sposta tutte le energie del getto di un evento nella stessa direzione. Queste distorsioni possono essere modellate cambiando i parametri nella simulazione MC alzando e abbassando il valore della sistematica di una deviazione standard e ricalcolando le distribuzioni: si ottengono tre misure della forma e della normalizzazione delle distribuzioni che vengono poi interpolate.

Nell'analisi TOP-11-004 vengono definiti una serie di errori sistematici che influiscono sul processo e vengono stimati i loro impatti sulla misura finale (Tabella 4.1). Le principali fonti di incertezze sistematiche sono dovute alle PDF, alla Jet Energy Scale (JES), alla correzione dell'energia tau e all'identificazione del tau.

Uno degli errori sistematici più impattanti sulla misura finale della sezione d'urto sui dati di CMS è la JES. Questo errore ha origine nella precisazione del modello della simulazione del detector il quale non riesce a riprodurre con fedeltà la misura dell'energia dei getti adronici (jets) che si opera nei dati sperimentali. I getti infatti sono costituiti da particelle cariche, di cui si riesce a definire l'energia nel tracker e da particelle neutre di cui invece non si riesce a calcolare l'energia; per questo motivo l'energia misurata del jets è diversa da quella reale. Come già detto nel Capitolo 3, per ridurre questo problema si

utilizza l'algoritmo “particle flow”, che usa contemporaneamente le informazioni provenienti dal tracker e dal calorimetro per assegnare le energie alle diverse particelle.

Per ottenere una stima più precisa si simulano i processi fisici di background e di segnale e si calcola l'energia attraverso una procedura MC su dati simulati per stimare le PDF di segnale e fondo. Il processo non si presenta sempre con valori costanti, ma è definito con una certa variazione; se si riuscisse a catturare esattamente questa variazione e a replicarla nelle simulazioni MC si risolverebbe il problema perfettamente. La procedura MC porta però con sé un'imprecisione intrinseca. La “jet energy scale” è il rapporto tra la risposta della ricostruzione dell'energia dei jets nei dati e la risposta della ricostruzione dell'energia dei jets ottenuti via Monte Carlo con i dati simulati. Quindi la misura dell'energia dei getti definisce un'incertezza nei modelli, che si riflette nell'output della rete neurale in quanto utilizza anche l'energia dei getti come variabile cinematica per discriminare segnale e fondo. Per la definizione delle altre sistematiche si rimanda direttamente all'analisi TOP-11-004 (CMS Collaboration 2013).

4.4 Analisi sui dati di Asimov

In questa sezione si vuole applicare la procedura INFERNO in configurazioni semplificate con i dati di CMS riprodotti da Layer, riferiti al processo $t\bar{t} \rightarrow \tau_h + \text{jets}$. In particolare, in un setup controllato, si vuole studiare l'effetto della porzione di dati tra quelli disponibili sulle performance e l'impatto dei parametri di tuning della rete neurale sull'efficienza delle stime.

Generalmente in HEP i dati veri si dispongono solo in una fase finale dell'analisi, dunque, per ottimizzare gli algoritmi si usa, piuttosto che il dataset vero, il dataset definito di Asimov che è un sostituto adatto per i dati osservati. Esso corrisponde alla somma tra il contributo previsto di segnale e quello previsto di background in cui i parametri di disturbo e il parametro di interesse sono settati ai loro valori nominali. Il dataset di Asimov si può intendere come il modello saturato, ossia quando i dati veri sono esattamente uguali ai dati simulati, attesi. In HEP i dati simulati del processo, quali quelli di Asimov,

Source	Rel. uncert. [%]
W/Z/tt backgr. cross-section uncert.	±3
Top-quark mass	±3
Renormalization/factorization scale	±2
Parton matching	±3
PDF	±5
τ_h trigger efficiency	±7
Pileup	+5 – 1
τ_h energy scale	±7
τ_h identification	±9
Jet energy scale	±11
Jet energy resolution	±2
Unclustered E_T^{miss}	±7
b-tagging	±3
Multijet background reweighting	±5
Syst. uncert.	±21
Stat. uncert. from fit and MC samples	±8
Stat. uncert. from trigger	±0.4
Total stat. uncert.	±8

Tabella 4.1: Incertezze sistematiche relative nella misura della sezione d’urto del processo $t\bar{t}$ dell’analisi TOP-11-004 (CMS Collaboration 2013).

sono talmente tanto simili ai dati veri che è giustificato il loro utilizzo nella fase di ottimizzazione dei modelli e delle procedure.

Le simulazioni degli esperimenti sono un esempio particolare di problema che può essere affrontato mediante il metodo Monte Carlo. La procedura consta nella simulazione del comportamento delle unità elementari, così da predire il comportamento globale del sistema. Ripetendo molte volte l'intera procedura si può determinare la risposta media del sistema. Poiché il metodo Monte Carlo è fondato su un approccio statistico, è necessario che la procedura venga ripetuta molte volte, la precisione della simulazione è tanto maggiore quanto è alto il numero di replicazioni.

Uno dei presupposti del metodo Monte Carlo è quello di saper descrivere il comportamento di ciascuna delle unità elementari che compongono il sistema che si vuole simulare. Regole di comportamento leggermente diverse di queste unità possono portare a differenze più o meno visibili nei risultati per l'intero sistema. Talvolta, le regole di comportamento delle particelle elementari non sono note con precisione e si è costretti a ricorrere all'uso di modelli. In alcuni casi, i modelli non sono descritti mediante formulazioni analitiche, ma si basano su dati sperimentali, che vengono riportati in forma tabulata. Dopo aver simulato, grazie ad informazioni note, un gran numero di dati del sistema in analisi, si applica su di essi il training dei modelli, si definisce un andamento medio del sistema grazie alle proprietà del metodo Monte Carlo e infine si genera il dataset di Asimov che è caratterizzato da valori esattamente pari ai valori medi ottenuti dal MC. E' prassi in HEP, quando si ottimizza una procedura con dati simulati MC, verificare la compatibilità di questi con i dati veri in termini del parametro di interesse. Nel nostro caso specifico, se il rapporto tra il numero di eventi in ciascun bin dei dati veri e dei dati MC è significativamente diverso da 1 allora si deve dubitare che le simulazioni e il processo medio MC siano un buon approssimatore dei veri dati (vedere la fascia bassa nei plot ad istogramma delle Figure 4.9 e 4.10).

Il codice originale dell'algoritmo INFERNO descritto in De Castro e Dorigo 2019 non risulta essere opportuno per includere le tipologie di sistematiche che si incontrano tipicamente in HEP, pertanto si utilizza la versione di INFERNO implementata in PYTORCH (Paszke et al. 2019) da Strong 2021, utilizzata anche nell'esempio sintetico presentato nel Capitolo 2.

Per tutti gli studi di performance che si presentano si caratterizzano da set di training di 5000 eventi di fondo QCD e 20000 eventi di segnale $t\bar{t}$, mentre un set di validazioni di 5600 eventi di background e 23000 di segnale. Il dataset di training è usato per addestrare la rete e stimarne i suoi parametri, mentre quello di validazione è usato per adattare i modelli. L'ampiezza dei batch è di 1000.

Per il training di INFERO viene implementata in PYTORCH una rete neurale a due strati con 10 nodi di output. L'input della rete consta delle stesse otto variabili di alto livello che sono utilizzate per il training del modello che minimizza la BCE citate nella sezione 4.2. Il modello INFERO è confrontato con un modello BCE ottimizzato anch'esso con 10 nodi finali. L'addestramento di entrambe le reti viene eseguito per 100 epoche con l'optimizzatore ADAM. Si noti che il classificatore di INFERO, a differenza di quello BCE, non definisce dei bin ordinati per default in quanto l'ordine può cambiare casualmente con l'inizializzazione della rete; è tuttavia possibile ordinarli, ad esempio in base al rapporto segnale-rumore per emulare un comportamento simile a quello di BCE.

Le statistiche sommario che derivano dalle previsioni del modello di INFERO e dalle previsioni, trasformate in istogrammi, del modello BCE sono poi utilizzate come input per l'adattamento della verosimiglianza profilo per tutti i processi rilevanti di segnale e background.

L'adattamento della verosimiglianza profilo viene eseguito con CABINETRY basato su PYHF, che ha l'obiettivo di stimare il parametro di interesse s che è il numero di eventi di segnale atteso.

Le incertezze dei parametri e i coefficienti di correlazione sono stimati con l'algoritmo HESSE basato sul pacchetto MINUIT; mentre per calcolare gli errori corretti della frazione di segnale μ viene usato l'algoritmo MINOS.

4.4.1 Analisi sulla porzione di dati

Si vuole analizzare l'effetto di diverse porzioni dei dati disponibili in termini di alcune quantità rilevanti. Il parametro di interesse è il numero di eventi di segnale stimato nel campione in esame. Solitamente, e dunque anche qui, il numero di eventi di segnale viene misurato in rapporto al suo valore previsto

dalla teoria, tale rapporto è definito μ . Un valore di μ stimato a 1 significa che si misura un numero di eventi di segnale pari al numero di eventi attesi dalla previsione della teoria. Dunque quando si parla di valore nominale per μ si intende esattamente 1, cioè il caso in cui si osservi esattamente quanto atteso.

Ci si aspetta che riducendo il numero di dati utilizzati per fare il training dei due modelli, INFERNO e BCE, si verifichi un aumento del parametro di *pseudo-significance*, del *minos error* per μ e dell'incertezza sui parametri di disturbo.

Il valore di *significance* o significanza, in HEP, è un numero che è in relazione biunivoca e monotona con il p-value: se p è il p-value e z la significanza, allora l'integrale da z a infinito di una Gaussiana standard è uguale a p . Per convenzione viene convertito nel corrispondente “numero di sigma” cioè il numero di deviazioni standard dalla media di una Gaussiana unitaria; questa conversione da p alle unità di sigma, "N-sigma", è indipendente dai dettagli dell'esperimento. Per esempio, se si ha un p-value pari a 0.159, esso corrisponde ad una standard deviation in quanto la probabilità che una Normale standard sia maggiore di 1 è 0.159. Mentre 0.00135 corrisponde a 3-sigma e 0.000000285 corrisponde a 5-sigma. In realtà ci si riferisce ad una pseudo-significanza in quanto è un'approssimazione della significanza dato che non si sta considerando la distribuzione dei dati ma una di riferimento. In questo particolare contesto il valore della *significance* è tanto più basso quanto i dati sono poco compatibili con l'ipotesi di essere una realizzazione di un modello che prevede solo il background.

