UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Facoltà di Ingegneria
Corso di Studi in Ingegneria delle Telecomunicazioni

tesi di laurea

Analisi delle prestazioni di algoritmi di Machine Learning per la classificazione del traffico di rete

Anno Accademico 2012/2013

relatore
Ch.mo prof. A. Pescapè

correlatori
Ch.mo dott. M. Brescia
Ing. G. Aceto

candidata
Luna Di Colandrea
matr. N40 000132

Ai miei genitori ,
e ai miei nonni

Ringraziamenti

Ringrazio il prof. Pescapè , il prof. Brescia e l'ingegner Aceto senza il cui aiuto, non sarebbe stata possibile la realizzazione di questa tesi . Ringrazio , inoltre, tutta la mia famiglia che mi ha sempre sostenuta e motivata, il mio fidanzato e i miei amici , senza i quali non sarei mai riuscita ad arrivare fin qui . Infine ringrazio anche tutte quelle persone che avevano detto che "non ce l'avrei mai fatta".

Indice

Introduzione

In questo elaborato di tesi si affrontano le tematiche relative alla classificazione del traffico di rete. Si è scelto di analizzare questo problema a causa dei suoi molteplici campi d'applicazione, come ad esempio nella sicurezza informatica, in cui è possibile utilizzare tecniche di classificazione del traffico basate sul controllo del payload per rilevare attacchi da entità malevole; oppure nel campo del traffic engineering, per studiare o forzare delle proprietà legate alla Quality of Service relative ad uno specifico servizio di rete.

Oggigiorno, infatti, il moderno traffico Internet ha una struttura ben più complicata di quanto i suoi utilizzatori possano immaginare; ciò fa di Internet un'importante quanto critica infrastruttura di comunicazione che offre servizi come la sicurezza, l'affidabilità, la privacy e il servizio multiplo di qualità, in un'architettura "best effort", originariamente prevista per il solo supporto di un ambiente di ricerca. Per poter garantire servizi di priorità, protezione o per impedire un certo traffico di rete, i providers necessitano, dunque, di implementare nuove tecnologie per la classificazione del traffico di rete.

Nonostante ciò non esiste nessun sistema scientificamente riproducibile di ricerca sulle caratteristiche del traffico Internet a livello mondiale, a causa della sensibilità e delle restrizioni tipiche dovute alla condivisione dei dati di traffico [8]. Uno dei problemi fondamentali di questa disciplina, dovuto a motivi di privacy, infatti, è proprio il recupero delle tracce di flusso su cui lavorare.

Nel presente lavoro, quindi, tra le varie metodologie di classificazione del traffico di rete proposte in letteratura si è scelto di focalizzare l'attenzione su quelle basate sul Machine Learning (ML). A tal proposito sono stati scelti e studiati due algoritmi innovativi di ML quali FMLPGA (Fast Multi Layer Perceptron with Genetic Algorithm) e GAME (Genetic Algorithm Modeling Experiment); è stata quindi condotta un'analisi volta a selezionare le caratteristiche del traffico utilizzabili come input per gli algoritmi e sono stati pianificati ed eseguiti esperimenti utilizzando tracce di traffico di rete reali su entrambi gli algoritmi, a cui

è seguita l'analisi e la discussione dei risultati ottenuti sia in termini qualitativi che quantitativi. La realizzazione degli esperimenti, inoltre, ha incluso la pre-elaborazione delle tracce di traffico, oggetto di analisi per l'estrazione dei dati di input processabili dagli algoritmi, utilizzando strumenti per l'elaborazione di tracce in formato pcap (tcpdump, wireshark), e strumenti per il data mining con algoritmi di Machine Learning (progetto DAME, Data Mining & Exploration).

Capitolo 1

La classificazione del traffico di rete

La classificazione del traffico di rete consiste nell'associare ad una sequenza di pacchetti scambiati tra due dispositivi dotati di NIC (Network Interface Card), e di due rispettive porte a livello di trasporto, la presunta applicazione che li ha generati. La sua importanza è dovuta, essenzialmente a due aree molto importanti : la qualità del servizio delle reti IP (QoS) e i sistemi di intercettazione legale (LI), [10].

Oggigiorno, a causa della congestione della rete Internet, la strategia comune utilizzata dai fornitori di servizi di rete è basata sul sotto-utilizzo (over provisioning) della capacità trasmissiva del collegamento. Per la maggior parte degli ISP (Internet Service Provider),però, questa soluzione è tutt'altro che conveniente in termini economici. Un meccanismo di tariffazione,si rende allora necessario, per differenziare i clienti con esigenze e consumi diversi per la QoS che ricevono. Ciò implica che tutti i sistemi QoS hanno un certo grado di classificazione del traffico IP implicito nel loro design. La classificazione del traffico, quindi, permette anche di sostenere un'implementazione di rete basata su una ripartizione non omogenea della QoS tra i vari utenti.

La Classificazione del traffico, inoltre, è anche un'importante soluzione per la richiesta emergente di intercettazioni legali (LI). I governi, di solito, chiedono l'attuazione di LI a vari livelli di astrazione. Tra gli ISP, allora, tecniche di classificazione del traffico offrono la possibilità di utilizzare modelli identificativi del traffico di rete e di identificare ,quindi, quali classi di applicazioni siano utilizzate da una persona soggetta a controllo, in un qualsiasi momento. A seconda, poi, del particolare schema di identificazione utilizzato, le

informazioni raccolte possono essere potenzialmente ottenute senza alcuna violazione della privacy del soggetto in questione.

I metodi per la classificazione del traffico proposti sia nella letteratura accademica che nel caso pratico sono : metodi port-based, metodi basati sul controllo del payload, metodi basati sul controllo del flusso e metodi basati su un approccio di Machine Learning.

1.1 *Port-based* classification

I metodi basati sul controllo delle porte sono stati i primi ad essere utilizzati. Essi si basano sull'ispezione dei 16bit del numero di porta del livello di Trasporto, usati dai server per determinare lo specifico processo che riceve un certo flusso di traffico. L'attribuzione dei numeri di porta è assegnata dall'Internet Assigned Numbers Authority (IANA).I primi studi condotti con questa metodologia si basavano sul fatto che a molti protocolli fosse assegnata una porta "ben conosciuta"(well-known port) registrata dallo IANA. Tuttavia questi metodi soffrono di alcune problematiche dovute a, [7] [8] :

- La proliferazione di nuove applicazione non standardizzate comporta l'utilizzo di porte non registrate dallo IANA;
- L'uso improprio da parte di progettisti e utenti delle "well-known port" per mascherare il loro traffico e aggirare filtri o firewall;
- L'inevitabile esaurimento degli indirizzi IPv4 e il conseguente utilizzo di NAT(Network Address Translation);
- Il protocollo FTP(File Transfer Protocol) in modalità passiva può volutamente non utilizzare le porte note, e assegnarle, quindi, dinamicamente;
- Il traffico P2P(Peer-to-Peer) con il suo notevole consumo di banda e le sue particolari caratteristiche di traffico tra cui l'utilizzo dinamico delle porte e le questioni dovute alla distribuzione di materiale protetto da copyright, costituisce un notevole problema per questa metodologia di classificazione.

Nonostante i suoi numerosi problemi, la classificazione basata sul numero di porta rimane ancora il metodo più semplice e veloce.

1.2 *Inspection of packet payload data* classification

Attualmente, un metodo di classificazione del traffico molto popolare è il metodo di classificazione basato sull'ispezione del carico del pacchetto, grazie alla sua relativamente elevata precisione. Sotto quest'aspetto i payloads dei pacchetti sono esaminati bit a bit per localizzare specifici streams di bit in cui siano presenti le firme identificative di un certo protocollo di rete. Se viene trovato un certo stream, con conseguente identificazione di una firma , il pacchetto può essere correttamente etichettato e classificato. La ricerca della firma può richiedere mirroring (passivo) o intercettazione (inline) del traffico. Generalmente questa tecnica è utilizzata per il traffico P2P e per la rilevazione delle intrusioni in una rete. Un metodo basato sull'ispezione del payload dei pacchetti è il Deep Packet Inspection (DPI); esso è uno dei metodi implementabili su GPU (Graphics Processing Unit).

La *Inspection of packet payload data classification* è considerata una tecnica affidabile per la classificazione del traffico Internet, ma pone pesanti sfide alla privacy; le politiche di privacy possono, infatti, impedire l'accesso o l'archiviazione del contenuto del pacchetto. Questa tecnica, inoltre, risulta proibitiva anche dal punto di vista tecnologico ed economico.Infatti, la crittografia, l'offuscamento del protocollo o l'incapsulamento (HTTP tunneling), aumentano la complessità computazionale e il costo dell'uso dei collegamenti a larga banda. Inoltre tale tecnica può risultare inefficace nel caso di nuove applicazioni di cui la firma non sia ancora conosciuta. Tali problematiche hanno motivato i ricercatori a trovare nuovi algoritmi discriminatori di classi di traffico di rete.

1.3 *Flow-based* classification

In questo approccio, delle tracce di flusso,che riassumono i tratti caratteristici dei flussi di rete, vengono raccolte solo le intestazioni. In questo caso, quindi, i problemi riscontati nel metodo precedente, quali la quantità di dati da analizzare, la crittografia e la riservatezza dei dati, non costituiscono alcun problema. Questa metodologia prevede che ogni traccia di flusso sia caratterizzata da un ben definito set di caratteristiche che assumono differenti valori a seconda della rispettiva classe di traffico di rete. Più precisamente, le caratteristiche sono informazioni che possono direttamente essere estratte dall'header IP. Si noti però

che nei metodi flow-based è richiesta la trasmissione di tutti i pacchetti del flusso prima che la classificazione possa essere eseguita.

1.4 Classification with *Machine learning*

L'applicazione del Machine Learning (ML) alla classificazione del traffico di rete mostra risultati promettenti, anche nel caso di traffico cifrato o oscurato e senza l'analisi del payload. Il primo controllore di traffico di rete basato su tale metodologia fu usato in una rete di Telecomunicazioni nel 1990. Successivamente, nel 1994, il ML fu utilizzato per la prima volta per un classificatore di un flusso Internet nel contesto del rilevamento delle intrusioni, [10].

La particolarità di questi sistemi riguarda il fatto che imparano da dati empirici ad associare automaticamente ad un oggetto la classe di traffico corrispondente. Un sistema basato sul ML prende in ingresso set di dati di istanze in cui ogni istanza è caratterizzata dai valori delle sue caratteristiche (note anche come attributi, features o discriminatori), che misurano diversi aspetti dell'istanza. Nel contesto della classificazione del traffico di rete,con il termine *classe* si intende il traffico IP causato da (o appartenente a) un'applicazione o da un gruppo di applicazioni,mentre con il termine *istanza,* di solito, si intendono più pacchetti appartenenti allo stesso flusso o biflusso. Le caratteristiche, invece, sono attributi tipicamente numerici, calcolati per più pacchetti appartenenti a singoli flussi o biflussi .

Il dataset d'ingresso, infine, viene presentato come una matrice di casi di caratteristiche, mentre l'uscita è la descrizione della conoscenza appresa (target).

Esistono due tipi di algoritmi di ML :

- Nel ML *supervisionato* le classi sono definite dall'utente, inglobate all'interno del dataset di training fornito in ingresso al sistema, associate ad ogni istanza;
- Nel ML *non supervisionato*, invece, il sistema identifica e classifica da solo le classi grazie a procedure di auto-organizzazione, come ad esempio il metodo di clustering.

Generalmente per classificare il traffico di rete si utilizzano tecniche con ML supervisionato anche se tecniche di apprendimento automatico senza supervisione risultano un modo promettente per far fronte ai continui cambiamenti del traffico rete. La questione principale, relativa alla valutazione della classificazione con ML supervisionato, riguarda il fatto che sia i dati di training che quelli di test devono essere correttamente etichettati; ovvero le classi di traffico dei dati di addestramento e di prova devono essere preventivamente note. Le prestazioni di tali classificatori, inoltre, dipendono non solo dalle diverse tipologie di algoritmi di apprendimento automatico utilizzati, ma anche dalla selezione delle caratteristiche di classificazione, ovvero i dati utilizzati per presentare ciascun flusso al sistema di apprendimento. Generalmente le caratteristiche includono proprietà comuni dei flussi, nonché proprietà più dettagliate come dimensione e inter-packet Time.

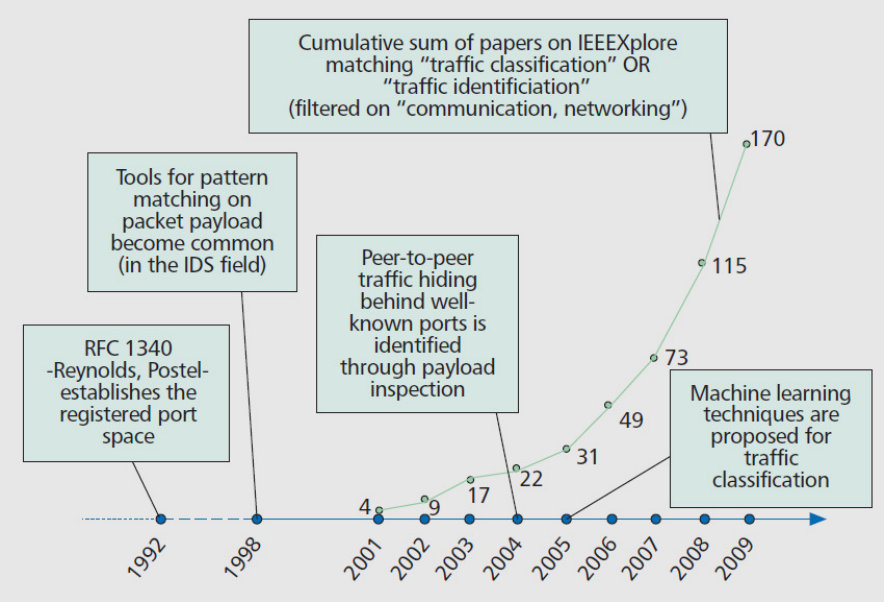


Figura 1.1 [8] Evoluzione delle metodologie di classificazione

1.5 Scalabilità e l'introduzione del GPGPU

La costante evoluzione di Internet comporta un'enorme crescita di infrastrutture e di banda dei collegamenti trasmissivi. Molte applicazioni per la classificazione del traffico di rete, inoltre, richiedono strumenti per lavorare in linea; i progettisti, quindi, si trovano davanti ad un necessario compromesso tra precisione, prestazioni e costi. Per non essere penalizzati da costi Hardware proibitivi, si è pensato ad esempio di lavorare su porzioni minori di payload. Le tecniche di classificazione con ML richiedono compromessi simili legati alla latenza di classificazione; la riduzione dei dati, in questo caso, è ottenuta limitando il numero di pacchetti usati per estrarre le caratteristiche di classificazione, e limitando l'insieme di quest'ultime. Idealmente la latenza è anche influenzata dalla velocità dello specifico algoritmo di apprendimento utilizzato.

A valle di ciò, nel prossimo decennio i sistemi di classificazione del traffico saranno progettati per lavorare su strutture hardware multi-core e many-core per architetture altamente parallele. Il paradigma General-purpose Graphic Process Unit (GPGPU) ha introdotto un nuovo paradigma di calcolo consentendo l'utilizzo di applicazioni computazionalmente intensive, raggiungendo un sostanziale miglioramento delle prestazioni a costi hardware più contenuti[8]. L'utilizzo di più core nel processo di classificazione ha accelerato i tempi di esecuzione, diminuendo la latenza. Gli algoritmi di ML come il Support Vector Machine (SVM), implementabili su strutture parallele, hanno contribuito a un miglioramento della scalabilità. Inoltre si può perseguire una struttura parallela anche allo strato più alto di un'architettura di classificazione del traffico di rete, utilizzando un multithreading della tipica sequenza di esecuzione.

1.6 Metriche per la valutazione della classificazione del traffico

Quando si vuole valutare una tecnica di classificazione del traffico di rete si usa come parametro *l'accuratezza predittiva,* come ad esempio il grado di precisione con cui il modello prende le decisioni.

Detta X la classe di traffico d'interesse, il suo classificatore deciderà se un flusso o un pacchetto apparterrà o meno alla classe X. Per verificare l'accuratezza del classificatore, quindi, si utilizzano parametri noti quali[10] :

- Falsi negativi (FN) : percentuale di membri della classe X erroneamente classificati come non appartenenti alla classe di interesse;
- Falsi positivi (FP) : percentuale dei membri di altre classi erroneamente classificati come appartenenti alla classe X;
- Veri positivi (TP) : percentuale dei membri della classe X correttamente classificati come appartenenti alla classe X;
- Veri negativi (TN) : percentuale dei membri di altre classi correttamente classificati come non appartenenti alla classe X;

Classified as →	X	$\overline{X}$
X	TP	FN
$\overline{X}$	FP	TN

Figura 1.2 [10]

Un buon classificatore di traffico risulta tale se esso minimizza i parametri di Falsi Negativi e Positivi.

In tale contesto esiste anche la *valutazione metrica* .Essa è definita come la percentuale di flussi o pacchetti correttamente classificati rispetto al totale. Nella letteratura riguardante la classificazione con ML, inoltre, si utilizzano due ulteriori parametri quali :

- Recall : percentuale di membri della classe X correttamente classificati come appartenenti alla classe X;
- Precisione : percentuale di quei casi che appartengono veramente alla classe X , tra tutti quelli classificati come appartenenti alla classe X.

Capitolo 2

Concetti base del Machine Learning

2.1 Note sul Data Mining

Negli ultimi anni, i progressi nel campo dell'acquisizione di dati digitali e l'incremento delle prestazioni dei dispositivi di immagazzinamento hanno portato alla crescita di grandi database. I dati contenuti in questi database vanno dalle transazioni bancarie e tabulati telefonici fino ad arrivare a dati tecnico/scientifici come cartelle cliniche, dati astronomici, ecc.

Si parte da un presupposto reale e fondamentale: viviamo in un mondo sommerso da un'infinità di dati. I dati possono essere di molti tipi: tabelle, immagini, grafici, osservati, simulati, calcolati da statistiche o acquisiti da diversi tipi di sistemi di monitoraggio. Inoltre la recente esplosione del World Wide Web e altre risorse ad alte prestazioni stanno rapidamente contribuendo alla proliferazione di enormi database di dati. Questa grande quantità di dati porta ad una domanda importante: in che modo si possono gestire, capire e usare in modo efficiente e completo? E' ormai ampiamente riconosciuto lo squilibrio cronico tra la crescita dei dati disponibili e la capacità di gestirli [20] e nella maggior parte dei casi, i dati acquisiti, non sono direttamente interpretabili e comprensibili; questo o perché sono oscurati da alcune informazioni ridondanti o da fonti di rumore, oppure perché hanno bisogno di essere fusi con altri dati.

Il "quarto paradigma della scienza" [20], pone in primis, ancora prima della rappresentazione o della strategia di memorizzazione, il problema della comprensione dei

dati. Questo infatti pone in termini scientifici una metodologia per ricavarsi informazioni utili a partire da una conoscenza senza pregiudizi e da dataset di qualsiasi tipo[14]. Di norma questo paradigma impone l'uso di strumenti informatici efficaci e versatili, in grado di colmare il divario tra le limitate capacità umane (in termini di tempo di elaborazione) ed una crescita costante della quantità e della complessità dei dati oggi in nostro possesso. In altre parole si necessita di un software in grado di replicare le alte capacità di apprendimento e adattamento del cervello umano, e allo stesso tempo riuscire a gestire un numero molto elevato di dati, caratteristica dei moderni calcolatori.

Queste due prerogative, sono, i punti cardine sui quali si basano due discipline di apprendimento: ***Data Mining (DM)*** e ***Machine Learning(ML)***.

Andando nel particolare, ci sono due concetti chiave che devono essere chiariti:

- Cosa si intende tecnicamente con il termine "apprendimento”?
- Come sfruttare le capacità computazioni delle macchine per eseguire esperimenti di apprendimento?

Generalmente l'aspetto importante non è tanto se un computer riesca ad apprendere o meno, bensì se sia in grado di rispondere in maniera corretta a specifiche domande. Questa capacità non è però sufficiente per affermare che un computer abbia imparato, soprattutto se si considera che il vero apprendimento è legato alla capacità di generalizzazione di un problema. In altre parole, il fatto che una macchina dia risposte corrette a domande note, utilizzate durante la fase di addestramento, è solo la fase preliminare della sua formazione completa. Quello che più interessa è il comportamento della macchina in situazioni imprevedibili, cioè in quelle domande mai poste durante la fase di "training".

Il Machine Learning è una disciplina scientifica che si occupa della progettazione e dello sviluppo di algoritmi che consentono ai computer di far evolvere i propri comportamenti basandosi su dati empirici.

Un algoritmo può usufruire di esempi (dati) per cogliere caratteristiche di interesse della loro distribuzione statistica. Questi dati costituiscono la cosiddetta Knowledge Base (KB), ovvero un insieme sufficientemente grande di esempi da utilizzare per l'addestramento dell'algoritmo, e per testarne le prestazioni.

I metodi di DM, d'altro canto, sono molto utili per ottenere informazioni a partire da piccoli dataset e, pertanto, possono essere utilizzati in maniera efficace per affrontare problemi di scala molto più piccola[11] .

2.2 Cos'è il Machine Learning

Con il termine ***"Machine Learning"*** si intende una delle discipline fondamentali dell'intelligenza artificiale che si occupa di elaborare algoritmi di ottimizzazione auto-adattivi, ossia in grado di plasmare se stessi a partire da una fase di addestramento su dati noti per lo specifico problema[16] .Nel particolare questa scienza prevede che una macchina possa modificare il suo modo di interagire con il mondo esterno grazie all'analisi approfondita di eventi noti. Da questa definizione si può dedurre che il ML rappresenta una forma di adattamento analoga a quella che avviene per gli organismi biologici i quali (in tempi ovviamente più lunghi), si adattano all'ambiente in cui si trovano.

Per effettuare l'apprendimento la macchina ha bisogno di due componenti:

- Un dataset contenente informazioni riguardanti il dominio di applicazione
- Un algoritmo di apprendimento in grado di estrarre le conoscenze dal dataset in analisi.

Il primo esperimento di ML fu effettuato negli anni '50, dallo scienziato americano "Arthur Lee Samuel",il quale progettò un programma in grado giocare a scacchi ad alti livelli [28]. La cosa sorprendente era che Samuel non era assolutamente un bravo giocatore di scacchi; egli infatti progettò un programma che effettuava centinaia di partite contro se stesso, durante le quali la macchina imparava quali fossero le mosse più o meno opportune da fare durante una partita.

Esistono due tipologie di modelli di ML:

- ***Modelli supervisionati***: Approccio "top-down" applicabile quando è noto il dominio del problema;

- ***Modelli non supervisionati***: Approccio "botton-up" che, diversamente dall'apprendimento supervisionato, costruisce modelli a partire da dati non appartenenti a classi predefinite.

2.3 I paradigmi di apprendimento : ML supervisionato e non

Nel ML supervisionato si ha un set di data point o osservazioni di cui si conosce il risultato (in questo caso la classe di appartenenza). Quest'ultimo fornisce alcuni livelli di supervisione che sono usati dal modello per regolare i parametri e per prendere delle decisioni, permettendo così la predizione dell'output corretto per dei nuovi dati.

I passi costituenti un processo di ML supervisionato consistono in[11]:

1) Realizzazione di un modello di input;
2) Costruzione del dataset di addestramento;
3) Addestramento;
4) Validazione;
5) Uso;

In un modello di ML non supervisionato , invece, non si fornisce alcun valore o label delle classi di appartenenza alle istanze in input. In questo caso dunque il sistema si auto-organizza, provando a creare sovra-densità (clusters) di dati che sono inerentemente simili (ad esempio in termini di una metrica prescelta). Non occorre sapere necessariamente cosa li renda simili, poiché gli algoritmi non supervisionati sono capaci di cercare relazioni nascoste tra i dati e raggrupparli di conseguenza.

Le fasi caratteristiche di un processo di ML non supervisionato sono[11]:

1) Selezione delle caratteristiche di allineamento;
2) Esecuzione;

3) Validazione;

In altre parole,mentre il ML supervisionato cerca di minimizzare l'errore di classificazione delle osservazioni (sulla base di informazioni pre-acquisite), un processo di ML non supervisionato cerca di creare gruppi di dati organizzati per similitudine.

2.4 Domini di applicazione[11]

La scelta di un modello di machine learning dovrebbe essere sempre accompagnata dalla definizione di un dominio funzionale. Infatti, all'interno dei due paradigmi di ML, definiti in precedenza, esistono diverse categorie di algoritmi specializzati nella risoluzione di problemi di ottimizzazione afferenti alle seguenti funzionalità:

- classificazione;
- regressione;
- riduzione delle dimensionalità dello spazio dei parametri;
- clustering;
- filtraggio adattivo;
- pattern recognition;

Nel presente lavoro, avendo impiegato algoritmi di ML specializzati in classificazione, focalizziamo l'attenzione su questa funzionalità.

2.4.1 Classificazione

La classificazione è una procedura nella quale singoli oggetti vengono assegnati a gruppi definiti, grazie a dataset contenenti informazioni inerenti agli oggetti stessi e sulla base di un training set di voci precedentemente etichettate [21]. Si intende con "classificatore" un

sistema che esegue una mappatura tra uno spazio di caratteristiche X e un set di etichette Y. Fondamentalmente un classificatore assegna un etichetta predefinita per ogni campione.

Un problema di classificazione può essere formalmente formulato come segue: dati i seguenti dati di training {(x_1, y_1),.,(x_n, y_n)}, (dove x_i sono vettori), un classificatore h:X->Y mappa un oggetto x ε X sulla sua etichetta di classificazione y ε Y.

Durante la classificazione possono verificarsi le seguenti situazioni:

- ***"Classificazione esatta"***: dato un pattern 'x' in input, il classificatore ritorna la sua etichetta 'y' (scalare);
- ***"Classificazione probabilistica":*** Dato un pattern 'x' in input, il classificatore restituisce un vettore 'y' che contiene la probabilità che 'y_i' sia l'etichetta giusta per 'x'. In altre parole cerchiamo, per ogni vettore, la probabilità che esso sia un membro della classe y_i (qualunque sia 'y_i').

Entrambi i casi possono essere applicati in esperimenti che prevedono due o più classi di classificazione. Le fasi di classificazione sono sostanzialmente tre:

- ***Fase di Training:*** durante questa fase l'algoritmo viene addestrato a restituire delle valutazione di qualche tipo, a partire da un insieme di pattern (istanze) di input, dotati di relativa label della classe di appartenenza;
- ***Fase di Testing:*** prevede una serie di test durante i quali l'algoritmo "addestrato" prende in ingresso pattern e vettori di "target" mai usati durante il training e restituisce dati statistici, matrici di confusione, errore globale, nonché le etichette di classificazione ricavate per ogni pattern;
- ***Fase di valutazione:*** In questa fase viene dato all'algoritmo un insieme di dati di cui non si conosce il valore di output. L'algoritmo restituisce le etichette di classificazione per ogni pattern di input in base alle sue capacità ottenute durante le due fasi precedenti.

A causa della natura "supervisionata" degli esperimenti di classificazione, le prestazioni del sistema possono essere misurate durante la fase di "testing" tramite l'utilizzo di un insieme di vettori di "target", i quali vengono confrontati con le etichette generate dall'algoritmo.

