UNIVERSITÀ
DEGLI STUDI
DI PALERMO

Un Sistema Bayesiano per il Rilevamento delle Intrusioni nelle Reti Private

Tesi di Laurea Magistrale in Ingegneria Informatica

F. Giaimo

Relatore: Prof. Giuseppe Lo Re
Correlatore: Ing. A. De Paola

Un Sistema Bayesiano per il Rilevamento delle Intrusioni nelle Reti di Calcolatori

Sommario

Nel presente lavoro è stata trattata la costruzione di un *Anomaly-based Network Intrusion Detection System*, cioè un sistema di rilevazione delle intrusioni nelle reti di calcolatori basato sulle anomalie che si discostano dall'evoluzione "normale" dell'ambiente monitorato.

Si è scelto di implementare questo sistema su una rete Bayesiana: un particolare classificatore nel quale si assume che le caratteristiche usate per i calcoli inferenziali, che sono ottenute a partire dai dati estratti dai flussi di rete, siano indipendenti tra loro data la classe di appartenenza.

Prima di poter testare il rilevatore Bayesiano su di un dataset sono state effettuate delle operazioni di *feature selection* atte a selezionare le caratteristiche migliori in termini di predittività della classe, così da ridurre l'onere computazionale del sistema ed ottenere al contempo rilevazioni più precise.

A tal scopo è stato implementato un algoritmo di ricerca basato su una euristica di tipo *best-first*, che ha consentito di ridurre la complessità della ricerca nello spazio delle caratteristiche.

Per la verifica sperimentale del sistema è stato utilizzato il dataset NSL-KDD, costruito specificatamente per essere utilizzato nell'ambito dell'*Intrusion Detection*. Il sistema Bayesiano ha ottenuto risultati soddisfacenti, mostrando tassi di rilevazione dell'ordine dell'80-90%.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo I

Introduzione

Un *Intrusion Detection System (IDS)* è un sistema software o hardware per la rilevazione delle intrusioni, dove per "intrusioni" si intende *qualunque insieme di azioni che cercano di compromettere l'integrità, confidenzialità o disponibilità di una risorsa* [28]. Quindi non ci si riferisce solo alle effettive intrusioni di utenti non riconosciuti o agli attacchi contro le risorse locali o remote ma anche ai tentativi da parte di utenti noti di accedere a dati, funzionalità o diritti per i quali non hanno le necessarie credenziali.

Il funzionamento degli IDS si basa sull'assunzione che le attività intrusive siano distinguibili dalle attività normali, e quindi riconoscibili [5].

Gli IDS possono operare nel contesto delle reti locali, proteggendole dagli attacchi provenienti dall'esterno e dalle infrazioni commesse dagli utenti interni alla rete, o nell'ambito di una singola macchina, monitorando le azioni intraprese dall'utente e dal sistema operativo, e stabilendo se queste costituiscano una minaccia all'integrità del sistema e dei suoi dati o se rappresentino un uso ammissibile del sistema.

Da un sistema del genere ci si aspettano alcune caratteristiche:

- essere tollerante ai guasti, e operare con la minima supervisione umana. L'IDS deve quindi essere capace di recuperare la sua operatività dopo guasti accidentali alla macchina o dopo attacchi volti a minarne le capacità;
- avere la capacità di resistere alle manomissioni in modo che un attaccante non sia in grado di disabilitarlo o modificarlo facilmente;
- generare un carico computazionale minimo al sistema, in modo da non interferire con le sue normali attività;
- essere configurabile, in modo da poter implementare le politiche di sicurezza al meglio;

- deve essere capace di adattarsi ai cambiamenti del sistema e del comportamento degli utenti;
- poter essere installato in modo semplice sul sistema, garantendo la portabilità su varie architetture ed essere semplice da usare per l'operatore;
- avere capacità di generalizzazione, in modo da riconoscere numerose varietà di attacco e non avere falsi positivi o falsi negativi.

Un IDS è composto da un agente che raccoglie dati dall'ambiente (la rete o la macchina locale) e li fornisce al componente che li processa, confrontandoli secondo una sua metrica con dei modelli di utilizzo conosciuti ed eventualmente con tracce di attacchi già noti. Se viene rilevato un possibile uso potenzialmente dannoso del sistema viene inviata una segnalazione agli organi competenti che possono essere gli amministratori preposti alla sicurezza che quindi avranno il compito di investigare sull'origine dell'anomalia, o routine automatiche che provvederanno all'arginamento della contingenza, o entrambi.

In questo lavoro di tesi si implementerà un *anomaly-based network* IDS, nella forma di un classificatore Bayesiano dotato di una semplice struttura nella quale la classe influenza le variabili che rappresentano le *feature* estratte dai dati e le cui distribuzioni di probabilità sono supposte indipendenti tra loro. Sarà applicata la *feature selection* tramite l'implementazione di una tecnica *greedy*, adottando quindi con un approccio più orientato al *data mining* che non alla semantica delle *feature*. Verranno infine confrontate le performance di rilevazione tra i set di *feature* ottenuti tramite l'algoritmo descritto e quelli suggeriti dall'algoritmo *RankSearch* fornito dal *framework* per il *data mining* Weka.

Il presente lavoro è così strutturato:

- il Capitolo II contiene una panoramica sugli *Intrusion Detection Systems*, le diverse categorie esistenti ed i metodi usati per implementarli;
- il Capitolo III descrive le reti Bayesiane, le loro caratteristiche e gli algoritmi più usati per effettuare l'addestramento e l'inferenza probabilistica con questi strumenti;
- il Capitolo IV descrive la parte sperimentale di questo lavoro di tesi, esponendo le caratteristiche degli strumenti utilizzati, l'algoritmo adottato ed i risultati sperimentali ottenuti;
- il Capitolo V contiene le conclusioni ed i possibili sviluppi futuri.

Capitolo II

Sistemi di Rilevamento delle Intrusioni

I sistemi di rilevazione delle intrusioni operano secondo un principio comune: l'individuazione grazie ad una qualche metrica, intesa in senso generalizzato, di comportamenti che non rientrano nella normalità, basando queste rilevazioni sui dati da essi acquisiti dal sistema che devono proteggere.

La categoria di appartenenza di questi è determinata dall'approccio e dagli strumenti usati per la ricerca di questi comportamenti scorretti. Tutte le tipologie di IDS, in ogni caso, adottano tecniche di *data mining*. Il *data mining* è definito come la ricerca semiautomatica di pattern, regole e legami ed eventi statisticamente significativi nei dati. Possiamo quindi asserire che il *data mining* si occupa di estrapolare conoscenza dai dati. Per questa sua finalità il *data mining*, nelle sue svariate forme, è stato più volte impiegato nell'ambito della rilevazione delle intrusioni per estrarre conoscenza dai dati raccolti riguardo le abitudini degli agenti operanti nell'ambiente.

Saranno descritte le tre tipologie comunemente accettate di IDS, ponendo particolare attenzione verso gli *Anomaly-based Intrusion Detection Systems*.

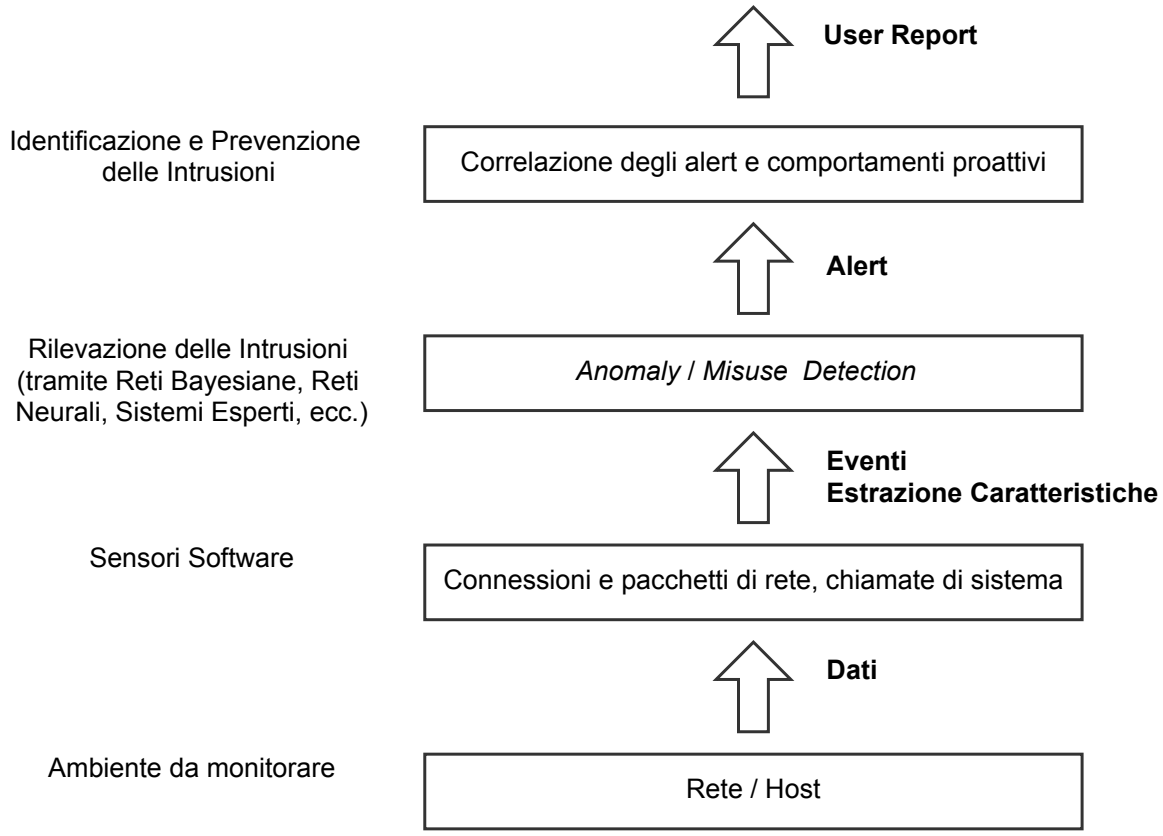

Figura 1: *Struttura generale di un IDS.*

2.1. Classificazione degli IDS

Esistono diverse classificazioni di IDS a seconda del particolare aspetto tenuto in considerazione per il confronto. Il principale criterio usato per classificare gli IDS è l'approccio usato per la rilevazione.

Gli IDS si dividono infatti in due categorie concettualmente ortogonali a seconda dell'approccio con cui cercano di rilevare le intrusioni: possono adottare una strategia detta di *signature* (o *misuse*) *detection* o di *anomaly detection*. Nella prima categoria si confrontano i dati raccolti con tracce di attacchi già noti, cercando una corrispondenza che confermi il fatto che ci sia un attacco in corso. Nel secondo caso si monitora invece il comportamento del sistema controllando che le sue modalità di utilizzo non subiscano deviazioni significative dall'uso considerato "normale".