Il *minos error* è una misura di errore sulla stima di μ , l'unico parametro di interesse. È calcolato tramite l'algoritmo MINOS che permette di considerare forme di verosimiglianza profilo non simmetriche attorno al massimo. Pertanto si ottengono degli intervalli di confidenza al 68% asimmetrici per la stima di μ , che riflettono la larghezza della verosimiglianza profilo. Per convenienza si valuta sempre la log-verosimiglianza relativa moltiplicata per due e cambiata di segno, in modo che i valori di μ che hanno valore 1 della curva corrispondono ad una deviazione standard. L'incertezza sui parametri di disturbo e i coefficienti di correlazione sono calcolati tramite l'algoritmo HESSE, del pacchetto MINUIT.

Il modello prevede quattro variabili di ingresso della rete neurale che so-

no: *aplanarity*, che descrive la topologia sferica dei prodotti del decadimento del quark top; *chargeEta*, che è il prodotto tra la carica del τ adronico e la pseudorapidità del candidato τ usata per spiegare la natura simmetrica della carica degli eventi $t\bar{t}$ in contrasto con gli eventi $W + \text{jets}$ prodotti nelle collisioni; *MET_met*, che corrisponde all'energia trasversa mancante; *deltaPhi-TauMet* che è l'angolo azimutale tra il candidato tau adronico e la direzione dell'energia trasversa mancante. Sono previsti inoltre solo due parametri di disturbo: *QCD_norm* disturbo sulla normalizzazione del background QCD e *TTjes_signal_jes* disturbo inerente alla jet energy scale dell'evento di segnale.

Per il parametro di normalizzazione del background di tipo QCD non si è imposto alcun vincolo e dunque il fit può regolarlo a qualsiasi valore. La forma della sistematica *TTjes_signal_jes* invece viene stimata dal modello.

Per entrambi i modelli, le reti neurali sono implementate con 100 epoch, un learning rate pari a 0.001 e 10 nodi di output.

Nelle Tabelle riportate in Figura 4.3 si definiscono i confronti tra le performance di INFERN0 e BCE in termini di significance per 4 diversi valori di sottocampionamento: 0.3, 0.5, 0.7 e 0.9. Questo significa che le procedure sono state implementate con il 30%, il 50%, il 70% e il 90% dei dati disponibili dal dataset di Asimov. Nel campione di Asimov si sono misurati in totale 3323 eventi; perciò i sottocampionamenti avranno un numero di eventi rispettivamente pari a 997, 1662, 2326, 2991. Si vede che, come ci si attendeva, all'aumentare del numero di dati i p-value si riducono rendendo l'ipotesi di solo background sempre meno compatibile con i dati.

BCE	p-value osservato	Significanza osservata
downsampling_factor = 0.3	4.5e-11	4.48
downsampling_factor = 0.5	7.33e-17	8.26
downsampling_factor = 0.7	4.57e-24	10.05
downsampling_factor = 0.9	6.01e-31	11.52

INFERNO	p-value osservato	Significanza osservata
downsampling_factor = 0.3	2.97e-13	7.20
downsampling_factor = 0.5	6.45e-21	9.31
downsampling_factor = 0.7	1.52e-27	10.81
downsampling_factor = 0.9	7.95e-35	12.25

Figura 4.3: Valori del p-value osservato e significance osservata, nel fit di BCE sopra e INFERNO sotto, per diversi fattori di sottocampionamento dei dati di Asimov.

In modo analogo si riportano nelle Tabelle di Figura 4.6 le incertezze sulle stime dei parametri del modello: il parametro di interesse μ e i due parametri di disturbo QDC_norm e $TTJets_signal_jes$. Si ricorda che per il parametro μ si stimano delle incertezze asimmetriche. Le quantità riportate indicano quanto l'adattamento dei modelli considera possibile che queste quantità varino nel campione. Le stime che si ottengono dei parametri del modello sono per tutti i casi circa pari a 1 per μ , 1 per QDC_norm e 0 per $TTJets_signal_jes$. All'aumentare della frazione di sottocampionamento, la precisione di stima sui parametri aumenta, questo perché un maggior numero di dati porta con sé un maggior quantitativo di informazione che permette una stima più precisa dei parametri del modello.

Nel confronto tra le due procedure, quella che si basa su INFERNO è migliore in termini di incertezza sul parametro di interesse rispetto a BCE, per tutte le ampiezze campionarie considerate.

BCE	QCD_norm +-	mu +-	TTJets_signal_jes +-
downsample_rate = 0.3	+ - 0.04	-0.19 + 0.22	+ - 0.99
downsample_rate = 0.5	+ - 0.03	-0.15 + 0.18	+ - 0.98
downsample_rate = 0.7	+ - 0.03	-0.13 + 0.16	+ - 0.97
downsample_rate = 0.9	+ - 0.02	-0.12 + 0.15	+ - 0.97

INFERNO	QCD_norm +-	mu +-	TTJets_signal_jes +-
downsample_rate = 0.3	+ - 0.04	-0.17 + 0.19	+ - 0.89
downsample_rate = 0.5	+ - 0.03	-0.14 + 0.15	+ - 0.84
downsample_rate = 0.7	+ - 0.03	-0.12 + 0.13	+ - 0.73
downsample_rate = 0.9	+ - 0.02	-0.11 + 0.12	+ - 0.72

Figura 4.4: Incertezza dei parametri del modello stimato da BCE sopra e INFERNO sotto, usando i dati di Asimov.

Un’immagine chiara del vantaggio dell’utilizzo di INFERNO nel caso in cui i dati siano influenzati dalle sistematiche considerate è rappresentata in Figura 4.5. Essa rappresenta la log-verosimiglianza profilo relativa moltiplicata per due e cambiata di segno, ottenuta dal fit delle statistiche sommario via BCE e INFERNO nel caso in cui si consideri un fattore di sottocampionamento pari a 0.9. Si vede che per INFERNO la curva è più stretta attorno al minimo, che si riflette in intervalli di confidenza meno ampi, ossia misure di μ più precise.

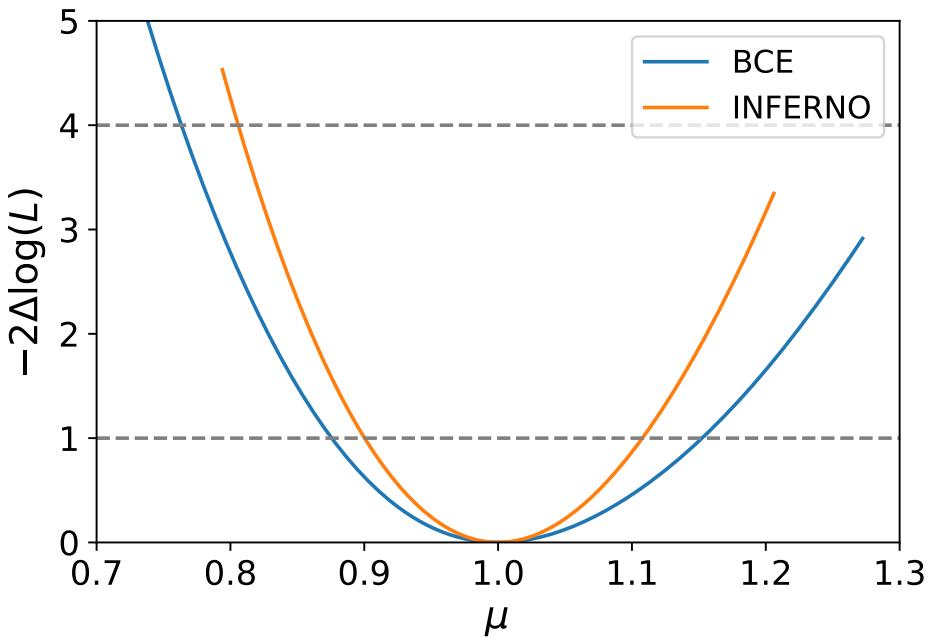


Figura 4.5: Log-verosimiglianza profilo relativa di μ moltiplicata per -2 usando il 90% dei dati disponibili nel caso di presenza di parametri di disturbo, calcolate tramite le procedure INFERNO e BCE.

4.4.2 Analisi sui parametri della rete neurale

Un aspetto non considerato nell'analisi precedente riguarda le scelte fatte sull'impostazione di coefficienti della rete necessari per gli algoritmi. I coefficienti qui considerati sono il numero di nodi di output della rete, ossia il numero di bin nella rappresentazione ad istogramma per entrambe le procedure e il valore di *temperature* per INFERNO. Ci si concentra dunque sull'impatto di questi iperparametri sui risultati delle procedure nel caso in cui non ci siano parametri di disturbo. Le variabili cinematiche inserite sono le medesime dell'analisi precedente.

Per quanto riguarda il numero di bin ci si aspetta che all'aumentare di questo, sarà meno verosimile che i dati provengano dal modello che comprende solo il background, perché aumenta la potenza statistica potendo suddividere gli eventi in più bin. Ci si attende dunque un'incertezza sul parametro di interesse che decresce all'aumentare del numero di bin.

La temperatura invece è un parametro tipico della funzione softmax delle reti neurali, utilizzata nell'approccio INFERNO. La funzione softmax, $\mathbf{R} \rightarrow [0, 1]$, permette, se si danno in pasto valori minori o uguali a 0, di ottenere delle probabilità, ossia valori in $[0, 1]$ normalizzate affiché sommino a 1. La temperatura è un iperparametro che viene applicato ai logaritmi, il logaritmo di una probabilità sta infatti nell'intervallo $[-\infty, 0]$, per influenzare le probabilità finali restituite dalla funzione softmax: una temperatura bassa, inferiore a 1, rende il modello più sicuro, una temperatura elevata, sopra a 1, rende il modello meno sicuro. Ossia con una temperatura alta si mettono maggiormente in evidenza i valori grandi e si nascondono, abbassandoli, quelli che sono lontani dal valore massimo. Nel caso della rete neurale la funzione softmax trasforma gli output numerici dell'ultimo strato della rete di classificazione in probabilità prendendo gli esponenti di ogni output e normalizzando con la somma degli esponenti di tutti gli output, in modo che l'intero vettore sommi a 1.

Per condurre l'analisi si procede in due step:

1. Si fissa la temperatura ad un valore pari a 0.1 e si implementano le due procedure 5 volte con numero di bin variabile appartenente al seguente vettore: (5, 8, 11, 14, 17, 20).
2. Si fissa il numero di bin a 10 e si implementa INFERNO 5 volte con temperatura variabile appartenente al seguente vettore: (0.001, 0.05, 0.1, 0.5, 0.9, 0.99).