Tuttavia l'errore di classificazione può non sempre essere considerato rappresentativo della qualità del classificatore stesso. Infatti, nel caso in cui il dataset sia "sbilanciato" (percentuali fortemente diverse di pattern per ogni classe del problema), il tasso di errore potrebbe essere elevato anche se il classificatore, da parte sua, risulta performante.

Per una corretta analisi delle prestazioni di classificazione può essere calcolata una matrice di confusione [25]: ogni colonna della matrice rappresenta le istanze di una classe prevista, mentre ciascuna riga rappresenta le istanze di una classe reale. Uno dei principali benefici della matrice di confusione è quello di verificare in modo semplice se il sistema stia mischiando due classi. In alternativa, si può utilizzare una procedura di validazione (molti modelli di classificazione non necessitano di questa procedura).La procedura di validazione è un processo che ha il compito di controllare la genuinità del classificatore. Questa può essere utilizzata per evitare un eccessivo "overfitting" e condizionare l'apprendimento del modello in base ad alcuni criteri oggettivi. Con "criteri oggettivi" si intendono criteri che non si basano sugli stessi dati utilizzati durante la fase di training. Se il sistema non soddisfa i criteri previsti, il sistema può essere modificato. Successivamente viene rieseguita la procedura di validazione, finché non si raggiungono i criteri richiesti (ad esempio il numero massimo di epoche). Sono diverse le procedure di valutazione utilizzabili: Una possibile soluzione è quella di utilizzare un intero dataset creato appositamente per la convalida; questo dataset può essere preparato direttamente dall'utente, o tramite metodi automatici. In altri casi (quando ad esempio il dataset di training ha una dimensione limitata) invece, è possibile utilizzare una tecnica di "validazione incrociata" (cross validation), ovvero, tramite un partizionamento di un campione di dati, in due sotto-insiemi dei quali, uno viene utilizzato per la fase di training, mentre il secondo viene utilizzato per confermare e convalidare il classificatore [23]. Esistono diversi tipi di "validazione incrociata" , ad esempio k-fold, leave-one-out ecc.

Si può concludere, riassumendo, che un comune esperimento di classificazione necessita di:

- Un dataset di training per "addestrare" il modello;
- Un dataset di test, che viene utilizzato per ottenere un giudizio finale sulla qualità del classificatore. I dati di questo dataset devono essere dati "nuovi", cioè non precedentemente utilizzati nell'addestramento;

- Un dataset di validazione che può essere fornito dall'utente, estratto casualmente dalla KB, oppure generato automaticamente tramite una procedura di validazione incrociata.

2.5 Metodologie di apprendimento supervisionato[11]

Uno dei più semplici modelli di apprendimento supervisionato è il *Perceptron*.Esso prevede che una rete neurale sia composta da un singolo neurone di output e N neuroni di input(*x*).La rete, dunque, impara modificando i pesi *w* secondo la funzione seguente, in cui H() è un operatore funzionale (denominata funzione di attivazione del neurone), selezionabile in fase di setup del modello. Nel caso del percettrone, tale operatore può essere scelto tra la funzione di Heaviside, sigmoide o tangente iperbolica (esempi in fig. 2.3).

$$y = H(\sum_{i=1}^{n} w_i x_i) \qquad (2.1)$$

Il più grande limite di questo modello è di essere capace di classificare correttamente gli input solo se le classi di riferimento sono linearmente separabili (figura x.2 caso "a") .

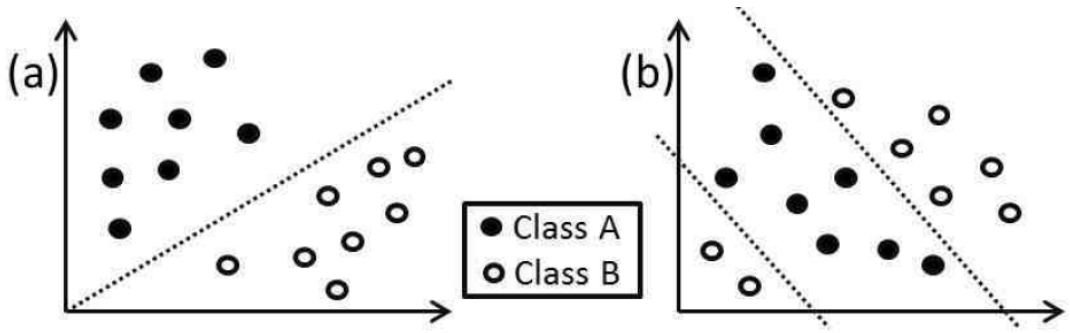


Figura 2.2 *[11] Esempio di classi linearmente separabili (A) e non (B)*

Per superare questo problema si è introdotto un nuovo modello chiamato *Multi-layer Perceptron* (MLP): esso è una rete composta da uno o più livelli nascosti di neuroni, totalmente connessi tra livelli di input e output. .

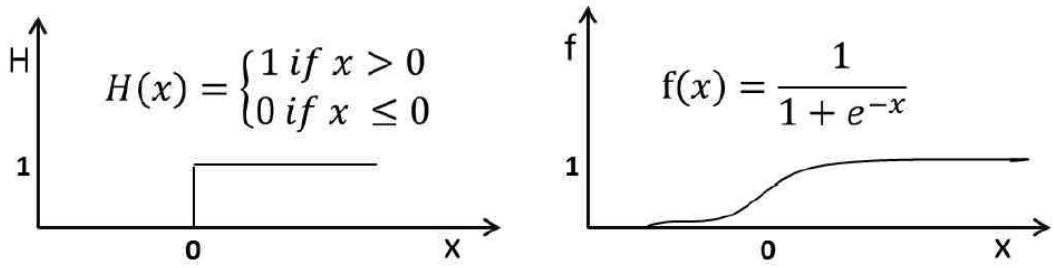


Figura 2.3 *[11]confronto tra funzione di Heavside (H) e sigomoidale (f))*

Inoltre se si definisce come funzione d'errore il valor quadratico medio (MSE) tra l'output atteso e l'output della rete,tale funzione può essere utilizzata per stimare il livello di apprendimento, ossia raggiungendo il minimo della funzione errore. Il processo di apprendimento più comunemente associato al modello MLP viene detto "*back propagation*",poiché l'errore viene retro propagato dall'output fino ai livelli di input, modificando i pesi in funzione del gradiente della funzione errore. Per minimizzare l'MSE , allora, gli algoritmi come l'MLP con back propagation si basano sul metodo *del Gradiente Discendente* (DGA), nelle sue versioni online e batch. Nella versione online i pesi sono aggiornati dopo ogni presentazione di un pattern di input, mentre nella versione batch, i pesi sono aggiornati ad ogni iterazione solo dopo l'intera presentazione del training set. Utilizzando la DGA standard, la direzione di ogni passo di aggiornamento viene calcolato attraverso il gradiente discendente dell'errore, mentre la lunghezza(del passo)è determinata dal tasso di apprendimento. Un approccio più sofisticato consiste nel muoversi verso la direzione negativa del gradiente (direzione line search) non da una lunghezza fissa, ma fino al raggiungimento di un minimo della funzione, lungo quella direzione. Ciò è possibile mediante il calcolo del gradiente discendente e analizzando la variazione del tasso di apprendimento. Il problema di *"linea di ricerca " ,quindi, è* in pratica, un problema di minimizzazione a singola dimensione.

Alle metodologie citate prima va aggiunto il SupportVector Machine (SVM) il quale è un potente strumento capace di classificare due classi di oggetti linearmente separabili. Per ogni classe il SVM identifica l'iperpiano che massimizza il margine di separazione. Con lo scopo di migliorare la ricerca del miglior margine tale algoritmo cerca per iperpiani che avvolgono ogni classe e allora cera il separatore tra loro .

Capitolo 3

Gli algoritmi genetici

3.1 FMLPGA

In questo capitolo si descrive l'architettura e le caratteristiche progettuali del modello ibrido denominato Multi Layer Perceptron con Algoritmo Genetico (MLPGA). Tale modello si basa sulla coniugazione delle proprietà architetturali della rete neurale MLP con quelle algoritmiche di un algoritmo genetico (GA). Quest'ultimo viene impiegato come sistema di addestramento (nel seguito training) del modello MLP, in modo da aggiornare i pesi della rete neurale. Il GA viene quindi a sostituire il classico algoritmo di training noto come Back Propagation (BP), comunemente usato per la rete MLP [16]. Come verrà approfondito nel seguito, l'addestramento con un GA introduce un meccanismo di ottimizzazione basato su proprietà sia statistiche che analitiche, fondando il processo di training sul concetto di evoluzione genetica piuttosto che sulle proprietà del gradiente discendente per minimizzare la funzione errore del paradigma supervisionato.

3.1.1 Modello architetturale MLP

Non esiste una singola e precisa definizione di "**Rete Neurale**", questo perché è lo stesso concetto di Rete Neurale ad essere molteplice; In ogni caso si può ricavare una definizione abbastanza generalizzata: "Una Rete Neurale è un circuito composto da un elevato numero

di oggetti elementari che hanno funzione di neurone. Ogni neurone opera su una singola informazione, e il loro lavoro congiunto (ma asincrono) determina l'output della rete."

Esistono diverse versioni di reti neurali che variano a seconda dell'interconnessione tra i neuroni, e le operazioni che i neuroni stessi applicano alle singole informazioni; In questo testo si tratta in particolare del modello di rete neurale che caratterizza la rete MLPGA: il MultiLayer Perceptron (MLP).

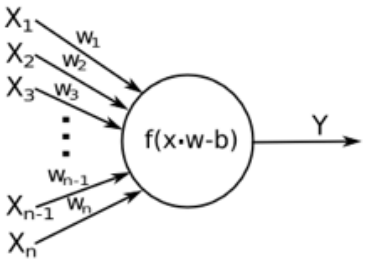


Figura 3.1

Nel 1957, lo studioso Frank Rosenblatt, inventò il ***Percettrone*****:** un modello di rete neurale basato sul paradigma di apprendimento supervisionato del machine learning[26]. Questo modello era composto da uno o più "neuroni": unità elementari in grado di elaborare un numero arbitrario di input, restituendo un valore di output.

Anche rappresentando una grande rivoluzione nel mondo del machine learning, questo modello fu messo subito da parte quando si realizzò che il Percettrone fosse in grado di distinguere e classificare esclusivamente classi completamente disgiunte tra loro nello spazio dei parametri. Per questo motivo il Percettrone fu accantonato per diverso tempo finché, verso la fine degli anni sessanta, dopo diverse ricerche si scoprì che l'utilizzo di due o più livelli di Perceptrons (MultiLayer Perceptron), se opportunamente "addestrati", avrebbe potuto approssimare anche le funzioni prima inclassificabili[27]. Queste nuove scoperte non convinsero però i ricercatori, che ritenevano eccessivamente complessa la fase di training per questa struttura.

Solamente negli ultimi decenni, con l'aumentare della potenza computazionale dei nuovi calcolatori, si è iniziato a rivalutare questo modello, facendolo diventare uno dei più noti e largamente diffusi. Identicamente alle reti neurali biologiche, anche la MLP è costituita da

un numero finito di neuroni organizzati in livelli (chiamati layer); inoltre ogni neurone è "completamente connesso" con le unità del layer precedente.

Le reti MLP grazie alla loro versatilità e duttilità possono essere utilizzate per l'approssimazione di qualsiasi funzione a n variabili, e la loro "universalità" viene confermata dal teorema di Kolmogorov [22] che afferma appunto che qualsiasi funzione continua reale di variabili reali, definita in un sistema n-dimensionale, può essere rappresentata come somma di funzioni continue a variabile singola (successivamente vedremo che la scomposizione in funzioni ad unica variabile è proprio il "modus operandi" del nostro modello).

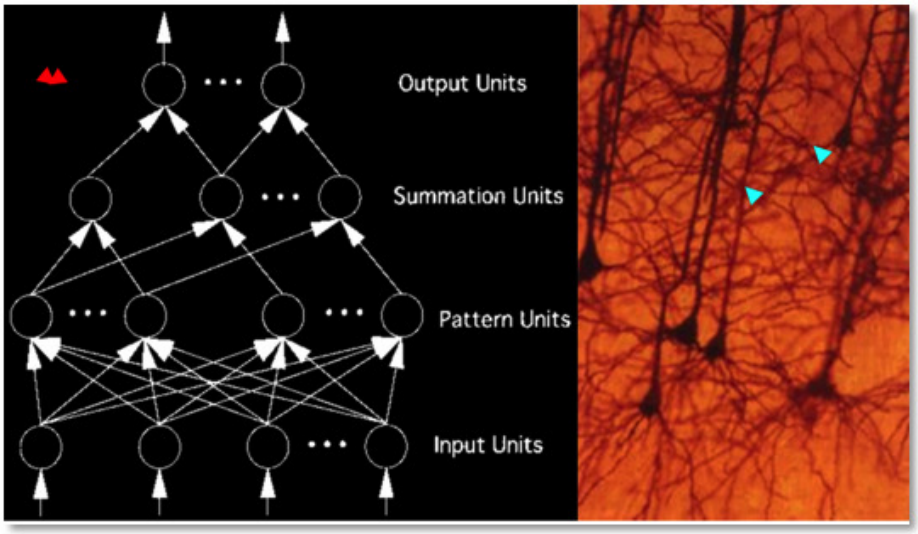


Figura 3.2 Esempio di rete neurale

Così come avviene per la struttura, anche il funzionamento della rete MLP riprende quello della classica rete neurale biologica. Infatti, così come l'impulso elettrico viaggia lungo gli assoni attraverso i neuroni, così nella MLP l'input (immagazzinato dallo strato denominato "input layer") viene propagato lungo tutta la rete, passando per ogni neurone, che lo elabora secondo regole che verranno illustrate a breve. L'output dei neuroni di ogni singolo layer verrà poi propagato ai neuroni del layer successivo, o rappresenterà l'output finale nel caso in cui il layer in questione sia l'ultimo della rete (output layer).La generazione degli output dei neuroni, oltre che dagli output dei neuroni del layer precedente, dipende da altri due fattori altrettanto importanti (se non più importanti): le funzioni di attivazione e i pesi.

3.1.2 La funzione di attivazione

La funzione di attivazione rappresenta la modalità con cui un neurone reagisce agli stimoli input ricevuti e la sua funzione è quindi di propagare agli strati successivi l'informazione elaborata.La giusta scelta e combinazione delle funzioni di attivazione può influenzare in maniera rilevante l'output finale della rete.

Le principali tipologie di funzioni di attivazione (usate opzionalmente nel nostro modello) sono quattro:

- **Funzione lineare**: E' la più semplice funzione di attivazione applicabile ad una rete neurale. Le funzioni lineari, da sole sono inappropriate per la giusta approssimazione della maggior parte delle funzioni, è dunque consigliato il suo uso associato ad altre funzioni non lineari;
- **Funzione a scalino:** E' la funzione di attivazione che più si avvicina ai comportamenti dei neuroni biologici quando questi vengono a contatto con stimoli esterni. Questa tipologia di funzione di attivazione viene scelta per la risoluzione di problemi che prevedono la classificazione tra classi ben definite;
- **Funzione sigmoidea:** E' una delle funzioni più utilizzate nelle reti neurali, ed è caratterizzata da una curva smussata e restituisce valori compresi nell'intervallo [0,1];
- **Funzione tangente iperbolica:** Questa funzione è molto simile alla sigmoidale tranne per il fatto che restituisce valori inclusi nell'intervallo [-1,1].

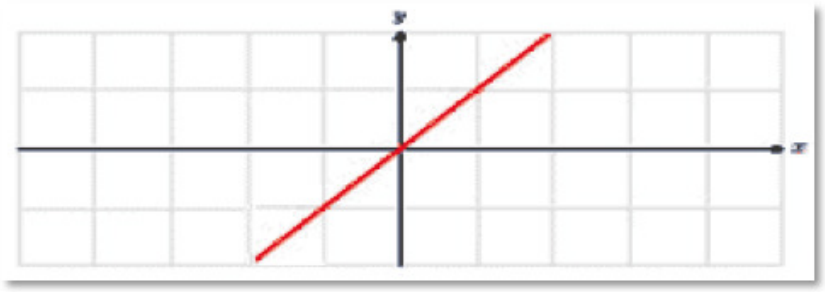


Figura 3.3

Funzione Lineare

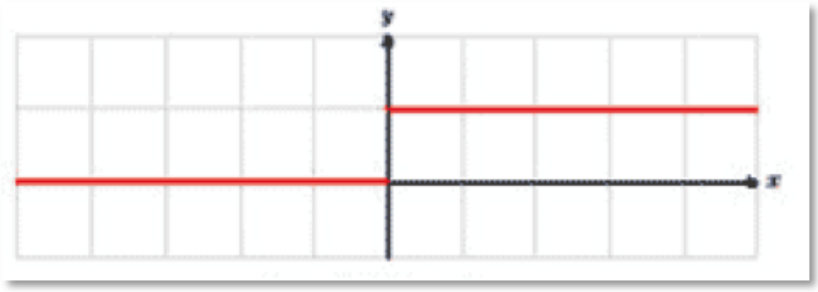


Figura 3.4

Funzione a gradino

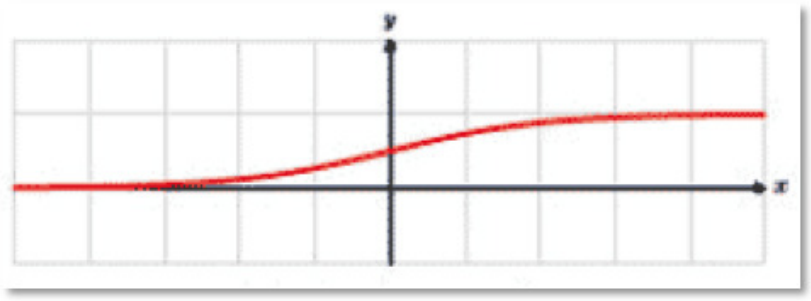


Figura 3.5

Funzione Sigmoide

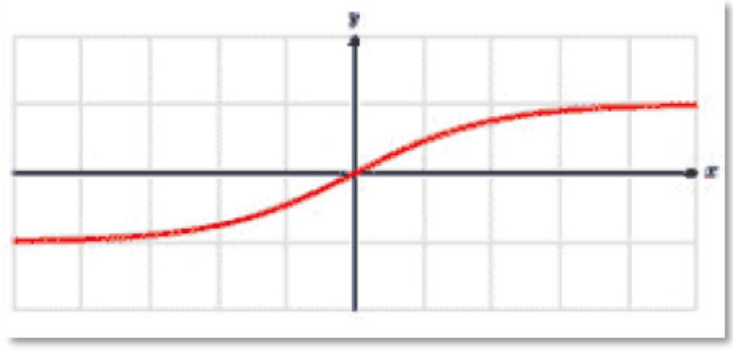


Figura 3.6

Funzione tangente iperbolica

3.1.3 I pesi

I pesi di una rete MLP sono valori numerici normalizzati, solitamente compresi nell'intervallo [-1,1] e costituiscono un punto fondamentale, se non il più importante nel funzionamento di una rete MLP. Questi valori caratterizzano gli archi di collegamento tra neuroni ed hanno il compito di calibrarne l'output. Il ruolo dei pesi è quello di fungere da "filtro di comunicazione" tra i neuroni, e osservando l'immagine sottostante si può intuire come vengono utilizzati questi "filtri": I valori di output dei neuroni del layer 0 vengono propagati al neurone del layer 1 e durante la propagazione ogni valore viene moltiplicato per il peso dell'arco che unisce i due neuroni. Il valore di output del neurone nel layer 1 sarà dunque il valore di output della propria funzione di attivazione il cui input sarà dato dalla sommatoria degli input ricevuti dal layer precedente.

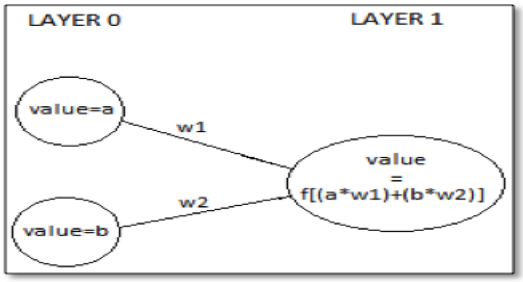


Figura 3.7 esempio di aggiornamento dei pesi in una rete MLP

3.1.4 Apprendimento con algoritmo genetico

L'algoritmo genetico è un metodo di computazione ed ottimizzazione dei dati ispirato al processo evolutivo darwiniano. Grazie alle sue capacità "evolutive", questo algoritmo è molto spesso utilizzato in modelli di data mining, o più in generale per determinare la soluzione di un problema quando il suo codominio non sia noto a priori. Come detto in precedenza, questo algoritmo di ottimizzazione si basa sul modello evolutivo presentato da Charles Darwin nella sua opera "*L'origine delle specie"* nel 1859 [17]. Darwin affermava che gli individui di una popolazione evolvono a causa di due semplici fattori: il caso e la necessità. E' infatti noto che si hanno delle piccole mutazioni naturali casuali dovute a piccoli errori genetici durate la riproduzione, ed inoltre le necessità ambientali (clima,alimentazione, ecc.), effettuano una naturale selezione degli individui con migliori capacità di sopravvivenza. Così come accade con l'evoluzione biologica proposta da Charles

Darwin, anche il modello proposto prevede una popolazione composta da esemplari, chiamati cromosomi, che evolvono durante tutta l'esecuzione dell'algoritmo. Ogni cromosoma è costituito a sua volta da un set di "geni" che costituiscono il DNA del cromosoma stesso.I DNA costituiscono il componente cardine dell'algoritmo genetico; ogni singolo DNA rappresenta infatti un possibile risultato finale, ed è proprio la modifica dei singoli geni all'interno dei DNA a determinare l'evoluzione degli individui della popolazione. Il funzionamento dell'algoritmo genetico può essere sintetizzato in quattro fasi:

- **L'inizializzazione della popolazione:** Il primo passo che viene fatto nell'utilizzo dell'algoritmo genetico consiste nella generazione casuale di una popolazione di cromosomi usando una qualsiasi distribuzione statistica;
- **Calcolo degli output:** Durante questa fase i DNA di ciascun cromosoma vengono impiegati all'interno di un modello di elaborazione dell'informazione (nel nostro caso il MultiLayer Perceptron), ricavando i valori di output;
- **Confronto dei risultati:** In questa fase vengono calcolate le differenze tra i risultati ottenuti tramite i DNA dei cromosomi, e il risultato atteso. Il valore ottenuto è chiamato "*fitness" o bontà della valutazione degli input da parte dei vari individui della popolazione.* Un valore di fitness viene dunque associata ad ogni cromosoma per caratterizzarne il suo grado di prestazioni. E' il parametro più importante in un GA e dipende dal problema in esame. Nel caso della classificazione del traffico di rete, la fitness è l'errore di classificazione. Pertanto il miglior individuo della popolazione sarà quello con fitness più bassa;
- **Evoluzione degli individui:** Durante questa fase vengono applicati (secondo regole che verranno spiegate in seguito), i cosiddetti "*operatori genetici"*, cioè funzioni che hanno lo scopo di modificare i DNA dei cromosomi, in modo che possano migliorare (o peggiorare) la propria fitness all'iterazione genetica successiva. Al termine di questa fase otteniamo quindi una nuova popolazione nata dall'evoluzione della precedente.

La seconda, terza, e quarta fase saranno eseguite ciclicamente sulle varie generazioni della popolazione, ed ogni ciclo è chiamato "*epoca".* L'algoritmo genetico potrebbe essere eseguito all'infinito, ma di norma si predispone che questo termini quando uno dei seguenti eventi si verifichi:

- Si trova un cromosoma la cui fitness sia minore o uguale ad un valore predeterminato (soglia di ottimizzazione);
- Si è raggiunto il numero massimo pre-impostato di epoche.

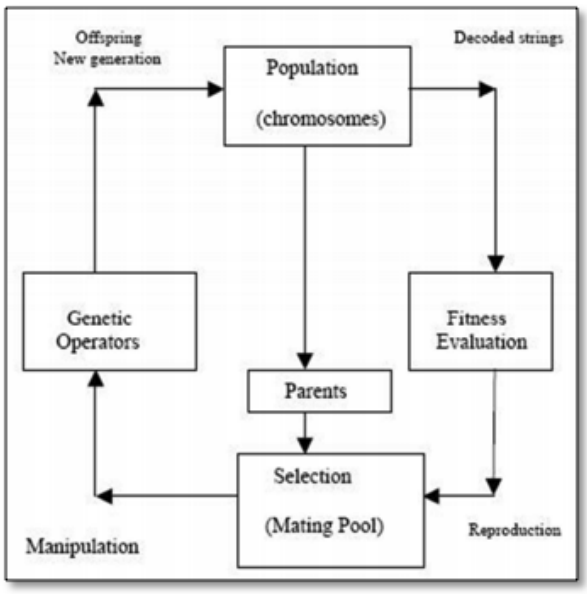


Figura 3.8 Ciclo evolutivo di un algoritmo genetico

Come detto in precedenza le operazioni di modifica dei DNA avviene tramite particolari funzioni chiamate "operatori genetici".L'algoritmo genetico prevede due tipi di operatori:

- Il crossover;
- La mutazione.

La funzione di ***crossover*** riprende il concetto di "crossover biologico" che prevede la generazione di un nuovo individuo (o più individui), partendo da un mix dei geni provenienti dal DNA di due individui diversi (chiamati "*genitori*").Nell'algoritmo genetico il crossover può essere implementato in diversi modi, tra i principali ci sono:

- **Single point crossover**: risulta essere il metodo di crossing-over più semplice, e consiste nel scegliere due individui, tagliare il loro DNA in un punto selezionato casualmente, chiamato "*crossingpoint*" (il punto deve essere lo stesso per entrambi

i genitori), e creare un nuovo individuo che abbia la "testa" del primo individuo, e la "coda" del secondo. Nell'immagine sottostante viene mostrato un esempio in cui il "crossingpoint" è uguale ad 1.

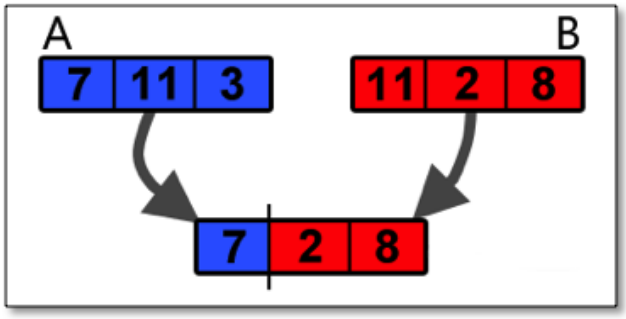


Figura 3.9Esempio di single point crossover

- **Double point crossover:** Questo metodo è una piccola variante della versione "*Single point crossover"* mostrata in precedenza; l'unica differenza consiste nel numero di *crossingpoint* che passa da uno a due. Nell'immagine sottostante è stata utilizzata la tecnica del Double point crossover in cui i due "crossingpoint" sono impostati rispettivamente a 1e2.