Un ulteriore criterio che può essere adottato per la classificazione degli IDS è la risposta che viene fornita in seguito alla rilevazione di un'intrusione. Quando un IDS reagisce ad una intrusione adottando comportamenti correttivi o proattivi, come per esempio risolvere una possibile vulnerabilità o disconnettere un utente o un servizio che si comporta in modo eccessivamente sospetto, si dice *attivo*, se invece si limita a generare un allarme è detto *passivo*.

Un'altra classificazione si basa invece sull'ambiente nel quale questi agenti operano, cioè la rete (*network-based* IDS) o la singola macchina (*host-based* IDS). I primi IDS vennero infatti ideati e sperimentati nella seconda metà degli anni '80 per la protezione di mainframe e dei relativi utenti. L'attenzione venne però spostata progressivamente alla sicurezza nelle reti ed i primi *network-based* IDS che affrontavano il problema non erano altro che *host-based* IDS che si scambiavano dati. Infine, con la crescita di Internet, l'ambito di rete divenne preponderante, rendendo sempre maggiore l'esigenza di proteggere le reti da attacchi che ne mettessero in crisi le funzionalità e aprendo la strada ai *network-based* IDS, il cui campo d'azione è la rete ed il flusso di dati che in essa scorre.

Esistono inoltre IDS sviluppati con un approccio ibrido tra le due categorie *host-based* e *network-based*, che sono composti da moduli dei due tipi che comunicano le loro rilevazioni ad un terzo modulo di decisione.

2.2. Signature-based Intrusion Detection System

I *signature-based* IDS (noti anche come *Knowledge-based Intrusion Detection System*) si basano su modelli di intrusioni note (la *signature* o firma di un attacco), confrontandoli con i dati che vengono raccolti e cercando una possibile corrispondenza.

Il rilevamento è deciso sulla base della conoscenza di un modello e delle tracce lasciate da un'intrusione e questi modelli devono essere noti a priori ed impiantati nell'IDS in fase di sviluppo; ciò implica che questo approccio sia molto efficace contro gli attacchi ben conosciuti e modellizzati, fornendo quindi un tasso molto basso di falsi positivi, ma sia più facilmente eludibile da attacchi nuovi i cui modelli non sono noti all'IDS. Nonostante questo, la loro generale accuratezza li rende la scelta più diffusa nei sistemi commerciali.

Questo implica il fatto che gli IDS di questo tipo debbano ricevere aggiornamenti regolari per poter affrontare attacchi che sfruttino nuove vulnerabilità. Le nuove definizioni sono in genere costruite tramite algoritmi di apprendimento. Inoltre esiste un problema di generalizzazione, cioè il fatto che le caratteristiche di un attacco dipendano fortemente da sistemi operativi, versioni e piattaforme dei sistemi coinvolti, e quindi gli IDS di tale genere devono essere strettamente legati al particolare ambiente per i quali sono indirizzati.

Il modulo decisionale dell'IDS è programmato con regole di rilevamento, che sono rappresentazioni di quali eventi e valori ci si deve aspettare in caso di intrusione, con un implicito *"Default Permit"* finale: se il comportamento osservato non ricade in una categoria considerata rischiosa per la sicurezza, allora lo si prende per lecito.

2.2.1. Sistemi Esperti

L'implementazione più diffusa è quella basata sui *Sistemi Esperti*, le cui regole modellano il comportamento intrusivo. Gli eventi rilevati nel sistema vengono tradotti in *fatti* associati a certi valori semantici ed il motore inferenziale trae le sue conclusioni applicando sui fatti riportati le sue regole che possono essere immaginate come una sequenza di *if-else*.

I linguaggi basati su regole e produzioni sono la scelta naturale per modellare le conoscenze di questi sistemi. Alcune limitazioni di questi rilevatori sono legate al problema della completezza: può essere difficile tradurre la conoscenza in regole e possono esserci più modi per sfruttare una certa vulnerabilità, facendo sì che siano necessarie sempre più regole. Un ulteriore aspetto da considerare è la mancanza di sequenzialità degli *statement* condizionali che deve essere in qualche modo risolta per rendere più accurata la rilevazione, introducendo dei meccanismi che permettano di verificare regole aggiuntive qualora un qualche controllo precedente abbia dato esito positivo.

Un numero elevato di regole porta ad un problema di performance: un sistema esperto deve acquisire i dati dall'ambiente, tradurli in fatti e poi processarli secondo le sue regole, e queste sono operazioni costose in termini computazionali. Tuttavia la facilità di gestione di questi sistemi li rende la scelta più diffusa in questo ambito.

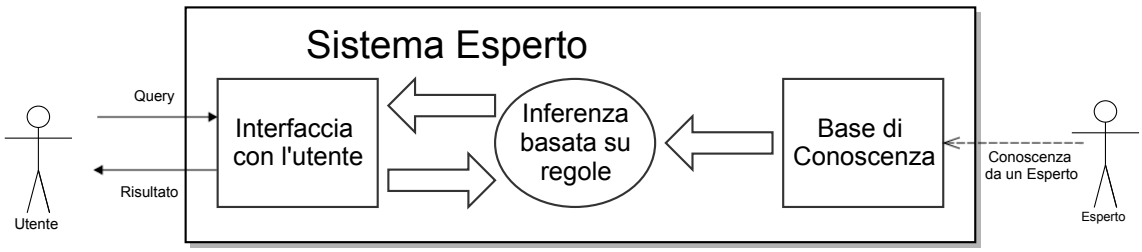

Figura 2: *Tipica struttura del Sistema Esperto. La conoscenza impiantata dagli esperti umani viene utilizzata per effettuare calcoli inferenziali su casi futuri.*

2.2.2. Modellazione di stato

Sistemi a modellazione di stato, che modellano il comportamento di un sistema come un insieme di stati che devono essere attraversati. Se la sequenza degli stati visitati non è quella attesa, si genera un allarme.

Questa tecnica viene utilizzata principalmente per quegli ambienti che hanno un comportamento facilmente modellizzabile, infatti si implementano prevalentemente nell'ambito degli *host-based* IDS, dove è il sistema di calcolo stesso ad essere monitorato.

2.2.3. *Pattern Matching*

Si tratta di sistemi che usano lo *string matching*, cioè cercano corrispondenze tra i messaggi scambiati tra sistemi sulla rete o derivanti dall'utilizzo del sistema da parte dell'utente, come comandi da console o le richieste di accesso a risorse o servizi privati, e sottostringhe note per essere usate durante particolari attacchi, come ad esempio combinazioni di comandi che si sa essere sfruttati per commettere infrazioni alle politiche di sicurezza.

Solo raramente questo approccio è implementato come unico mezzo di rilevazione, infatti esso è generalmente affiancato a sistemi più complessi, come i Sistemi Esperti, permettendo di aumentarne le capacità predittive grazie all'analisi dei contenuti dei pacchetti di rete o dei comandi di sistema. Il problema di cui soffrono queste tecniche è il tempo di analisi di grandi moli di dati ma sono stati proposti algoritmi sempre più efficienti per la ricerca di match tra stringhe [12].

2.3. Anomaly-based Intrusion Detection System

Gli *anomaly-based* IDS assumono che si possa rilevare un'intrusione osservando deviazioni significative dai comportamenti attesi dal sistema o dagli utenti. Questi rilevatori operano costruendo un modello di comportamento lecito dell'oggetto in esame (il sistema, un utente, ecc.), scegliendo una metrica ed una distanza secondo tale metrica oltre la quale una attività che si discosti dal comportamento "normale" è da considerarsi sospetta e segnalando infine qualunque attività che si discosti sufficientemente dall'uso considerato legittimo.

I vantaggi di questo tipo di rilevatori è che hanno maggiori possibilità di rilevare un attacco non noto a priori, sono meno legati alla specifica piattaforma e rilevano più facilmente gli abusi di privilegi. Questi vantaggi si pagano però al prezzo di una minore accuratezza generale dovuta al fatto che i modelli che l'IDS deduce possono non essere sempre comprensivi di tutti i possibili utilizzi legittimi del sistema. Inoltre bisogna considerare che i comportamenti degli utenti possono cambiare col tempo, rendendo a volte necessario un aggiornamento dei profili.

Alcuni di questi IDS (a volte chiamati *ad auto-apprendimento*) possono costruire questi modelli dagli esempi forniti nella fase iniziale della loro installazione o anche direttamente dai dati raccolti dal sistema 'sul campo' e non solo in fase di sviluppo, così da adattarsi meglio ai profili di utilizzo nel particolare ambiente dove dovrà operare. Dagli esempi e dalla osservazione delle operazioni che vengono effettuate essi deducono un modello delle attività lecite contro il quale valutare in futuro i comportamenti che vengono costantemente rilevati. Altri sistemi invece hanno una 'descrizione' (sotto

forma per esempio di un set di regole o statistiche), fornita in fase di progetto o di configurazione, del comportamento ammissibile.

Lo svantaggio di questi sistemi, cioè i tassi di precisione generalmente non ottimi, si manifesta qualora le soglie di rilevazione non siano appropriate per il contesto di utilizzo. La scelta di soglie adeguata che bilancino i falsi positivi senza ignorare le operazioni illecite può non essere semplice in contesti di rete molto dinamici.

Per aumentare l'efficienza di questo tipo di IDS è possibile affiancare al generico modulo *anomaly-based* anche un riconoscitore di attacchi noti, in modo da includere le funzionalità dei *signature-based* IDS ed ottenere così un rilevatore composito, potenzialmente più accurato dei due sistemi singolarmente presi.

2.3.1. Statistiche semplici

Alcuni *anomaly-based* IDS, i meno sofisticati, si basano su semplici statistiche costruite a partire dall'osservazione di alcune variabili significative (per esempio gli orari di login e logout, il livello medio di utilizzo delle risorse, il numero di inserimenti falliti delle password). Quando il rilevatore osserva un comportamento non più in linea con le statistiche viene lanciato un allarme.

Lo svantaggio di questi rilevatori è che una simile rappresentazione non riproduce abbastanza fedelmente il sistema da monitorare. Una evoluzione di questi IDS è stata ottenuta facendo sì che il rilevatore costruisse modelli a lungo e breve termine delle attività da controllare, in modo da particolarizzare maggiormente l'analisi. Anche in questo caso i profili d'uso richiedono aggiornamenti costanti per seguire le naturali alterazioni dei comportamenti del sistema o degli utenti.