All'esito di ogni fit dei modelli si estrae il valore stimato dell'incertezza sul parametro di interesse μ . Come per l'analisi precedente le incertezze sono calcolate con l'algoritmo MINOS per tenere conto della possibile asimmetria della verosimiglianza profilo.

La procedura è interamente implementata sul 90% dei dati di Asimov disponibili e si riporta l'esito di un unico training per ciascun caso in esame in quanto si è verificato che le fluttuazioni sono molto piccole e dunque si riescono ad interpretare con sufficiente precisione i risultati anche riportando gli esiti di un unico addestramento dei modelli. La funzione di perdita di INFERNO corrisponde alla varianza approssimata $\sigma^2(s)$ del numero atteso di eventi di segnale s . Il corrispondente valore per BCE è ottenuto trasformando in istogramma le previsione di BCE dopo ogni epoca e utilizzando questi istogrammi

come input per l'algoritmo INFERNO che ne calcola la matrice di varianza e covarianza approssimata. Entrambi i modelli, per tutti i casi considerati nell'addestramento convergono ad un valore simile di $\sigma^2(s)$.

Le Tabelle di Figura 4.6 e 4.7 riportano gli esiti dei due step dell'analisi definiti sopra.

INFERNO

Numero di bin	$\mu_{\text{hat}} -$	$\mu_{\text{hat}} +$	ampiezza
5	-0.096	+0.099	0.195
8	-0.096	+0.099	0.195
11	-0.097	+0.100	0.197
14	-0.097	+0.100	0.197
17	-0.096	+0.100	0.196
20	-0.096	+0.099	0.195

BCE

Numero di bin	$\mu_{\text{hat}} -$	$\mu_{\text{hat}} +$	ampiezza
5	-0.113	+0.116	0.229
8	-0.101	+0.107	0.208
11	-0.101	+0.104	0.205
14	-0.097	+0.100	0.197
17	-0.098	+0.100	0.198
20	-0.096	+0.099	0.195

Figura 4.6: Incertezza del parametro di interesse μ e ampiezza dell'intervallo di confidenza al 68%, ottenute con INFERNO e BCE con diversi valori di numero di bin e temperatura fissata a 0.1.

INFERNO

Temperatura	mu_hat -	mu_hat +	ampiezza
0.001	-0.096	+0.099	0.195
0.05	-0.095	+0.098	0.193
0.1	-0.098	+0.101	0.199
0.5	-0.097	+0.100	0.197
0.9	-0.096	+0.098	0.194
0.99	-0.095	+0.098	0.193

Figura 4.7: Incertezza del parametro di interesse μ e ampiezza dell’intervallo di confidenza al 68% ottenute con INFERNO a diversi valori di temperatura e numero di bin impostato a 10.

Per determinare i valori del numero di bin e di temperatura migliori, tra quelli considerati, si utilizza l’ampiezza della log-verosimiglianza profilo relativa moltiplicata per due e cambiata di segno ad altezza 1, calcolata nelle tabelle come la somma dei valori assoluti degli scostamenti verso l’alto e verso il basso stimati. Esso corrisponde ad un intervallo di confidenza di livello 0.68.

Si vede che il valore migliore di numero di bin tra quelli considerati per INFERNO è 20, 5, oppure 8 con un’ampiezza dell’intervallo di confidenza pari a 0.195; mentre per BCE il numero di bin migliori è 20 con un’ampiezza di intervallo di 0.195. Si nota anche che, mentre per BCE all’aumentare del numero di bin l’ampiezza decresce, per INFERNO l’andamento non è decrescente ed è molto più stabile, come si vede più chiaramente in Figura 4.8.

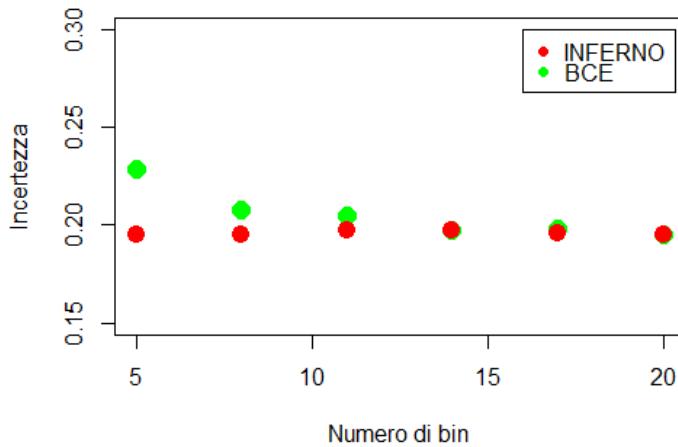


Figura 4.8: Ampiezza dell’intervallo di confidenza di livello 68% per μ calcolata con INFERNO e BCE al variare del numero di bin dell’istogramma di output con temperatura fissata a 0.1.

Per quanto riguarda la temperatura si osserva che per INFERNO si ottiene un’incertezza minore per valori di temperatura pari a 0.05 e 0.99.

Si riportano, in Figura 4.9 e 4.10 gli esiti dell’adattamento dei modelli ai dati di Asimov e i profili delle log-verosimiglianze nei casi di numero di bin preferibile con i vincoli su temperature e numero di bin utilizzati precedentemente nei due step dell’analisi.

La log-verosimiglianza profilo relativa moltiplicata per due e cambiata di segno è ottenuta dal fit delle statistiche sommario via BCE e INFERNO. I grafici che riportano gli histogrammi invece contengono i valori "osservati", dal dataset di Asimov, raffigurati come dei puntini neri, e la quantità di eventi classificati come segnale e fondo per ogni bin dalla rete neurale rappresentate con le barre colorate. Quello che si riporta nella fascia in basso dei grafici con gli histogrammi è una misura di quanto, in ogni bin, il rapporto tra il numero di eventi di segnale "osservati" nel dataset di Asimov e quelli simulati sia pari a 1, con una certa incertezza (barre verticali) dovute all’incertezza dei dati e del modello.

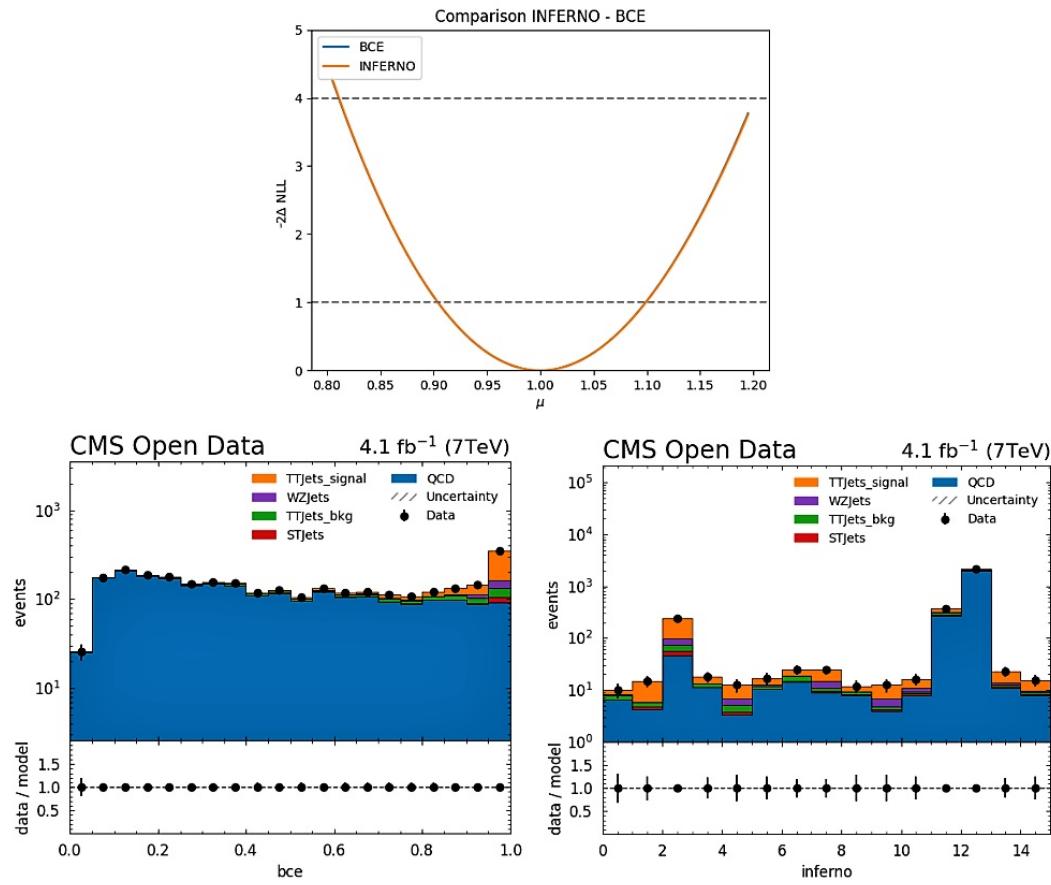


Figura 4.9: Caso con 20 bin e temperatura 0.1. Sopra: confronto tra le trasformazioni delle log-verosimiglianze profilo per μ ottenuta tramite l'adattamento con BCE e INFERNO con i dati di Asimov. Sotto: concordanza tra i dati e le simulazioni per BCE a sinistra e INFERNO a destra nel caso di assenza di parametri di disturbo.

Si noti che per INFERNO in Figura 4.9 non ci sono esattamente 20 bin, questo perchè la procedura ha stimato un numero di eventi pari a 0 su 5 bin che dunque non vengono riportati.