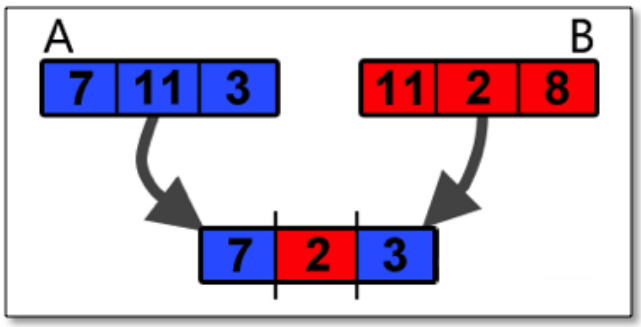


Figura 3.10 Esempio di double point crossover

- **Mixing Crossover:** E' un'evoluzione delle precedenti tecniche, e nasce dalla consapevolezza statistica che è molto improbabile che, all'interno di un DNA, due geni "utili" si trovino in posizioni contigue. La tecnica proposta, infatti, invece di selezionare segmenti di geni contigui, prevede che il DNA risultante sia la copia esatta di uno dei due genitori al quale siano stati modificati n geni (il numero viene scelto casualmente) scelti a caso. Nell'esempio riportato in basso viene illustrato il meccanismo della tecnica appena spiegata.

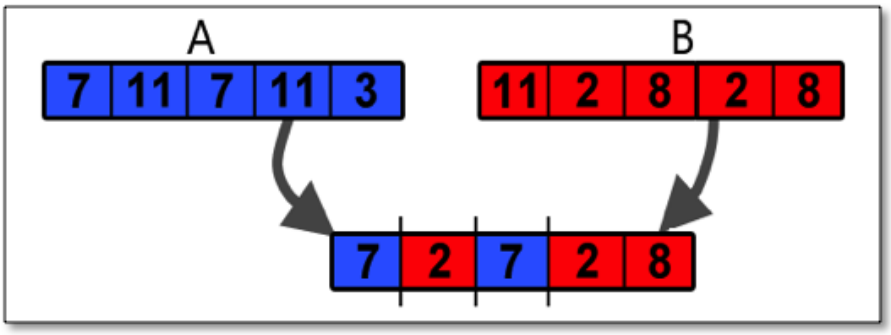


Figura 3.11 Esempio di mixing crossover

A differenza del *crossover* che prevede l'utilizzo di due individui genitori, la ***mutazione,*** invece, esegue una piccola modifica di uno (o più) geni all'interno del DNA sottoposto. Anche se negli anni sono state implementate tantissime versioni di questo operatore, in questo testo verrà illustrata esclusivamente la più semplice e più usata versione di mutazione: la "*single gene unbiased mutation*" (conosciuta anche come **mutazione standard**).La *mutazione standard* prevede la selezione casuale di un gene all'interno del DNA e la sua sostituzione con uno generato tramite un generatore di valori casuali.

Figura 3.12 Esempio di mutazione

Le regole che gestiscono le modalità di applicazione del crossover e della mutazione sono molteplici, ma in questo testo descriveremo esclusivamente le principali, che sono:

- Fitting;
- Roulette v_1;
- Roulette v_2;
- Ranking.

Prima però di introdurre queste regole, bisogna prima chiarire un concetto comune a queste tre regole: **l'elitismo.** L'*elitismo* si basa sul concetto di "elite" ed è una strategia evolutiva che va a completare il processo di sostituzione della popolazione. Quando creiamo una nuova popolazione con crossover e mutazione abbiamo una grande probabilità di perdere il miglior cromosoma (o i migliori). L'elitismo semplice è un metodo

che lascia inalterati il miglior cromosoma o i pochi migliori nella nuova popolazione e poi prosegue con la logica espressa in precedenza. L'elitismo può far crescere rapidamente le performance dell'algoritmo genetico, perché evita la perdita della migliore soluzione trovata. Tale tecnica prevede ovviamente che la popolazione ad ogni epoca sia stata ordinata in ordine decrescente di fitness, permettendo l'immediata estrazione degli individui di elite (i primi della lista ordinata).

Il **fitting** è una delle più semplici regole ed al contempo riesce a generare risultati in taluni casi soddisfacenti .Il suo modus operandi è il seguente:

- Considera "*elitico"* il 10% della popolazione con il fitness più alto;
- Il resto della popolazione viene incrociato (crossover) con uno degli individui elitici preso a caso, dopodiché, al risultato di questo incrocio viene applicato l'operatore di *mutazione*.

Roulette è un metodo di selezione proporzionale al valore di fitness che prevede che gli accoppiamenti tra gli individui della popolazione avvengano tramite l'utilizzo di una roulette (virtuale) che restituisce un altro individuo della popolazione secondo due fattori precisi:

- il peso della fitness dell'individuo all'interno dell'intera popolazione;
- Il caso;

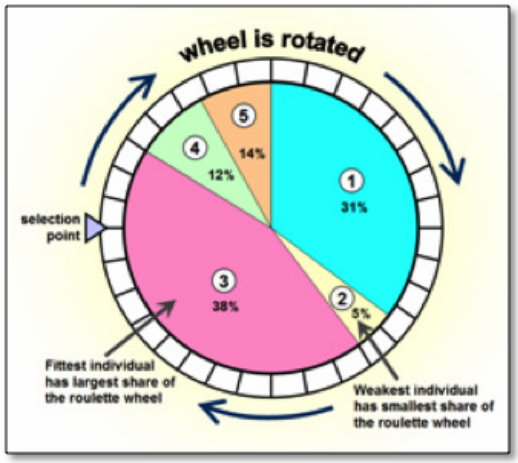


Figura 3.13 Esempio Roulette

Ad ogni elemento della popolazione viene assegnato un determinato numero di caselle della ruota, che è dovuto al rapporto tra la sua fitness e quella totale (somma delle fitness di tutti gli individui).

La ruota assegnerà quindi un maggior numero di caselle all'elemento della popolazione con il valore di fitness migliore, ed il minor numero di caselle (che può essere anche zero!) all'elemento con la peggior fitness. In breve questa regola effettua i seguenti passi:

- Considera "elitici" i primi x elementi con il valore di fitness migliore (il valore 'x' viene impostato in fase di 'starting' dell'algoritmo);
- Il resto della popolazione viene incrociato (crossover) con l'individuo risultate dall'utilizzo della ruota della roulette, dopodiché, al risultato di questo incrocio, viene applicato l'operatore di *mutazione*.

Con il termine Roulette v_2 si intende proprio il meccanismo sopra citato, mentre con Roulette v_1 si intende una variante del metodo standard che considera solo gli elementi la cui distanza da un certo punto è inferiore a una certa soglia pre-impostata.

$$\left|\frac{f1}{\varepsilon} - \frac{f2}{\varepsilon}\right| < \varepsilon \qquad (3.1)$$

Infine il metodo **ranking (o torneo)** prevede semplicemente accoppiamenti tra un certo numero (deciso dall'utente), di vari individui della popolazione, selezionati casualmente, dai quali viene selezionato l'elemento con la fitness migliore.

3.1.5 Modello ibrido MLPGA

Tenendo presente le considerazioni fatte per il MultiLayer Perceptron e per l'Algoritmo genetico, si mostra, ora, come questi vengano combinati insieme, ottenendo così il modello **MLPGA.** In pratica i DNA dei cromosomi della popolazione rappresentano i possibili pesi della rete neurale, e l'evoluzione di questi cromosomi ha lo scopo di trovare il set di pesi (il DNA) che, innestato all'interno della rete, restituisca un output con il minor errore possibile.

L'esecuzione di questo algoritmo comporta i seguenti passi:

1. Generazione di una popolazione random di cromosomi. Ogni cromosoma rappresenta un intero set di pesi di una rete MLP;
2. Viene eseguita la "Forward propagation" su tutti i pattern del dataset di input generando un risultato di output. Questo passaggio viene ripetuto per ogni set di pesi (ogni singolo cromosoma della popolazione);
3. Calcolo dell'errore di training;
4. Generazione di una nuova popolazione di cromosomi applicando gli operatori genetici e i metodi precedentemente mostrati;
5. Ripetizione dei passi da 2 a 4 finché non si verifichi un evento di terminazione (si è raggiunto il numero massimo di *epoche* oppure si è trovato un cromosoma con un fitness minore di un valore soglia inizialmente settato).

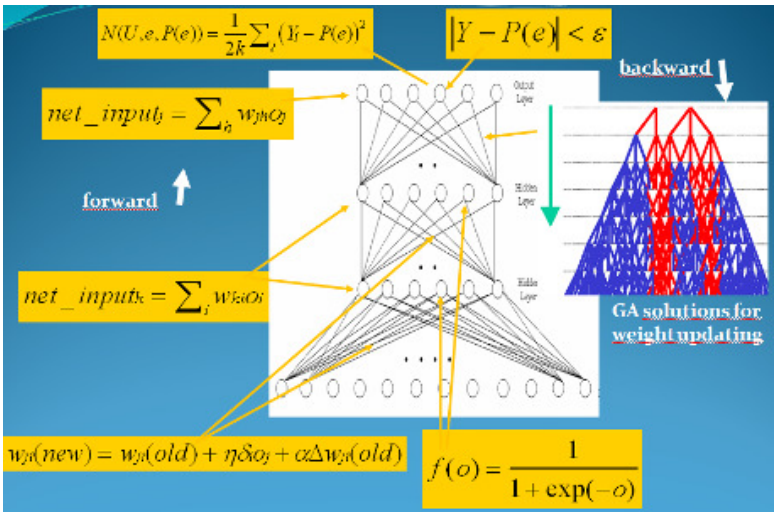


Figura 3.14 Esempio di rete MLPGA

3.1.6 Casi d'uso

L'MLPGA prevede complessivamente tre casi d'uso:

- Train;
- Test;
- Run.

Il caso d'uso "***Train***" prevede l'addestramento del modello tramite l'utilizzo di un dataset di input e relativi target. Al termine di questa fase, viene salvato all'interno di un file, i miglior risultato ricavato, ovvero la matrice dei pesi tramite la quale la MLP restituisce l'output con il miglior valore di fitness.

Nel caso d'uso "***Test***" l'utente ha la possibilità di verificare la qualità dei pesi ricavati durante la fase di Train. In questa fase è possibile utilizzare o lo stesso input dataset utilizzato durante l'addestramento, oppure usare un input dataset composto da pattern mai utilizzati (scelta consigliata); Gli output ricavati dalla rete, infine, verranno confrontati infine con i target del dataset.

Dopo aver addestrato e testato la nuova rete MLP, l'utente potrà eseguire il caso d'uso "***Run***". A differenza dei casi precedenti, in questa fase non è previsto l'uso di target, perché il suo unico scopo è semplicemente quello di eseguire il modello come se questo fosse una funzione analitica standard, su dati nuovi e sconosciuti.

3.1.7 Gestione dei file di configurazione

Gli input e gli output del software implementato sono file di testo che possono essere suddivisi in:

- File di configurazione;
- File di input;
- File di output.

I file di configurazione contengono le direttive generali per quanto riguarda la scelta del caso d'uso, la morfologia della rete MLP e la costituzione della popolazione. Questa tipologia di file si suddivide a loro volta in tre categorie:

- ***Network configuration file***, contenente le informazioni per la configurazione della rete MLP;
- ***Experiment configuration file,*** contenente informazioni dettagliate riguardanti il caso d'uso da eseguire;

- ***General configuration file***, contenente il caso d'uso da eseguire (TRAIN, TEST o RUN), e i nomi del "*Experiment configuration file*" e del "*Network configuration file*".

I file di input contengono i dataset di input e di target utili all'esecuzione del modello.***I file di output*** contengono i risultati ottenuti durate il caso d'uso eseguito.

Esistono sostanzialmente cinque file di output:

- ***Weights file***: contenente la matrice dei pesi ricavati durante il "training";
- ***Error file***: contenente l'andamento dell'errore durante tutta la fase di "Train";
- ***Log file***: contenente le informazioni generali riguardanti l'esperimento appena concluso;
- ***Output file:*** contiene l'output della rete MLP ottenuto tramite l'utilizzo dei pesi contenuti nel "Weights file";
- ***Statistical file:*** sono file contenenti dati statistici relativi alla "Confusion Matrix" per la classificazione.

3.1.7.1 General configuration file

Questo file è un documento testuale composto da 4 righe :

- Riga1 : contiene la directory in cui il programma può trovare i file di configurazione della rete neurale e il file relativo alla configurazione dell'esperimento;
- Riga2 : contiene il nome del Network configuration file;
- Riga3 : contiene il caso d'uso TRAIN/TEST/RUN;
- Riga4 : contiene il nome dell' Experiment configuration file.

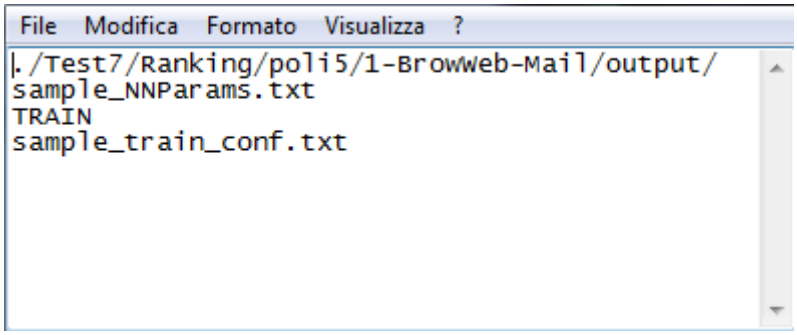


Figura 3.15 Esempio di General configuration file

La figura 3.14 mostra un esempio di General configuration file . Questo è il file che si da come input al programma.

3.1.7.2 Network configuration file

Questo file testuale contiene i parametri della rete neurale ed è composto da 8 righe così composte :

- Riga1 : numero di caratteristiche;
- Riga2 : numero di livelli nascosti intermedi;
- Riga3 : numero di nodi del primo livello nascosto;
- Riga4 : numero di nodi del secondo livello nascosto;
- Riga5 : numero di nodi output (classi targets);
- Riga6 : tipo di funzione di attivazione dei neuroni del primo livello hidden a scelta tra quelle in 3.1.2;
- Riga7 : tipo di funzione di attivazione dei neuroni del secondo livello hidden a scelta tra quelle in 3.1.2;
- Riga8 : tipo di funzione di attivazione dei neuroni del livello output a scelta tra quelle in 3.1.2;

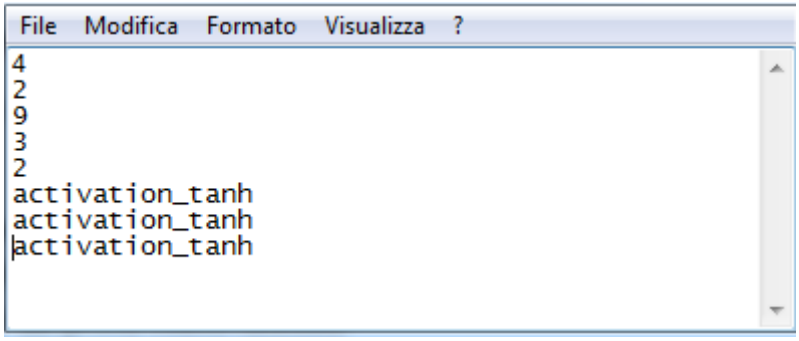


Figura 3.16 Esempio di Network configuration file

La figura 3.15 mostra un esempio di Network configuration file. Questo file risulta identico nei tre casi d'uso.

3.1.7.3 Experiment configuration file

Questo file nel caso d'uso TRAIN è costituito da ben 17 righe testuali così costituite :

- Riga1 : tipo di esperimento CLASSIFICATION/REGRESSION;
- Riga2 : nome del dataset di input;
- Riga3 : tipo di generazione casuale GRANDOM/DRANDOM/NRANDOM;
- Riga4 : tipo di funzione di selezione;RANKING/FITTING/ROULETTEv_1/ROULETTEv_2;
- Riga5 : soglia d'errore;
- Riga6 : numero di iterazioni;
- Riga7 : frequenza dell'errore parziale di output;
- Riga8 : probabilità di crossover;
- Riga9 : probabilità di mutazione;
- Riga10 : numero cromosomi per popolazione;
- Riga11 : elitismo;
- Riga12 : numero cromosomi per il RANKING;
- Riga13 : nome del file di output relativo ai pesi;
- Riga14 : nome del file di output relativo agli errori;
- Riga15 : nome del file di output relativo gli output;
- Riga16 : nome del file di output relativo ai log;

- Riga17 : nome del file di output relativo alle informazioni statistiche.

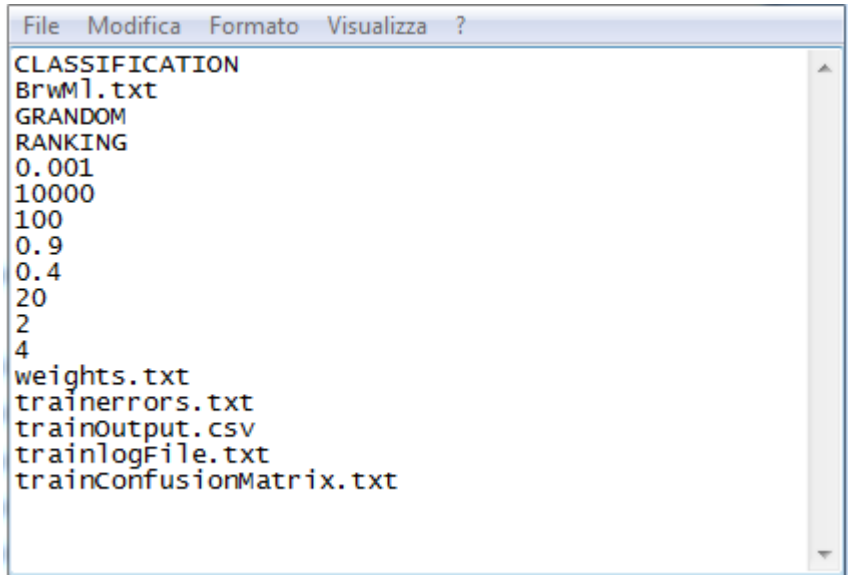


Figura 3.17 Esempio di exsperiment configuration file

La figura 3.16 mostra un esempio di Experiment configuration file nel caso d'uso TRAIN. Nei casi d'uso TEST e RUN , questo file è composto solo da 7 righe testuali relative al tipo di esperimento, nome del dataset e ai nomi dei file di output.

3.1.8 Modello FMLPGA

Il modello implementato risulta essere in media discretamente efficiente, anche se in presenza di dati particolarmente "rumorosi" o complessi, le sue prestazioni possono risultare degradate rispetto a modelli MLP con differenti regole di apprendimento [11].In realtà la vera limitazione di questo modello è, come detto in precedenza, l'eccessivo tempo impiegato per effettuare la fase di training. E' stato proprio questo il problema principale che ha spinto a cercare un'alternativa alla classica implementazione seriale in modo da ridurre i tempi computazionali della fase di training. Per risolvere i problemi di performance illustrati nel capitolo precedente, questa implementazione utilizza la tecnologia Nvidia GPU in ambiente CUDA per parallelizzare il lavoro del modello MLPGA, riducendo così i tempi di esecuzione dello stesso.Dalla parallelizzazione di MLPGA nasce il modello chiamato FMLPGA (Fast MLPGA). La fase più lenta del caso training è il calcolo dei valori di fitness dei singoli individui della popolazione; il calcolo di questi valori è infatti affidato ad un doppio

loop (sulla dimensione della popolazione e sul numero di pattern del dataset), all'interno del quale viene eseguita la funzione forward della rete.

Questo approccio porta all'esecuzione di un elevatissimo numero di istanze seriali della funzione forward (anche oltre 100000), che penalizza molto le performance del modello.Ciò ha portato alla riprogettazione dell'algoritmo di training di MLPGA in modo tale da sostituire il doppio loop precedentemente illustrato con un'unica funzione che effettui il forwarding in parallelo su N reti, e dai loro risultati ricavarci i valori di fitness.

La gestione dello spazio di memoria all'interno dell'ambiente CUDA è una fase delicata della programmazione e per evitare inutili allocazioni/deallocazioni di variabili e strutture dati nel continuo scambio d'informazioni tra CPU host e GPU device, che influenzerebbero negativamente le prestazioni del modello, il sistema alloca tutte le strutture dati e le variabili utilizzate nell'ambiente CUDA nella fase di startup del modello.

Andando nel particolare, nella versione GPU vengono allocate le seguenti risorse:

- Un array contenente tutti i pattern del dataset "input";
- Un array contenente tutti i pattern del dataset "target";
- Un array contenente tutti i DNAdella popolazione;
- N reti Multi Layer Perceptron;
- Un array di dimensione N il cui scopo è quello di immagazzinare gli errori parziali delle singole reti, dove "N" è un valore intero equivalente al prodotto tra il numero di individui della popolazione, e il numero di pattern del dataset.

Una volta terminata l'allocazione delle variabili e delle strutture dati sulla GPU, viene configurata la "CUDA Grid".Con il termine "CUDA Grid" si intende la griglia virtuale composta dai thread che eseguono il lavoro parallelo.La "CUDA Grid", in questo modello, è costituita da un numero di blocchi pari al numero di pattern del dataset, ed ogni blocco sarà a sua volta composto da un numero di thread pari al numero di individui della popolazione. Ogni thread esegue la funzione di forward su una specifica coppia pattern/DNA attraverso una delle N reti allocate in fase di startup. Nell'immagine sottostante si mostra la morfologia della CUDA Grid per dataset di dimensione 4 e una popolazione di dimensione 8.

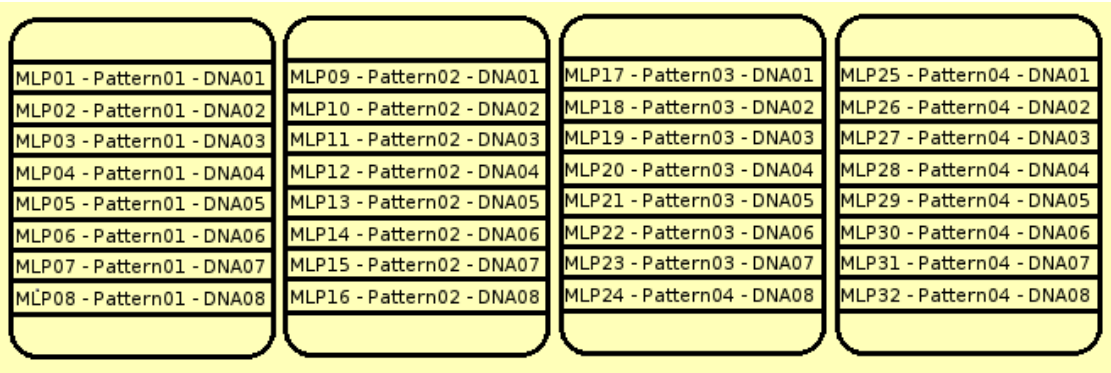


Figura 3.18 Esempio CUDA grid

Una volta allocate le variabili sulla GPU e settate le dimensioni della CUDA Grid, il modello è pronto per affrontare la fase di Training. Questa tipologia di implementazione del modello MLPGA (chiamata FMLPGA), offre prestazioni fino a 10 volte superiori alla classica implementazione seriale, ma di questo argomento ci occuperemo nel capitolo dedicato ai test.

3.2 GAME

Come si è detto precedentemente con il termine genetico si intende un algoritmo basato su metodi ispirati alla naturale evoluzione in accordo con la teoria di Darwin. Questi algoritmi sono strumenti potenti,utilizzati per risolvere i problemi in cui lo spazio dei parametri non sia ben definito e per trovarne la soluzione migliore[2].Per lo scopo di questa Tesi è stato utilizzato oltre al modello FMLPGA anche l'algoritmo GAME. Esso è un algoritmo genetico puro, appositamente progettato per risolvere i problemi di ottimizzazione, in relazione a funzionalità di regressione o di classificazione, su Massive Data Sets (MDS).

Dato un insieme di dati generico con N caratteristiche , t target e detto pat uno schema generico di ingresso del set di dati,con $pat = (f_1, \cdots, f_N, t)$e *g(x)* funzione generica reale, la rappresentazione di una generica caratteristica f_i, di un generico pattern espressa con una sequenza polinomio di grado *d* è:

$$G(f_i) \cong a_0 + a_1 g(f_i) + \cdots + a_d g^d(f_i) \quad (3.1)$$

Quindi, il k-esimo pattern (pat_k) con N caratteristiche può essere rappresentato da:

$$Out(pat_k) \cong \sum_{i=1}^{N} G(f_i) \cong a_0 + \sum_{i=1}^{N} \sum_{j=1}^{d} a_j g^j(f_i) \quad (3.2)$$

Mentre il target k-esimo *tk*, relativo al pattern k-esimo, che può essere utilizzato per valutare l'errore di approssimazione del modello di ingresso rispetto al valore atteso, può essere espresso come :

$$E_k = (t_k - Out(pat_k))^2 \quad (3.3)$$

Al fine di valutare l'idoneità dei modelli , come estensione della (3.3) si utilizza l'errore quadratico medio (MSE) o la radice quadrata dell'errore quadratico medio (RMSE) che posso essere espressi come :

$$MSE = \frac{\sum_{k=1}^{NP}(t_k - Out(pat_k))^2}{NP} \quad (3.4)$$

$$RMSE = \sqrt{\frac{\sum_{k=1}^{NP}(t_k - Out(pat_k))^2}{NP}} \quad (3.5)$$

A valle di ciò si definiscono[2]:

1) La funzione di fitness come l'espressione (3.2) . Essa è in grado di valutare la bontà di un cromosoma rispetto ad un altro;
2) La matrice (a0, ..., aM) definisce M geni del cromosoma generico (normalizzati tra -1 e +1);
3) L'espressione (3.3) dà l'errore standard per valutare il livello di forma dei cromosomi;
4) La popolazione (genoma) è composto da un numero di cromosomi imposti dalla scelta della funzione *g (x)* della sequenza polinomiale;

5) Il numero di cromosomi è dato dall'espressione :

$$NUM_{CHROMOSOMES} = (d \cdot N) + 1 \qquad (3.6)$$

$$NUM_{GENES} = (d \cdot B) + 1 \qquad (3.7)$$

Dove N è il numero delle caratteristiche del modello e B è un fattore moltiplicativo che dipende dalla scomposizione in fattori del polinomio (ad esempio seni e coseni di uno sviluppo polinomiale trigonometrico).

3.2.1 Implementazione di un algoritmo genetico in un'architettura Multi-Core

In un algoritmo genetico una possibile soluzione è data da un vettore di geni che rappresenta il DNA di ogni singolo cromosoma . Gli aspetti cruciali di attuazione, allora, sono[2]:

- La classe Chromosome deve poter maneggiare il suo proprio stack array <double>DNA con tutti i geni e il relativo stack array vector<double> output Data;
- la classe Popolazione deve poter gestire lo stack array vector<Chromosome*> popv, che è una matrice con righe corrispondenti al numero di cromosomi e colonne corrispondenti al vettore DNA di ciascun cromosoma;
- la classe GasControl si occupa della popolazione oggetto *P, e delle matrici dello stack relative all'ingresso e ai modelli di riferimento, rispettivamente, vettore <vettore <double>> InputData e vector<vector<double>> targetData;

Con i costrutti GasControl :: GasControl e GasControl :: next , otteniamo rispettivamente l'inizializzazione della popolazione e l'attuazione dell'evoluzione della popolazione durante l'allenamento.