2.3.2. Sistemi Esperti

Esistono anche rilevatori basati su sistemi esperti che confrontano il comportamento rilevato con un insieme di regole che possono essere stabilite programmaticamente. Questo avviene nel caso in cui siano in vigore rigide politiche di sicurezza, come per esempio accade in ambito aziendale o che queste regole siano imparate dal rilevatore attraverso l'osservazione, producendo statistiche che saranno successivamente convertite in regole. Infine si opera come di consueto verificando che gli agenti nell'ambiente da controllare operino nel rispetto delle regole dedotte e segnalando le operazioni che le infrangano.

2.3.3. Artificial Neural Network

Le implementazioni più numerose sono forse quelle in cui il modulo di rilevazione è basato sulle reti neurali (ANN, *Artificial Neural Network*).

Le reti neurali sono modelli di apprendimento costruiti interconnettendo un insieme

di unità logiche secondo una particolare topologia che ne determina il comportamento. Questi strumenti sono in grado di imparare tramite esempi le relazioni tra dati di ingresso ed uscita basandosi anche su set di dati non completi o rumorosi e di astrarle applicandole ad un nuovo e più ampio insieme di dati in input, calcolandone le relative uscite.

Le reti in questione vengono addestrate con i dati relativi al traffico ammesso e a quello non ammesso, facendo sì che la rete sappia classificare correttamente ciò che si troverà in ingresso una volta in fase operativa. Anche in presenza di attacchi nuovi, con i quali non vi era stato alcun addestramento, la rete segnala di trovarsi in una situazione non prevista poiché i dati che riceve non rispecchiano la normale attività del sistema. Le reti neurali offrono numerosi vantaggi: possono essere aggiornate semplicemente e automaticamente per tenere il passo con l'evoluzione dei comportamenti del sistema, imparano relazioni non lineari tra i dati che sarebbero difficili da esprimere altrimenti e sono uno strumento ampiamente accettato e compreso; tuttavia per grandi moli di dati, come quelli che possono essere generati su reti aziendali di grandi dimensioni, può esistere un problema computazionale.

In base a come avviene l'addestramento dividiamo le ANN in due categorie: ad *addestramento supervisionato* ed *addestramento non supervisionato*.

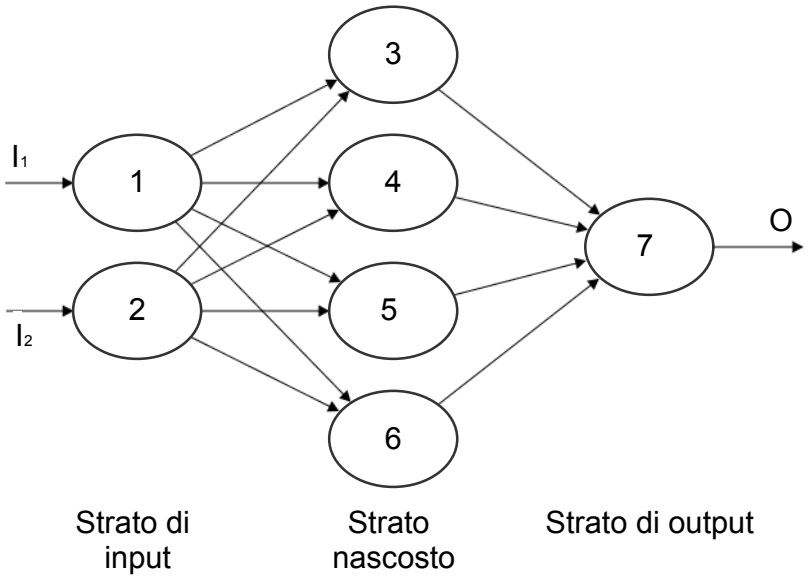

Figura 3: *Esempio di rete neurale multistrato. Ad ogni arco è associato un peso.*

Addestramento supervisionato

Delle prima categoria fanno parte le *recurrent* ANN e le *feed-forward* ANN, cioè reti neurali senza retroazione nelle quali i calcoli e risultati parziali vengono propagati

semplicemente dall'input all'output della rete. Di queste abbiamo le *Multi-layered Feed Forward Neural Networks (MLFF)* e le *Radial Basis Function Neural Networks (RBF)*.

Le reti MLFF sono le più semplici, costituite da neuroni basati su soglie di somme pesate, mentre le reti RBF si basano sul calcolo delle distanze tra gli input ed i centri delle radial basis function rappresentate dai neuroni degli strati nascosti. Le RBF hanno ottenuto performance migliori delle MLFF [41], mostrando alta capacità di rilevazione, un basso tasso di falsi positivi e richiedendo un tempo di addestramento molto inferiore.

Delle *recurrent* ANN fanno parte le reti di Elman, che prendono in input oltre ai dati di ingresso anche dati dagli strati intermedi del tempo precedente, e le *Cerebellar Model Articulation Controller* ANN. Entrambe le tipologie hanno ottenuto risultati migliori delle semplici MLFF [4].

Addestramento non supervisionato

Nella categoria delle ANN con addestramento non supervisionato un importante esempio sono le *Self Organizing Maps*.

Le SOM sono reti classificatrici *feed-forward* che raggruppano gli oggetti in input assegnandoli ad un singolo neurone della rete, che di norma ha una dimensionalità bassa (spesso si tratta di reti bidimensionali a singolo strato). Questa capacità di *clustering* le rende promettenti candidate per il riconoscimento delle intrusioni, in quanto il traffico lecito è generalmente prevedibile e ripetuto nel tempo e quindi sarà raggruppato in pochi ampi settori, mentre le attività vietate avranno una topologia molto più sparsa e meno popolosa.

Una importante caratteristica delle SOM è la conservazione della topologia: se due dati in ingresso sono simili, verranno mappati su due neuroni vicini o sullo stesso neurone: questo ha portato al tentativo di sviluppo di un approccio visuale per il riconoscimento delle attività della rete da proteggere, proiettando su una griglia a colori lo stato dei neuroni di una SOM bidimensionale per permettere l'ispezione visiva delle attività di rete [16].

In numerosi studi è emerso che reti SOM gerarchiche ben costruite hanno ridotto l'ambiguità nelle reti a singolo livello dove attività diverse venivano mappate nello stesso neurone, ottenendo un evidente aumento della precisione delle rilevazioni [17].

2.3.4. Reti Bayesiane

Un ulteriore approccio è l'utilizzo di reti Bayesiane. Dato che l'elevato numero di falsi positivi degli *anomaly-based* IDS nasce dall'incertezza insita nel metodo in cui avviene la discriminazione tra comportamento permesso e quello anomalo, è stato proposto l'utilizzo di una rete di Bayes, uno strumento che modella l'incertezza in un sistema rappresentato come un insieme di variabili casuali correlate tra loro.

In un Grafo Orientato Aciclico, cioè un grafo dotato di archi orientati e privo di cicli diretti, ad ogni nodo è associata una variabile casuale e la sua Tabella delle Probabilità Condizionate che esprime il legame di dipendenza condizionale nei confronti delle variabili rappresentate dai nodi genitori. Viene quindi applicato un algoritmo che simula un 'ragionamento' su questi nodi: a partire dai nodi di evidenza, come per esempio quelli legati alla presenza di certe caratteristiche estratte dai pacchetti della rete, si calcolano le probabilità che certe ipotesi di interesse per la rilevazione siano verificate, adottando un processo inferenziale che ricalca il ragionamento diagnostico.

2.3.5. Logica Fuzzy

La logica Fuzzy è una logica polivalente, basata sulla teoria degli *insiemi sfumati*, nella quale ogni preposizione può avere un *grado di verità* che può essere compreso tra 0 e 1. Una *funzione di appartenenza* fornirà la corrispondenza tra la proposizione ed il grado di appartenenza al particolare insieme.

La logica Fuzzy permette di definire un set di funzioni che classifichino i dati da analizzare in modo da stabilire se descrivano un comportamento ammesso o si tratti di infrazioni alle politiche di sicurezza. Un vantaggio di questo approccio è la flessibilità, poiché in questo modo si ha un IDS espresso come un set di regole associative sulle quali possono intervenire esperti umani o tecniche automatiche di adattamento [40].

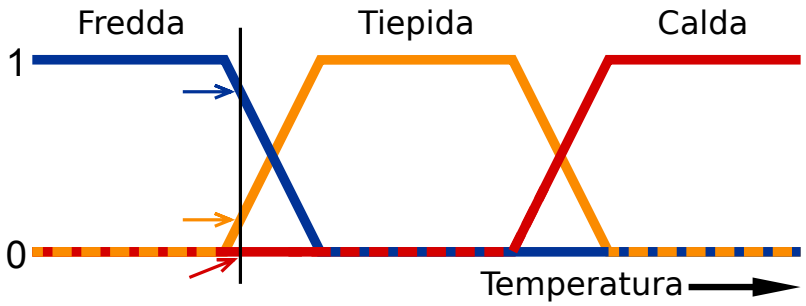

Figura 4: *Esempio di funzioni di appartenenza Fuzzy.*

2.3.6. Programmazione Genetica

La Programmazione Genetica è basata sugli Algoritmi Evolutivi, cioè tecniche di apprendimento automatico basate sulle dinamiche della "selezione naturale". Nella Programmazione Genetica un programma è rappresentato da un cromosoma, cioè un insieme di geni che indicano punti o caratteristiche salienti del suo codice. Partendo da un insieme di possibili programmi candidati si applicano *operatori evolutivi* quali il *crossover*, che combina casualmente due cromosomi per crearne di nuovi, o la *mutazione*, che causa una alterazione arbitraria in un cromosoma. Procedendo in questo modo

si creano casualmente nuove istanze di programmi ad ogni iterazione, chiamata *epoca*, rimpiazzando quelle che secondo il verdetto di una apposita funzione di *fitness* sono meno adatte a risolvere il problema.

Nell'ambito della rilevazione delle intrusioni la Programmazione Genetica è stata utilizzata in diversi modi, per esempio nella ricerca del miglior insieme di caratteristiche da sfruttare per la rilevazione, per l'ottimizzazione dell'apprendimento di regole da parte di classificatori Fuzzy [18], cioè impiegando la capacità degli algoritmi genetici di trovare "buone" soluzioni nella ricerca di regole Fuzzy sempre più precise nel classificare i comportamenti malevoli, o per la ricerca della struttura e dei pesi ottimali per la classificazione effettuata con reti neurali.