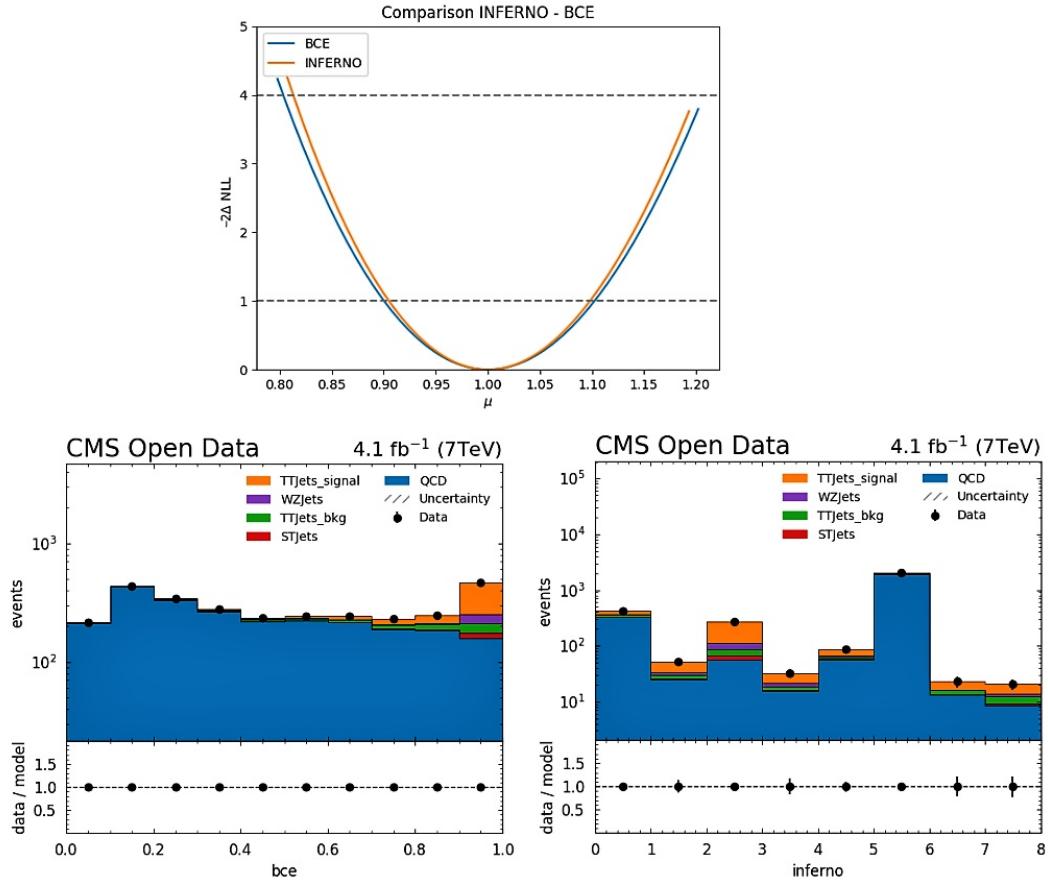


Figura 4.10: Caso con 10 bin e temperatura per INFERNO 0.9. Sopra: confronto tra le trasformazioni delle log-verosimiglianze profilo per μ ottenuta tramite l'adattamento con BCE e INFERNO con i dati di Asimov. Sotto: concordanza tra i dati e le simulazioni per BCE a sinistra e INFERNO a destra nel caso di assenza di parametri di disturbo.

La misura della potenza di segnale senza sistematiche nel dataset di Asimov basata sulle statistiche sommario stimate con INFERNO e BCE sono molto simili tra loro, con un leggero vantaggio per la procedura INFERNO nel secondo caso analizzato. Questo è un comportamento atteso, poiché un classificatore addestrato con BCE dovrebbe essere ottimale per l'inferenza se non sono presenti parametri di disturbo e INFERNO dovrebbe essere ottimale per l'inferenza per costruzione. Entrambe le procedure stimano correttamente il vero valore del parametro μ anche se è evidente come suddividano in modo

molto differente gli eventi di segnale nei vari bin.

Gli studi condotti da Layer (Layer 2022) mettono in luce il fatto che più parametri di disturbo di tipo "ShapeNorm" influenti vengono introdotti nei dati, più INFERNO stima il parametro di interesse μ con incertezze minori rispetto a quelle di BCE. Il motivo è che INFERNO è costruito per tenere conto, già nella fase di training della rete neurale del successivo passo di inferenza statistica, ottimizzando i coefficienti della rete in modo da minimizzare la varianza stimata del parametro di interesse. Inoltre la sua implementazione è tale da decorrelare il più possibile i parametri di disturbo con quello di interesse, consentendo una buona riduzione dell'incertezza sulla misura di μ .

Conclusioni

Il Large Hadron Collider (LHC) è il più grande e potente acceleratore di particelle al mondo situato presso il CERN di Ginevra, il più grande laboratorio di fisica delle particelle. Gli strumenti che si utilizzano al fine di cercare nuova fisica sono gli acceleratori di particelle, che accelerano le particelle fino ad energie molto elevate e i rivelatori, che permettono di vedere e scoprire nuove particelle che si creano durante le collisioni. Un aspetto cruciale per misurare nuova fisica è l'utilizzo di classificatori del tipo reti neurali per distinguere gli eventi di segnale da quelli di fondo. Gli eventi osservati infatti sono caratterizzati da una molteplicità di caratteristiche che rende necessaria la riduzione dimensionale e dunque la costruzione di potenti statistiche riassuntive bidimensionale tramite i classificatori. Le statistiche riassuntive sono poi utilizzate come input per un'analisi statistica inferenziale per il calcolo di quantità di interesse come le sezioni d'urto.

La potenza della classificazione è degradata dall'incompleta conoscenza delle proprietà del segnale e del fondo; questo si riflette in modelli imprecisi caratterizzati da incertezze sistematiche. Generalmente questo tipo di errore viene preso in considerazione solo nella fase inferenziale, producendo delle misure con alte incertezze sistematiche.

La nuova tecnica INFERNO proposta da De Castro e Dorigo 2019, risolve il problema degli alti errori sistematici sulle misure di interesse tenendo conto già nella fase di costruzione delle statistiche sommario dell'obiettivo finale di inferenza. INFERNO costruisce una statistica riassuntiva ottimizzando una rete neurale che minimizza l'incertezza attesa del parametro di interesse. Per tenere conto in questa fase dei parametri di disturbo si utilizza un algoritmo di interpolazione tra le statistiche sommario nominali e variazioni verso l'alto

e verso il basso delle sistematiche.

Al fine di testare l'algoritmo INFERNO e mettere in luce le sue proprietà si è condotta un'estesa analisi su un campione di dati sintetici. Questo ha permesso di mostrare che la procedura ha delle performance migliori in termini di ampiezza degli intervalli di confidenza per il parametro di interesse rispetto ad un approccio basato sulla *binary cross entropy* (BCE). In aggiunta vengono condotti degli studi su un set di dati veri, pubblici, estratti da CMS Open Data e riprodotti da Layer 2022. I dati si riferiscono ad un'analisi sistematica del 2013 dell'esperimento CMS (CMS Collaboration 2013) per misurare la sezione d'urto del processo $t\bar{t} \rightarrow \tau_h + \text{jets}$. L'applicazione di INFERNO a tali dati ha permesso di valutarne attraverso confronti alcuni suoi aspetti. Le analisi qui proposte, insieme a quelle condotte da Layer 2022 dimostrano che INFERNO ha il potenziale per mitigare l'effetto dei parametri di disturbo riducendo l'ampiezza degli intervalli di confidenza calcolati nel processo inferenziale in analisi vere di LHC.

In conclusione, l'algoritmo INFERNO per la costruzione di statistiche sommario può apportare dei grandi vantaggi alle analisi di fisica delle particelle che sono affette da incertezze sistematiche che influenzano le forme delle distribuzioni degli eventi osservati. INFERNO permette dunque di utilizzare tecniche di apprendimento automatiche, come le reti neurali, le cui ottimizzazioni, attraverso apposite funzioni di perdita, sono allineate con l'obiettivo finale del problema, che è una stima inferenziale di un certo parametro di interesse.

Riferimenti

- ATLAS Collaboration, CMS Collaboration e LHC Higgs Combination Group (2011). *Procedure for the LHC Higgs boson search combination in Summer 2011*. Rapp. tecn. Geneva: CERN. URL: <https://cds.cern.ch/record/1379837>.
- Cabibbo, Nicola (1963). «Unitary symmetry and leptonic decays». In: *Physical Review Letters* 10.12, p. 531.
- Charnock, Tom, Guilhem Lavaux e Benjamin D. Wandelt (2018). «Automatic physical inference with information maximizing neural networks». In: *Physical Review D* 97.8, p. 083004.
- Chau, Ling-Lie e Wai-Yee Keung (1984). «Comments on the parametrization of the Kobayashi-Maskawa matrix». In: *Physical Review Letters* 53.19, p. 1802.
- CMS Collaboration (2013). «Measurement of the $t\bar{t}$ production cross section in the $\tau+jets$ channel in pp collisions at $\sqrt{s} = 7$ TeV». In: *The European Physical Journal C* 73.4, p. 2386. ISSN: 1434-6052. DOI: [10.1140/epjc/s10052-013-2386-x](https://doi.org/10.1140/epjc/s10052-013-2386-x).
- (2016). *Simulated dataset TTJets_TuneZ2_7TeV-madgraph-tauola in AOD-SIM format for 2011 collision data (SM Inclusive)*. CERN Open Data Portal. DOI: [10.7483/OPENDATA.CMS.ZBGF.H543](https://doi.org/10.7483/OPENDATA.CMS.ZBGF.H543).
- CMS collaboration et al. (2016). «Reconstruction and identification of τ lepton decays to hadrons and ν_τ at CMS». In: *Journal of Instrumentation* 11.01, P01019. DOI: [10.1088/1748-0221/11/01/p01019](https://doi.org/10.1088/1748-0221/11/01/p01019).
- Conway, John S. (2011). *Incorporating Nuisance Parameters in Likelihoods for Multisource Spectra*. arXiv: [1103.0354 \[physics.data-an\]](https://arxiv.org/abs/1103.0354).

- Cristinziani, Markus e Martijn Mulders (2017). «Top-quark physics at the Large Hadron Collider». In: *Journal of Physics G: Nuclear and Particle Physics* 44.6, p. 063001. ISSN: 1361-6471. DOI: [10.1088/1361-6471/44/6/063001](https://doi.org/10.1088/1361-6471/44/6/063001).
- De Castro, Pablo e Tommaso Dorigo (2019). «INFERNO: Inference-aware neural optimisation». In: *Computer Physics Communications* 244, pp. 170–179.
- Elvira, V. Daniel (2017). «Impact of detector simulation in particle physics collider experiments». In: *Physics Reports* 695, 1–54. ISSN: 0370-1573. DOI: [10.1016/j.physrep.2017.06.002](https://doi.org/10.1016/j.physrep.2017.06.002).
- Elwood, Adam e Dirk Krücker (2018). *Direct optimisation of the discovery significance when training neural networks to search for new physics in particle colliders*. arXiv: [1806.00322 \[physics.data-an\]](https://arxiv.org/abs/1806.00322).
- Heinrich, Lukas e Nathan Simpson (2020). *pyhf/neos: initial zenodo release*. Ver. 0.0.2. DOI: [10.5281/zenodo.3697981](https://doi.org/10.5281/zenodo.3697981).
- Kobayashi, Makoto e Toshihide Maskawa (1973). «CP-violation in the renormalizable theory of weak interaction». In: *Progress of theoretical physics* 49.2, pp. 652–657.
- Lassila-Perini, Kati et al. (2021). «Using CMS Open Data in research – challenges and directions». In: *EPJ Web of Conferences* 251, p. 01004. ISSN: 2100-014X. DOI: [10.1051/epjconf/202125101004](https://doi.org/10.1051/epjconf/202125101004).
- Layer, Lukas (2022). *llayer/cmsopen: INFERNO for CMS Open Data*. Ver. 0.0.1. DOI: [10.5281/zenodo.6080791](https://doi.org/10.5281/zenodo.6080791).
- Lundberg, Scott M. e Su-In Lee (2017). «A Unified Approach to Interpreting Model Predictions». In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 4765–4774. URL: <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>.
- Maki, Ziro, Masami Nakagawa e Shoichi Sakata (1962). «Remarks on the unified model of elementary particles». In: *Progress of Theoretical Physics* 28.5, pp. 870–880.
- Mobs, Esma (2019). «The CERN accelerator complex-2019. Complexe des accélérateurs du CERN-2019». In: *CERN-GRAFICS-2019-002*. General Photo. URL: <https://cds.cern.ch/record/2684277>.