La popolazione di cromosomi, originariamente creata da una generazione casuale, è sostituita ad ogni ciclo da una nuova, ottenuta applicando operatori genetici, cercando di evolvere verso la migliore popolazione (soluzione).Esistono tre tipi di generazione casuale:

• RANDOM: genera numeri pseudo-casuali i cui valori sono compresi in [-1, +1];
• GRANDOM: che genera valori casuali a seguito della distribuzione normale in [-1, +1];
•DRANDOM: che genera pseudo-casuali valori in [0, 1];

Successivamente i cromosomi sono valutati in termini qualificativi per risolvere un problema specifico, la cui soluzione iniziale sono i cromosomi della prima popolazione casuale. La valutazione viene effettuata dalla funzione di fitness. Essa assegna un punteggio ad ogni cromosoma. Il passo successivo, affidato sempre alla funzione di fitness, è l'evoluzione (riproduzione) della popolazione. La riproduzione viene fatta utilizzando tipici operatori genetici, come *crossover* e *mutazione casuale*. In pratica, la riproduzione avviene selezionando i cromosomi più forti ed "eliminando" gli altri.

Finora, la funzione di fitness consiste nell'errore, ottenuto come differenza assoluta tra l'uscita polinomiale ed il valore obiettivo per ciascun modello. A causa del fatto che in questa trattazione siamo interessati a trovare il valore minimo dell'errore, la funzione di fitness è calcolata come l'opposto dell'errore (ossia 1-errore) e il problema si riduce a trovare il cromosoma che porti al raggiungimento del valore massimo della funzione fitness. A ogni ciclo di evoluzione ci sono tre opzioni di configurazione, che possono essere scelte dall'utente :

1) tipo di funzione di errore : MSE , TMSE (Soglia MSE) o RMSE (Root MSE);
2) tipo di selezione: la funzione di selezione da utilizzare per estrarre alcuni cromosomi che saranno , poi , usati per realizzare l' evoluzione della popolazione. E'possibile scegliere tra RANKING, ROULETTE e FITTING;
3) Un altro meccanismo di evoluzione della popolazione è l'*Elitism.*

Esisto , inoltre , due operatori genetici. Essi sono utilizzati per ingranare i geni tra i cromosomi selezionati, ossia crossover e mutazione.

L'oggetto di classe Chromosome è identificato da un vettore con valori compresi in [-1, +1], mentre il costruttore Chromosome::Cromosoma crea un cromosoma assegnandogli valori casuali[2].
L'esecuzione dell'algoritmo , quindi , inizia con il prendere come input il numero di cromosomi della popolazione e la dimensione dei cromosomi, con le seguenti formule :

- Population ::Crossover implementa l'operatore genetico crossover;
- Population :: Mutation implementa l'operatore genetico mutation;
- Population :: getChromosomeFromRankTournament() implementa il criterio di selezione RANKING, fornendo il cromosoma vincitore (candidato con la migliore forma fisica, cioè con il minor valore d'errore);
- Population :: getChromosomeFromRouletteTournament() implementa il già citato criterio di selezione ROULETTE, fornendo il cromosoma vincitore (candidato con la migliore probabilità di forma fisica);
- Population ::Best() e Population:Worst() sono utilizzati per estrarre, rispettivamente il peggior ed il miglior candidato all'interno della popolazione attuale, utile per l'ordinamento del vettore cromosoma.
- Population ::Next() genera una nuova popolazione.

3.2.2 Casi d'uso

I casi d'uso previsti per l'esecuzione sono relativi ad un tipico modello di modalità di apprendimento automatico:

•TRAIN: consiste nella presentazione del set di dati di addestramento al fine di costruire e conservare la migliore popolazione dell'algoritmo genetico;

•TEST: utilizzato per verificare e convalidare le prestazioni di apprendimento;

•RUN: è la modalità di esecuzione;

•FULL:un caso di flusso di lavoro.

3.2.3 Gestione dei file di configurazione

GAME permette la realizzazione di due esperimenti denominati REGRESSION e CLASSIFICATION. La scelta dell'esperimento viene effettuata nei file di input da dare al programma .

Ogni esperimento è articolato in tre fasi : TRAIN , TEST , RUN .Oltre a queste tre modalità di esecuzione separate, GAME ne prevede ancora un'altra, denominata FULL, che fondamentalmente racchiude le tre prima citate.

3.2.3.1 Training

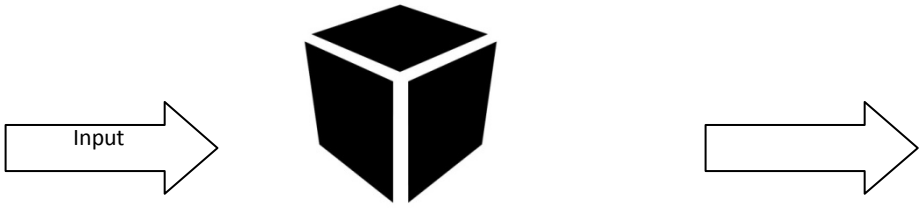


Figura 3.19

La modalità training prende come input un set di dati di cui si conosce il relativo risultato; con tali dati addestra il sistema,tramite una procedura di apprendimento supervisionato. Dopo l'addestramento, il sistema sarà capace di prendere delle decisioni da solo riguardanti gli input forniti, dando il corretto risultato in riferimento al tipo di esperimento scelto .

Per far questo , la fase di training necessita di tre file di input di tipo testuale :

1) Un file contenente il data set di addestramento;
2) Un file di configurazione;
3) Un file contenente i parametri della modalità di esecuzione scelta.

Il primo è il file contenente l'insieme dei dati per l'addestramento del sistema. Si può pensare ai dati da processare come ad una matrice, in cui ogni riga corrisponde a un nuovo pattern mentre, detto N il numero di colonne, detto K il numero di caratteristiche di ciascun pattern, avremo : k colonne per definire le caratteristiche, seguite da (N-k) colonne riferite ai risultati (si ricorda che nella fase di training i risultati di ogni pattern sono noti).Schematicamente possiamo pensare a una situazione del genere :

	Caratteristica	Caratteristica	Target
Pattern			
Pattern			
Pattern			

Tabella 3.1 Generico schema di un dataset

Si noti che nella scrittura del file contenente i dati di training, vanno inseriti in forma tabellare solo i dati, senza specificare né il nome/numero del pattern, né il numero di caratteristiche, né il numero di target.
Il secondo file di input è il file in cui è possibile specificare il tipo di esperimento da realizzare (CLASSIFICATION/REGRESSION), il caso d'uso (TRAINING , TEST , RUN), e il file contenente i parametri relativi al caso d'uso . Esso sarà quindi costituito da tre righe :

- Riga 1 : tipo di esperimento da realizzare REGRESSION/CLASSIFICATION;
- Riga 2 : caso d'uso TRAINING/TEST/RUN;
- Riga 3 : percorso della directory relativa al file contenente i parametri del caso d'uso .

Un esempio di questo file testuale potrebbe essere :

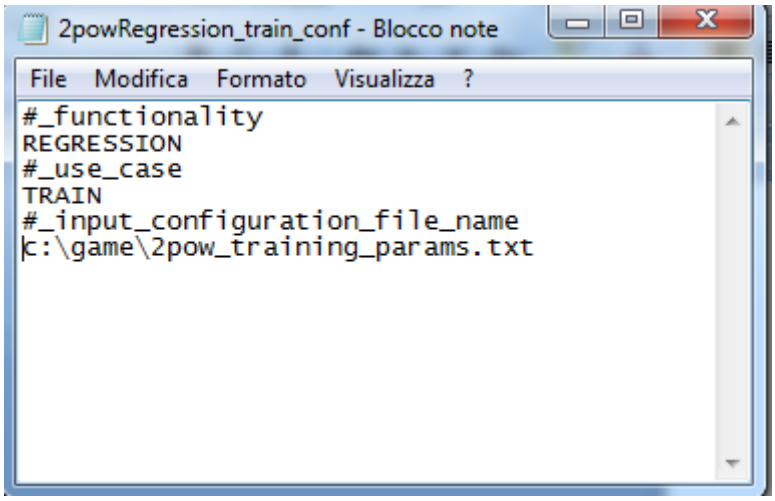


Figura 3.20sempio file di configurazione

Le stringhe precedute dal simbolo # possono essere anche omesse, essendo la loro funzione solo descrittiva del parametro che le segue alla riga successiva.

Nel terzo file di input, invece, si specificano tutti i parametri relativi alla modalità di utilizzo, in questo caso quelli relativi al Training. Esso è un file testuale costituito da 20 righe così definite:

- Riga 1 : percorso del file contenente i dati di training;
- Riga 2 :Specificazione del modo di generazione casuale della popolazione iniziale tra GRANDOM /DRANDOM/NORANDOM;
- Riga 3 : nome della popolazione di training;
- Riga 4 : numero delle caratteristiche di input;
- Riga 5 : numero di targets;
- Riga 6 : grado del polinomio;
- Riga 7 : tipo della funzione polinomio;
- Riga 8 : tipo della funzione errore (funzione di fitness) TMSE/RMSE/MSE;
- Riga 9 : soglia d'errore opzionale;
- Riga 10 : tipo della funzione di selezione;
- Riga 11 : soglia d'errore;
- Riga 12 : massimo numero di iterazioni;
- Riga 13 : frequenza dell'errore parziale di output;
- Riga 14 : crossover rate;
- Riga 15 : mutation rate;
- Riga 16 : numero di tournament chromosomes;
- Riga 17 : elitism rate;
- Riga 18 : percorso e nome del file in cui salvare la popolazione di output;
- Riga 19 : percorso e nome del file in cui salvare la lista degli errori;
- Riga 20 :percorso e nome del file in cui salvare l'output.

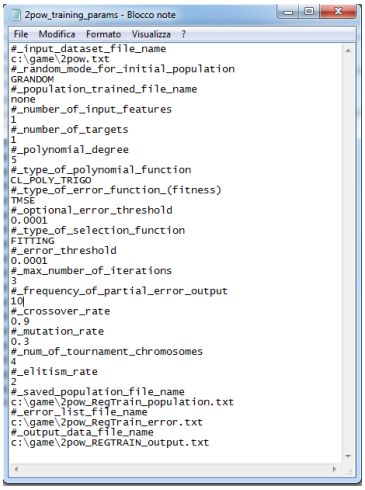


Figura 3.21 Esempio file relativo ai parametri

In uscita alla fase di training saranno restituiti otto file di tipo testuale :

- ➔ tre dei quali sono i file prima creati (e lasciati vuoti) di cui abbiamo passato il percorso della loro locazione al programma nelle righe 18 , 19 e 20 del file di input riguardante i parametri del caso d'uso. Dopo l'esecuzione questi tre file conterranno rispettivamente :
  - La popolazione di output;
  - La lista degli errori eventualmente riscontrati durante l'esecuzione;
  - L'output dell'esecuzione.
- ➔ Due file di log (generati automaticamente dal programma);
- ➔ Un file contenente i targets interni (generato automaticamente dal programma);
- ➔ Un file denominato <tipo esperimento>_MLPGA_train_confmat.txt contenente i risultati in termini statistici dell'esperimento (generato automaticamente dal programma);
- ➔ Un file denominato <tipo_esperimento>_trained_GAME_internal_params.txt contente parametri interni.Questo file sarà poi utilizzato nella fase si Test (generati automaticamente dal programma).

3.2.3.2 Test

I dati di input da dare nella fase di test sono simili, se non addirittura uguali, a quelli del caso d'uso TRAIN. Essi quindi comprendono :

1) Un file testuale comprendente i dati di test;

2) Un file di configurazione che specifica il tipo di esperimento e il caso d'uso , in questo caso TEST;
3) Un file contenente i parametri relativi al TEST;

Il file al punto uno è concettualmente identico al caso TRAIN.
Il secondo file è l'unico file che si fornisce direttamente in ingresso a GAME durante il caso d'uso TEST. Esso è formato da tre righe contenenti:

- Riga 1 : tipo di esperimento REGRESSION/CLASSIFICATION;
- Riga 2 : Caso d'uso , in questo caso TEST;
- Riga 3 : percorso del file contenente i parametri dell'esecuzione TEST.

Nel terzo file, invece, bisogna specificare tutti i parametri necessari a una corretta esecuzione del programma durante la fase TEST. Esso sarà sempre di tipo testuale e sarà composto da 4 righe :

- Riga 1 : percorso del file contenente il data set di test;
- Riga 2 : percorso riferito al file contenente la popolazione di input (questo file non è altro che il file di output relativo alla popolazione ottenuto in uscita alla fase di TRAIN);
- Riga 3 :percorso del file contenente i parametri di configurazione interna (si riferisce al file ottenuto in uscita alla fase di TRAIN sotto il nome di <tipo_esperimento>_trained_GAME_internal_params.txt);
- Riga 4 : percorso del file in cui salvare i dati di output (questo file va creato precedentemente e lasciato vuoto; sarà poi riempito durante l'esecuzione del programma).

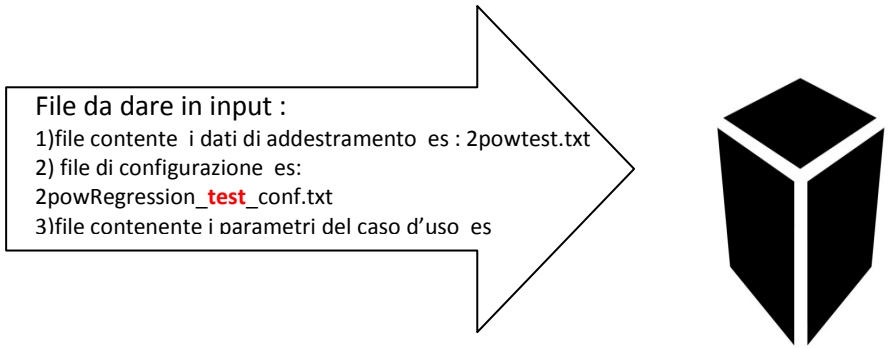


Figura 3.22 Schema riassuntivo file input

Una volta dati questi file di input, il sistema restituirà cinque file di output ; di cui:

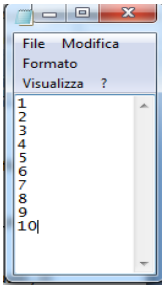
- Uno contenente i file di output (nella directory specificata nel file relativo ai parametri di test);
- Due file di log (generati automaticamente dal programma);
- Un file contenente i targets interni (generato automaticamente dal programma);
- Un file <tipo esperimento>_MLPGA_test_confmat.txt, contenente i risultati in termini statistici dell'esperimento (generato automaticamente dal programma);

3.2.3.3 Run

Il caso d'uso RUN è concettualmente molto simile al caso TEST. Anch'esso infatti è costituito da tre file di input del tipo:

1) Un file testuale comprendente i dati su cui vogliamo effettuare l'esperimento;
2) Un file di configurazione che specifica il tipo di esperimento e il caso d'uso , in questo caso RUN;
3) Un file contenente i parametri relativi al RUN;

A differenza dei primi due casi, in cui il file contenente i dati da sottomettere poteva anche essere uguale, in questo caso invece, la struttura dell'elaborato testuale si presenta differentemente. Infatti, non sono presenti le colonne relative ai target. Questo è ovvio, se si pensa al RUN come il vero e proprio svolgimento dell'esperimento su dati nuovi e non noti .



	Caratteristica
Pattern	
Pattern	
Pattern	

Figura 3.23 Esempio dataset caso RUN

Gli altri due file (quelli relativi alla configurazione e ai parametri) sono pressoché identici al caso TEST con le dovute eccezioni, riguardanti la sostituzione del caso d'uso da TEST →RUN e il cambio di alcune directory.Si propongono , a tal riguardo , alcuni esempi .

Esempio file di configurazione RUN :

```
#_functionality
REGRESSION
#_use_case
RUN
#_input_configuration_file_name
../run/input/2pow_run_params.txt
```

Figura 3.24 Esempio file di configurazione caso RUN

Esempio file di parametri RUN :

```
#_nome_del_dataset_di_test
../run/input/2powRun2.txt
#_nome_della_popolazione_di_input
../game_esetest/training/output/training/2pow_RegTrain_population.txt
#_nome del_file_di_configurazione_interna
../game_esetest/training/output/training/regression_trained_GAME_internal_params.txt
#_nome_del_file_output_test
../run/output/2pow_REGRUN_output.txt
```

Figura 3.25 Esempio file relativo ai paramtri di RUN

Una volta inseriti questi tre file di input il sistema restituirà tre file di output :

- Un file contenente il risultato dell'esperimento, contenuto nella directory specificata nel file relativo ai parametri di input.
- Due file di log.

Capitolo 4
Presentazione del Dataset

I dati usati per lo svolgimento di questo elaborato di Tesi sono stati messi a disposizione sotto gentile concessione dall'università di Brescia. Essi sono stati raccolti tramite il meccanismo del Ground Truth (GT) system . Ogni traccia di flusso è fornita di un file di log in cui sono specificati :

- il numero di porta a livello Trasporto;
- il risultato dell'analisi DPI;
- il nome dell'applicazione che ha generato il flusso, restituito da GT;

Inoltre per garantirne l'anonimato esse sono state "maneggiate" da Crypto-pan nel seguente modo[1] :

- sopra l'header del livello di trasporto;
- crittografia del prefisso dell' indirizzo ip per preservarne l'anonimato;
- mappatura dell'ip in diverse tracce con la medesima chiave;
- l'indirizzo originale può essere recuperato con una chiave;

[1] http://www.ing.unibs.it/ntw/tools/traces/

Una volta ottenute le tracce si cerca di applicare una procedura euristica per identificare il protocollo a livello applicazione che ha generato ciascun flusso di rete.

Le tracce sono state raccolte presso un router di bordo area dell'università di Brescia mentre il traffico è stato generato da un insieme di 20 workstations sulle quali girava GT client daemon.Il traffico è stato, quindi, raccolto eseguendo TCPdump (strumento per l'elaborazione di tracce in formato pcap)sul router della Facoltà, collegato alla rete Internet tramite un collegamento dedicato a 100Mb/s in uplink.La fase di raccolta delle tracce è durata tre giorni di lavoro consecutivi[2] (09/30, 10/01 e 10/02). Durante questa fase sono stati raccolti dall'università di Brescia circa 27 GB di dati in cui sono stati riconosciuti i seguenti protocolli di livello applicazione : Web (HTTP e HTTPS), posta (POP3, IMAP4, SMTP e le loro varianti SSL), Skype, il traffico generato dal peer-to-peer, come BitTorrent e eDonkey, e ad altri protocolli (FTP, SSH, e MSN).Oltre a un'analisi identificativa del protocollo applicativo su questo insieme di dati è stata eseguita un'ulteriore analisi relativa alla perdita dei pacchetti durante la fase di acquisizione.Un ulteriore test è stato eseguito per verificare se ci fossero state ulteriori perdite, causate ad esempio dal sovraccarico delle code FIFO usate per il trasferimento dei dati DMA tra la scheda di rete e il kernel. Il risultato di queste analisi ha mostrato che nel caso peggiore le perdite riportate dal TCPdump sono inferiori all'1% mentre la percentuale in Byte dei segmenti mancanti è inferiore al 0.22%; questo permette di poter considerare trascurabili, in questo caso, le perdite[3].

4.1 Realizzazione del Dataset

Il Networking e le tecniche di classificazione spesso si basano su un concetto di Ground Truth per valutare l'efficacia di una soluzione; per valutare ciò si osserva come il fenomeno in studio rifletta la Ground Truth. Il termine "Ground Truth", quindi, si riferisce a informazioni certe o note riguardanti il fenomeno da analizzare.

[2] http://www.ing.unibs.it/ntw/tools/traces/
[3] http://www.ing.unibs.it/ntw/tools/traces/

Per questo motivo dal file pcap contente 4.190.465 pacchetti sono stati estratti due file: uno riferito alla Ground Truth contenente 21.917 biflussi, e uno riferito all'organizzazione dei pacchetti in biflussi contente 23.467 biflussi.Con il termine biflusso si intende un insieme di pacchetti identificati dalla stessa quintupla : ip sorgente, ip destinatario, porta sorgente , porta destinatario e protocollo a livello trasporto , in una comunicazione bidirezionale (mittente e destinatario posso essere scambiati). In questo elaborato si farà riferimento a biflussi relativi al protocollo di trasporto TCP. Questa prima operazione permette di estrarre i Time Stamp iniziali di ogni biflusso.

La composizione del file Ground Truth è costituita da 8 campi, mentre quella riferita al file dei biflussi è costituita da 11:

- Ground Truth→Time Stamp: IP sorgente:IP destinatario: Porto sorgente: Porto destinatario: Protocollo a livello applicazione: Applicazione: Protocollo a livello trasporto;
- Biflussi→Time Stamp:IP sorgente:IP destinatario: Porto sorgente:Porto destinatario: Protocollo a livello Trasporto: Time Elapsed: Bytes: UpPackets: DownPackets: Up+DownPackets.

Si noti , inoltre, che la quadrupla iniziale, uguale in entrambi i file, è identificativa del biflusso.

In questo elaborato di Tesi, si è scelto di studiare come quattro caratteristiche riferite a informazioni temporali e quantitative possano contribuire a un'operazione di classificazione e/o identificazione del traffico di rete . Le caratteristiche scelte , riferite ai biflussi, sono :

1) Time Elapsed;
2) Byte;
3) Up Packets;
4) Down Packets.

Esse andranno a costituire le features(caratteristiche) nei dataset per GAME e FMLPGA. Come descritto nel capitolo riguardante gli algoritmi genetici, però, solo nel caso d'uso RUN

il dataset è costituito unicamente dalle features.Negli altri due casi ,TRAIN e TEST, è necessario fornire al programma anche i valori target, ossia i dati relativi ai risultati noti. Essendo questo un elaborato riferito alla classificazione del traffico di rete si è deciso di scegliere come target le applicazioni riportate nel file Ground Truth. Dall'analisi di quest'ultimo, sono state rilevate ben 31 applicazioni relative al livello Applicativo della suite di protocolli TCP/IP, identificate come di seguito :

Applicazione	Target Numerico
Mail	1
Bittorent.exe	2
firefox-bin	3
Skype	4
DashboardClient	5
Safari	6
privoxy	7
PubSubAgent	8
iTunes	9
SVCHOST.EXE	10
thunderbird-bin	11
firefox	12
gnome-panel	13
adeona_client.exe	14
ical	15
SSH	16
Transmission	17
Vmnet-natd	18
Octoshapeclient.exe	19
firefox.exe	20
synergy	21
dashboardadvisor	22
Microsoft	23
GoogleSoftwareup	24
amule	25
ocspd	26
opera	27
Skype.exe	28
SubmitReport	29
Btdna.exe	30
Thundrbird.exe	31

Tabella 4.1 Tabella delle applicazioni

Il cui significato è :

1) **MAIL :** servizio e-mail generico;
2) **Bittorent.exe:** applicazione p2p;
3) **firefox-bin:** browser web;
4) **Skype:** messaggistica istantanea , videochiamate , filesharing;
5) **DashboardClient:** applicazione per eseguire Widget Dashboard;
6) **Safari:** browser web;
7) **Privoxy**: programma di proxy web. Ha funzionalità di filtro per la protezione della privacy, per la modifica dei dati delle pagine web, per la gestione dei cookies, per il controllo degli accessi e per la rimozione selettiva di contenuti come annunci, banner e pop-up. Può essere personalizzato e può essere usato sia per sistemi stand-alone che per reti multi-utente;
8) **PubSubAgent:** recupera e gestisce i feed in background;
9) **iTunes:** lettore multimediale;
10) **SVCHOST.EXE:** "lanciatore" di servizi di Windows oppure, in altri casi, sistema che crea le condizioni per cui i Servizi di Windows possano girare;
11) **thunderbird-bin:** servizio e-mail;
12) **firefox:** browser web;
13) **gnome-panel:** servizi desktop;
14) **adeona_client.exe:** processo generico;
15) **iCal:** pianificazione eventi;
16) **ssh**:Secure Shell , sicurezza;
17) **Transmission:** applicazione p2p;
18) **Vmnet-natd:** processo che virtualizza la scheda di rete per renderla disponibile alla macchina virtuale, mettendola così in grado di navigare;
19) **Octoshapeclient.exe :** monitoraggio applicazioni;
20) **firefox.exe:** browser web;
21) **synergy:** condivisione mause e tastiera su più computer;
22) **dashboardadvisor :** applicazione per business;
23) **Microsoft:** processo generico;

24) **GoogleSoftwareUp:** applicazione in background per il sistema operativo Mac che aiuta a garantire che si abbia sempre la versione più aggiornata, stabile, e la versione più sicura del software di Google installato;
25) **amule:** applicazione p2p;
26) **ocspd:** demone che viene utilizzato dal framework di sistema per effettuare le verifiche del certificato di protezione;
27) **opera:** browser web;
28) **skype.exe:** messaggistica istantanea , videochiamate , filesharing;
29) **submitReport :** processo generico;
30) **Btdna.exe:** applicazione p2p;
31) **Thunderbird.exe:** servizio e-mai;l

La realizzazione del dataset generico nei due casi d'uso principali degli algoritmi, quindi, si ottiene associando a ciascuna quadrupla di features del file biflussi la corrispondente applicazione del file Ground Truth . Infine si otterrà un file testuale costituito da 35 colonne; di cui le prime 4 costituiscono le features e le rimanenti 31 colonne costituiscono i 31 target prima citati (il numero di colonna,escluse le colonne delle features, corrispondente al numero di applicazione) ; in queste ultime colonne sarà presente il numero "1" solo se le caratteristiche corrispondenti si riferiscono a quell'applicazione altrimenti sarà presente il numero "0".
A partire da questo dataset generale multi-classe, è possibile ottenere tutti gli altri dataset desiderati. Per ulteriori informazioni riguardanti la creazione di un dataset utilizzando un foglio elettronico consultare l'Appendice A :"Creazione di un dataset con Libre Office Calc".

4.2 Anomalie riscontrate

Le prime due anomalie riscontrate sono state trovate durante la semplice visualizzazione a video del file di Ground Truth. In alcuni biflussi di questo file, infatti, in corrispondenza del risultato del filtraggio dell'università di Brescia (campo protocollo a livello applicazione) "NoMACH", si presenta un errato collocamento delle caratteristiche che lo seguono.

L'errore è stato corretto manualmente posizionando in maniera conforme al resto dei dati le caratteristiche di tali biflussi. Osservando ancora il file di Ground Truth , però, si evince anche che alcuni biflussi presentano come risultato del filtraggio ben due risultati, con conseguente errata collocazione delle categorie seguenti. L'errore è stato risolto assegnando un unico risultato al biflusso, assegnando come protocollo quello più vicino alla caratteristica presente nel biflusso . Si è notato , inoltre , che questo tipo di errore è presente solo per due tipi di applicazione : Skype e Transmission , con maggiore frequenza in quest'ultima.