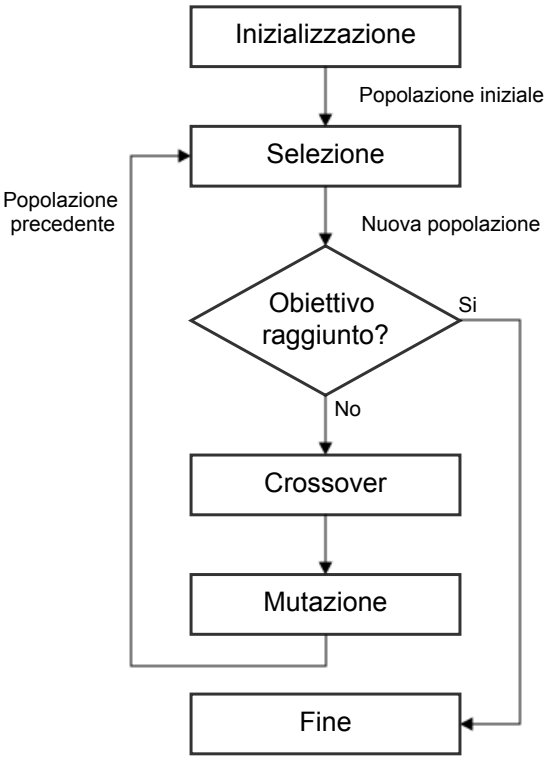

Figura 5: *Diagramma di flusso per gli Algoritmi Genetici.*

2.4. Sistemi Ibridi

Per superare le limitazioni che hanno i sistemi delle due categorie *signature-based* e *anomaly-based*, sono stati implementati IDS che formano la loro decisione sulla base di modelli di entrambe le classi, prendendo quindi in considerazione sia il comportamento normale del sistema, sia il comportamento atteso da un attaccante. Dato che parte del rilevatore è *anomaly-based*, lo si deve addestrare con esempi di traffico normale e non, mentre il modulo per la *misuse detection* è normalmente basato su regole, e costituisce

la parte predominante del sistema composto. I sistemi ibridi offrono due importanti vantaggi:

- generalmente i sistemi di questa classe hanno maggiori probabilità di riconoscere gli attacchi e possono qualificare meglio gli allarmi che lanciano rispetto ai sistemi delle classi semplici [8];
- questa strategia permette di evitare che un utente malevolo possa introdurre comportamenti scorretti nel sistema diluendoli nel tempo, facendo sì che il modulo *anomaly-based* abbia il tempo di adattarsi alla condotta illegale e finendo per non rilevarla quando un attacco vero e proprio è in corso [2].

Per queste motivazioni l'approccio ibrido è adottato da gran parte degli *Intrusion Detection System* disponibili, di cui è fornita una lista non esaustiva nella Sezione 2.7.

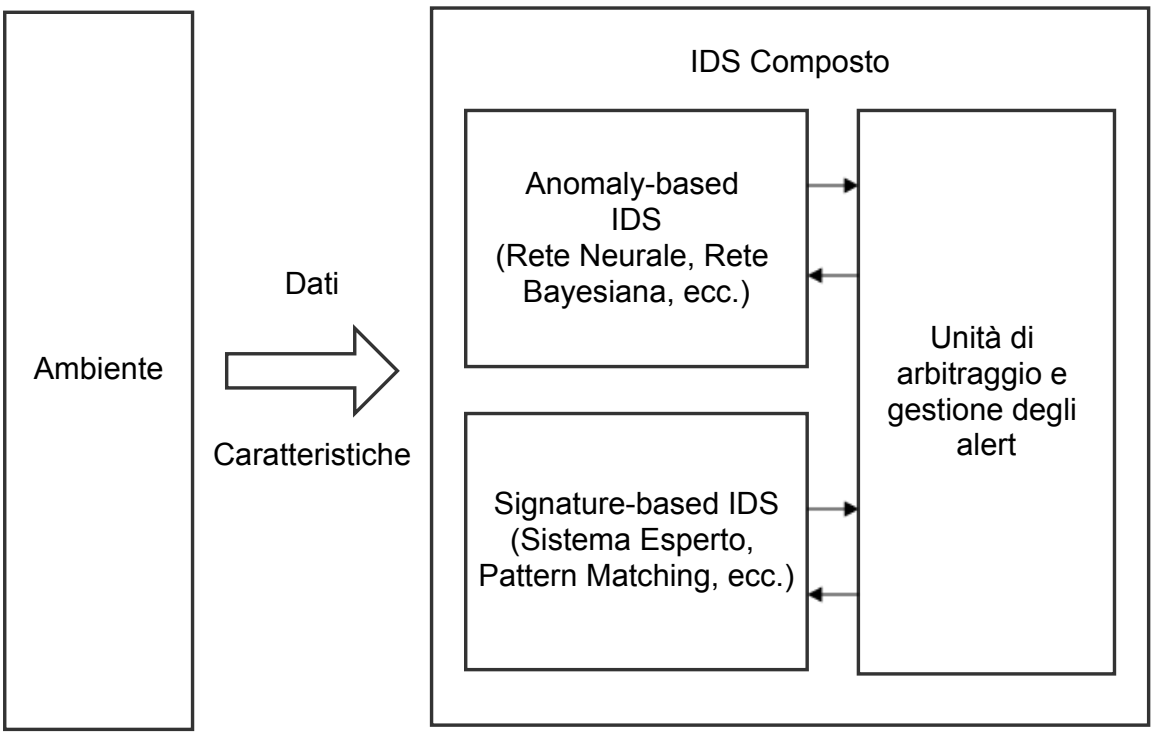

Figura 6: *Struttura generale di un IDS ibrido.*

2.5. Acquisizione dei dati

Sistemi di intrusione che operano su ambiti diversi acquisiranno informazioni diverse. Ricordando che la principale differenza è l'ambiente d'azione dell'IDS, descriviamo brevemente le principali fonti di dati di *auditing*.

2.5.1. Host-based IDS

Gli *host-based* IDS acquisiscono le informazioni su cui basare le decisioni da classici strumenti di amministrazione di un sistema, come ad esempio le *system call*, i sistemi per il controllo della quantità di risorse utilizzate, gli strumenti per il controllo delle attività degli utenti registrati e delle modifiche effettuate sul registro ed il monitoraggio dei diversi file di log di sistema. Dalle informazioni raccolte da queste fonti possono essere costruite delle statistiche, come il numero di tentativi di *login* effettuati nell'ambito di una apposita finestra temporale, che possono fornire una indicazione efficace per la rilevazione di tentativi di manomissione del sistema o comunque di attività anomale.

2.5.2. Network-based IDS

I *network-based* IDS si basano sull'analisi dei pacchetti di rete, acquisiti al momento della ricezione da parte dell'*host* o prima della loro ritrasmissione (operazione detta *sniffing*). L'analisi può essere condotta controllando due porzioni diverse dei pacchetti: l'*header* o intestazione del pacchetto, o il *payload* o contenuto. Ognuno degli approcci può riconoscere attacchi che l'altro non potrebbe rilevare poiché effettuati su livelli dello stack TCP/IP diversi. Tuttavia rispetto gli attacchi che sfruttano vulnerabilità dei protocolli di rete o trasporto per danneggiare l'obiettivo, quelli ai livelli applicativi sono sempre più pericolosi e frequenti dato che con la grande varietà di applicazioni e servizi *web-based* esistono moltissimi modi per tentare di danneggiare il sistema e carpire dati sensibili, laddove le modalità degli attacchi diretti ai protocolli di rete o trasporto sono per la maggior parte noti e le falle da essi sfruttate sono state ormai ampiamente corrette o comunque almeno in parte arginate [9].

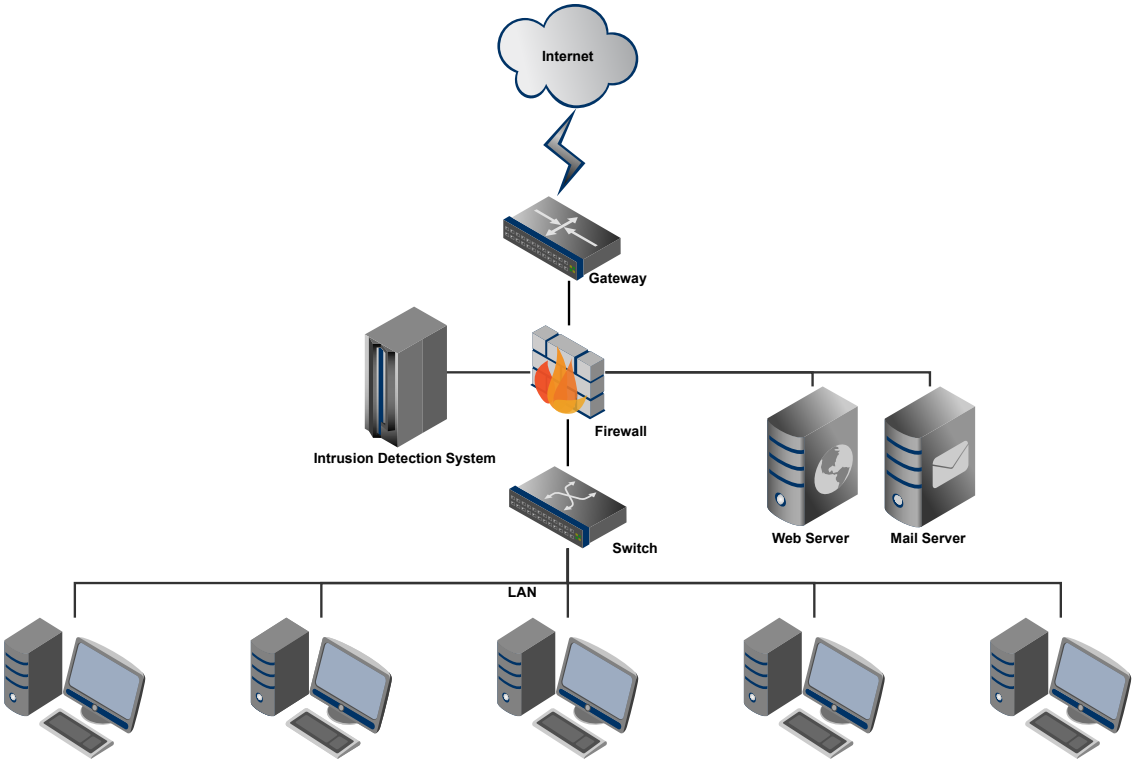

Figura 7: *Tipico posizionamento di un IDS all'interno della topologia di una rete aziendale.*

Sistemi Header-based

Gli *Header-based Network* IDS, effettuano controlli sulle intestazioni dei pacchetti, verificandone quindi la correttezza e l'evoluzione. Sono possibili infatti attacchi che aggrediscono i protocolli di comunicazione, sfruttando le loro proprietà intrinseche o le loro implementazioni.