- Murtagh, Bruce A. e Michael A. Saunders (1978). «Large-scale linearly constrained optimization». In: *Mathematical Programming* 14.1, pp. 41–72. ISSN: 1436-4646. DOI: [10.1007/BF01588950](https://doi.org/10.1007/BF01588950).
- Neal, Radford et al. (2008). «Computing likelihood functions for high-energy physics experiments when distributions are defined by simulators with nuisance parameters». In: *PHYSTAT-LHC Workshop on Statistical Issues for LHC Physics*, pp. 111–118. DOI: [10.5170/CERN-2008-001.111](https://doi.org/10.5170/CERN-2008-001.111).
- Paszke, Adam et al. (2019). «PyTorch: An Imperative Style, High-Performance Deep Learning Library». In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 8024–8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- Pontecorvo, Bruno (1968). «Neutrino experiments and the problem of conservation of leptonic charge». In: *Sov. Phys. JETP* 26.984-988, p. 165.
- Sakuma, Tai e Thomas McCauley (2014). «Detector and event visualization with SketchUp at the CMS experiment». In: *Journal of Physics: Conference Series*. Vol. 513. 2. IOP Publishing, p. 022032.
- Sirunyan, Abel M. et al. (2017). «Particle-flow reconstruction and global event description with the CMS detector». In: *Journal of Instrumentation* 12.10, P10003–P10003. ISSN: 1748-0221. DOI: [10.1088/1748-0221/12/10/p10003](https://doi.org/10.1088/1748-0221/12/10/p10003).
- Strong, Giles (2021). *GilesStrong/pytorch_inferno: v0.2.2*. Ver. v0.2.2. DOI: [10.5281/zenodo.5040810](https://doi.org/10.5281/zenodo.5040810).
- Therhaag, Jan (2010). «TMVA Toolkit for multivariate data analysis in ROOT». In: *PoS ICHEP2010*, p. 510. DOI: [10.22323/1.120.0510](https://doi.org/10.22323/1.120.0510).
- Wunsch, Stefan et al. (2021). «Optimal statistical inference in the presence of systematic uncertainties using neural network optimization based on binned Poisson likelihoods with nuisance parameters». In: *Computing and Software for Big Science* 5.1, pp. 1–11.
- Xia, Li-Gang (2019). «QBDT, a new boosting decision tree method with systematical uncertainties into training for High Energy Physics». In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 930, pp. 15–26.

Appendice A

Codice dati sintetici

L'algoritmo originale e' stato implementato in TENSORFLOW 1. Qui viene utilizzata la versione in PYTORCH implementata da Strong 2021.

```
#####
#IMPORT OF NECESSARY PACKAGES
import os
if 'google.colab' in str(get_ipython()):
    os.system('pip install torch==1.9.0 fastcore==1.3.20 pytorch_inferno
              ==0.2.2')

from pytorch_inferno.data import get_paper_data
from pytorch_inferno.utils import init_net
from pytorch_inferno.model_wrapper import ModelWrapper
from pytorch_inferno.inferno import VariableSoftmax, PaperInferno
from torch import nn, optim
from pytorch_inferno.callback import LossTracker, SaveBest, EarlyStopping
from fastcore.all import partiallier
from torch.distributions import Normal
from pytorch_inferno.inferno import InfernoPred
from pytorch_inferno.inference import bin_preds, get_shape, get_paper_syst_
    shapes, calc_profile
from pytorch_inferno.plotting import plot_likelihood, plot_preds
from pytorch_inferno.callback import PredHandler
import pandas as pd
```

```

import numpy as np
import torch
from torch import Tensor
import matplotlib.pyplot as plt
#####
##### #IMPLEMENTATION OF USEFUL FUNCTIONS
in_edges=bins)def compute_nll(model:ModelWrapper, is_inferno:bool,r_vals
=[-0.2,0,0.2],l_vals=[2.5,3,3.5]) -> Tensor:
    preds = model._predict_dl(test, pred_cb=InfernoPred() if is_inferno
        else PredHandler())
    df = pd.DataFrame({'pred':preds.squeeze()})
    df['gen_target'] = test.dataset.y
    df.head()
    bins = np.linspace(0,10,11) if is_inferno else np.linspace(0,1,11)
    bin_preds(df, bins=bins)
    #plot_preds(df, bin_edges=bins)
    f_s,f_b = get_shape(df,1),get_shape(df,0)
    b_shapes = get_paper_syst_shapes(bkg, df, model=model, r_vals = r_vals,
        l_vals = l_vals, pred_cb=InfernoPred() if is_inferno else
        PredHandler(), bins=bins)
    nll = calc_profile(f_s_nom=f_s, **b_shapes, n_obs=1050, mu_scan=mu_scan
        , mu_true=50, n_steps=100, shape_aux=[Normal(0,1), Normal(0,1)])
    return nll

#Plot of loss
def plot_loss(lt):

```

```

plt.plot(lt.losses["trn"], label="train")
plt.plot(lt.losses["val"], label="val")
plt.ylabel(r"$\sigma^2(\mu)$")
plt.xlabel(r"epoch")
plt.legend(loc="upper right")
plt.show()

#Functions to fit models and calculate nll
def fit_bce( data, std_r, std_l ):

    net2 = nn.Sequential(nn.Linear(3,100), nn.ReLU(),
                         nn.Linear(100,100), nn.ReLU(),
                         nn.Linear(100,1),   nn.Sigmoid())
    init_net(net2)
    model_bce = ModelWrapper(net2)
    lt_bce = LossTracker()
    model_bce.fit(100, data=data, opt=partial(optim.Adam, lr=1e-5), loss=
                  nn.BCELoss(),
                  cbs=[lt_bce, SaveBest('weights/best_ie.h5'), EarlyStopping
                        (100)])
    plot_loss(lt_bce)
    r_vals = [0.-std_r, 0. , 0.+std_r]
    l_vals = [3.-std_l, 3. , 3.+std_l]
    print('r_vals', r_vals)
    print('l_vals', l_vals)
    nll_bce = compute_nll(model_bce, False , r_vals = r_vals, l_vals = l_
                           vals)
    return nll_bce

def fit_inferno( data, std_r, std_l ):

    net = nn.Sequential(nn.Linear(3,100), nn.ReLU(),

```

```

        nn.Linear(100,100),nn.ReLU(),
        nn.Linear(100,10),
        VariableSoftmax(0.1))

init_net(net)
model_inf = ModelWrapper(net)
lt_inf = LossTracker()
model_inf.fit(100, data=data, opt=partial(optim.Adam, lr=1e-5), loss=
    None,
    cbs=[PaperInferno(float_r=True, float_l=True, shape_aux=[

        Normal(0, std_r),
        Normal(0, std_l)], nonaux_b_norm=False), lt_inf, SaveBest('

            weights/best_ie.h5'), EarlyStopping(100)])]

plot_loss(lt_inf)
r_vals = [0.-std_r, 0. , 0.+std_r]
l_vals = [3.-std_l, 3. , 3.+std_l]
print('r_vals', r_vals)
print('l_vals', l_vals)
nll_inf = compute_nll(model_inf, True , r_vals = r_vals, l_vals = l_
    vals)
return nll_inf

#####
#####IMPLEMENTATION OF EACH CASE
#####
results_bce = []
results_inf = []
for i in range(10):
    print('Run number:', i)
    data, test = get_paper_data(200000, bs=2000, n_test=1000000) #generate a
        new sample of data
    mu_scan = torch.linspace(20,80,61)
    bkg = test.dataset.x[test.dataset.y.squeeze() == 0]