Le altre anomalie , invece , sono state registrate durante la fase di creazione del dataset di cui si mostra un diagramma di flusso riassuntivo di seguito :

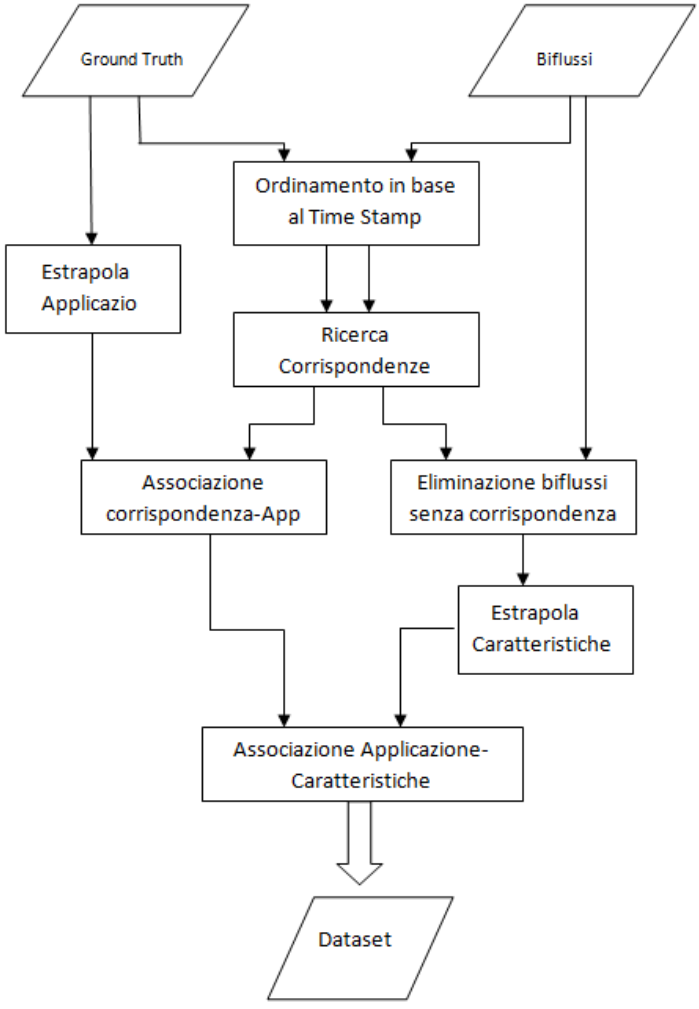


Figura 4.1 Diagramma di flusso della creazione di un dataset con foglio di calcolo

Dalla realizzazione dei dataset di input si evincono, infatti, due anomalie relativi ai biflussi estratti :

1) alcuni biflussi presentano caratteristiche mancanti;
2) alcuni biflussi presenti nel file dei biflussi estratti non trovano alcuna corrispondenza con i biflussi presenti nella ground truth.

4.2.1Caratteristiche assenti

Un biflusso in cui si evidenzia l'assenza di una caratteristica si presenta come in figura 4.1 :

9	1254319104.505322	245.234.7.168	53.235.30.39	51306	80	tcp	810612	4497	21	7	28
10	1254309271.400365	245.234.7.145	93.133.137.63	1799	36895	tcp	1.78528	68	3	3	6
11	1254321441.317510	245.234.7.181	173.228.7.99	58325	55813	tcp	0	0	1		1
12	1254321442.235297	245.234.7.181	173.228.7.99	58325	55813	tcp	45.4461	3191	13	10	23
13	1254311924.865496	245.234.7.180	53.235.30.39	43763	80	tcp	0.465746	3832	10	8	18
14	1254316201.140018	245.234.7.181	140.129.48.239	51882	64161	tcp	37.3874	3111	13	11	24

Figura 4.2 Esempio biflusso con caratteristica assente

Si noti che in genere un biflusso che presenta tale proprietà, oltre a presentare l'assenza di una caratteristica, presenta come altri parametri i valori 0/1.

Da una prima analisi visiva sul file dei biflussi estratti, si evince che prima o dopo il biflusso anomalo è presente un altro biflusso, identificato dalla stessa tupla, ma con Timestamp diverso, in cui sono presenti i valori reali riferiti alle caratteristiche del biflusso.

Continuando a visualizzare a video il file , inoltre , è possibile notare interi gruppi di biflussi con caratteristica mancante.

83	1254362617.165703	245.234.7.177	40.170.249.40	52929	80	tcp	29888	52452	27	40	67
84	1254326151.759975	245.234.7.180	184.62.7.243	35490	80	tcp	151013	20759	27	26	53
85	1254316304.244163	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
86	1254316305.178480	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
87	1254316306.181039	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
88	1254316307.183437	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
89	1254316308.185995	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
90	1254316309.188328	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
91	1254316311.193110	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
92	1254316315.202710	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
93	1254316323.221985	245.234.7.181	191161163130	51969	46960	tcp	846993	884	17	1	18
94	1254314703.702470	245.234.7.168	53.235.30.46	65342	80	tcp	0.719124	9317	16	14	30
95	1254321624.369580	245.234.7.161	53.65.88.252	59549	80	tcp	7.36971	906	6	3	9
96	1254305454.218157	245.234.7.175	181.66.245.217	63847	80	tcp	71.3541	137357	37	104	141

Figura 4.3 Insieme di biflussi con caratteristica assente

Anche in questo caso essi sono caratterizzati dalla stessa tupla, ma con Timestamp diverso e sono seguiti, o preceduti, da un altro biflusso con identica tupla, ma con i valori effettivi delle caratteristiche.

85	1254316304.244163	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
86	1254316305.178480	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
87	1254316306.181039	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
88	1254316307.183437	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
89	1254316308.185995	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
90	1254316309.188328	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
91	1254316311.193110	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
92	1254316315.202710	245.234.7.181	191161163130	51969	46960	tcp	0	0	1		1
93	1254316323.221985	245.234.7.181	191161163130	51969	46960	tcp	846993	884	17	1	18

Figura 4.4 Esmpio biflusso con caratteristica

Per capire il perchè di questa anomalia si è effettuata un'analisi sulla traccia di flusso fornita dall'università di Brescia. Aperto il file pcap in Wireshark (strumento per l'elaborazione di tracce di flusso in formato pcap), quindi, si procede con la ricerca dei pacchetti contenuti nel biflusso anomalo .

Essendo un biflusso identificato dalla tupla ip sorgente, ip destinatario, porta sorgente, porta destinatario) , si è utilizzato in Wireshark il filtro :

ip.src == xxx.yyy.zzz.kkk&&ip.dst == qqq.eee.rrr.ttt&&udp/tcp.port == X &&udp/tcp.port == Y

Esso permette di individuare tutti i pacchetti riferiti al biflusso cercato. Effettuando questa operazione, si nota che il flusso è composto da due o più pacchetti uguali in cui cambia solo il time Epoch.

A valle di ciò possiamo dire che il fenomeno "della caratteristica mancante " è dovuto probabilmente alla presenza di pacchetti duplicati nella traccia.

4.2.2 Corrispondenza non trovata

La seconda anomalia riscontrata è quella della corrispondenza mancante : ovvero il numero di biflussi estratti (23.467), non corrisponde al numero di biflussi contenuti nella Ground Trouth (21.917). Per vedere quali biflussi siano senza corrispondenza, è stato utilizzato il foglio elettronico utilizzato per la creazione del dataset in cui c'è tutta la procedura di

ricerca delle corrispondenze. Facendo, anche in questo caso, una prima analisi visiva sul file, si nota che i file senza corrispondenza possono essere di tre tipi :

- biflusso isolato senza corrispondenza;
- biflusso senza corrispondenza con caratteristica mancante (incluso caso con repliche);
- biflussi senza corrispondenza con caratteristica mancante e uguale Timestamp.

Per capire il perchè di questa anomalia si è effettuata , anche in questo caso, un' analisi in wireshark . Aprendo il file pcap della traccia di flusso dell'università di Brescia si utilizza lo stesso filtro di prima per ricercare , qualche biflusso senza corrispondenza .
Effettuando questo filtraggio si evince che il time stamp del biflusso anomalo non corrisponde al Time Epoch del primo pacchetto del flusso , ma a quello del secondo.

4.3 Analisi statistica

Una volta ottenuto il dataset generico è possibile condurre un'analisi dal punto di vista statistico per vedere com'è composto. Come detto in precedenza nel file Ground Truth si hanno a disposizione i risultati del filtraggio svolto dall'università di Brescia. Essi indicano il protocollo a livello applicativo assegnato al biflusso. Una prima analisi che si può fare, quindi, è vedere in che modo i protocolli siano distribuiti rispetto alle applicazioni.

	http	imap	pop3	smtp	skype	bittorr	edonk	ssl	ssh
firefox-bin	X							X	
Firefox.exe	X							X	
firefox	X							X	
Safari	X							X	
opera	X							X	
Skype	X				X		X	X	
Skype.exe					X		X		
Mail	X		X	X				X	
thunderbird-bin	X	X						X	
Thundrbird.exe	X							X	
DashboardClient	X								
PubSubAgent	X								
iTunes	X							X	

	http	imap	pop3	smtp	skype	bittorr	edonk	ssl	ssh
ical	X								
Bittorent.exe	X				X	X	X		
Transmission	X				X	X	X		
amule					X		X		
Btdna.exe	X								
privoxy	X								
ssh									X
ocspd	X								
GoogleSoftware								X	
SVCHOST.EXE	X							X	
Octoshapeclient	X								
synergy	X								
Microsoft	X								
SubmitReport								X	
Adeona-client.exe	X				X				
Dashboardvisor	X								
Gnome-panel	X								
VMNET-NTD	X								

Tabella 4.2 Tabella dei protocolli riferiti ad ogni applicazione

La tabella 4.2 mostra,per ogni applicazione, i vari protocolli a livello applicativo riscontrati durante il filtraggio. E' possibile notare come molte applicazioni che svolgono la stessa funzione siano caratterizzate anche dagli stessi protocolli; ad esempio firefox, firefox-bin, firefox.exe, opera, safari,tutti browser web, sono caratterizzati dai protocolli http e ssl(securesocket layer). Mentre per applicazioni più generali come Mail è possibile distinguere, al suo interno, vari protocolli che identificano servizi di posta elettronica differenti. Un caso particolare è, invece, Skype che, essendo un'applicazione multifunzione a protocolli proprietari e pacchetti cifrati, potrebbe permetterci, vedendo quale protocollo utilizzi, l'identificazione del servizio offerto.

Una seconda analisi che può essere condotta sul dataset è di tipo quantitativo. Come detto in precedenza, il dataset è composto da 31 applicazioni/targets. Essendo i dati utilizzati biflussi estratti da un insieme di pacchetti internet, la distribuzione dei 31 target sarà sicuramente non omogenea. E' bene, dunque, capire in che modo essi siano distribuiti e in che percentuale.

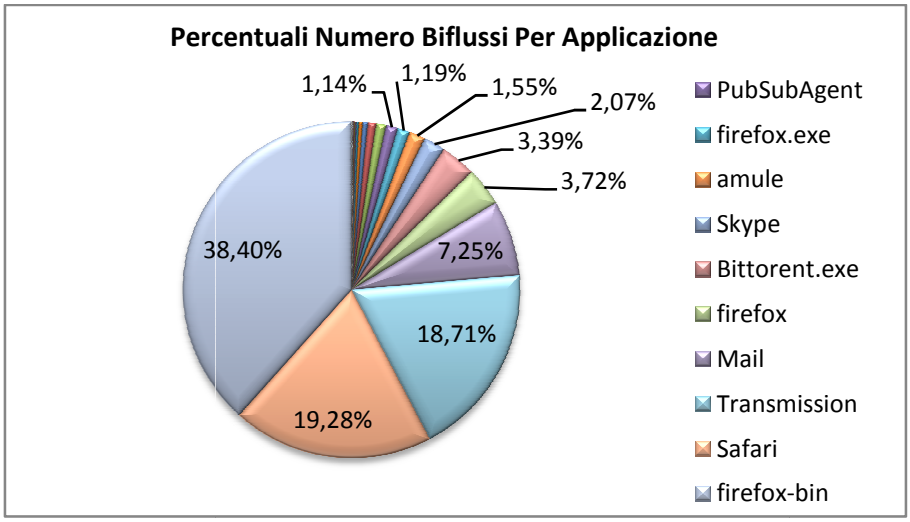


Figura 4.5 Grafico delle percentuali dei numeri di biflussi per applicazione

La figura 4.4 mostra le percentuali (riferite al numero di biflussi) con cui sono presenti le applicazioni nel dataset. Non sono state riportate le applicazioni con una percentuale inferiori al'1% . Da questo grafico, quindi, si può notare un forte squilibrio tra i vari target. Si può pensare, allora, di unire più classi di applicazione affini tra loro, come ad esempio tutti i browser web o tutte le applicazioni p2p, in una classe e di considerare tutte le classi, al di sotto di una percentuale di biflussi rilevante, come un'unica classe denominata Other in modo tale da realizzare un primo bilanciamento dei targets.

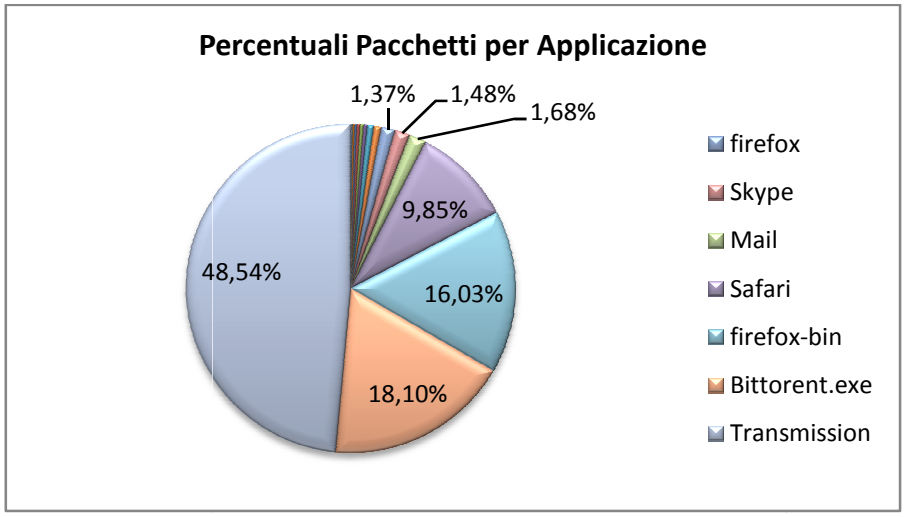


Figura 4.6 Grafico delle percentuali dei pacchetti per applicazione

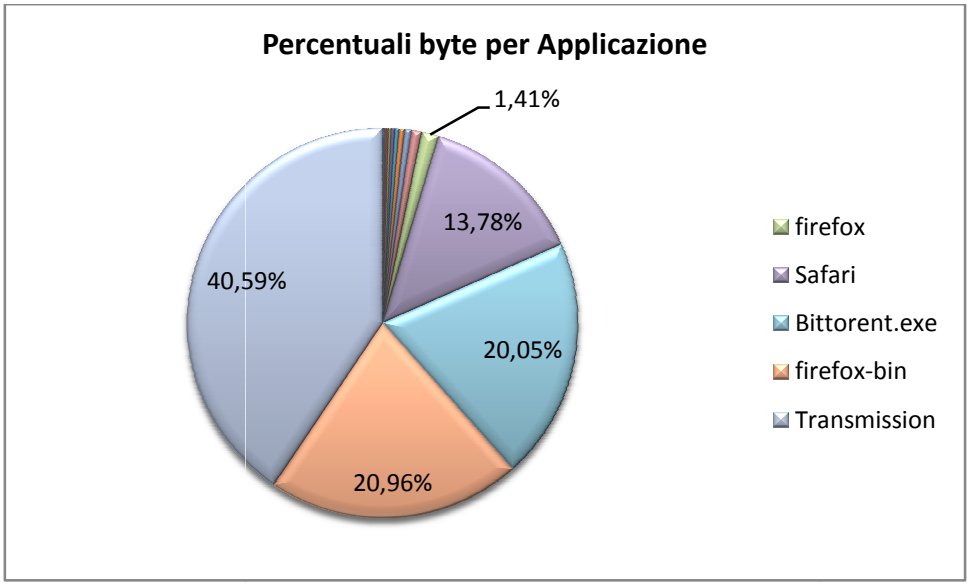


Figura 4.7 Grafico byte per applicazione

In figura 4.5 e 4.6 sono mostrate, per una più completa visione del dataset, i grafici relativi alle percentuali riferite al numero di pacchetti e al numero di byte per ogni applicazione. Anche in questo caso non sono state riportate le applicazioni con percentuali inferiori all'1%.

Capitolo 5

Risultati Sperimentali

Come anticipato nella premessa, lo scopo ultimo di questo lavoro di tesi consiste nella sperimentazione di tecniche di machine learning per la classificazione del traffico di rete. A tal proposito sono stati utilizzati i modelli FMLPGA e GAME, di cui è disponibile un'implementazione tramite il team del progetto DAME[4].L'impiego di modelli ibridi ed algoritmi genetici di questo tipo rappresentano un carattere innovativo nel settore della classificazione del traffico di rete.
Sono stati realizzati, quindi, vari esperimenti atti a studiare: le capacità di classificazione degli algoritmi al variare delle classi di traffico, la velocità di apprendimento al variare del numero di biflussi e la congruenza dei risultati.

5.1 I dataset utilizzati

Per la realizzazione degli esperimenti il dataset originario di 31 classi target è stato diviso in 6 sottoclassi. Per realizzare ciò le classi di traffico sono state raggruppate per attinenza e numero di biflussi, ottenendo la seguente suddivisione del dataset :

Classe	Sotto Classi contenute	Identificativo

[4] http://dame.dsf.unina.it/

Browser Web	fire-fox, firefox.bin, firefox.exe, safari, opera	1
Skype	Skype, skype.exe	2
P2P	bittorent.exe, amule, Transmission,Brdna.exe	3
TrafficoCifrato	privoxy, ssh, ocspd	4
Mail	mail, thundrbird-bin, thundrbird.exe	5
Other	classirimanenti	6

Tabella 5.1 Tabella delle nuove classi ottenute per accorpamento

Dal punto di vista statistico il dataset suddiviso in sei classi risulta distribuito come il seguente grafico

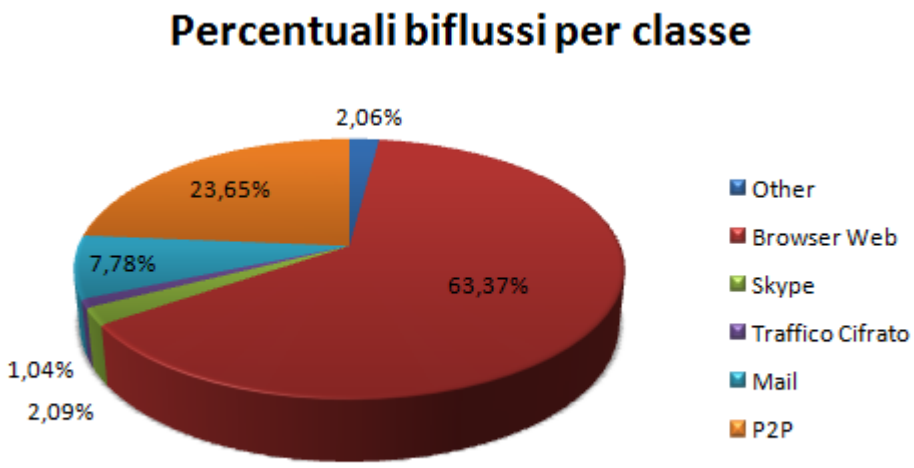


Figura 5.1 Visione statistica del nuovo dataset

Una volta ottenuto questo dataset è stata eliminata la classe Other in modo da focalizzare l'attenzione sulle classi di interesse. Dall'insieme di dati così ottenuto sono stati ricavati dieci dataset bi-classe. Lo scopo di questa operazione è quello di voler studiare il confronto tra due possibili classi di traffico; se si considerano tutti i possibili confronti di due elementi che si possono ottenere da un insieme di cinque, il numero che si ottiene è dieci che è proprio il numero di dataset creati.

- Browser Web vs Mail;

- Browser Web vs Skype;
- Browser Web vs P2P;
- Browser Web vs Traffico Cifrato;
- Mail vs P2P;
- Mail vs Skype;
- Mail vs Traffico Cifrato;
- Traffico Cifrato vs P2P;
- Traffico Cifrato vs Skype;
- Skype vs P2P.

Una volta creati questi nuovi insiemi di dati contenenti solo due classi target, essi sono stati bilanciati in modo tale da non contaminare il campione, ovvero il numero di biflussi appartenenti a una determinata classe è uguale al numero di biflussi presenti nell'altra. I dataset così costituiti sono stati inseriti in DAMEWARE[5], un servizio basato su web, per l'elaborazione di dataset di ML, per poter mescolare in modo random i biflussi e per poter dividere ogni dataset nelle due percentuali per la fase di test e di train, garantendo una distribuzione bilanciata dei vari tipi di istanze (pettern) nei due insiemi di training e test.

5.2 Lettura dei risultati e la matrice di confusione

I risultati forniti da un algoritmo genetico, in genere, sono forniti in termini statistici. Di norma, in esperimenti basati su tecniche di ML, nella fase di training la statistica ha significato solo per valutare il livello di apprendimento del modello rispetto al dataset fornito in input.Nella fase di Test, invece, le prestazioni qualitative vengono dedotte proprio dall'analisi dei risultati, dato che al sistema vengono sottomessi biflussi che l'algoritmo non conosce e su cui non sia stato addestrato; dunque è in questa fase che il grado di generalizzazione dei classificatori empirici ha un'importanza oggettiva[29].
In genere si fa uso dei seguenti indicatori statistici in riferimento ai risultati :

[5] http://dame.na.astro.it:8080/MyDameFE/

- *Classification accuracy:* la percentuale totale di biflussi correttamente classificati, rispetto alla totalità dei biflussi del dataset. Con riferimento alla matrice di confusione, tale indicatore si ottiene tramite somma dei valori della diagonale principale, in percentuale rispetto alla totalità del campione;
- *Purezza:* la percentuale di biflussi relativi ad ogni singola classe correttamente riconosciuti come tali dal modello (i valori singoli della diagonale principale rispetto ad ogni classe);
- *Contaminazione:* il complemento della purezza. Ossia la percentuale di biflussi, nella base di conoscenza,etichettati come non appartenenti ad una classe, ma erroneamente attribuiti a quella classe dal modello.

Entrambi i casi d'uso, train e test, forniscono in output le matrici di confusione, nei file classification_MLPGA_train/test_confmat per GAME e in train/test_confusionmatrix.txt per FMLPGA. La matrice di confusione è l'elemento chiave per l'interpretazione dei risultati di classificazione. Questa matrice va letta per righe : ogni riga si riferisce ad una classe target, mentre ogni colonna riporta la classe output del modello. Per ogni singola classe target (riga) la somma dei valori nelle sue colonne corrisponde al numero totale di biflussi di quella classe. Il valore sulla diagonale principale, invece, indica sempre il corretto numero di biflussi perfettamente classificati dal modello per ogni classe target (classification accuracy).

class	class 1 (output)	class 2 (output)
class 1 (target)	20	5
class 2 (target)	2	40

Figura 5.2 Esempio matrice di confusione

5.3 Test qualitativi su GAME

La prima serie di esperimenti effettuata sui dataset con il modello GAME, di cui si riportano gli esiti di seguito, sono volti allo studio degli aspetti qualitativi. Per aspetti qualitativi si intendono le percentuali di giusta classificazione di ogni classe di traffico. Per questa prima esperienza sono state utilizzate prima le percentuali 70%-30% e poi quelle 80%-20% di pattern, rispettivamente per la fase di train e test .

Dataset	N° Biflussi Train (70%)	N° Biflussi Test (30%)
BrowserWeb-Mail	1043	447
BrowserWeb-Skype	261	113
BrowserWeb-P2P	806	346
BrowserWeb-TrafCifr	172	74
Mail-P2P	1843	791
Mail-Skype	541	233
Mail-TrafficoCifrato	268	115
TrafficoCifrato-P2P	237	103
TrafficoCifrato-Skyppe	287	123
Skype-P2P	343	147

Tabella 5.2 Numero di biflussi percentuali 70-30

Dataset	N° Biflussi Train (80%)	N° Biflussi Test (20%)
BrowserWeb-Mail	1192	298
BrowserWeb-Skype	921	231
BrowserWeb-P2P	300	74
BrowserWeb-TrafCifr	197	49
Mail-P2P	2107	527
Mail-Skype	619	155
Mail-TrafficoCifrato	306	77
TrafficoCifrato-P2P	392	98
TrafficoCifrato-Skyppe	270	70
Skype-P2P	328	82

Tabella 5.3 Numero di biflussi percentuali 80-20

Dall'analisi di questo esperimento si vuole studiare quali classi di traffico, confrontate con tutte le altre, risultino più facilmente classificabili dall'algoritmo e quali, invece, risultino maggiormente difficili da classificare. Per perseguire questo scopo si sono svolti vari step . Il primo passo è consistito nell'eseguire un test su dataset non bilanciati e quindi utilizzando

tutti i biflussi disponibili per le due classi. Ovviamente essendo la distribuzione delle classi nel dataset generale (figura5.1) non uniforme, i risultati risultano fortemente sbilanciati a favore della classe dominante (quella con più biflussi). In questo caso siamo in presenza di una forte contaminazione dei dati, dovuta al fatto che la classe, la cui presenza sia maggiore nel dataset, eclissa completamente l'altra classe, ottenendo, nella maggior parte dei casi, percentuali di classificazione superiori al 90% di cui però si è riusciti a identificare correttamente solo i biflussi della classe dominante. La conseguenza è quindi l'ottenimento di percentuali di classificazione bassissime (0% -3%) nell'altra. Si è proseguito, allora, effettuando i test sui dataset bilanciati, di cui si riporta il numero di biflussi utilizzati nelle tabelle5.2 e 5.3 . In questo caso non ci si aspetta una forte contaminazione dei dati, visto che le classi sono state precedentemente bilanciate.

L'esperienza consiste nell'effettuare, per ogni dataset, le fasi di train e test variando i parametri riferiti a:

- grado dell'espansione polinomiale : 5 , 8 ,10;
- tipo di funzione di selezione : Roulette, Fitting, Ranking.

Il numero di iterazioni , invece, è stato fissato a 1000 in tutti i casi. Di seguito si riportano gli altri parametri utilizzati, riferiti alla configurazione della fase di train:

- Modo di generazione della popolazione iniziale : GRANDOM (distribuzione gaussiana);
- Numero delle caratteristiche (features) di input : 4;
- Numero dei targets (numero di classi) : 2;
- Grado del polinomio : 5, 8, 10;
- Tipo della funzione polinomio : CL_POLY_TRIGO (espansione trigonometrica);
- Tipo della funzione errore :TMSE (MSE a soglia);
- Soglia d'errore opzionale: 0.1;
- Tipo di funzione di selezione: Roulette,Fitting, Ranking;
- Soglia d'errore : 0.001;
- Numero massimo di iterazioni : 1000;
- Frequenza di visualizzazione dell'errore parziale di output: 10;
- Crossover rate : 0.9 (90%);
- Mutation rate: 0.3 (30%);

- Numero di tournament chromosomes: 4 (usato nel ranking);
- Elitism rate: 2 (2 cromosomi di elite).

Si è scelto, quindi, di studiare le prestazioni di classificazione del modello non solo dal punto di vista delle classi di traffico, ma anche al variare del grado del polinomio e del tipo di funzione di selezione. Si è scelto , invece, di utilizzare come modalità di generazione della popolazione iniziale GRANDOM poiché esso risulta essere , tra le varie disponibili, la tecnica con i più alti livelli di casualità.

Una volta definiti i parametri, sono state effettuate tre prove identiche(tre per il caso 70%-30% e tre per il caso 80%-20%), utilizzando gli stessi dataset e gli stessi parametri per verificare l'attendibilità dei risultati. I risultati finali sono quindi riportati in media sulle tre prove.