Si riporta l'esempio dell'attacco *teardrop*, un tipo di attacco *Denial of Service (DoS)*, nel quale si sfrutta una vulnerabilità presente in alcune vecchie implementazioni dello stack TCP/IP che non effettuavano controlli accurati nel caso di riassemblaggio di pacchetti IP frammentati quando gli indici indicavano una sovrapposizione. L'attacco consiste proprio nell'invio di pacchetti IP frammentati sovrapposti costruiti ad arte: il sistema ricevente, se non effettua gli opportuni controlli, tenterà di riunirli in un unico pacchetto, ma la sovrapposizione farà sì che il sistema tenterà di allocare un blocco di memoria di dimensione negativa, causando un errore che manderà in crash il sistema operativo.

Sistemi Payload-based

Nell'approccio *Payload-based*, si controlla il contenuto del pacchetto, quindi i dati afferenti ai livelli applicativi, cercando di riconoscere comandi o dati tipici di un attacco. Un modo per effettuare questi controlli è per esempio l'utilizzo di tecniche di *pattern*

matching per trovare comandi o caratteri che possano potenzialmente costituire un rischio per il sistema ricevente [24].

Un esempio ben noto e molto pericoloso di questo tipo di attacchi è l'*SQL injection*, nel quale si sfruttano controlli poco accurati su dati ricevuti in input da applicazioni che si basano su un database SQL con l'effetto di poter 'iniettare' e far eseguire una query al sistema anche senza averne i privilegi.

2.6. Riduzione dei dati

Prima dell'analisi vera e propria delle caratteristiche estratte dall'ambiente, siano esse statistiche, chiamate di sistema o pacchetti estratti dalla rete, è quasi sempre necessario effettuare un *preprocessing* su queste informazioni prima di elaborarle con il Sistema di Rilevazione.

La riduzione dei dati da fornire agli IDS è spesso un fattore strategico. Dato che un IDS si troverà a dover processare una grande mole di dati anche se il suo ambiente operativo è circoscritto ad una sola macchina o ad una rete di piccole dimensioni, sarà necessario fornire all'IDS solo quei dati che gli saranno effettivamente utili, eliminando quelle informazioni ininfluenti o fuorvianti che ne degraderebbero le prestazioni aumentandone comunque il tempo di computazione, aspetto fondamentale negli IDS ai quali sono sono richieste prestazioni in tempo reale.

La riduzione dei dati può avvenire in più modi. I dati reputati non influenti possono essere filtrati, mantenendo solo i dati potenzialmente utili. In alternativa si può considerare l'utilizzo di una tecnica di *clustering* per non dover lavorare con le caratteristiche del dato vero e proprio ma con quelle del suo cluster di appartenenza. Infine si può applicare una tecnica di *feature selection* per decidere quali siano le fonti di dati più significative nell'ottica della rilevazione, scartando le altre.

2.6.1. Filtraggio

Il filtraggio opera per ridurre la quantità di dati prima che questi arrivino all'IDS. Alcuni di questi dati possono infatti non essere utili per la rilevazione e quindi essere eliminati prima di essere processati.

Questo metodo ha il vantaggio di diminuire i requisiti di memoria dato che le informazioni scartate non saranno salvate nei file di log e non saranno oggetto di analisi, consentendo di ridurre il tempo di elaborazione ed aumentare le prestazioni di rilevazione. Chiaramente il filtraggio va gestito con attenzione poiché c'è il rischio di scartare dati utili.

2.6.2. Clustering

Il *clustering* viene applicato per trovare pattern nascosti e correlazioni tra dati, in modo da individuare le *feature* più idonee ad essere utilizzate.

Il *clustering* può essere utile per la riduzione dei dati poiché applicando a monte del rilevatore un modulo che associ al dato il suo *cluster* di appartenenza è possibile modificare l'IDS in modo che lavori sulla base sulle caratteristiche dei *cluster* invece che sulle *feature* dei dati veri e propri.

2.6.3. Feature selection

Nell'ambito della classificazione, alcune caratteristiche dei dati possono influire negativamente sul processo di rilevazione. Possono essere infatti presenti *feature* che esprimono false correlazioni, o sono ridondanti.

Le tecniche di *feature selection* permettono di eliminare quelle caratteristiche che riducono l'accuratezza del rilevatore, cercando nello spazio delle *feature* quelle più efficaci nel classificare i dati. Il compito di questi algoritmi non è semplice poiché non esistono modelli o funzioni che catturino esattamente tutte le relazioni tra le *feature* e gli attacchi.

Il modo ottimo per costruire un *ranking* delle caratteristiche in base alla loro correlazione con la classe del record analizzato è l'*Analisi delle Componenti Principali*.

Analisi delle Componenti Principali

L'Analisi delle Componenti Principali (Principal Component Analysis, PCA) è un metodo molto utilizzato quando un particolare aspetto di interesse non sia direttamente quantificabile e si abbia accesso solo a indicatori indiretti del problema.

Questo metodo cerca di ridurre la dimensionalità dei dati di partenza, visti come un vettore di variabili casuali $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ le cui n componenti sono correlate tra loro, operando proprio sulle loro intercorrelazioni in modo da rendere più agevoli le successive analisi. Per ottenere questo obiettivo, la PCA opera trasformando il set di variabili originario in un nuovo set in cui ogni nuova variabile è funzione delle variabili di partenza.

Le nuove variabili devono rispettare tre importanti proprietà:

- sono variabili statisticamente scorrelate;
- vengono ottenute in ordine decrescente di varianza, avendo come prima variabile quella a varianza maggiore, la seconda con varianza immediatamente inferiore alla prima e così via;

- la varianza totale del dataset originario è uguale alla varianza totale delle nuove variabili.

Queste nuove variabili, chiamate proprio *componenti principali*, sono definite come particolari combinazioni lineari delle n variabili di partenza $X_1, X_2, ..., X_n$. La prima componente Y_1 sarà quella combinazione lineare nel vettore $\mathbf{X}$ con la maggiore varianza possibile

$$Y_1 = \boldsymbol{\alpha_1 X} = (\alpha_{11}X_1 + \alpha_{12}X_2 + ... + \alpha_{1n}X_n)^T = \sum_{i=1}^{n} \alpha_{1i}X_i \ .$$

La seconda componente principale Y_2 sarà un'altra combinazione lineare, anch'essa cercata con la massima varianza possibile posto che sia del tutto scorrelata da Y_1, e così via. Chiaramente potranno essere trovate al più n componenti principali, ma generalmente il loro numero è minore. [22]

Il calcolo delle componenti procede come segue:

1. si organizzano i K record in forma matriciale, scrivendoli come vettori riga, ottenendo quindi una matrice $\boldsymbol{D}$ di dimensioni $K \times n$;

2. si centrano le componenti del vettore iniziale sul valore medio, cioè si calcola la media per ognuna delle n componenti e la si sottrae ad ogni rispettiva singola variabile X_i con $1 \leq i \leq n$, ottenendo quindi la matrice $\boldsymbol{D'}$;

3. a partire dalla matrice $\boldsymbol{D'}$, ora centrata sullo 0, si calcola la matrice di covarianza $\boldsymbol{C}$, di dimensione $n \times n$;

4. si trova la matrice $\boldsymbol{A}$ degli autovettori di $\boldsymbol{C}$. Questi autovettori saranno i vettori $\boldsymbol{\alpha_i}$ con $1 \leq i \leq n$.

Si ha che gli autovalori associati agli autovettori rappresentano la varianza di quella particolare combinazione. Ordinando quindi in senso decrescente gli autovettori in base agli autovalori si ottengono le componenti Y con l'ordinamento ricercato [33].

L'importanza del metodo sta nel fatto che in dati ad alta dimensionalità si trova che le componenti che influiscono maggiormente sulle variabili di partenza sono poche rispetto al totale, permettendo di utilizzarle senza le altre per ricostruire i dati iniziali con una approssimazione generalmente accettabile, risparmiando di conseguenza sul salvataggio, codifica o trasmissione di un gran numero di componenti poco influenti.

Nell'ambito dell'*Intrusion Detection* la PCA permette di trasformare il dominio delle caratteristiche di partenza in un dominio di nuove variabili dalle quali estrarre quelle maggiormente significative ed ignorando quelle la cui influenza rispetto la classe di appartenenza del record non è abbastanza elevata.

2.7. IDS open-source

Esistono implementazioni sia commerciali che *open-source* di *network-based* IDS. A causa del grande problema degli *anomaly-based* IDS, cioè l'elevato numero di falsi positivi, la maggior parte di queste implementazioni adotta approcci *signature-based*, giovando anche della facilità di manutenzione ed estensibilità di una struttura basata su regole. Tuttavia in molti casi i due approcci vengono affiancati, costituendo in effetti un sistema composto. Si riporta a titolo di esempio una breve descrizione delle caratteristiche di tre tra i più usati *Network Intrusion Detection Systems* multipiattaforma rilasciati con licenza *open-source*.

2.7.1. Snort

Tra i più noti *Intrusion Detection Systems* troviamo sicuramente Snort. Snort è un sistema di rilevazione e prevenzione delle intrusioni di rete sviluppato dalla Sourcefire. Esso permette di combinare gli approcci *signature-based* ed *anomaly-based* per rilevare le minacce e può intraprendere azioni contro l'attaccante in caso queste siano state abilitate. Questa sua capacità giustifica la definizione di *Intrusion Prevention System* [34].

2.7.2. Suricata

Un altro IDS è Suricata, anch'esso ibrido, sviluppato dalla *Open Information Security Foundation (OISF)*. A differenza di Snort, Suricata supporta il multithreading, ma non prevede l'attuazione di strategie atte a bloccare un attacco rilevato [36]. Tuttavia questo IDS, grazie allo *string matching* effettuato sui campi dei pacchetti, può riconoscere traffico appartenente ad un certo protocollo a prescindere dalla porta di provenienza, permettendo di scrivere regole riguardanti il protocollo e non la porta.

2.7.3. Bro

Un ulteriore IDS è Bro, sviluppato dal *Network Research Group al Lawrence Berkley National Lab* e dall'*International Computer Science Institute*. Anche questo è un IDS passivo, la cui particolarità sta nel fatto che la filosofia col quale è stato costruito lo rende più simile ad un *framework* nell'ambito della rilevazione delle intrusioni. Oltre alle classiche funzionalità di analisi dei pacchetti è dotato infatti di un suo linguaggio, simile al Python, e di una *standard library* di funzionalità già implementate che permettono all'utente di sviluppare programmi per analizzare il traffico in modo totalmente personalizzabile. Infine le sue caratteristiche di scalabilità lo rendono utilizzabile anche su reti ad alta velocità e ad alto volume di traffico [3].