```

```
std_r = 0.2 #part to modify
std_l = 0.5 #part to modify

nll_inf = fit_inferno( data, std_r= std_r, std_l= std_l )
nll_bce = fit_bce( data, std_r= std_r, std_l= std_l)

std_bce, std_inf = plot_likelihood({'BCE':nll_bce, 'INFERNO':nll_inf}, mu
_scan)

results_bce.append(std_bce)
results_inf.append(std_inf)

results_bce
results_inf
#####
#####
##### #First part : definition of useful functions
import os
if 'google.colab' in str(get_ipython()):
    os.system('pip install torch==1.9.0 fastcore==1.3.20 pytorch_inferno
==0.2.2')

from pytorch_inferno.data import get_paper_data
from pytorch_inferno.utils import init_net
from pytorch_inferno.model_wrapper import ModelWrapper
from pytorch_inferno.inferno import VariableSoftmax, PaperInferno
from torch import nn, optim
from pytorch_inferno.callback import LossTracker, SaveBest, EarlyStopping
from fastcore.all import partialler
from torch.distributions import Normal
from pytorch_inferno.inferno import InfernoPred
from pytorch_inferno.inference import bin_preds, get_shape, get_paper_syst_
shapes, calc_profile
from pytorch_inferno.plotting import plot_likelihood, plot_preds
```

```
from pytorch_inferno.callback import PredHandler
import pandas as pd
import numpy as np
import torch
from torch import Tensor
import matplotlib.pyplot as plt
from pytorch_inferno.inference import calc_nll, calc_grad_hesse
from typing import Optional, Union, List, Dict
from torch.distributions import Distribution

def compute_nll(model:ModelWrapper, is_inferno:bool,r_vals=[-0.2,0,0.2],l_
    vals=[2.5,3,3.5]) -> Tensor:
    preds = model._predict_dl(test, pred_cb=InfernoPred() if is_inferno
        else PredHandler())

    df = pd.DataFrame({'pred':preds.squeeze()})
    df['gen_target'] = test.dataset.y
    df.head()

    bins = np.linspace(0,10,11) if is_inferno else np.linspace(0,1,11)
    bin_preds(df, bins=bins)
    #plot_preds(df, bin_edges=bins)
    f_s,f_b = get_shape(df,1),get_shape(df,0)

    b_shapes = get_paper_syst_shapes(bkg, df, model=model, r_vals = r_vals,
        l_vals = l_vals, pred_cb=InfernoPred() if is_inferno else
        PredHandler(), bins=bins)

    nll = calc_profile(f_s_nom=f_s, **b_shapes, n_obs=1050, mu_scan=mu_scan
        , mu_true=50, n_steps=100, shape_aux=[Normal(0,1), Normal(0,1)])
    return nll

#Plot of loss
def plot_loss(lt):
```

```

plt.plot(lt.losses["trn"], label="train")
plt.plot(lt.losses["val"], label="val")
plt.ylabel(r"$\sigma^2(\mu)$")
plt.xlabel(r"epoch")
plt.legend(loc="upper right")
plt.show()

def newton_optimizer(mu_best:float, f_s_nom:Tensor, f_b_nom:Tensor, n_obs:
    int, mu_true:int,
    f_s_up:Optional[Tensor]=None, f_s_dw:Optional[Tensor]=None
    ,
    f_b_up:Optional[Tensor]=None, f_b_dw:Optional[Tensor]=None
    ,
    shape_aux:Optional[List[Distribution]]=None,
    s_norm_aux:Optional[List[Distribution]]=None, b_norm_aux:
        Optional[List[Distribution]]=None, nonaux_b_norm:bool=
            False,
    n_steps:int=100, lr:float=0.1, verbose:bool=True) ->
    Tensor:

    r'''Compute profile likelihoods for best mu value, optimising on full
    hessian.'''
    for f in [f_s_nom, f_s_up, f_s_dw, f_b_nom, f_b_up, f_b_dw]: # Ensure
        correct dimensions
        if f is not None and len(f.shape) < 2: f.unsqueeze_(0)
    # Cases where nuisance only causes up xor down variation
    if (f_s_up is None and f_s_dw is not None): f_s_up = torch.repeat_
        interleave(f_s_nom, repeats=len(f_s_dw), dim=0)
    if (f_s_dw is None and f_s_up is not None): f_s_dw = torch.repeat_
        interleave(f_s_nom, repeats=len(f_s_up), dim=0)
    if (f_b_up is None and f_b_dw is not None): f_b_up = torch.repeat_
        interleave(f_s_nom, repeats=len(f_b_dw), dim=0)
    if (f_b_dw is None and f_b_up is not None): f_b_dw = torch.repeat_

```

```

    interleave(f_s_nom, repeats=len(f_b_up), dim=0)
if f_s_up is not None and f_b_up is not None and len(f_s_up) != len(f_b
_up):
    raise ValueError("Shape variations for signal & background must
        have the same number of variations. \
            Please enter the nominal templates for nuisances
            that only affect either signal or background.
        ")

# Norm uncertainties
if s_norm_aux is None: s_norm_aux = []
if b_norm_aux is None: b_norm_aux = []
# Compute nuisance indeces
poi_idx = 0
n_alpha = 1 + np.max((len(f_b_up) if f_b_up is not None else 0, len(f_s
_up) if f_s_up is not None else 0))
shape_idxs = list(range(1,n_alpha))
print("shape_idxs", shape_idxs)
s_norm_idxs = list(range(n_alpha, n_alpha+len(s_norm_aux)))
n_alpha += len(s_norm_aux)
b_norm_idxs = list(range(n_alpha, n_alpha+len(b_norm_aux)+nonaux_b_norm
))
n_alpha += len(b_norm_aux)+nonaux_b_norm

b_true = n_obs-mu_true

get_nll = partialler(calc_nll, s_true=mu_true, b_true=b_true,
                     f_s_nom=f_s_nom, f_s_up=f_s_up, f_s_dw=f_s_dw,
                     f_b_nom=f_b_nom, f_b_up=f_b_up, f_b_dw=f_b_dw,
                     s_norm_aux=s_norm_aux, b_norm_aux=b_norm_aux, shape
                     _aux=shape_aux)

# Initialize the parameters
alpha = torch.zeros((n_alpha), requires_grad=True, device=f_b_nom.
device)
#mu = torch.tensor(mu_best).to(f_b_nom.device)
with torch.no_grad(): alpha[poi_idx] = mu_best

```

```

print("alpha", alpha)

# List to store relevant info
param_val = []

# print("Mu", mu)
for i in range(n_steps): # Newton optimise nuisances
    nll = get_nll(shape_alpha=alpha[shape_idxs], mu=alpha[poi_idx], s_
                  norm_alpha=alpha[s_norm_idxs], b_norm_alpha=alpha[b_norm_idxs])
    #print("nll", nll)
    grad, hesse = calc_grad_hesse(nll, alpha, create_graph=False)
    #print("grad", grad)
    #print("hesse", hesse)
    # Step is learning rate * gradient * covariance matrix
    cov = torch.inverse(hesse)
    step = lr*grad.detach()@cov
    #print("step", step)
    # Avoid too large steps
    step = torch.clamp(step, -100, 100)
    # Apply the step to the parameters
    alpha = alpha-step
    print("alpha", alpha)
    # Store the parameter infos
    param_val.append({ 'step':i, 'alpha':alpha.detach().numpy(), 'cov':
                       cov.detach().numpy() })

# Last nll evaluation
nll = get_nll(shape_alpha=alpha[shape_idxs], mu=alpha[poi_idx], s_norm_
               alpha=alpha[s_norm_idxs], b_norm_alpha=alpha[b_norm_idxs]).detach()
return nll, param_val

def best_fit_values(mu_best:float, model:ModelWrapper, is_inferno:bool,r_
                    vals=[-0.2,0,0.2],l_vals=[2.5,3,3.5]) -> Tensor:
    preds = model._predict_dl(test, pred_cb=InfernoPred() if is_inferno
                               else PredHandler())

```

```

df = pd.DataFrame({'pred':preds.squeeze()})
df['gen_target'] = test.dataset.y
df.head()

bins = np.linspace(0,10,11) if is_inferno else np.linspace(0,1,11)
bin_preds(df, bins=bins)
f_s,f_b = get_shape(df,1),get_shape(df,0)
b_shapes = get_paper_syst_shapes(bkg, df, model=model, r_vals = r_vals,
    l_vals = l_vals, pred_cb=InfernoPred() if is_inferno else
    PredHandler(), bins=bins)
nll, param_val = newton_optimizer(mu_best=mu_best, f_s_nom=f_s, **b_
    shapes, n_obs=1050, mu_true=50, n_steps=100, shape_aux=[Normal(0,1)
    , Normal(0,1)])
return nll, param_val

def get_best_fit(nll, mu_scan):
    return mu_scan[np.argmin(nll)]

def correlation_from_covariance(covariance):
    v = np.sqrt(np.diag(covariance))
    outer_v = np.outer(v, v)
    correlation = covariance / outer_v
    correlation[covariance == 0] = 0
    return correlation

#####
#Creazione dataset c_i con cui eseguo l'i-esimo run per ogni caso
data, test = get_paper_data(200000, bs=2000, n_test=1000000)
#####

#####
#Second part : extraction of values from INFERNO and from BCE with data
sample c_i

```

```

#Part to modify
std_r = 0.2
std_l = 1.4

#Inferno model training
net = nn.Sequential(nn.Linear(3,100), nn.ReLU(),
                    nn.Linear(100,100),nn.ReLU(),
                    nn.Linear(100,10), VariableSoftmax(0.1))

init_net(net)
model_inf = ModelWrapper(net)
lt_inf = LossTracker()
model_inf.fit(100, data=data, opt=partialiller(optim.Adam,lr=1e-4), loss=None
              ,
              cbs=[PaperInferno(float_r=True, float_l=True, shape_aux=[Normal(0,std_r), Normal(0,std_l)], nonaux_b_norm=False),
                   lt_inf,SaveBest('weights/best_ie.h5'),EarlyStopping(100)])
              

#BCE model training
net2 = nn.Sequential(nn.Linear(3,100),    nn.ReLU(),
                     nn.Linear(100,100), nn.ReLU(),
                     nn.Linear(100,1),   nn.Sigmoid())

init_net(net2)
lt_bce = LossTracker()
model_bce = ModelWrapper(net2)
model_bce.fit(100, data=data, opt=partialiller(optim.Adam,lr=1e-4), loss=nn.
              BCELoss(),
              cbs=[lt_bce,SaveBest('weights/best_ie.h5'),EarlyStopping(100)
                   ])

mu_scan = torch.linspace(20,80,61)
bkg = test.dataset.x[test.dataset.y.squeeze() == 0]

#The changes in inference