Gli esperimenti sono stati eseguiti su una macchina linux Intel Xeon CPU E5405 @ 2,00GHz. Dopo ogni esecuzione della fase di train e test il programma restituisce a video delle informazioni relative alla tempistica di esecuzione e al best error ottenuto, mentre nel file classification_GAME_train/test_cofmat vengono salvate le matrici di confusione. Per lo studio dei risultati tutte queste informazioni sono state salvate in specifiche tabelle relative al dataset in uso, di cui si riporta un esempio di seguito :

TRAINING		
	P(class1 P2P	P(class2 MI)
target class 1	737	177
target class 2	78	851
total classification		86,16%
class 1 classification		80,63%
Class 2 classification		91,60%
Train Time	32:28.47	
Best error	0,122595	
threshold	0,5	
N° Patterns	1843	

TEST		
	P(class1 P2P	P(class2 MI)
target class 1	326	77
target class 2	33	355
total classification		86,09%
class 1 classification		80,89%
Class 2 classification		91,49%
Train Time	00:06.00	
threshold	0,5	
N° Patterns	791	

Figura 5.3 Esempio tabelle dei risultati

Per avere, invece, un'idea più generale e complessiva dell'intero esperimento si riportano i risultati su una media di tre prove, relativi alle percentuali di classificazione complessiva dei vari casi studiati, e al tempo di train, sia nel caso 70%-30% che per quello 80%-20%.

NETWORK TRAFFIC CLASSIFICATION - TEST BASED ON GAME												
70% train – 30% test				*these columns report classification accuracy percentages (average on two classes)*								
V	P	CASE	FUNC	Br-Ml	Br-PP	Br-Sk	Br-TC	Ml-PP	Ml-Sk	Ml-TC	PP-Sk	PP-TC	Sk-TC
CPU	5	TRAIN	ROULET.	75,65	66,01	62,2	68,6	78,71	79,73	74,63	63,17	70,6	71,43
			FITTING	75,65	66,42	63,6	68,99	78,93	79,42	74,01	63,75	70,89	72,71
			RANK.	75,49	66,09	62,45	68,02	78,8	79,42	74,63	63,36	71,31	72,36
	8	TRAIN	ROULET.	85,62	67,74	61,81	70,93	85,93	88,91	85,2	66,76	78,62	75,15
			FITTING	85,84	68,36	63,09	73,26	85,84	88,66	85,82	66,38	77,5	75,91
			RANK.	86	68,16	61,17	72,87	85,3	88,72	85,32	66,18	77,78	75,03
	10	TRAIN	ROULET.	85,75	67,99	62,71	70,35	86,47	88,66	86,57	66,86	77,64	75,84
			FITTING	86,45	68,78	64,37	71,9	86,56	88,72	87,31	68,51	81,29	76,31
			RANK.	85,91	67,2	63,09	69,77	86,42	88,42	85,57	67,83	78,45	75,73
		NUM PATTERNS		1043	806	261	172	1843	541	268	343	237	287
	5	TEST	ROULET.	73,75	63,68	57,82	68,02	79,43	75,54	73,62	62,13	63,43	73,17
			FITTING	74,05	63,29	57,52	68,02	79,35	75,97	73,04	62,59	64,4	71,82
			RANK.	74,05	63,2	57,82	67,12	79,48	76,25	72,75	62,36	63,11	72,09
	8	TEST	ROULET.	81,95	66,76	57,23	61,71	85,93	87,84	80	69,84	63,75	72,9
			FITTING	81,06	67,44	55,46	63,96	85,5	87,41	80,87	70,75	63,43	72,63
			RANK.	81,43	67,34	59	60,81	84,91	87,7	80,87	70,52	61,17	72,63
	10	TEST	ROULET.	81,58	66,47	59,29	64,86	85,25	86,84	80,58	68,03	64,08	73,98
			FITTING	81,8	67,44	58,7	66,22	85,46	88,41	83,19	68,25	62,78	74,25
			RANK.	82,48	66,57	58,7	65,32	85,38	87,41	80	69,55	63,43	72,63
		NUM PATTERNS		447	346	113	74	791	233	115	147	103	123

Tabella 5.4 Risultati GAME 70-30

NETWORK TRAFFIC CLASSIFICATION - TEST BASED ON GAME												
70% train -- 30% test				*execution time in seconds on a host Intel Xeon CPU E5405 @ 2,00GHz*								
V	P	CASE	FUNCTION	Br-Ml	Br-PP	Br-Sk	Br-TC	Ml-PP	Ml-Sk	Ml-TC	PP-Sk	PP-TC	Sk-TC
CPU	5	TRAIN	ROULET.	665	521	191	110	1164	343	168	251	154	193
			FITTING	663	518	188	108	1156	340	164	253	151	191
			RANKING	668	521	192	110	1162	342	165	251	154	193
		NUM PATTERNS		1192	921	300	197	2107	619	306	392	270	328
	8	TRAIN	ROULET,	1114	880	320	184	1936	823	271	423	252	318
			FITTING	1109	885	320	181	1936	569	269	423	1592	317
			RANKING	1108	883	321	182	1958	570	269	423	253	318
		NUM PATTERNS		1192	921	300	197	2107	619	306	392	270	328
	10	TRAIN	ROULET.	1445	1141	414	236	2505	732	346	545	323	407
			FITTING	1437	1134	411	234	2515	734	346	543	322	406
			RANKING	1443	1132	414	236	2504	743	345	543	324	408
		NUM PATTERNS		1192	921	300	197	2107	619	306	392	270	328

Tabella 5.5 Risultati tempi GAME 70-30

NETWORK TRAFFIC CLASSIFICATION - TEST BASED ON GAME												
80% train -- 20% test				*these columns report classification accuracy percentages (average on two classes)*								
V	P	CASE	FUNC	Br-Ml	Br-PP	Br-Sk	Br-TC	Ml-PP	Ml-Sk	Ml-TC	PP-Sk	PP-TC	Sk-TC
CPU	5	TRAIN	ROULET.	74,97	65,94	62,22	68,36	79,15	78,62	74,73	63,44	72,22	71,44
			FITTING	74,80	65,94	62,11	68,02	78,93	78,78	75,16	63,18	71,85	71,44
			RANK.	74,44	65,91	62,33	68,70	79,09	78,41	75,16	62,84	71,23	71,24
	8	TRAIN	ROULET.	84,62	67,61	61,78	72,08	85,59	87,51	84,64	66,75	74,57	74,70
			FITTING	84,56	67,39	61,44	71,40	85,48	87,94	84,64	68,45	75,31	73,58
			RANK.	84,93	67,72	61,89	73,27	85,94	87,83	84,20	67,77	75,56	72,97
	10	TRAIN	ROULET.	84,76	68,08	63,22	72,08	85,92	87,88	86,71	67,77	76,79	75,51
			FITTING	85,15	67,75	62,22	72,76	86,58	87,61	77,47	68,28	77,78	75,00
			RANK.	84,56	67,14	61,44	72,42	86,43	88,21	86,71	67,86	76,79	74,80
		NUM PATTERNS		1192	921	300	197	2107	619	306	392	270	328
	5	TEST	ROULET.	78,19	62,63	53,15	64,63	78,68	78,06	73,16	63,61	62,38	76,83
			FITTING	78,30	62,63	54,05	65,31	78,49	78,28	73,59	64,63	61,43	77,64
			RANK.	77,74	62,77	52,70	64,63	78,56	78,92	72,73	64,63	62,86	77,64
	8	TEST	ROULET.	82,77	69,12	57,66	59,86	86,34	89,89	82,25	70,75	60,00	76,83
			FITTING	83,22	69,26	57,21	59,86	86,02	91,40	82,68	71,43	61,90	76,42
			RANK.	83,56	69,26	54,95	59,86	86,40	90,69	80,09	70,75	62,38	75,61
	10	TEST	ROULET.	83,33	68,69	59,01	61,90	84,76	89,89	83,55	73,13	62,86	77,24
			FITTING	83,89	72,01	61,71	61,22	85,20	89,89	83,98	71,09	59,05	76,42
			RANK.	84,34	69,99	59,46	59,86	85,39	90,11	80,52	71,09	58,57	76,42
		NUM PATTERNS		298	231	74	49	527	155	77	98	70	82

Tabella 5.6 Risultati GAME 80-20

NETWORK TRAFFIC CLASSIFICATION - TEST BASED ON GAME												
80% train -- 20% test				*execution time in seconds on a host Intel Xeon CPU E5405 @ 2,00GHz*								
V	P	CASE	FUNCTION	Br-Ml	Br-PP	Br-Sk	Br-TC	Ml-PP	Ml-Sk	Ml-TC	PP-Sk	PP-TC	Sk-TC
CPU	5	TRAIN	ROULET.	760	597	219	128	1328	393	189	284	175	219
			FITTING	757	592	218	125	1330	388	186	280	172	218
			RANKING	760	593	218	128	1321	392	185	281	174	217
		NUM PATTERNS		1192	921	300	197	2107	619	306	392	270	328
	8	TRAIN	ROULET,	1258	1011	368	212	2216	651	306	478	288	366
			FITTING	1263	1005	365	209	2223	654	302	476	286	361
			RANKING	1271	1009	367	210	2234	650	304	477	287	363
		NUM PATTERNS		1192	921	300	197	2107	619	306	392	270	328
	10	TRAIN	ROULET.	1637	1293	476	272	2875	844	392	614	369	468
			FITTING	1640	1303	469	269	2866	838	390	610	365	462
			RANKING	1637	1298	473	271	2868	840	391	613	369	465
		NUM PATTERNS		1192	921	300	197	2107	619	306	392	270	328

Tabella 5.7

5.3.1 Analisi dei risultati

Dai risultati si evince che le prestazioni migliori si hanno con l'utilizzo del polinomio di grado 10, mentre a parità di condizioni la funzione di selezione più performante durante la fase di train nel caso 70%-30% è la Fitting.Con essa si riscontrano le percentuali di classificazione più alte : nel caso dell'utilizzo del polinomio di grado 5, circa l'80% delle sue percentuali risulta più alta rispetto ai risultati ottenuti con le altre due funzioni.Nel caso di utilizzo di un polinomio di grado 8 si ottiene circa il 50% e con un polinomio di grado dieci si arriva al 90%. Nella modalità d'uso test (ben più significativa),la funzione più performante risulta essere la funzione Roulette rispetto alla Fitting su due casi (grado 5 e 8) su tre (grado 10). Nel caso 80%-20%, invece la funzione più performante nel caso d'uso train risulta essere la Roulette .

Dal puto di vista delle classi di traffico il confronto che ha ottenuto complessivamente le percentuali di classificazione più alte è stato Mail vs Skype. Queste due classi risultano quindi , facilmente classificabili tra loro.Ciò significa che strutturalmente sono abbastanza diverse da poter consentire una buona identificazione dei biflussi. Da un'analisi protocollare (tabella 4.2) questo comportamento potrebbe essere attribuito al fatto che queste due classi contengono in esse biflussi che appartengono a più protocolli : infatti la classe Mail racchiude biflussi relativi a ben cinque protocolli (http, imap, pop3, smtp, ssl), mentre la classe Skype ne racchiude quattro (http, skype, edonkey, ssl). Inoltre tra questi protocolli solo due sono in comune nelle due classi (http , ssl), mentre i protocolli più specifici delle classi sono presenti solo in una o nell'altra. Questa distribuzione di protocolli potrebbe contribuire, quindi, alla diversificazione delle due classi permettendo all'algoritmo una facile discriminazione dei dati. Per una maggior comprensione del risultato si riportano nello specifico i risultati riferiti a questo dataset nel caso di polinomio 10.

Mail– Skype					
TRAINING			TEST		
	P(class1 Sk)	P(class2 Ml)		P(class1 Sk)	P(class2 Ml)
target class 1	219	47	target class 1	103	18
target class 2	15	260	target class 2	7	105

total classification		88,54%	total classification		89,27%
class 1 classification		8233,08%	class 1 classification		85,12%
Class 2 classification		9454,55%	Class 2 classification		93,75%
Train Time	12:13.90		Train Time	00:00,01	
Best error	0,101463		threshold	0,5	
threshold	0,5		N° Patterns	233	
N° Patterns	541				

Figura 5.4 Esempio tabella risultati Mail-skype

Le classi, invece, che hanno ottenuto i risultati peggiori sono state BrowserWeb-Skype. Queste classi evidentemente risultano troppo simili tra loro e quindi difficilmente classificabili. Effettuando un'analisi simile a quella effettuata nel caso Mail-Skype si nota che gli unici due protocolli, a cui si riferiscono i biflussi, che sono contenuti nella classe BrowserWeb (http , ssl) sono in comune con la classe Skype ; questo potrebbe contribuire a rendere le classi molto simili tra loro e quindi difficilmente identificabili.

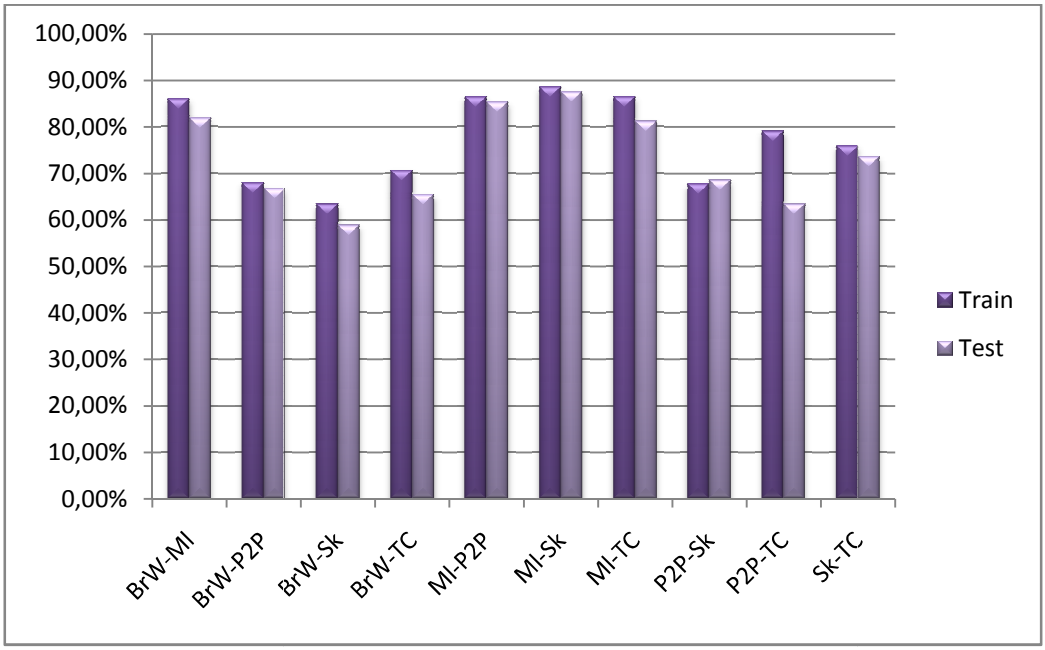


Figura 5.5 Grafico percentuali di classificazione poli 10

Il grafico in figura 5.3 mostra le percentuali di classificazione per ogni coppia di classe di traffico nel caso di utilizzo di polinomio di grado 10 (percentuali ottenute mediando sulle tre funzioni di selezione relative al caso di grado 10) nei due casi train e test.

Dal punto di vista delle tempistiche di esecuzione, invece, la funzione di selezione più performante risulta essere la Fitting.

5.4 Test quantitativi su GAME

In questa seconda serie di test si è interessati a studiare le velocità di esecuzione di GAME su CPU dell'algoritmo. Essendo le tempistiche relative alla fase di test trascurabili (nel nostro caso si aggirano tra i 0-10 secondi), l'attenzione si è focalizzata soprattutto a quelli relativi alla fase di train. I tempi della fase di train, infatti, sono molto importanti poiché in genere possono essere medio-lunghi ; la cosa importante , però, è che una volta speso il tempo per la fase dell'addestramento dell'algoritmo, fase eseguita una sola volta, le fasi successive di utilizzo sono altresì caratterizzate dal tempo ottenuto durante la fase di test che è ragionevolmente più breve.

Per effettuare, allora, questa serie di test prestazionali relativi al training, si è scelto il dataset contenente più biflussi (Mail-P2P) e sono stati ricompattati il dataset di train e quello di test in un unico file. Questo file è stato inserito, poi, in DAMEWARE e ne sono stati ricavati altri quattro, contenenti rispettivamente il 25%, 50%, 75% ed il 100% dei biflussi del dataset di origine. Per ognuno di questi quattro dataset sono stati ricavati i dataset di train e test nelle percentuali 70%-30%.Una volta sistemati i dati si è proseguito con le loro varie esecuzioni su GAME. I risultati in termini temporali sono riportati di seguito .

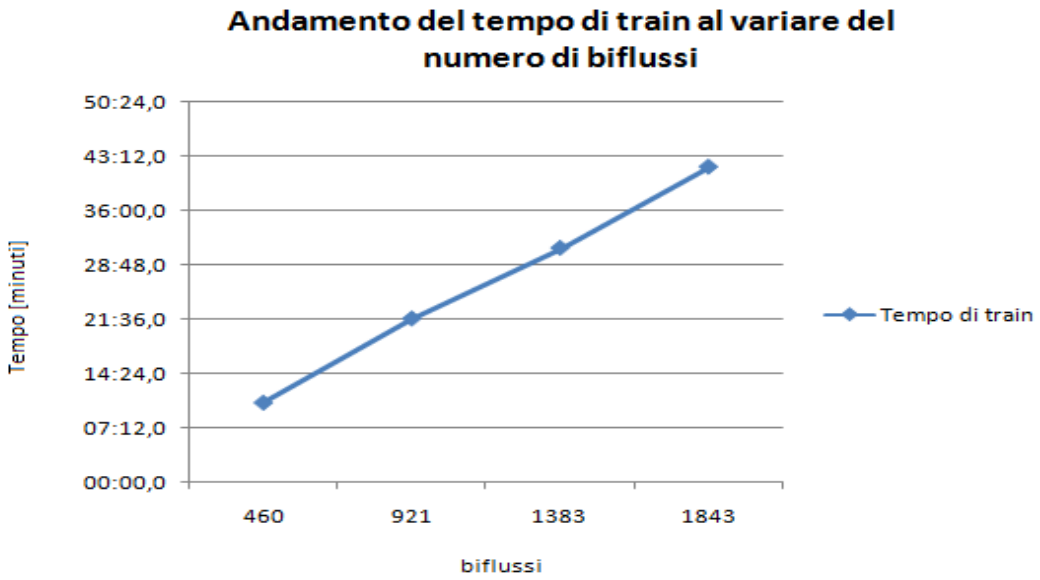


Figura 5.6 Grafico dei tempi GAME

Dalla figura 5.6 Si può notare che l'andamento del tempo di train all'aumentare del numero di biflussi della fase train si può considerare lineare.

5.5 Test qualitativi su FMLPGA

Per FMLPGA sono stati svolti esperimenti qualitativi analoghi al caso GAME . Per questo modello di ML, tuttavia, si è scelto di studiarne le prestazioni al solo variare della funzione di selezione, che come detto nel capitolo 3, può essere scelta tra ROULETTE_v_1, ROULETTE_v_2, FITTING e RANKING. Inoltre per questo modello è disponibile l'implementazione anche su GPU e quindi i test sono stati eseguiti su entrambe le piattaforme. I parametri utilizzati per tali esperimenti relativi alla configurazione della rete neurale sono :

- numero di caratteristiche (features) input : 4;
- numero di livelli nascosti intermedi: 2;
- numero di neuroni del primo livello nascosto : 9;

- numero di neuroni del secondo livello nascosto : 3;
- numero di neuroni output (classi target) : 3;
- tipo di funzione di attivazione dei neuroni del primo livello hidden : activation_tanh;
- tipo di funzione di attivazione dei neuroni del secondo livello hidden: activation_tanh;
- tipo di funzione di attivazione dei neuroni del livello output : activation_tanh.

Mentre i parametri riferiti alla configurazione di Train sono :

- tipo di esperimento: CLASSIFICATION;
- tipo di generazione casuale: GRANDOM;
- tipo di funzione di selezione: RANKING, FITTING, ROULETTE_v_1, ROULETTE_v_2;
- soglia d'errore: 0.001;
- numero di iterazioni: 10000;
- frequenza di visualizzazione dell'errore parziale di output:100;
- probabilità di crossover: 0.9 (90%);
- probabilità di mutazione: 0.4 (40%);
- numero cromosomi per popolazione:20;
- elitismo: 2;
- numero cromosomi per il RANKING: 4;

I test sono stati realizzati su un host i7-quad core 3.4 GHz CPU e GPU device Geforce460 a 336 cores. I risultati in termini di percentuali di classificazione e tempi di esecuzione sono riportati di seguito .

NETWORK TRAFFIC CLASSIFICATION - TEST BASED ON FMLPGA												
70% train -- 30% test			*classification accuracy percentages (average on two classes)*									
V	CASE	FUNC	Br-Ml	Br-PP	Br-Sk	Br-TC	Ml-PP	Ml-Sk	Ml-TC	PP-Sk	PP-TC	Sk-TC
CPU	TRAIN	ROULV1	92,43	84,42	95,4	87,21	95,28	96,86	92,16	83,09	91,56	96,17
		ROULV2	92,72	85,24	96,93	85,47	94,79	98,15	97,39	87,46	94,94	97,21
		FITTING	90,52	84,86	92,34	86,05	93,65	95,56	96,64	86,88	88,19	93,73
		RANKING	91,45	84,12	93,45	85,69	94,12	96,32	95,41	86,98	91,54	94,54
	NUM PATTERNS		1044	806	261	172	1843	541	268	343	237	287
	TEST	ROULV1	89,51	82,08	97,35	83,78	94,31	93,56	90,43	81,63	78,64	94,31
		ROULV2	91,07	80,64	95,58	83,78	94,18	94,85	91,3	87,07	85,44	95,12
		FITTING	87,72	82,08	90,27	81,08	92,54	93,13	90,43	89,12	72,82	92,68
		RANKING	89,32	81,54	92,65	82,09	93,47	93,51	90,11	88,32	73,12	93,44
	NUM PATTERNS		448	346	113	74	791	233	115	147	103	123
GPU	TRAIN	ROULV1	90,71	84,49	96,93	77,91	93,22	97,6	94,78	88,63	95,36	95,82
		ROULV2	92,43	85,24	92,34	87,79	94,9	97,23	91,79	87,17	94,51	95,82
		FITTING	91,28	84,24	90,04	86,63	92,13	97,41	98,13	77,26	87,76	95,82
		RANKING	91,02	84,05	92,54	79,54	92,66	97,11	92,55	79,17	89,65	94,87
	NUM PATTERNS		1044	806	261	172	1843	541	268	343	237	287
	TEST	ROULV1	88,84	80,64	96,46	79,73	91,91	95,28	89,57	91,16	86,41	92,68
		ROULV2	90,63	83,82	90,27	86,49	93,3	93,13	90,43	85,03	84,47	93,5
		FITTING	89,73	81,21	92,92	85,14	92,54	95,28	94,78	81,63	71,84	95,93
		RANKING	88,64	81,64	91,45	78,41	92,18	94,77	91,24	80,55	89,21	93,98
	NUM PATTERNS		448	346	113	74	791	233	115	147	103	123

Tabella 5.8 Risultati FMLPGA 70-30

NETWORK TRAFFIC CLASSIFICATION - TEST BASED ON FMLPGA											
70% train -- 30% test			*execution time in seconds*								
V	CASE	FUNC	Br-Ml	Br-PP	Br-Sk	Br-TC	Ml-PP	Ml-Sk	Ml-TC	PP-Sk	PP-TC	Sk-TC
CPU	TRAIN	ROULV1	270	228	108	48	498	370	171	195	157	192
		ROULV2	296	231	72	47	507	359	175	191	155	194
		FITTING	266	201	109	47	562	399	177	193	145	184
		RANKING	299	234	111	51	562	401	178	200	161	198
	NUM PATTERNS		1044	806	261	172	1843	541	268	343	237	287
GPU	TRAIN	ROULV1	32	21	8	7	59	14	9	11	8	10
		ROULV2	32	21	8	7	60	20	9	11	9	9
		FITTING	30	19	7	6	59	14	7	9	7	8
		RANKING	34	24	9	8	63	21	11	12	11	12
	NUM PATTERNS		1044	806	261	172	1843	541	268	343	237	287

Tabella 5.9 Risultati tempoi FMLPGA 70-30

NETWORK TRAFFIC CLASSIFICATION - TEST BASED ON FMLPGA											
80% train -- 20% test			*classification accuracy percentages (average on two classes)*								
V	CASE	FUNC	Br-Ml	Br-PP	Br-Sk	Br-TC	Ml-PP	Ml-Sk	Ml-TC	PP-Sk	PP-TC	Sk-TC
CPU	TRAIN	ROULV1	91,78	85,02	91	86,8	93,36	95,8	93,14	81,38	91,85	96,34
		ROULV2	91,86	84,8	92,67	84,26	92,69	95,64	96,41	90,05	95,56	96,65
		FITTING	90,69	74,7	90,67	69,04	91,08	94,99	94,12	82,14	92,59	93,9
		RANKING	91,28	84,47	92	86,8	92,17	96,93	93,46	89,03	94,44	95,43
	NUM PATTERNS		1192	921	300	197	2107	619	306	392	270	328
	TEST	ROULV1	91,61	78,79	91,89	79,59	93,74	94,19	93,51	78,57	71,43	91,46
		ROULV2	91,95	81,82	90,54	75,51	93,36	93,55	92,21	86,73	87,14	92,68
		FITTING	89,6	71,43	91,89	73,47	90,89	90,32	93,51	79,59	72,86	92,68
		RANKING	89,93	79,65	91,89	81,63	91,46	93,55	92,21	85,71	81,43	91,46
	NUM PATTERNS		298	231	74	49	527	155	77	98	70	82
GPU	TRAIN	ROULV1	90,86	84,91	96,67	85,28	91,93	95,64	94,44	91,33	90	96,95
		ROULV2	91,36	85,78	92	87,31	91,69	96,77	94,44	84,18	88,15	96,34
		FITTING	90,69	84,36	94	83,25	92,22	95,64	92,81	90,56	90,74	95,73
		RANKING	92,11	84,78	94,65	86,87	91,11	95,12	92,87	88,56	89,54	91,84
	NUM PATTERNS		1192	921	300	197	2107	619	306	392	270	328
	TEST	ROULV1	90,27	79,65	97,3	81,63	92,6	92,9	94,81	93,88	71,43	91,46
		ROULV2	91,61	81,82	91,89	85,71	91,84	94,84	93,51	83,67	68,57	92,68
		FITTING	90,27	80,95	90,54	79,59	92,03	94,84	93,51	92,86	70	92,68
		RANKING	91,28	80,82	93,15	84,1	91,45	92,74	92,98	87,62	70,32	90,14
	NUM PATTERNS		298	231	74	49	527	155	77	98	70	82

Tabella 5.10 Risultati FMLPGA 80-20

NETWORK TRAFFIC CLASSIFICATION - TEST BASED ON FMLPGA											
80% train -- 20% test			*execution time in seconds*								
V	CASE	FUNC	Br-Ml	Br-PP	Br-Sk	Br-TC	Ml-PP	Ml-Sk	Ml-TC	PP-Sk	PP-TC	Sk-TC
CPU	TRAIN	ROULV1	319	276	191	137	605	418	194	227	175	208
		ROULV2	317	277	190	130	607	416	192	226	175	207
		FITTING	315	270	185	125	601	413	188	222	172	204
		RANKING	321	278	192	139	612	421	195	229	178	209
	NUM PATTERNS		1192	921	300	197	2107	619	306	392	270	328
GPU	TRAIN	ROULV1	37	23	10	7	73	17	10	11	9	10
		ROULV2	37	23	10	8	72	16	10	10	8	9
		FITTING	35	21	8	6	71	15	8	10	7	8
		RANKING	39	28	12	10	76	20	13	14	13	14
	NUM PATTERNS		1192	921	300	197	2107	619	306	392	270	328

Tabella 5.11 Risultati tempi FMLPGA 80-20

5.5.1 Analisi dei risultati

Come è possibile notare della tabelle 5.8 e 5.9 non è possibile distinguere una coppia di classi vera e propria che abbia ottenuto le percentuali di classificazione più alte rispetto alle altre come in GAME; in questo caso infatti, i risultati delle percentuali di classificazioni risultano più omogenei e meglio distribuiti tra loro. Inoltre è possibile notare che con i parametri utilizzati con questo modello di ML si ottengono risultati più soddisfacenti rispetto a GAME, raggiungendo risultati più che soddisfacenti per tutti i dataset .