Capitolo III

Reti Bayesiane per la Rilevazione delle Intrusioni

Il vantaggio di operare con strumenti probabilistici è quello di poter gestire naturalmente situazioni in cui non tutte le informazioni sono disponibili a priori o sono prive di incertezza. In questo contesto tra i più noti strumenti troviamo le reti Bayesiane.

3.1. Concetti di base

Una rete Bayesiana permette di applicare tecniche di inferenza statistica per diagnosticare le cause di un evento data l'evidenza dell'effetto, di inferire la probabilità di un effetto data la presenza di una causa e di risolvere problemi di classificazione del tipo $\max_{classe} Pr(classe|dati)$ o di *decision making* se fornita di una funzione di costo/utilità.

3.1.1. Grafo Diretto Aciclico

Come precedentemente accennato, una rete Bayesiana è un modello grafico che rappresenta un insieme di variabili casuali e le loro dipendenze condizionali, rappresentandole con l'ausilio di un grafo orientato aciclico, cioè un grafo dotato di nodi ed archi orientati ma privo di cicli diretti, ovvero a partire da qualunque nodo del grafo non è possibile incrociare nuovamente il nodo di partenza muovendosi secondo il verso degli archi della rete.

I nodi rappresentano variabili ai cui possibili stati, che devono essere mutualmente esclusivi, è associato un certo valore di probabilità, mentre gli archi tra nodi indicano una relazione di dipendenza condizionale tra le variabili rappresentate dai nodi: se due nodi non sono direttamente connessi, sono condizionatamente indipendenti. Ai nodi che hanno genitori, cioè che sono collegati ad almeno un arco che punta su di loro, sono associate le TPC (Tabelle di Probabilità Condizionata) cioè tabelle che contengono le probabilità dei valori del nodo condizionate alle possibili combinazioni di valori dei nodi

genitori, esprimendo quindi i possibili valori di $Pr(A|B_1, B_2..., B_n)$ per il nodo A con n genitori $\{B_1, B_2..., B_n\}$. In quei nodi che non hanno genitori saranno invece presenti delle Tabelle di Probabilità A Priori che esprimeranno semplicemente le probabilità marginali per ogni valore della variabile del nodo.

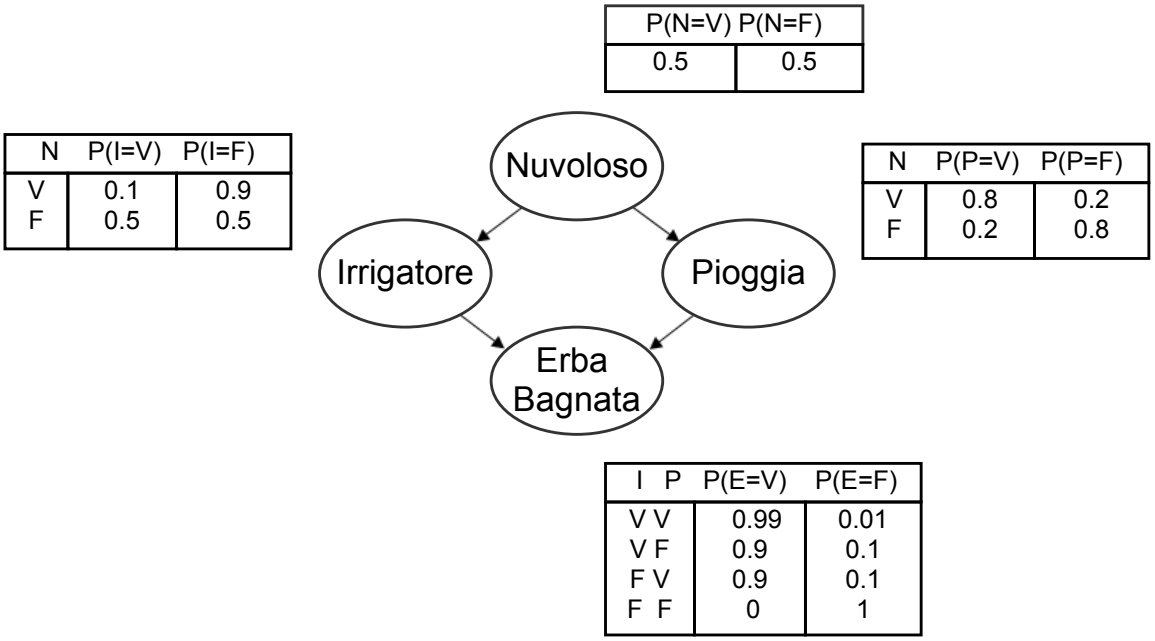

Figura 8: *Esempio di rete Bayesiana. Si notino l'assenza di cicli diretti, le Tabelle di Probabilità Condizionata associate alle variabili rappresentate dai nodi e come la dimensionalità delle Tabelle aumenti al crescere del numero dei genitori del nodo.*

3.1.2. Regola del concatenamento

Una importante proprietà probabilistica è la *regola del concatenamento (chain rule)* che esprime la relazione che intercorre tra la probabilità congiunta di un insieme di k variabili e le loro rispettive probabilità condizionate:

$$Pr(V_1, V_2, ..., V_k) = \prod_{i=1}^{k} Pr(V_i|V_{i-1}, ..., V_1) \ .$$

Grazie a questa regola e sfruttando il fatto che un nodo rappresenta una variabile che dipende solo dalle variabili dei nodi genitori, è possibile esprimere la probabilità congiunta di tutte le variabili nella rete secondo la formula

$$Pr(X_1, X_2, ..., X_n) = \prod_{i=1}^{m} Pr(X_i|parents(X_i)) \ ,$$

dove m indica il numero di nodi nella rete.

3.1.3. *Inferenza Bayesiana con il Teorema di Bayes*

L'inferenza sulle reti Bayesiane è un processo di inferenza statistica nel quale si sfrutta il Teorema di Bayes per stimare ed aggiornare la probabilità di una ipotesi via via che vengono raccolte nuove evidenze. Il teorema permette di esprimere una probabilità condizionata nei termini della probabilità condizionata opposta, pesandola opportunamente secondo la formula

$$Pr(I|E) = \frac{Pr(E|I)Pr(I)}{Pr(E)} ,$$

dove

- I indica una ipotesi la cui probabilità deve essere dedotta a partire dalle osservazioni.
- E rappresenta l'evidenza dei nuovi dati presi in esame.
- $Pr(I|E)$ è la probabilità dell'ipotesi condizionata all'effettiva rilevazione dell'evidenza.
- $Pr(I)$ indica la probabilità a priori dell'ipotesi prima che venga osservato l'evento E.
- $Pr(E|I)$ è la verosimiglianza, cioè probabilità di ottenere l'evidenza E posto che si sia verificata l'ipotesi I.
- $Pr(E)$ rappresenta la probabilità di rilevare l'evento E a prescindere dal verificarsi o meno dell'ipotesi I.

Il teorema quindi ci permette di stimare la probabilità di una certa ipotesi a partire dalle evidenze e del loro legame statistico con l'ipotesi stessa. Questo è un grande aiuto nella rilevazione delle intrusioni perché diventa possibile dedurre la probabilità di essere sotto attacco ragionando sulle caratteristiche che vengono estratte dal traffico nella rete.

3.2. Classificatori Naive

Nel problema della classificazione abbiamo a che fare con un insieme di esempi omogenei caratterizzati ognuno da un certo numero n di attributi ed associati ad un determinato valore c di una classe C nella forma $E = (x_1, x_2, ..., x_n)$, dove x_i è il valore dell'attributo X_i. In fase di inferenza saranno forniti alla rete degli esempi che dovranno essere classificati. Secondo la regola di Bayes la probabilità di assegnare un valore c all'esempio $E = (x_1, x_2, ..., x_n)$ è data da

$$Pr(c|E) = \frac{Pr(E|c)Pr(c)}{Pr(E)} .$$

Si ricorda inoltre che l'evidenza E viene associata alla classe $c_+ \in C = \{c_+, c_-\}$ da un classificatore Bayesiano se

$$f_b(E) = \frac{Pr(C = c_+|E)}{Pr(C = c_-|E)} \geq 1 \ .$$

Date le ipotesi semplificative che esistano solo due valori per C (per esempio c_+ per 'attacco' e c_- per 'normale') e che le probabilità degli attributi siano indipendenti data la classe, cioè

$$Pr(E|c) = Pr(x_1, x_2, ..., x_n|c) = \prod_{i=1}^{n} Pr(x_i|c) \ ,$$

definiamo un classificatore Bayesiano *naive* con la funzione

$$f_{bn}(E) = \frac{Pr(C = c_+)}{Pr(C = c_-)} \prod_{i=1}^{n} \frac{Pr(x_i|C = c_+)}{Pr(x_i|C = c_-)} \ .$$

Una rete Bayesiana *naive* per la classificazione è una rete dove sono state assunte delle ipotesi esemplificative sugli attributi, chiamati anche *feature* o caratteristiche. Nello specifico ognuno di essi è supposto indipendente dagli altri data la classe di appartenenza. La struttura di una tale rete è rappresentata in Figura 9: il nodo rappresentante la variabile di classe si lega a tutti gli altri nodi con un legame di dipendenza condizionale, cioè con un arco nel grafo, in quanto la classe è supposta l'unica causa che incide sul valore delle altre variabili. Le caratteristiche invece non presentano archi uscenti [42].

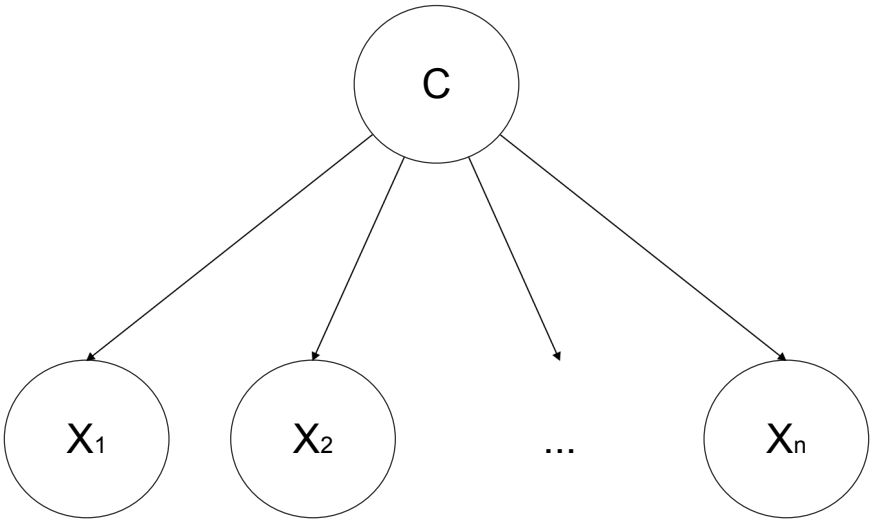

Figura 9: *Struttura di un classificatore naive Bayesiano. La classe C influisce su tutti gli attributi X, che sono indipendenti tra loro.*

3.3. Inferenza

L'inferenza probabilistica che ha luogo nelle reti Bayesiane consiste nel trovare le distribuzioni di probabilità di una o più variabili di interesse date le osservazioni effettuate su un insieme di altre variabili, dette *di evidenza*.