```

```
r_vals = [0. - std_r, 0., 0. + std_r]
l_vals = [3. - std_l, 3., 3. + std_l]
nll_inf = compute_nll(model_inf, True , r_vals = r_vals, l_vals = l_vals)
nll_bce = compute_nll(model_bce, False, r_vals = r_vals, l_vals = l_vals)

# Get best fit value for mu
mu_best_inf = get_best_fit(nll_inf, mu_scan)
mu_best_bce = get_best_fit(nll_bce, mu_scan)

# Minimize NLL at best fit value
nll_i, param_val_i = best_fit_values(mu_best_inf, model_inf, True, r_vals=r
    _vals, l_vals =l_vals)
nll_b, param_val_b = best_fit_values(mu_best_bce, model_bce, False, r_vals=
    r_vals, l_vals =l_vals)
param_df_i = pd.DataFrame(param_val_i)
param_df_b = pd.DataFrame(param_val_b)

# Print cov matrix for last step of optimization
#Inferno
alpha_post_i = param_df_i.iloc[-1]["alpha"]
cov_post_i = param_df_i.iloc[-1]["cov"]
corr_post_i = correlation_from_covariance(cov_post_i)
print('INFERNO')
print(alpha_post_i)
print(cov_post_i,'\\n')
print(corr_post_i, '\\n')

#Bce
alpha_post_b = param_df_b.iloc[-1]["alpha"]
cov_post_b = param_df_b.iloc[-1]["cov"]
corr_post_b = correlation_from_covariance(cov_post_b)
print('BCE')
print(alpha_post_b)
print(cov_post_b,'\\n')
```

```
print(corr_post_b, '\n')

#Plot loss
print('Loss function of INFERNO')
plot_loss(lt_inf)
print('Loss function of BCE')
plot_loss(lt_bce)
#####
#####

#####
#First part : definition of useful functions
import os
if 'google.colab' in str(get_ipython()):
    os.system('pip install torch==1.9.0 fastcore==1.3.20 pytorch_inferno
              ==0.2.2')

from pytorch_inferno.data import get_paper_data
from pytorch_inferno.utils import init_net
from pytorch_inferno.model_wrapper import ModelWrapper
from pytorch_inferno.inferno import VariableSoftmax, PaperInferno
from torch import nn, optim
from pytorch_inferno.callback import LossTracker, SaveBest, EarlyStopping
from fastcore.all import partiallier
from torch.distributions import Normal
from pytorch_inferno.inferno import InfernoPred
from pytorch_inferno.inference import bin_preds, get_shape, get_paper_syst_
    shapes, calc_profile
from pytorch_inferno.plotting import plot_likelihood, plot_preds
from pytorch_inferno.callback import PredHandler
import pandas as pd
import numpy as np
import torch
from torch import Tensor
import matplotlib.pyplot as plt
from pytorch_inferno.inference import calc_nll, calc_grad_hesse
```

```
from typing import Optional, Union, List, Dict
from torch.distributions import Distribution

def compute_nll(model:ModelWrapper, is_inferno:bool,r_vals=[-0.2,0,0.2],l_
    vals=[2.5,3,3.5]) -> Tensor:
    preds = model._predict_dl(test, pred_cb=InfernoPred() if is_inferno
        else PredHandler())

    df = pd.DataFrame({'pred':preds.squeeze()})
    df['gen_target'] = test.dataset.y
    df.head()

    bins = np.linspace(0,10,11) if is_inferno else np.linspace(0,1,11)
    bin_preds(df, bins=bins)
    #plot_preds(df, bin_edges=bins)
    f_s,f_b = get_shape(df,1),get_shape(df,0)

    b_shapes = get_paper_syst_shapes(bkg, df, model=model, r_vals = r_vals,
        l_vals = l_vals, pred_cb=InfernoPred() if is_inferno else
        PredHandler(), bins=bins)

    nll = calc_profile(f_s_nom=f_s, **b_shapes, n_obs=1050, mu_scan=mu_scan
        , mu_true=50, n_steps=100, shape_aux=[Normal(0,1), Normal(0,1)])
    return nll

#Plot of loss
def plot_loss(lt):

    plt.plot(lt.losses["trn"], label="train")
    plt.plot(lt.losses["val"], label="val")
    plt.ylabel(r"$\sigma^2(\mu)$")
    plt.xlabel(r"epoch")
    plt.legend(loc="upper right")
    plt.show()
```

```

def newton_optimizer(mu_best:float, f_s_nom:Tensor, f_b_nom:Tensor, n_obs:
    int, mu_true:int,
        f_s_up:Optional[Tensor]=None, f_s_dw:Optional[Tensor]=None
        ,
        f_b_up:Optional[Tensor]=None, f_b_dw:Optional[Tensor]=None
        ,
        shape_aux:Optional[List[Distribution]]=None,
        s_norm_aux:Optional[List[Distribution]]=None, b_norm_aux:
        Optional[List[Distribution]]=None, nonaux_b_norm:bool=
        False,
        n_steps:int=100, lr:float=0.1, verbose:bool=True) ->
    Tensor:

r'''Compute profile likelihoods for best mu value, optimising on full
hessian.'''
for f in [f_s_nom, f_s_up, f_s_dw, f_b_nom, f_b_up, f_b_dw]: # Ensure
    correct dimensions
    if f is not None and len(f.shape) < 2: f.unsqueeze_(0)
# Cases where nuisance only causes up xor down variation
if (f_s_up is None and f_s_dw is not None): f_s_up = torch.repeat_
    interleave(f_s_nom, repeats=len(f_s_dw), dim=0)
if (f_s_dw is None and f_s_up is not None): f_s_dw = torch.repeat_
    interleave(f_s_nom, repeats=len(f_s_up), dim=0)
if (f_b_up is None and f_b_dw is not None): f_b_up = torch.repeat_
    interleave(f_s_nom, repeats=len(f_b_dw), dim=0)
if (f_b_dw is None and f_b_up is not None): f_b_dw = torch.repeat_
    interleave(f_s_nom, repeats=len(f_b_up), dim=0)
if f_s_up is not None and f_b_up is not None and len(f_s_up) != len(f_b
    _up):
    raise ValueError("Shape variations for signal & background must
        have the same number of variations. \
        Please enter the nominal templates for nuisances
        that only affect either signal or background.

```

```

        ")
# Norm uncertainties
if s_norm_aux is None: s_norm_aux = []
if b_norm_aux is None: b_norm_aux = []
# Compute nuisance indeces
poi_idx = 0
n_alpha = 1 + np.max((len(f_b_up) if f_b_up is not None else 0, len(f_s_up) if f_s_up is not None else 0))
shape_idxs = list(range(1,n_alpha))
print("shape_idxs", shape_idxs)
s_norm_idxs = list(range(n_alpha, n_alpha+len(s_norm_aux)))
n_alpha += len(s_norm_aux)
b_norm_idxs = list(range(n_alpha, n_alpha+len(b_norm_aux)+nonaux_b_norm))
n_alpha += len(b_norm_aux)+nonaux_b_norm

b_true = n_obs-mu_true

get_nll = partialiller(calc_nll, s_true=mu_true, b_true=b_true,
                      f_s_nom=f_s_nom, f_s_up=f_s_up, f_s_dw=f_s_dw,
                      f_b_nom=f_b_nom, f_b_up=f_b_up, f_b_dw=f_b_dw,
                      s_norm_aux=s_norm_aux, b_norm_aux=b_norm_aux, shape_aux=shape_aux)

# Initialize the parameters
alpha = torch.zeros((n_alpha), requires_grad=True, device=f_b_nom.device)
#mu = torch.tensor(mu_best).to(f_b_nom.device)
with torch.no_grad(): alpha[poi_idx] = mu_best

print("alpha", alpha)

# List to store relevant info
param_val = []

# print("Mu", mu)

```

```

for i in range(n_steps): # Newton optimise nuisances
    nll = get_nll(shape_alpha=alpha[shape_idxs], mu=alpha[poi_idx], s_
                  norm_alpha=alpha[s_norm_idxs], b_norm_alpha=alpha[b_norm_idxs])
    #print("nll", nll)
    grad, hesse = calc_grad_hesse(nll, alpha, create_graph=False)
    #print("grad", grad)
    #print("hesse", hesse)
    # Step is learning rate * gradient * covariance matrix
    cov = torch.inverse(hesse)
    step = lr*grad.detach()@cov
    #print("step", step)
    # Avoid too large steps
    step = torch.clamp(step, -100, 100)
    # Apply the step to the parameters
    alpha = alpha-step
    print("alpha", alpha)
    # Store the parameter infos
    param_val.append({ 'step':i, 'alpha':alpha.detach().numpy(), 'cov':_
                      cov.detach().numpy() })

# Last nll evaluation
nll = get_nll(shape_alpha=alpha[shape_idxs], mu=alpha[poi_idx], s_norm_
               alpha=alpha[s_norm_idxs], b_norm_alpha=alpha[b_norm_idxs]).detach()
return nll, param_val

def best_fit_values(mu_best:float, model:ModelWrapper, is_inferno:bool,r_
                    vals=[-0.2,0,0.2],l_vals=[2.5,3,3.5]) -> Tensor:
    preds = model._predict_dl(test, pred_cb=InfernoPred() if is_inferno
                               else PredHandler())

    df = pd.DataFrame({'pred':preds.squeeze()})
    df['gen_target'] = test.dataset.y
    df.head()

    bins = np.linspace(0,10,11) if is_inferno else np.linspace(0,1,11)
    bin_preds(df, bins=bins)

```

```
f_s,f_b = get_shape(df,1),get_shape(df,0)
b_shapes = get_paper_syst_shapes(bkg, df, model=model, r_vals = r_vals,
                                  l_vals = l_vals, pred_cb=InfernoPred() if is_inferno else
                                  PredHandler(), bins=bins)
nll, param_val = newton_optimizer(mu_best=mu_best, f_s_nom=f_s, **b_
                                   shapes, n_obs=1050, mu_true=50, n_steps=100, shape_aux=[Normal(0,1)
                                   , Normal(0,1)])
return nll, param_val

def get_best_fit(nll, mu_scan):
    return mu_scan[np.argmin(nll)]

def correlation_from_covariance(covariance):
    v = np.sqrt(np.diag(covariance))
    outer_v = np.outer(v, v)
    correlation = covariance / outer_v
    correlation[covariance == 0] = 0
    return correlation

#####
#####! Da fare per salvare ed importare il dataset !!
#Creo un unico dataset che user per fare le 5 implementazioni per
#ciascuno dei 5 casi e lo salvo per poterlo recupere quando esco dalla
#sessione
import numpy as np
from pytorch_inferno.data import *
from pytorch_inferno.data import get_paper_data

from google.colab import drive

def save_array(dl, name):
    np.save(name + '_x.npy', dl.dataset.x)
    np.save(name + '_y.npy', dl.dataset.y)

def load_array(name):
```

```

x = np.load(name + '_x.npy')
y = np.load(name + '_y.npy')
return x, y

def save_dl(data, test, prefix="dataset0"):
    save_array(data.trn_dl, prefix + "_train")
    save_array(data.val_dl, prefix + "_validation")
    save_array(test, prefix + "_test")

def reinit_dl(bs=2000, prefix="dataset0"):
    x_train, y_train = load_array(prefix + "_train")
    x_val, y_val = load_array(prefix + "_validation")
    x_test, y_test = load_array(prefix + "_test")
    trn_dl = WeightedDataLoader(DataSet(x_train, y_train), batch_size=bs,
        shuffle=True, drop_last=True)
    val_dl = WeightedDataLoader(DataSet(x_val, y_val), batch_size=2*bs,
        shuffle=True)
    data = DataPair(trn_dl, val_dl)
    tst_dl = WeightedDataLoader(DataSet(x_test, y_test), batch_size=2*bs)
    return data, tst_dl
#####
######Da fare una volta
######Salvare il dataset
drive.mount('/content/gdrive')
data, test = get_paper_data(200000, bs=2000, n_test=1000000)

# Save arrays
save_dl(data, test, prefix="/content/gdrive/MyDrive/dataset0")
#####

#####
# Load the arrays again and reinitialize the dataloaders
drive.mount('/content/gdrive')
data, test = reinit_dl(prefix="/content/gdrive/MyDrive/dataset0")