Dal punto di vista dell'architettura di esecuzione, si può notare che le percentuali di classificazione risultano nella maggior parte dei casi comparabili sia nel caso CPU che GPU, mentre il tempo di esecuzione (come previsto), risulta fortemente ridotto nel caso GPU. Da quest'ultima osservazione possiamo dedurre che a classificazioni comparabili nel caso delle due architetture di esecuzione è preferibile scegliere un'esecuzione su GPU che su CPU, in modo da ottimizzare i tempi della fase di Train, che come abbiamo detto in precedenza, risultano essere fortemente lunghi.

5.6 Test quantitativi su FMLPGA

Anche per FMLPGA sono stati eseguiti dei testi quantitativi per analizzarne il tempo di train. Analogamente a GAME anche qui si è scelto di studiare l'andamento della tempistica utilizzando la funzione di selezione Fitting e dividendo il dataset relativo alle classi Mail-P2P nelle percentuali 25%, 50%, 75% e 100%.I risultati , come atteso, mostrano un andamento lineare. Per poter apprezzar meglio i risultati del caso GPU il grafico sottostante riporta i valori temporali in secondi.

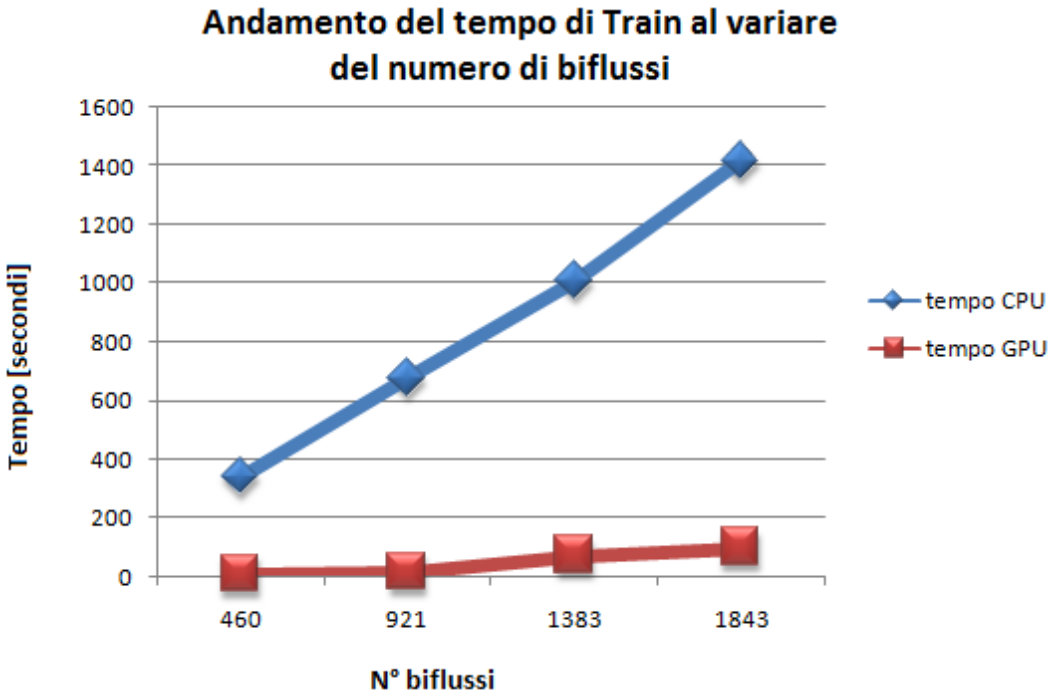


Figura 5.7 Andamento temporale GPU vs CPU

5.6.1 Calcolo dello Speed Up

Per poter apprezzare meglio il guadagno in termini temporali nell'utilizzare la versione GPU dell'algoritmo, rispetto alla versione CPU, si è proceduto al calcolo dello Speed Up. Con il termine Speed Up si intende a quanto un algoritmo parallelo è più veloce di un corrispondente algoritmo sequenziale..

	N° biflussi			
	460	921	1383	1843
tempo CPU[s]	341	679	1008	1418
tempo GPU [s]	10	17	74	100

Tabella 5.12 Tabella dei tempi

Per poter calcolare , quindi, questo parametro bisogna effettuare il rapporto tra i valori temporali ottenuti per la CPU e quelli ottenuti per la GPU , sommare tutti i valori ottenuti e dividere per quanti ne sono ; facendo questi calcoli si ottiene uno Speed Up di circa 25x. In

pratica ciò significa che, a parità di setup dell'esperimento, la fase di training eseguita su GPU impiega 25 volte meno rispetto al tempo impiegato su CPU.

Conclusioni e sviluppi futuri

In questo elaborato di Tesi è stato mostrato come sia possibile utilizzare tecniche di Machine Learning nel contesto della classificazione del traffico di rete. Sono state analizzate varie classi di traffico in modo da capire quali siano le più adatte ad un'analisi di questo tipo e quali meno.Inoltre sono stati proposti due approcci con due tecniche basate sul Machine Learning, nella fattispecie un modello ibrido FMLPGA (Fast Multi Layer Perceptron addestrato con un Algoritmo Genetico) ed un algoritmo genetico puro (GAME, Genetic Algorithm Modeling Experiment), in modo da avere una visione più ampia del problema ed analizzando i trend di variazione delle prestazioni utilizzando algoritmi differenti. E' stata, poi, svolta un'analisi volta a studiare i tempi di esecuzione degli algoritmi,. Avendo inoltre a disposizione la doppia implementazione su CPU e GPU del modello FMLPGA, sono stati mostrati non solo i risultati di classificazione ma anche quelli relativi al confronto del tempo di esecuzione (speed up).

Sulla base dei risultati descritti si evince come effettivamente il Machine Learning possa essere soddisfacentemente utilizzato per la classificazione del traffico di rete, ottenendo prestazioni comparabili a tecniche tradizionali, non basate su metodi empirici. Il vantaggio di tali metodologie sta proprio nel rendersi indipendenti da analisi approfondite o da modelli teorici per la caratterizzazione delle componenti caratterizzanti il traffico di rete, delegando al modello empirico il compito di individuare e discriminare le principali peculiarità delle varie applicazioni sul canale.Inoltre utilizzando una tale metodologia, una volta eseguita off-line la fase di Training particolarmente time consuming , le fasi vere e proprie di utilizzo risultano molto veloci, alla stregua dell'esecuzione di una normale funzione analitica.

Sviluppi futuri di questa tematica saranno: lo studio dell'implementazione di GAME per la classificazione del traffico di rete su GPU , la sperimentazione di altri algoritmi di ML applicati alla classificazione del traffico di rete, lo studio di altre caratteristiche di input da

dare agli algoritmi, l'analisi degli algoritmi descritti in questo testo applicati ad altre tracce di traffico,nonchè lo studio di altre classi target, oltre quelle proposte in questo elaborato.

La vastità dell'argomento propone innumerevoli spunti da poter approfondire nei prossimi anni.

Appendice A:Come creare un dataset di input con un foglio elettronico

Una traccia di flusso è un insieme di righe, dove ogni riga è un flusso. Nel nostro caso, più precisamente, si parla di biflusso. La differenza tra *flusso* e *biflusso* sta nel fatto che il flusso riserva una riga alla comunicazione unidirezionale tra ip sorgente a ip destinatario e un'altra alla comunicazione inversa, sempre unidirezionale tra ip destinatario a ip sorgente.Il biflusso invece riserva un'unica riga ad una comunicazione bidirezionale ip sorgente/ip destinatario.
Come detto nel capitolo 5, dal file pcap,fornito dall'università di Brescia, sono stati estratti i file di *GroundTruth* e il file *Bifluss.,* Supponendo che le relative anomalie presenti nei due file siano già state corrette, si può procedere alla creazione di un dataset generico contenente tutte le applicazioni target nel modo di seguito illustrato.

Per creare un dataset di input la prima cosa da fare è decidere le caratteristiche e i target. In questo elaborato di Tesi sono stati scelti i seguenti parametri :
→**Caratteristiche** : Time Elapsed , Bytes , unPackets , downPackets (contenutenel file Biflussi);
→ **Targets :** Applicazioni (contenute nel file GroundTruth).

Essendo il numero di biflussi contenuti nel file di Biflussi (23468) non coincidente con quello contenuto nel file di GroundTruth (21917), bisogna verificare quali biflussi siano presenti in entrambi i file in modo da poter relazionare le caratteristiche al target.

I vari passi da seguire per la creazione del DATA SET di INPUT, allora, sono :
- Identificazione dei biflussi contenuti in entrambi i file;
- Eliminazione delle righe senza corrispondenza;
- Calcolo per l'associazione caratteristiche → target;
- Conversione formato;
- Creazione del file;

•Salvataggio.

Per lo svolgimento dei punti appena citati è stato utilizzato LibreOfficeCalc.Per aprire i file relativi alla groundtruth e ai biflussi è necessario salvarli con l'estensione CSV. Questo passaggio ci permette di poter aprire un file tratto da un DataBase in un foglio elettronico. Una volta effettuata questa operazione è possibile aprire i due file con Office Calc. Dopo aver cliccato, quindi, sul file da aprire, il programma ci chiederà di identificare il carattere separatore dei campi che nel nostro caso è ":" in modo da assegnare ogni campo ad una colonna.

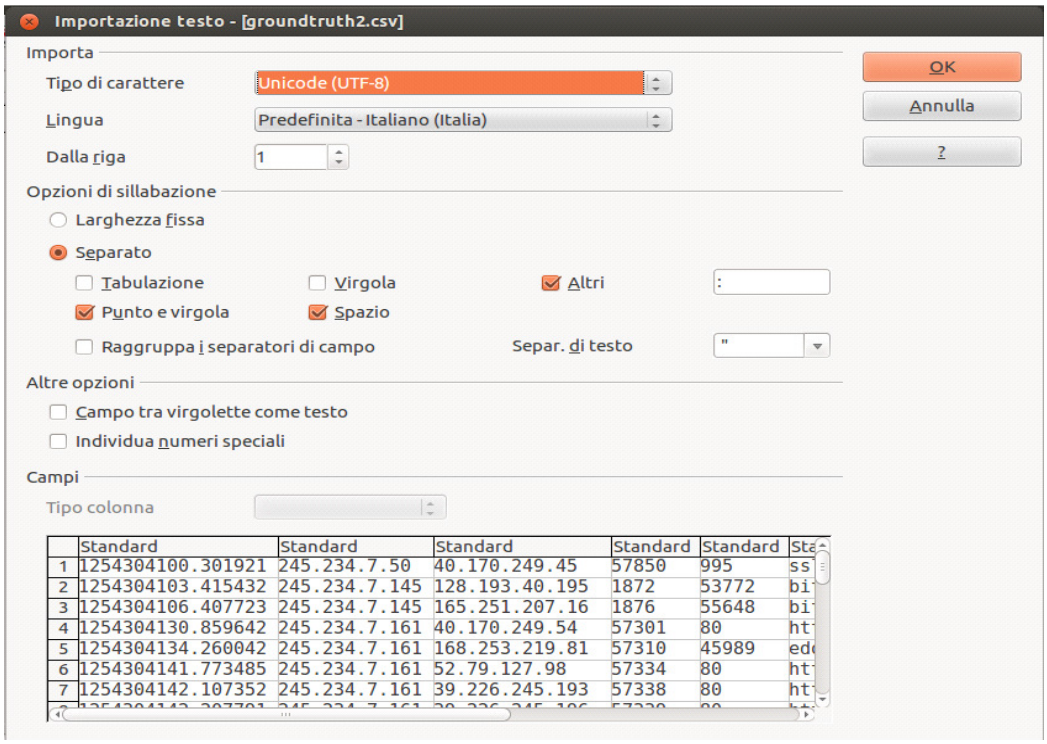


Figura AppendiceA.1 Esempio apertura di un file CSV

Eseguita questa operazione è possibile visualizzare i due file aperti in due fogli di calcolo. Una volta aperti, è opportuno ordinare i dati in modo crescente rispetto al Time Stamp nel seguente modo: selezionare tutte le colonne relative ai dati → menù dati → ordina → rispetto la colonna contenente i Time Stamp (ad esempio A) → crescente.

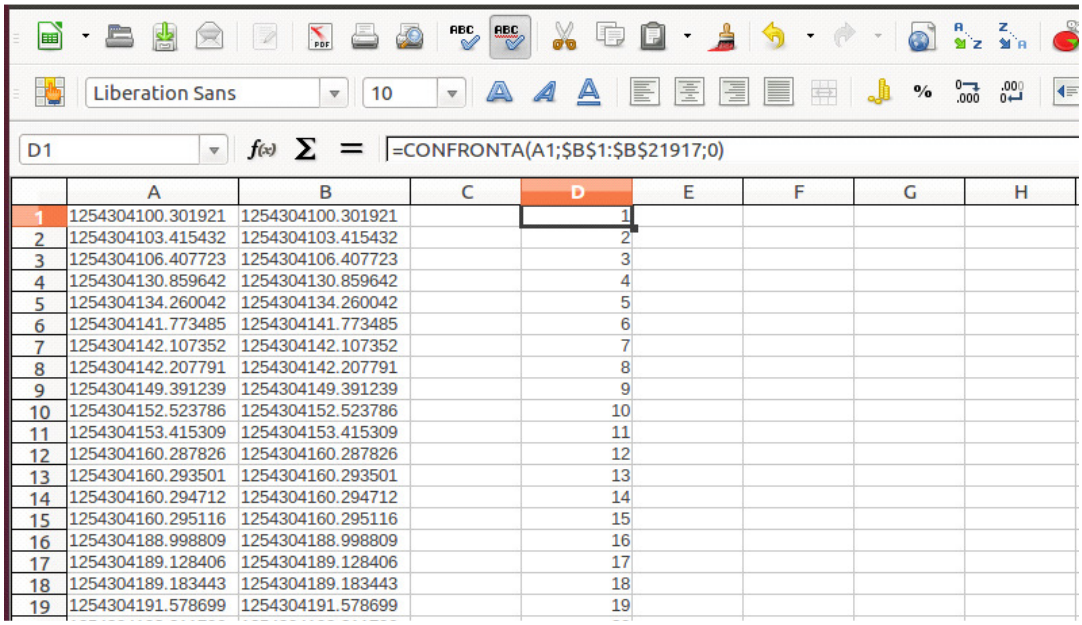
Identificazione dei biflussi contenuti in entrambi i file

Come detto in precedenza il numero dei biflussi nei due file non coincide. Occorre quindi individuare quali di essi siano contenuti in entrambi. Per far ciò, aperto un nuovo foglio di calcolo, si copia nella prima colonna, identificata come A, la colonna relativa ai Time Stamp del file biflussi, mentre nella seconda colonna si copia la colonna relativa ai Time stamp del file GroundTruth. Bisogna, a questo punto, individuare quali Time Stamp della colonna A siano contenuti nella colonna B. A questo proposito si è pensato di utilizzare la funzione CONFRONTA, la cui sintassi è :

f= CONFRONTA(valore da cercare ; intervallo di ricerca ; 0)

- valore da cercare : è il valore che si vuole trovare; dovendo variare riga dopo riga, il suo valore sarà un riferimento semplice in modo tale che "trascinando la formula " nelle righe successive, cambi di volta in volta il numero della riga mentre la lettera rimarrà costante . Es. A1;
- intervallo di ricerca : è lo spazio di celle in cui la funzione eseguirà la ricerca. Sapendo già quale sia lo spazio in cui bisogna ricercare i Time Stamp e non volendo che cambi in base al Time Stamp da ricercare, questo parametro sarà composto da due riferimenti assoluti (non cambieranno, quindi, al trascinare della formula su altre celle), uno identificante l'inizio dell'intervallo e l'altro la fine. Es. $B$1:$B$21917;
- 0 : questo parametro rappresenta la scelta dell'esecuzione di una ricerca precisa ovvero, nel caso in cui la ricerca non trovi nessuna corrispondenza, la funzione restituirà il valore #N/D.

La funzione CONFRONTA restituirà come output il numero della riga della colonna B in cui sia contenuto il Time Stamp corrispondente a un biflusso della colonna A, mentre restituirà il valore #N/D se non abbia trovato alcuna corrispondenza .



	A	B	C	D	E	F	G	H
1	1254304100.301921	1254304100.301921		1				
2	1254304103.415432	1254304103.415432		2				
3	1254304106.407723	1254304106.407723		3				
4	1254304130.859642	1254304130.859642		4				
5	1254304134.260042	1254304134.260042		5				
6	1254304141.773485	1254304141.773485		6				
7	1254304142.107352	1254304142.107352		7				
8	1254304142.207791	1254304142.207791		8				
9	1254304149.391239	1254304149.391239		9				
10	1254304152.523786	1254304152.523786		10				
11	1254304153.415309	1254304153.415309		11				
12	1254304160.287826	1254304160.287826		12				
13	1254304160.293501	1254304160.293501		13				
14	1254304160.294712	1254304160.294712		14				
15	1254304160.295116	1254304160.295116		15				
16	1254304188.998809	1254304188.998809		16				
17	1254304189.128406	1254304189.128406		17				
18	1254304189.183443	1254304189.183443		18				
19	1254304191.578699	1254304191.578699		19				

Figura AppendiceA.2 Esempio funzione confronta

Eliminazione delle righe senza corrispondenza

Trovati i biflussi con corrispondenza, bisogna eliminare quelli contrassegnati dal valore #N/D. Si apre, allora, un nuovo foglio di calcolo e si incollano nelle prime quattro colonne, le colonne relative alle caratteristiche scelte dal file biflussi. Accanto ad esse, poi, si incolla la colonna D ottenuta al punto precedente. Fatto ciò si seleziona quest'ultima colonna → menù Dati → filtro → filtro standard → scegliere la colonna per il filtraggio; essa sarà la colonna (D del punto precedente), contenente le indicazioni sulle righe del file ground truth → condizione <> (diverso) → parametro del filtraggio #N/D.

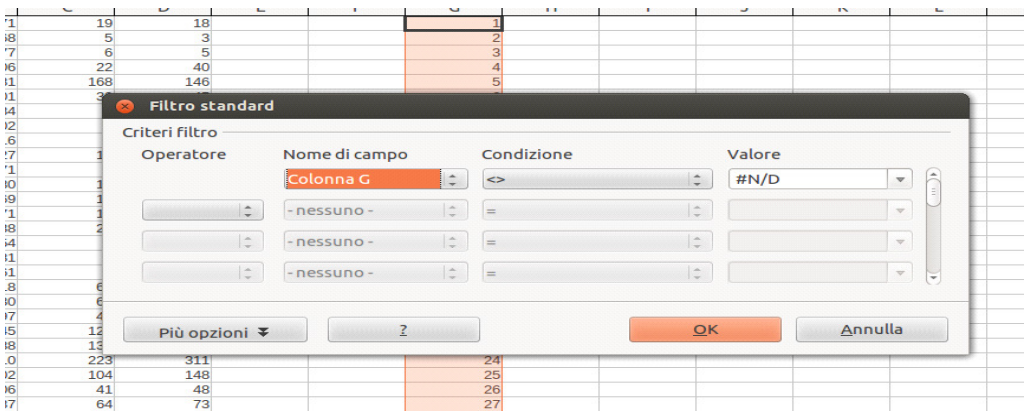


Figura AppendiceA.3 Esempio filtro

Questa operazione permette di eliminare le righe contrassegnate dal parametro #N/D.

Calcolo associazione

Ottenute tutte e sole le righe con corrispondenza dei due file, bisogna capire come relazionare il numero di riga ottenuto al punto 1) al corrispondente valore contenuto nella colonna Applicazione del file ground truth. Aperto un nuovo foglio elettronico, quindi, si incolla nella prima colonna, quella relativa al campo applicazioni della Ground truth. Successivamente riportiamo in questo file anche la colonna relativa alle righe, opportunamente filtrata, ottenuta al punto precedente. Da questa operazione preliminare si nota che le due colonne hanno lunghezza diversa: ovvero la colonna relativa ai numeri di riga risulta più corta rispetto a quella delle applicazioni (21917). Questo è dovuto al fatto che anche alcuni Time Stamp contenuti nel file ground truth non hanno trovato nessuna corrispondenza nel file biflussi .
Fatto ciò si utilizzano due funzioni : INDIRIZZO e INDIRETTO. Esse sono state utilizzate rispettivamente per ottenere gli indirizzi di riferimento delle celle, composti dal numero di riga ottenuto al punto uno e dalla colonna applicazioni del ground truth, per poter utilizzare come riferimento l'indirizzo così ottenuto. La sintassi di queste due funzioni è la seguente:

f=INDIRIZZO(riga ; numero colonna)

f=INDIRETTO(cella)

Analizziamo la funzione INDIRIZZO. Nel parametro riferito alla riga si inserisce il valore contenuto nella colonna contente i riferimenti di riga filtrata; mentre nel campo numero di colonna si inserisce il numero 1, poiché la colonna in cui sono state inserite le applicazioni è la prima da sinistra. Questa funzione restituirà un rifermento assoluto del tipo $F$2 che è a tutti gli effetti un indirizzo di cella in un foglio di calcolo. Si utilizza, quindi, la funzione INDIRIZZO per relazionare li Time Stamp (che hanno una relazione in entrambi i file) con la relativa applicazione contenuta nel file Groud Truth. In pratica l'indirizzo ottenuto prende per ogni Time Stamp del file biflussi (su cui si è effettuato prima il filtraggio) l'informazione

sul numero di riga corrispondente nell'altro file (Ground Truth) in modo da restituire l'indirizzo della cella contenente l'applicazione.

Per rendere effettivamente utilizzabile l'indirizzo restituito dalla funzione INDIRIZZO si utilizza la funzione INDIRETTO la quale usa il risultato della funzione INDIRIZZO come vero e proprio indirizzo di cella e ne restituisce, infine, il contenuto di tale cella (nel nostro caso l'applicazione). Nel campo della funzione INDIRETTO, quindi, inseriremo il risultato della funzione precedente .

	A	B	C	D	E	F
1	Mail		1		$A$1	Mail
2	bittorrent.exe		2		$A$2	bittorrent.exe
3	bittorrent.exe		3		$A$3	bittorrent.exe
4	firefox-bin		4		$A$4	firefox-bin
5	Skype		5		$A$5	Skype
6	firefox-bin		6		$A$6	firefox-bin
7	firefox-bin		7		$A$7	firefox-bin
8	firefox-bin		8		$A$8	firefox-bin
9	bittorrent.exe		9		$A$9	bittorrent.exe
10	firefox-bin		10		$A$10	firefox-bin
11	firefox-bin		11		$A$11	firefox-bin
12	Mail		12		$A$12	Mail
13	Mail		13		$A$13	Mail
14	Mail		14		$A$14	Mail
15	Mail		15		$A$15	Mail
16	firefox-bin		16		$A$16	firefox-bin
17	firefox-bin		17		$A$17	firefox-bin
18	firefox-bin		18		$A$18	firefox-bin
19	firefox-bin		19		$A$19	firefox-bin
20	firefox-bin		20		$A$20	firefox-bin
21	firefox-bin		21		$A$21	firefox-bin
22	firefox-bin		22		$A$22	firefox-bin
23	firefox-bin		23		$A$23	firefox-bin
24	firefox-bin		24		$A$24	firefox-bin
25	firefox-bin		25		$A$25	firefox-bin
26	firefox-bin		26		$A$26	firefox-bin
27	firefox-bin		27		$A$27	firefox-bin
28	firefox-bin		28		$A$28	firefox-bin

Figura AppendiceA.4 Esempio funzione indiretto

Nella figura *Appendice1.4* sono riportate nella colonna A le applicazioni della Ground Truth, nella colonna C le informazioni relative alle associazioni, nella colonna E i risultati della funzione INDIRIZZO e nella colonna F quelli della funzione INDIRETTO. A questo punto bisogna convertire in formato numerico le applicazioni. Si copia e si incolla, allora, la colonna F in una nuova colonna e si effettuano continue operazioni di sostituzione nome/numero come riportato nella tabella4.1 del capitolo 4.

Fatto ciò si apre un nuovo foglio elettronico e si copiano le colonne delle caratteristiche del file Biflusso (filtrate dalle righe senza corrispondenza), nelle prime quattro colonne; mentre nella quinta si incolla la colonna delle applicazioni “numeriche “.

Conversione nel formato di GAME/FMLPGA

Per poter sottomettere un dataset di input in GAME o in FMLPGA, però, ogni applicazione deve essere codificata solo con il valore1(appartiene a quel target) oppure 0(non appartiene al target). Nell'ultimo foglio di calcolo creatosi inseriscono N colonne (dove N è dato dal numero di applicazioni Targets) contenenti degli IF, i quali, restituiranno 1 solo se il numero presente nella colonna applicazioni numeriche appartiene alla applicazione desiderata , altrimenti restituiscono 0.

I2 =SE(E2=4;1;0)

A	B	C	D	E	F	G	H	I	J	K	L	M
1.08194	2371	19	18	9	0	0	0	0	0	0	0	0
1.10498	68	5	3	4	0	0	0	1	0	0	0	0
2.60271	1377	6	5	4	0	0	0	1	0	0	0	0
166.92	51706	22	40	1	1	0	0	0	0	0	0	0
1836.35	15081	168	146	3	0	0	1	0	0	0	0	0
114994	60801	30	45	1	1	0	0	0	0	0	0	0
15784	2384	6	6	1	1	0	0	0	0	0	0	0
15.6835	3102	7	6	1	1	0	0	0	0	0	0	0
3.50259	716	7	6	4	0	0	0	1	0	0	0	0
133662	2627	10	6	1	1	0	0	0	0	0	0	0
5.44287	771	7	5	1	1	0	0	0	0	0	0	0
0.363456	2380	19	18	9	0	0	0	0	0	0	0	0
0.444104	269	17	12	7	0	0	0	0	0	0	1	0
1.52776	2371	19	19	9	0	0	0	0	0	0	0	0
0.739326	2538	23	20	9	0	0	0	0	0	0	0	0
241867	1254	6	5	1	1	0	0	0	0	0	0	0
241736	1381	6	5	1	1	0	0	0	0	0	0	0

Figura AppendiceA.5 Esempio dataset

A questo punto si termina la creazione del dataset selezionando tutte le colonne e incollandole in un nuovo foglio elettronico (operazione effettuata con incolla speciale -> incolla tutto tranne formule). Si prosegue con l'eliminazione della colonna "E" contenente le "applicazioni numeriche " e con il salvataggio del foglio attivo in formato CSV. L'ultimo passo consiste nell'aprire il file appena creato con un editor di testo e sostituire al parametro separatore dei campi (ad esempio ";") uno spazio, ed alle eventuali virgole il punto . Il dataset è pronto per essere usato.