Per fare questo si ricorre al meccanismo della propagazione dell'evidenza, cioè l'aggiornamento e propagazione delle informazioni sulle distribuzioni di probabilità dai nodi di evidenza fino ai nodi di interesse. Questo è possibile poiché

$$Pr(X|E) = \alpha Pr(X, E) = \alpha \sum_{Y} Pr(X, Y, E) \ ,$$

dove X indica il set di nodi di interesse, E il set di nodi di evidenza, Y l'insieme degli altri nodi nella rete e α la costante di normalizzazione. Quindi la rete Bayesiana può trovare una risposta alle richieste che le vengono presentate semplicemente calcolando somme di prodotti di probabilità condizionate. Questo semplice approccio ha un grave difetto nella sua complessità temporale, che per n variabili binarie può diventare $O(n2^n)$ [31], rendendolo quindi impraticabile per reti complesse.

Si potrebbe pensare di modificare l'algoritmo per evitare la ripetizione di calcoli già effettuati in precedenza, conservando quindi i termini dei prodotti per riutilizzarli successivamente qualora non vengano modificati dalle evidenze raccolte. La versione più usata di questo approccio è l'algoritmo *Variable Elimination* ma nonostante il risparmio di operazioni il caso generale per nodi binari resta a complessità esponenziale $O(2^n)$[31].

L'inferenza nelle reti Bayesiane è infatti in generale un problema *NP-hard* [6], quindi non percorribile senza opportune restrizioni che permettano di ridurre la dimensionalità del problema, come ad esempio l'impiego di strutture poco connesse. Una di queste possibili strutture di rete è il *polytree* o *multi-albero*, nella quale tra due nodi esiste un solo percorso (senza considerare gli orientamenti degli archi) e dove quindi ogni nodo blocca il cammino tra i suoi antenati ed i suoi discendenti, permettendo di sfruttare una importante proprietà, la *d-separazione*.

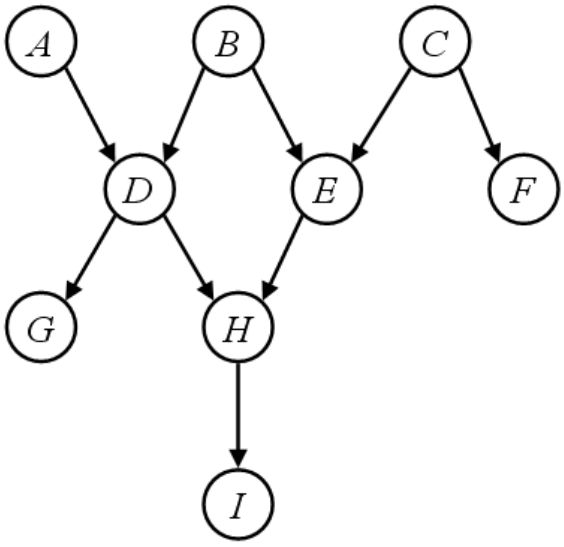

Figura 10: *Struttura di un multi-albero: tra un nodo ed i suoi discendenti esiste un solo percorso.*

3.3.1. D-separazione

Una proprietà che permette di rendere più trattabili i calcoli è la d-separazione delle variabili. La d-separazione riduce il numero di nodi coinvolti nell'inferenza, permettendo di non considerare quelle variabili che grazie all'indipendenza condizionale non aggiungono informazione. Dati due nodi nella rete Bayesiana, N_A ed N_B, possiamo asserire che sono condizionatamente indipendenti dato un insieme di nodi Γ se su ogni percorso non orientato nella rete tra N_A e N_B vi è almeno un nodo N_D che rispetta una delle seguenti proprietà:

- $N_D \in \Gamma$ ed entrambi gli archi del percorso partono da N_D;
- $N_D \in \Gamma$ e un arco del percorso incide su N_D, ed un altro parte da esso;
- $\{N_D \cup \Delta\} \notin \Gamma$ ed entrambi gli archi del percorso incidono su N_D (Δ indica l'insieme dei discendenti di N_D).

Se una di queste proprietà è verificata per ogni possibile percorso non orientato tra N_A e N_B, allora diciamo che Γ *d-separa* i due nodi e che quindi essi sono condizionatamente indipendenti dato l'insieme di evidenza Γ. Adottando topologie opportune per la rete Bayesiana, è possibile sfruttare questa proprietà per coinvolgere un minor numero di nodi nei processi inferenziali.

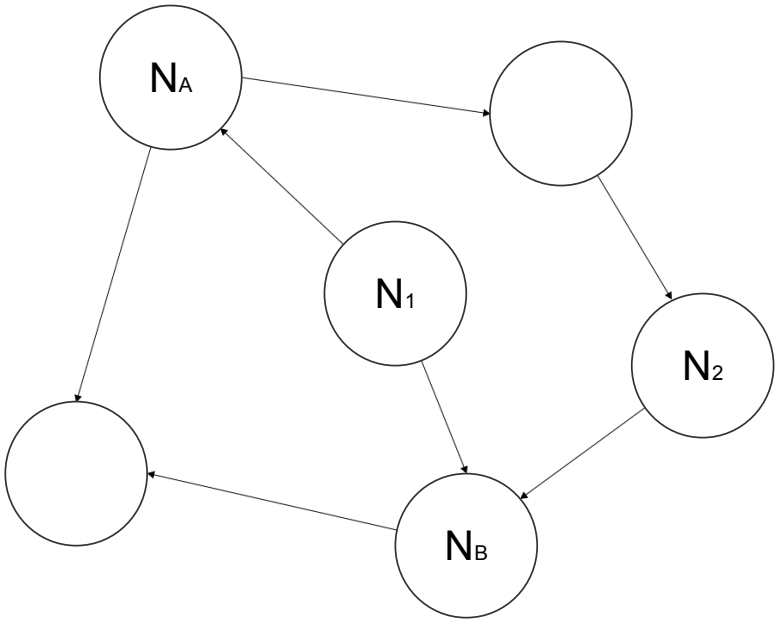

Figura 11: *Esempio di d-separazione. I nodi N_A e N_B sono d-separati dato l'insieme $\Gamma = \{N_1, N_2\}$.*

3.3.2. Inferenza esatta

Il semplice *belief updating* tramite eliminazione di variabile, grazie alla struttura particolare e la d-separazione, richiede nei multi-alberi una complessità lineare con la profondità della rete, ma come si è potuto vedere non è applicabile a topologie più complesse.

Esistono comunque altri algoritmi per i calcoli inferenziali, alcuni dei quali permettono l'utilizzo di strutture di rete anche più complesse dei multi-alberi. Verranno esposti brevemente gli algoritmi *Message passing* e *Junction tree*.

Message Passing generalizzato

Il *Message passing* generalizzato è un algoritmo per l'inferenza esatta sviluppato per i multi-alberi [30]. In questo algoritmo si ha che ogni nodo passa a genitori e figli un messaggio costituito da valori di probabilità aggiornati secondo l'evidenza raccolta.

Un vantaggio dell'algoritmo è il fatto che i calcoli da effettuare sono locali, cioè ogni nodo ha bisogno solo dei messaggi dei nodi direttamente connessi per poter aggiornare il proprio messaggio e poi inviarlo. Si ha però che al crescere del numero di genitori dei nodi di una rete l'algoritmo diventa rapidamente intrattabile. Un altro aspetto che ne riduce l'efficienza è il bisogno di specificare in anticipo la *query* probabilistica, rieseguendo l'algoritmo da zero per ogni richiesta. Questo può essere evitato costruendo delle strutture dati di servizio ed è l'idea alla base dell'algoritmo Junction Tree Propagation.

Junction Tree Propagation

L'algoritmo *Junction Tree Propagation* generalizza i concetti alla base del *Message passing*, compilando la rete in un insieme di strutture dati che raccolgono le probabilità dei nodi e che permettono l'esecuzione di più *query* evitando la riesecuzione dell'algoritmo per ognuna di esse.

Più in dettaglio, il metodo procede costruendo un albero non orientato dalla struttura molto più semplice della rete originale, chiamato appunto *Junction Tree*, i cui nodi sono *cluster* di variabili presi tra le variabili della rete Bayesiana, dove un *cluster* è definito come un sottoinsieme completamente connesso di nodi del grafo originario. Ognuno di questi *super-nodi* avrà una nuova distribuzione di probabilità che verrà utilizzata per i calcoli inferenziali.

Partendo dall'osservazione che si possono considerare le Tabelle di Probabilità Condizionata nelle reti Bayesiane come funzioni potenziali, l'idea alla base dell'algoritmo è quella di rappresentare la distribuzione di probabilità associata ad un grafo come il

prodotto dei potenziali dei suoi cluster:

$$Pr(X) = \frac{1}{Z} \prod_{C} \Psi_C(X_C) \ ,$$

dove:

- X rappresenta l'insieme delle variabili della rete Bayesiana, cioè il suo dominio;
- X_C è l'insieme delle variabili del cluster C.

I *cluster* devono essere costituiti in modo che, dati due *cluster* C_1 e C_2, ogni altro super-nodo sul cammino che li unisce contenga la loro intersezione $C_1 \cap C_2$. Inoltre ad ogni arco del *Junction Tree* deve essere associato una struttura chiamata *separatore*, contenente le variabili dell'intersezione dei due super-nodi che unisce.