```

```
#####
#####
#Second part : eseguo questa cella una volta per ogni caso, e mi
    restituisce gli esiti di 5 run sullo stesso dataset
#extraction of values from INFERNO and from BCE with data sample c_i
#da svolgere 5 volte per ogni caso

#Part to modify
std_r = 1
std_l = 0.5

for i in range(5):
    print('Run number:', i+1)

#Inferno model training
net = nn.Sequential(nn.Linear(3,100), nn.ReLU(),
                    nn.Linear(100,100),nn.ReLU(),
                    nn.Linear(100,10), VariableSoftmax(0.1))

init_net(net)
model_inf = ModelWrapper(net)
lt_inf = LossTracker()
model_inf.fit(100, data=data, opt=partialiller(optim.Adam,lr=1e-4), loss=
    None,
    cbs=[PaperInferno(float_r=True, float_l=True, shape_aux=[

        Normal(0,std_r), Normal(0,std_l)], nonaux_b_norm=False),
         ,
         lt_inf,SaveBest('weights/best_ie.h5'),EarlyStopping(100)])


#BCE model training
net2 = nn.Sequential(nn.Linear(3,100),   nn.ReLU(),
                     nn.Linear(100,100), nn.ReLU(),
                     nn.Linear(100,1),   nn.Sigmoid())

init_net(net2)
lt_bce = LossTracker()
```

```

model_bce = ModelWrapper(net2)
model_bce.fit(100, data=data, opt=partialler(optim.Adam,lr=1e-4), loss=nn
    .BCELoss(),
    cbs=[lt_bce,SaveBest('weights/best_ie.h5'),EarlyStopping
        (100)])

```



```

mu_scan = torch.linspace(20,80,61)
bkg = test.dataset.x[test.dataset.y.squeeze() == 0]

```



```

#The changes in inference
r_vals = [0. - std_r, 0., 0. + std_r]
l_vals = [3. - std_l, 3., 3. + std_l]
nll_inf = compute_nll(model_inf, True , r_vals = r_vals, l_vals = l_vals
    )
nll_bce = compute_nll(model_bce, False, r_vals = r_vals, l_vals = l_vals)

```



```

# Get best fit value for mu
mu_best_inf = get_best_fit(nll_inf, mu_scan)
mu_best_bce = get_best_fit(nll_bce, mu_scan)

# Minimize NLL at best fit value
nll_i, param_val_i = best_fit_values(mu_best_inf, model_inf, True, r_vals
    =r_vals, l_vals =l_vals)
nll_b, param_val_b = best_fit_values(mu_best_bce, model_bce, False, r_
    vals=r_vals, l_vals =l_vals)
param_df_i = pd.DataFrame(param_val_i)
param_df_b = pd.DataFrame(param_val_b)

# Print cov matrix for last step of optimization
#Inferno
alpha_post_i = param_df_i.iloc[-1]["alpha"]
cov_post_i = param_df_i.iloc[-1]["cov"]

```

```
corr_post_i = correlation_from_covariance(cov_post_i)
print('INFERNO')
print(alpha_post_i)
print(cov_post_i, '\n')
print(corr_post_i, '\n')

#Bce
alpha_post_b = param_df_b.iloc[-1]["alpha"]
cov_post_b = param_df_b.iloc[-1]["cov"]
corr_post_b = correlation_from_covariance(cov_post_b)
print('BCE')
print(alpha_post_b)
print(cov_post_b, '\n')
print(corr_post_b, '\n')

#Plot loss
print('Loss function of INFERNO')
plot_loss(lt_inf)
print('Loss function of BCE')
plot_loss(lt_bce)
#####
#####

#Code that I have to run only the first time
import os
if 'google.colab' in str(get_ipython()):
    os.system('pip install torch==1.9.0 fastcore==1.3.20 pytorch_inferno
              ==0.2.2')

from pytorch_inferno.data import get_paper_data
from pytorch_inferno.utils import init_net
from pytorch_inferno.model_wrapper import ModelWrapper
from pytorch_inferno.inferno import VariableSoftmax, PaperInferno
from torch import nn, optim
from pytorch_inferno.callback import LossTracker, SaveBest, EarlyStopping
```

```
from fastcore.all import partialler
from torch.distributions import Normal
from pytorch_inferno.inferno import InfernoPred
from pytorch_inferno.inference import bin_preds, get_shape, get_paper_syst_
    shapes, calc_profile
from pytorch_inferno.plotting import plot_likelihood, plot_preds
from pytorch_inferno.callback import PredHandler
import pandas as pd
import numpy as np
import torch
from torch import Tensor
import matplotlib.pyplot as plt

def compute_nll(mu_true, n_obs, model:ModelWrapper, is_inferno:bool,r_vals
    =[-0.2,0,0.2],l_vals=[2.5,3,3.5]) -> Tensor:
    preds = model._predict_dl(test, pred_cb=InfernoPred() if is_inferno
        else PredHandler())

    df = pd.DataFrame({'pred':preds.squeeze()})
    df['gen_target'] = test.dataset.y
    df.head()

    bins = np.linspace(0,10,11) if is_inferno else np.linspace(0,1,11)
    bin_preds(df, bins=bins)
    #plot_preds(df, bin_edges=bins)
    f_s,f_b = get_shape(df,1),get_shape(df,0)

    b_shapes = get_paper_syst_shapes(bkg, df, model=model, r_vals = r_vals,
        l_vals = l_vals, pred_cb=InfernoPred() if is_inferno else
        PredHandler(), bins=bins)
    nll = calc_profile(f_s_nom=f_s, **b_shapes, n_obs=n_obs, mu_scan=mu_
        scan, mu_true=mu_true, n_steps=100, shape_aux=[Normal(0,1), Normal
        (0,1)])
    return nll
```

```
#Plot of loss
def plot_loss(lt):

    plt.plot(lt.losses["trn"], label="train")
    plt.plot(lt.losses["val"], label="val")
    plt.ylabel(r"$\sigma^2(\mu)$")
    plt.xlabel(r"epoch")
    plt.legend(loc="upper right")
    plt.show()

#Functions to fit models and calculate nll
def fit_bce( data, std_r, std_l, mu_true, n_obs ):

    net2 = nn.Sequential(nn.Linear(3,100), nn.ReLU(),
                         nn.Linear(100,100), nn.ReLU(),
                         nn.Linear(100,1), nn.Sigmoid())
    init_net(net2)
    model_bce = ModelWrapper(net2)
    lt_bce = LossTracker()

    #Aumento il numero di iterazioni perch non giunge a convergenza
    model_bce.fit(130, data=data, opt=partialler(optim.Adam,lr=1e-5), loss=
                  nn.BCELoss(),
                  cbs=[lt_bce,SaveBest('weights/best_ie.h5'),EarlyStopping
                       (100)])
    plot_loss(lt_bce)

    r_vals = [0.-std_r, 0. , 0.+std_r]
    l_vals = [3.-std_l, 3. , 3.+std_l]
    print('r_vals', r_vals)
    print('l_vals', l_vals)

    nll_bce = compute_nll(mu_true=mu_true, n_obs=n_obs, model=model_bce, is
                          _inferno=False , r_vals = r_vals, l_vals = l_vals)
```

```

    return nll_bce

def fit_inferno( data, std_r, std_l, mu_true, b_true, n_obs ):

    net = nn.Sequential(nn.Linear(3,100), nn.ReLU(),
                        nn.Linear(100,100),nn.ReLU(),
                        nn.Linear(100,10),
                        VariableSoftmax(0.1))
    init_net(net)
    model_inf = ModelWrapper(net)
    lt_inf = LossTracker()

    #Aumentiamo le epochhe
    model_inf.fit(130, data=data, opt=partiallier(optim.Adam,lr=1e-5), loss=
        None,
        cbs=[PaperInferno(b_true=b_true, mu_true=mu_true,float_r=
            True, float_l=True, shape_aux=[Normal(0,std_r),
            Normal(0,std_l)], nonaux_b_norm=False), lt_inf,SaveBest('
            weights/best_ie.h5'),EarlyStopping(100)])
    plot_loss(lt_inf)
    r_vals = [0.-std_r, 0. , 0.+std_r]
    l_vals = [3.-std_l, 3. , 3.+std_l]
    print('r_vals', r_vals)
    print('l_vals', l_vals)
    nll_inf = compute_nll(mu_true=mu_true,model=model_inf, is_inferno=True
        , r_vals = r_vals, l_vals = l_vals, n_obs=n_obs)
    return nll_inf
#####
#####We compute 5 time the case r = N(0,0.2) and llambda=N(0,0.5)
#B=1000
S = 30
results_bce = []

```

```
results_inf = []
for i in range(4):
    print('Run number:', i+1)
    data, test = get_paper_data(200000, bs=2000, n_test=1000000) #generate a
        new sample of data
    mu_scan = torch.linspace(S-50,S+50,61) #####da cambiare
    bkg = test.dataset.x[test.dataset.y.squeeze() == 0]

    std_r = 0.2
    std_l = 0.5

    mu_true = S #part to modify
    b_true = 1000
    n_obs = mu_true + b_true

    nll_inf = fit_inferno( data, std_r= std_r, std_l= std_l, b_true=b_true,
        mu_true=mu_true, n_obs =n_obs )
    nll_bce = fit_bce( data, std_r= std_r, std_l= std_l, mu_true=mu_true, n_
        obs=n_obs)

    std_bce, std_inf = plot_likelihood({'BCE':nll_bce, 'INFERNO':nll_inf}, mu
        _scan)
    results_bce.append(std_bce)
    results_inf.append(std_inf)
#####
#####
```

Appendice B

Codice dati di CMS

Il codice si basa sulla versione in PYTORCH implementata da Strong 2021 che utilizza i dati riprodotti da Layer 2022.

```
import inferno_opendata
import inferno_config
import numpy as np

args = inferno_config.args

# Data reduction
for fact in [0.3, 0.5, 0.7, 0.9]:
    args["outpath"] = "/home/centos/mount_point/data_red/fact_"+str(fact)
    args["shape_syst"] = ["jes"]
    args["temperature"] = 0.1
    args["bins"] = 10
    args["downsample_factor"] = fact
    samples = inferno_opendata.run_cmsopen(args, epochs = 100, do_fit =
        True)

# Bins
for bins in [5, 8, 11, 14, 17, 20]:
    args["outpath"] = "/home/centos/mount_point/bins/bins_"+str(bins)
```

```
args["shape_syst"] = ["jes"]
args["temperature"] = 0.1
args["bins"] = bins
args["downsample_factor"] = fact
samples = inferno_opendata.run_cmsopen(args, epochs = 100, do_fit =
True)

# Temperature
for temp in [0.001, 0.05, 0.1, 0.5, 0.9, 0.99]:
    args["outpath"] = "/home/centos/mount_point/temp/temp_"+str(temp)
    args["shape_syst"] = ["jes"]
    args["temperature"] = temp
    args["bins"] = 10
    args["downsample_factor"] = 0.9
    samples = inferno_opendata.run_cmsopen(args, epochs = 100, do_fit =
True)
```