Appendice B:GPU Computing

GPGPU è un acronimo che sta per General Purpose Computing on Graphics Processing Units, inventato da Mark Harris nel 2002[2]. Inizialmente i processori grafici avevano un'unica funzione fissa riguardante le applicazioni grafiche. Oggigiorno, invece, sono strumenti complessi e sofisticati sempre più performanti dal punto di vista della capacità di calcolo e della programmabilità. Negli ultimi anni, un tipico utilizzo delle GPU è come co-processore di CPU nell'elaborazione dei dati; in tal senso la GPU viene utilizzata per applicazioni che richiedono un'elevata potenza elaborativa. Questo è possibile dato che la natura intrinseca dei processori grafici è una struttura many-core che permette di ottenere, con molti algoritmi, prestazioni notevolmente superiori rispetto a quelle ottenute con una CPU multi-core, raggiungendo , quindi, un'elaborazione parallela anziché in serie. Grazie alla ricerca in vari campi applicativi, il GPU computing è in grado di ridurre notevolmente i tempi di esecuzione di un'applicazione scientifica .

Passaggio da un'elaborazione sequenziale ad una parallela

Nel 1965 Gordon Moore propose una legge empirica nota come "Moore's law" la quale affermava che il numero di transistor in un microprocessore sarebbe raddoppiato ogni 18 - 24 mesi. Successivamente negli anni novanta tale legge è stata utilizzata per descrivere l'aumentare della velocità e della potenza in una CPU. A una maggiore produzione di CPU, inoltre, è seguita una quasi lineare diminuzione dei costi. Questa rapida crescita di prestazioni ha comportato, da un punto di vista degli utenti, una crescente richiesta di elevate prestazioni computazionali e affidabili applicazioni multi-tasking, mentre dal punto di vista degli sviluppatori, una richiesta di un miglioramento delle prestazioni hardware per ottimizzare la velocità di applicazione. Il trade off di questa situazione è dato dal fatto che i vincoli fisici imposti dalla termodinamica iniziano a causare rilevanti problemi in termini energetici e di energia dissipata all'interno del microprocessore. Per controllare gli effetti termici si riducono, quindi, le frequenze di clock per distribuire i carichi di lavoro su unità di

elaborazione diverse disposte sullo stesso chip. Si passa quindi da un approccio progettuale sequenziale ad uno parallelo.

Il motivo per cui le GPU permettono un sostanziale miglioramento delle prestazioni è dato dal fatto che la loro natura è intrinsecamente parallela; infatti mentre le CPU hanno una struttura composta da uno o più cores, i processori grafici sono composti da centinaia o addirittura migliaia di multi-processors/core. Il problema di questa situazione è dovuto al fatto che la maggior parte degli algoritmi sono realizzati per poter lavorare su CPU e quindi hanno una struttura di tipo seriale, ovvero tali algoritmi sono visti come uno stream sequenziale di istruzioni che sono svolte una per volta dal microprocessore. Solo quando si completa l'esecuzione di un'istruzione la CPU passa a quella successiva.

Una struttura parallela, invece, necessita di algoritmi paralleli. Purtroppo non tutti gli algoritmi sono parallelizzabili. Un tale tipo di elaborazione vede un algoritmo come uno stream di processi paralleli. Per ottenere questo risultato l'architettura di un processore grafico divide un problema in tanti sotto-problemi più piccoli, ognuno dei quali è eseguito parallelamente e indipendentemente dagli altri. Infine i vari risultati vengono ricombinati insieme.

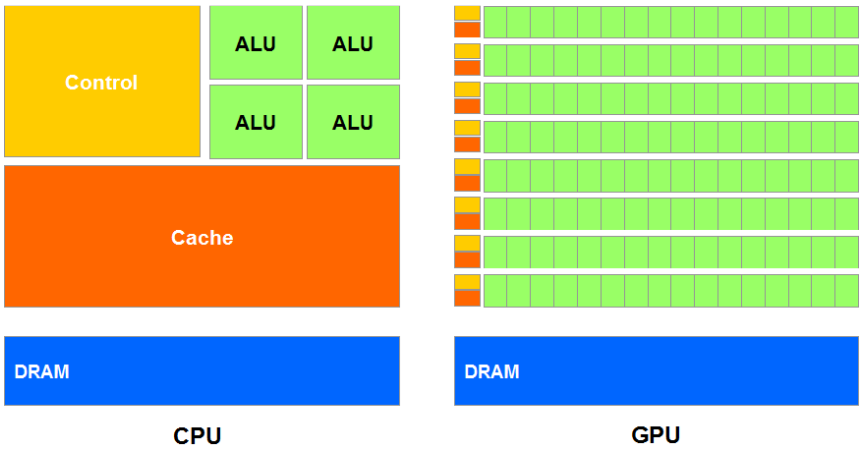


Figura AppendiceB.1 [2]

Molte aziende tra cui ATI Technologies Inc. (acquisita da AMD nel 2006) e NVIDIA Corp. hanno scelto di seguire la strada many-core . Alla base del concetto many-core è posto il miglioramento della velocità di esecuzione delle applicazioni parallele. Essi inizialmente

erano composti da circa una decina di nuclei più piccoli di una CPU fino a raggiungere dimensioni di centinaia di nuclei in pochi anni.

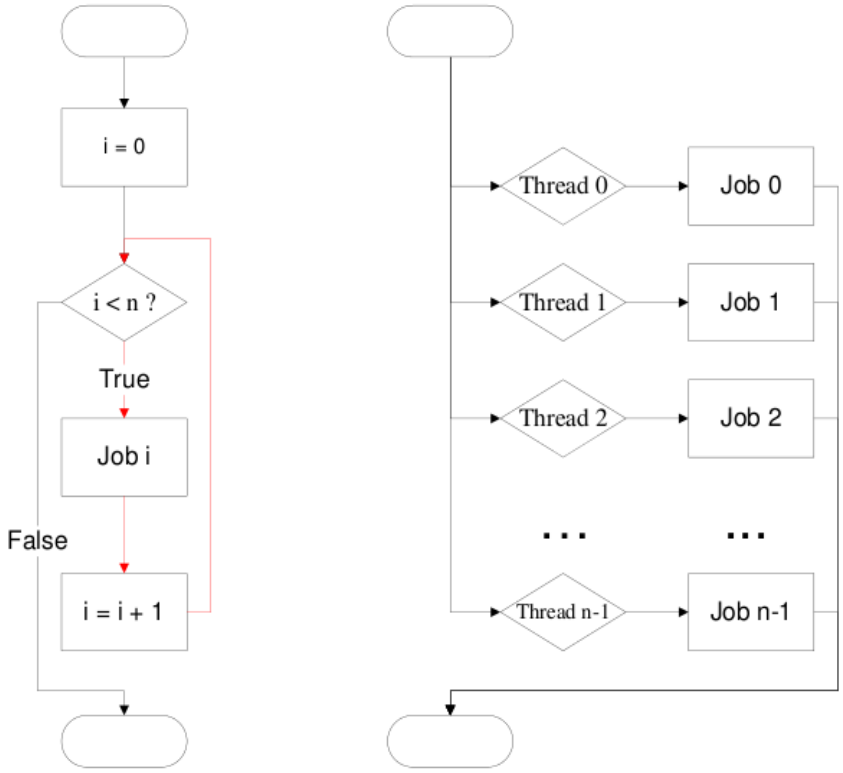


Figura AppendiceB.2 [2] Esecuzione seriale vs parallela

Per studiare le prestazioni di un codice quando si cerca di passare da una struttura seriale a una parallela si usa la **legge di Amdahl**:

$$S = \frac{1}{1-F+\frac{F}{N}} \qquad (2.1)$$

Dove N è il numero di thread paralleli disponibili mentre F è la percentuale del tempo di esecuzione che può essere parallelizzata. Questa legge, quindi, pone un limite superiore alle prestazioni di un codice in cui una frazione deve essere eseguita necessariamente in modo seriale.

Campi di impiego del GPU computing

Oggigiorno sono molteplici i campi di utilizzo di una tale tecnologia; essi in genere sono caratterizzati dall'impiego di massicce quantità di dati da analizzare. Di seguito sono riportati alcuni esempi :

Campo astrofisico : per verificare che i codici astrofisici portino a un effettivo miglioramento, se eseguiti su un'architettura GPU, si dovrebbe effettuare un'analisi costi-benefici. Usato nella giusta maniera o nell'applicazione giusta, la GPU può essere considerata un potente strumento per i processi astronomici che richiedono enormi volumi di dati . In questo contesto è stato sviluppato un algoritmo genetico multi-purpose GAME implementato su tecnologia GPGPU/CUDA, nonché un modello ibrido FMLPGA.

Classificazione del traffico di rete : in questo campo le GPU sono usate per garantire alta velocità nei metodi di aggregazione a supporto dei criteri di classificazione. Ad esempio il metodo di classificazione del traffico basato sul DeepPacketInspection (DPI) è implementabile su GPU mentre una semplice classificazione port-based è applicabile su GPU soltanto dopo un processo DPI .

Programmazione in parallelo: CUDA

Le GPU sono strumenti programmabili e una delle più famose piattaforme di elaborazione in parallelo è CUDA. Esso permette ai programmatori la parallelizzazione dei task scegliendo tra linguaggi di programmazione ad alto livello come C , C++ , Fortran oppure in standard aperti come OpenAcc.

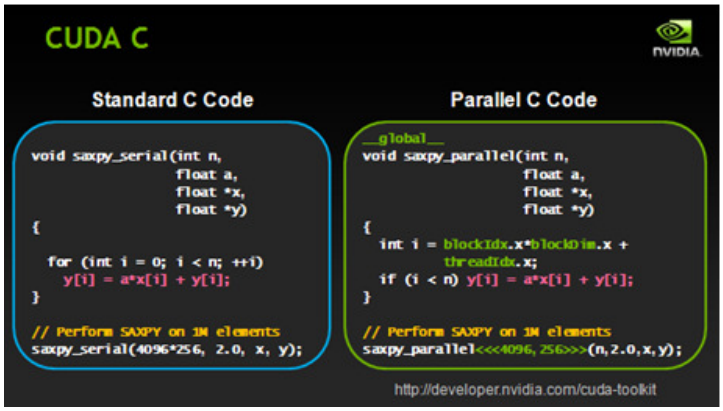


Figura AppendiceB.3 (http://developer.nvidia.com/cuda-toolkit)

Introdotta da NVIDIA nel novembre 2006, CUDA è un'architettura di elaborazione parallela che sfrutta le potenzialità delle GPU NVIDIA. Essa permette l'implementazione di software tramite linguaggi di alto livello come C/C++ e inoltre include l'Istruzione CUDA Set Architecture (ISA) e il motore di calcolo parallelo nella GPU. L'architettura CUDA ,inoltre, offre i seguenti vantaggi :

- Buona scalabilità;
- Supporto di elaborazioni parallele. Questo è molto importante quando in un codice siano presenti parti non parallelizzabili. Allora è più naturale che la CPU svolga queste istruzioni mentre la GPU si concentri sul codice parallelizzabile;
- Fornisce API (Application Programming Interface) sia di basso che di alto livello;
- Presenta numerose librerie di funzioni;
- Fornisce estensioni all'ambiente C;

CUDA : Modello Architetturale

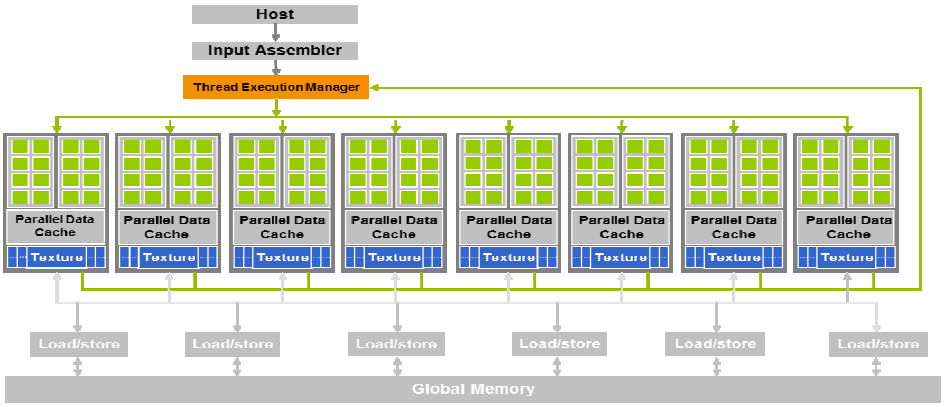


Figura AppendiceB.4[2] Modello architetturale GPU

In un'architettura CUDA la GPU è connessa al sistema ed alla CPU tramite un bus PCI. Essa è costituita, come possiamo vedere dalla figura appendiceB.4, da numerosi multiprocessori in threading elevato detti Streaming Multi Processor (SMs); essi, a loro volta, sono costituiti da 8 Stream Processors (SPs), ciascuno dei quali può eseguire un'operazione aritmetica fondamentale come addizione, sottrazione ecc,[6], su numeri interi o a virgola mobile a singola precisione a 32 bit (le odierne GPU di tipo Kepler garantiscono anche la doppia precisione in floating point). Si noti che mentre il numero di SMs può cambiare a seconda del tipo di GPU utilizzata, il numero degli SPs è fisso indipendentemente dal tipo di architettura parallela in uso. Ogni Stream Processor è sostituito da un unità multiply–add (MAD) e un'unità moltiplicativa addizionale. Inoltre ognuno dei multiprocessori è dotato di due unità per operazioni speciali che permettono di svolgere operazioni in virgola mobile come la radice quadrata.

Oggi ogni GPU viene fornita di: una memoria globale, accessibile da tutti i componenti, pari a 4 GB di Graphics Double Data Rate (GDDR) DRAM, della cache per le istruzioni e per i dati, e infine un blocco di decodifica delle istruzioni. Si noti, inoltre, che in CUDA l'host (CPU) e i dispositivi hanno memorie separate e che ogni thread ha una propria memoria locale privata. Ogni blocco di thread ha una memoria condivisa visibile da tutti i thread nel blocco. Tutti i thread hanno accesso alla stessa memoria globale.

Mentre la GPU (device) può accedere alla sua global memory (memoria privata),l'host ha solo la possibilità di spostare i dati in questa memoria. Inoltre ciascun multiprocessore può poi accedere alla propria shared memory, che invece non può essere gestita dall'host. I dati contenuti in shared memory hanno una visibilità limitata al singolo blocco di thread CUDA.

Nvidia propone per CUDA un modello architetturale di tipo SIMT ovvero Single Instruction Multiple Thread, poiché le stesse funzioni vengono eseguite da thread diversi. Le istruzioni vengono decodificate ogni 4 cicli di clock mentre ciascun processore può eseguire un'istruzione ad ogni ciclo di clock.

Modello di programmazione parallela seguito da CUDA

Con l'avvento delle GPU ci si è posti il problema di sviluppare applicazioni software che potessero sfruttare le potenzialità di un'elaborazione parallela: questo è il principio che NVIDIA ha posto alla base dell'architettura CUDA .

Tre sono le astrazioni chiave di CUDA[6]: la gestione gerarchica dei gruppi di thread, la memoria condivisa e la barriera di sincronizzazione. Esse ci forniscono una buona parallelizzazione dei dati e dei thread; inoltre aiutano il programmatore a suddividere il problema in sotto problemi più piccoli che posso essere eseguiti in modo parallelo dai thread, inoltre anche i sotto problemi posso essere ulteriormente suddivisi in pezzi più piccoli. Questo modo di operare garantisce un'intrinseca scalabilità del modello.

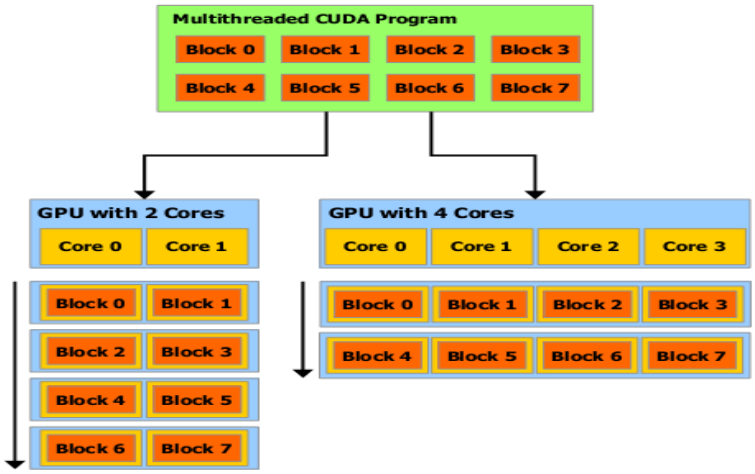


Figura AppendiceB.5[2] Modello esecutivo

Un'applicazione CUDA è composta da parti seriali eseguite dalla CPU denominata **host**, e da parti parallele dette **kernel,** che vengono invece eseguite dalla GPU detta **device**.

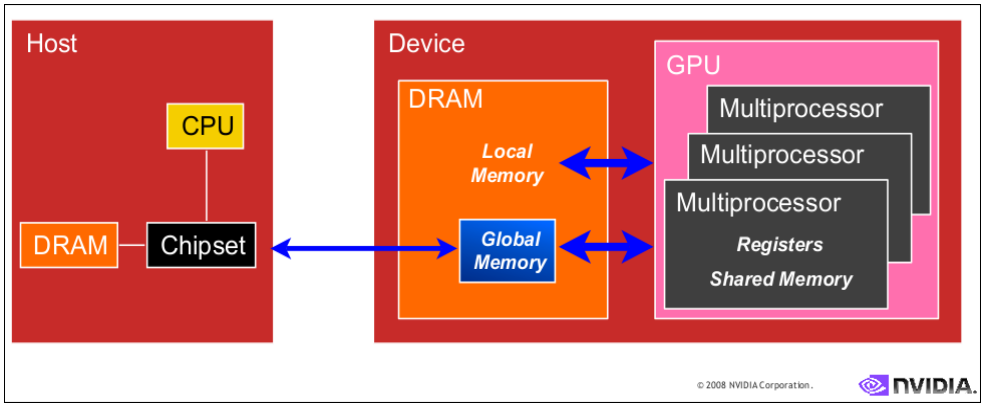


Figura AppendiceB.6 (www.gpgpu.it/joomla/cuda-tutorial/tutorial-cuda-prima-parte.html)

Oltre a gestire le parti seriali del codice la CPU è responsabile della gestione e dell'allocazione delle istruzioni e dei dati.

Ogni qualvolta un dato debba essere processato, esso viene prima allocato nella memoria dell'Host e successivamente viene copiato nel Device. Dopo la fase di processing, invece, il risultato segue il percorso invero dalla memoria del Device a quella dell'Host.

Quando deve essere processata la parte parallelizzabile, ovvero il Kernel, esso viene eseguito da un array di threads (ognuno identificato da un proprio ID). Quando una GPU si trova ad eseguire un kernel, un gran numero di thread sono generati per implementare l'elaborazione parallela; essi sono vengono chiamati collettivamente **griglia.** Quest'ultima a sua volta può essere suddivisa in più blocchi e ogni blocco può essere, poi, assegnato a un distinto multiprocessore. Un thread può, quindi, appartenere ad un unico blocco e il suo ID diventa univoco in tutto il kernel.

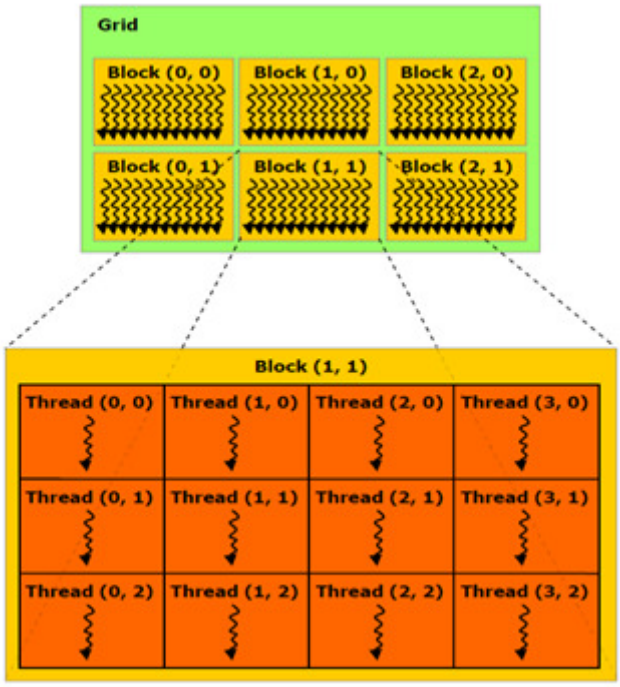


Figura AppendiceB.7 [2] Esempio CUDA GRID

Mentre le griglie sono eseguite sequenzialmente tra loro, i blocchi e i thread sono eseguiti in parallelo. Il numero di thread in esecuzione dipende dalla loro organizzazione in blocchi e dalle loro richieste in termini di risorse rispetto alle risorse disponibili nella GPU. Quando

finisce l'esecuzione di tutti i thread e quindi si è conclusa l'esecuzione di una griglia, l'elaborazione va avanti nell'host mentre nel device viene eseguita la griglia successiva.

Bibliografia

1. **Dangarra, Prof Jack.***Innovativativa computing laboratory (www.nvidia.it/object/GPU-computing-it.html).* University of Tennesee : s.n.

2. **Garofalo, Mauro Università degli studi di Napoli Federico II facoltà di ingegneria informatica.**Tesi in : *GPU Computing for Machine Learning Algorithms.*

3. **Aananthakrishnan, Sriram.***Parallel Programming for GPUs .* University of Utah : s.n.

4. **Wu, Haizhen.***Parallel Computing using GPUs.*

5. **Feamster, Nick.***Internet Relativism and the hunt for elasive Ground truth.*

6. *www.gpgpu.it/joomla/cuda-tutorial/tutorial-cuda-prima-parte.html.*

7. **Murat Soysal, Ece Guran Schmidt.***Machine learning algorithms for accurate flow-based network traffic.* s.l. : Performance Evaluation, 2010.

8. **Alberto Dainotti and Antonio Pescapé, University of Napoli Federico II ; Kimberly C. Claffy, University of California San Diego.***Issues and Future Directions in Traffic Classification.* s.l. : IEEE Network, 2012. 0890-8044/12/$25.00.

9. **G´eza Szab´o, Istv´an G´odor, Andr´as Veres, Szabolcs Malomsoky, S´andor Moln´ar.***Traffic Classification over Gbit Speed with Commodity Hardware.*

10. **T.T.T.Nguyen, G.Armitage.***A Survey of Techniques for Internet Traffic Classification using Machine Learning.* Swinburne University of Technology Melbourne, Australia : IEEE Communications Surveys and Tutorials, 2008.

11. **Brescia, Massimo.***NEW TRENDS IN E-SCIENCE:MACHINE LEARNING AND KNOWLEDGE DISCOVERY IN DATABASES.* National Institute for Astrophysics, Astronomical Observatory of Capodimonte , Naples : Chapter 1679 of Volume “Horizons in Computer Science Research”, NOVA, 2012.

12. **S. Cavuoti, M. Brescia, G. Longo.***Data mining and Knowledge Discovery Resources for Astronomy in the Web 2.0 Age.*

13. **Brescia, M., Cavuoti, S., Paolillo, M., Longo, G., Puzia, T.***The Detection of Globular Clusters in Galaxies as a data mining problem.* 2012. MNRAS, 421, 2, 1155-1165.

14. **Brescia, M., Longo, G.***Astroinformatics, data mining and the future of astronomical research. To appear in the Proceedings of the 2-nd International Conference on Frontiers in Diagnostic Technologies , Nuclear Instruments and Methods in Physics Research A.* s.l. : NIMA Elsevier Journal, 2012. arXiv:1201.1867.

15. **Aha, W., Kibler, D. e Albert, M.K.***Instance-Based Learning Algorithms.* Boston MA, USA : Machine Learning, Kluwer Academic Publishers Vol. 6, 1991.

16. **Bishop, C.M.***Pattern Recognition and Machine Learning.* s.l. : Springer, 206. ISBN 0-387-31073-8.

17. **Darwin, C.***On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.* London : s.n., 1859.

18. **Djorgovski, S.G., Longo, G., Brescia, M., Donalek, C., Cavuoti, S., Paolillo, M., D'Abrusco, R., Laurino, O.,Mahabal, A., Graham, M.***Data Mining and Exploration (DAME): New Tools for Knowledge Discovery in Astronomy.* s.l. : American Astronomical Society, AAS Meeting 219, 145.12, 2012.

19. **Hastie, T., et al.***The elements of statistical learning: data mining, inference and prediction.* s.l. : The Mathematical Intelligencer, Springer New York, Vol. 27, pp. 83-85, 2005.

20. **Hey, T., Tansley, S. e Tolle, K.***The Fourth Paradigm: Data-Intensive Scientific Discovery.* s.l. : Microsoft Research, Redmond Washington, 2009. ISBN-10: 0982544200, 2009.

21. **Kotsiantis, S. B.***Supervised Machine Learning: A Review of Classification Techniques, Proceeding of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering.* s.l. : IOS Press Amsterdam, The Netherlands, 2007, Vol. 160, pp. 3-24.

22. **Lorentz, G.G., Golitschek,M, Makovoz Y.,.***Constructive approximation: Advanced problems.* s.l. : Vol.304 Springer, Berlin, 1996.

23. **F., Mosteller e J.W., Turkey.***Data analysis, including statistics.* s.l. : In Handbook of Social Psychology. Addison-Wesley, Reading, MA, 1968.

24. **NVIDIA Corp.***CUDA C Best Practices Guide.* s.l. : NVIDIA Corporation, Distribution, ed. 4.1, 2012.

25. **Provost, F., Fawcett, T. e Kohavi, R.***The Case Against Accuracy Estimation for Comparing Induction Algorithms, Proceedings of the 15th International Conference on Machine Learning.* s.l. : Morgan Kaufmann. pp. 445-553, 1998.

26. **Rosenblatt, F.,.***The Perceptron - a perceiving and recognizing automaton.* s.l. : Report 85-460-1, Cornell Aeronautical Laboratory, 1957.

27. **Rumelhart, D., Hinton, G. e and Williams, R.,.***Learning internal representations by error propagation.* s.l. : In Parallel Distributed Processing, MIT Press, Cambridge, MA, chapter 8., 1986.

28. **Samuel, A.L.***Memorial Resolution.* s.l. : Stanford University, Historical society, 1990.

29. **Brescia, Massimo.***Classificazione Applicazioni con modello MLPQNA Test Report Rev. 1.* 2013 : s.n.

30. **Rottondi, Cristina , Politecnico di Milano facoltà di ingegneria delle Telecomunicazioni .** Tesi in : *CLASSIFICAZIONE DEL TRAFFICO INTERNET CON ATTRIBUTI PER-SOURCE.*