La rappresentazione in termini di potenziale di un intero *Junction Tree* è quindi il prodotto dei potenziali dei *cluster* diviso il prodotto dei potenziali dei separatori:

$$Pr(\mathbf{X}) = \frac{\prod_{C} \Psi_C(\mathbf{X}_C)}{\prod_{S} \Phi_C(\mathbf{X}_S)} \ .$$

Una volta stabiliti i cluster e aggiornati i valori di probabilità delle nuove strutture per assicurare la consistenza dei valori contenuti, si può procedere al calcolo di $Pr(x_{query}|e)$ con l'incorporamento dell'evidenza nelle funzioni potenziali e la marginalizzazione secondo le variabili di interesse per trovare la probabilità congiunta della variabile richiesta e l'evidenza $Pr(x_{query}, e)$, normalizzando infine secondo $Pr(e)$ ed ottenendo così la probabilità risultante $Pr(x_{query}|e)$. Volendo rispondere ad una nuova query $Pr(x_{query}|e_{new})$, non sarà necessario ricostruire il *Junction Tree*, ma basterà aggiornare le funzioni potenziali [21].

Le complessità dei passi di costruzione dell'algoritmo sono generalmente lineari. La triangolarizzazione ottima, operazione necessaria se si vogliono ottenere i *cluster* di dimensione minima, è un problema *NP-completo*, ma avendo l'accorgimento di adottare criteri *greedy* per questo compito è possibile costruire un *junction tree* in tempo polinomiale, benché non è detto che sia il migliore possibile per la rete di partenza. L'inferenza è una operazione che richiede lo scambio di un numero di messaggi lineare rispetto al numero di *cluster*, ma ognuno di questi passaggi di informazioni richiede la marginalizzazione sulle distribuzioni di probabilità dei *super-nodi*, che ha complessità esponenziale nel numero di righe delle tabelle ad essi associati.

3.3.3. Inferenza approssimata

Si ricorre ad algoritmi di inferenza approssimata quando si ha a che fare con reti troppo ampie e connesse per poter effettuare l'inferenza esatta.

Per l'approssimazione dell'inferenza si usano algoritmi per la simulazione stocastica nei quali si generano una sequenza di campioni da una certa distribuzione di probabilità. Da questi campioni si calcola una probabilità a posteriori approssimata che si mostra convergere alla probabilità attesa al crescere del numero di campioni. In questo modo la precisione dell'approssimazione dipende solo dal numero di campioni calcolati e non dalla struttura della rete.

Questi algoritmi si dividono in due categorie principali: i metodi di campionamento stocastico e gli algoritmi che implementano tecniche *Markov Chain Monte Carlo (MCMC).*

Campionamento stocastico

In questa famiglia di metodi si generano campioni indipendenti gli uni dagli altri basandosi sulle probabilità delle variabili della rete. In base al metodo usato per costruire i campioni si hanno più varianti di questo approccio. La prima tecnica sviluppata è stata il *logic sampling* [20], nella quale si eseguono simulazioni del mondo descritto dalla rete Bayesiana seguendo gli archi della rete assegnando valori casuali alle variabili incontrate, secondo un modo di procedere definito *forward sampling.* Se questi valori assegnati sono in contrasto con le evidenze il campione viene eliminato, altrimenti lo si usa per misurare la frequenza dei valori assunti dalle variabili di interesse e questa frequenza rispetto al totale sarà il risultato dell'inferenza. Questo metodo, benché molto semplice, ha un importante svantaggio nel fatto che all'aumentare del numero di nodi di evidenza o in caso di combinazioni poco probabili di questi nodi, il numero di campioni che sopravviveranno alla selezione decrescerà in modo esponenziale rispetto al totale, richiedendo quindi sempre più tempo e sforzo computazionale per effettuare inferenza.

Per superare questo problema è stato introdotto il *Likelihood weighting* [15] che procede come il metodo precedente secondo l'approccio *forward sampling.* Una volta arrivato ai nodi di evidenza non assegna valori casuali ma i loro valori noti, associando al campione un peso calcolato sulla base della probabilità che le variabili di evidenza hanno di assumere quel valore. Il procedimento continua finché i pesi non soddisfano una funzione obiettivo. Questo metodo è generalmente molto più veloce del *logic sampling* e può trattare reti molto ampie, ma per evidenze poco probabili può richiedere comunque molto tempo per convergere [19].

Esistono anche tecniche basate sul *backward sampling*, dove il metodo parte dai nodi di evidenza e si muove a ritroso seguendo gli archi della rete. Questo metodo funziona meglio del *forward sampling* per evidenze poco probabili ma possono esserci casi in cui entrambi gli approcci convergano con difficoltà [27]. Sono stati sviluppati negli anni numerose varianti di questi approcci che ne aumentano l'accuratezza e la rapidità di

convergenza, usando tecniche di campionamento per importanza.

Campionamento Markov Chain Monte Carlo

Nel campionamento basato su tecniche MCMC si generano campioni che non sono indipendenti gli uni gli altri, sfruttando tecniche come il campionamento Metropolis-Hastings o quello di Gibbs che ne è una specializzazione. Sono tecniche che funzionano bene quando tra le probabilità condizionate non sono presenti valori estremi.

Questi metodi sono possibili grazie al fatto che per una catena di Markov X aperiodica ed irriducibile, con probabilità di transizione $Pr(x, y)$ e con distribuzione stazionaria π, si ha che

$$\lim_{N\to\infty} Pr^N(x, y) = \pi(X) \ ,$$

quindi anche se gli stati sono tra loro dipendenti la media campionaria degli stati della catena approssima il valore atteso della distribuzione stazionaria π.

Il campionamento Metropolis-Hastings è usato in statistica per generare campioni casuali che rispettino una data distribuzione di probabilità, costruendo una catena di Markov irriducibile e aperiodica che abbia la distribuzione ricercata come sua distribuzione stazionaria.

Per spiegare semplicemente il metodo si supponga di voler campionare una distribuzione di probabilità $f(x)$. Questa è la *distribuzione target* che possiamo arrivare ad approssimare a meno di una costante di proporzionalità. L'unico requisito che si chiede alla distribuzione target è la possibilità di calcolare il rapporto tra due valori da essa estratti.

L'algoritmo di Metropolis-Hastings procede come segue: si supponga al passo t di avere un valore corrente x_t ed una distribuzione proposta $q(x|x_t)$, a questo punto si eseguano le operazioni:

- si campioni un valore x^* da $q(x|x_t)$
- si calcoli la *probabilità di accettazione*:

  $$Pr(x_t, x^*) = min\left\{1, \frac{f(x^*)}{f(x_t)}\frac{q(x_t|x^*)}{q(x^*|x_t)}\right\} \ .$$

  Questa probabilità ci indica quanto il campione x^* è 'verosimile' rispetto la distribuzione target ed il valore precedente.
- in base alla probabilità di accettazione il campione corrente x^* viene accettato, ponendo $x_{t+1} = x^*$, o scartato, nel qual caso si pone $x_{t+1} = x_t$.

L'algoritmo può iniziare con un valore x_0 scelto casualmente dalla distribuzione $q(x)$ poiché l'efficacia del metodo non dipende dal valore iniziale ma solo dal numero di

campioni che vengono calcolati, ottenendo stime sempre più precise al crescere della sequenza di valori ottenuti.

Grazie a questo metodo è possibile campionare una distribuzione non nota che nel caso dell'inferenza Bayesiana è la distribuzione di probabilità che desideriamo ottenere dalla rete.

3.4. Apprendimento dei parametri

Lo scopo dell'apprendimento è quello di estrarre dal dataset informazioni sulle dipendenze delle variabili in modo da ottenere una struttura di rete e dei parametri, cioè le Tabelle di Probabilità Condizionata, che descrivano al meglio l'insieme di dati.

Quando la struttura della rete è nota si devono apprendere solo le Tabelle associate alle variabili. Distinguiamo due casi a seconda della completezza delle informazioni a nostra disposizione per effettuare il training della rete.

3.4.1. Dati completi

Nell'ipotesi di avere dati completi sul problema, si può adottare un approccio frequentistico per stabilire i valori delle Tabelle di Probabilità Condizionata $Pr(X_i|parents(X_i))$, cioè il rapporto tra il numero di volte in cui il nodo assume quel particolare valore X_i posto che i genitori si trovino negli stati considerati ed il numero di volte in cui i genitori assumono gli stati considerati a prescindere del valore del nodo.

$$Pr(X_i|parents(X_i)) = \frac{count(X_i|parents(X_i))}{count(parents(X_i))} \ .$$

Si procede in maniera analoga per le reti *naive*, dove interessa trovare il valore $Pr(x_i|c)$, contando le occorrenze dei valori delle variabili presenti nella rete e calcolandone il rapporto rispetto al totale. Se quindi il dataset contiene n esempi appartenenti alla classe c e di questi solo n_i possiedono il valore x_i per la *feature* X_i, allora

$$Pr(x_i|c) = \frac{n_i}{n} \ .$$

Si noti che con questa formula può verificarsi che $Pr(x_i|c) = 0$ se $n_i = 0$ cioè possono essere assegnate probabilità nulle se non ci sono esempi nel dataset in cui l'attributo $X_i = x_i$. Per evitare che questo avvenga, si somma alla probabilità una piccola quantità positiva in modo da rendere il valore x_i molto difficile da ottenere, ma non impossibile. Questo semplice metodo fornisce valori esattamente coerenti al dataset fornito.

3.4.2. Dati mancanti

Purtroppo nei casi reali non sempre si può avere disponibilità di dati completi, in quanto generalmente si ha a che fare con problemi i cui dati sono affetti da rumore o con dataset i cui record possono avere una o più variabili nascoste o con dati mancanti.

Per ricostruire le Tabelle di Probabilità Condizionata le due tecniche più usate sono la *Stima della Massima Verosimiglianza (Maximimum-Likelihood Estimation (MLE))* e l'Algoritmo *Expectation-Maximization.*

Stima della Massima Verosimiglianza

Il problema è il trovare le ipotesi $\boldsymbol{\theta}$ che a partire dai dati D massimizzino la stima della variabile X

$$Pr(X|D) = \sum_i Pr(X|D, \theta_i) Pr(\theta_i|D) = \sum_i Pr(X|\theta_i) Pr(\theta_i|D) \ .$$

Dato che risolvere questo problema in modo esatto è un compito troppo arduo si ricorre ad approssimazioni, limitandosi quindi a trovare quelle ipotesi θ_i che massimizzino $Pr(\theta_i|D)$. Usando la regola di Bayes abbiamo che

$$Pr(\theta_i|D) = \frac{Pr(D|\theta_i) Pr(\theta_i)}{Pr(D)} \ ,$$

il cui denominatore è costante, il che permette di limitarsi a massimizzare il numeratore, che è composto da due termini: la probabilità di osservare il dataset D qualora l'ipotesi θ_i sia verificata, e la probabilità a priori dell'ipotesi stessa. Dato che per la maggior parte dei problemi non si hanno informazioni specifiche per prediligere un particolare modello θ_i rispetto gli altri, si usa spesso una distribuzione uniforme. Resta quindi da massimizzare il termine $Pr(D|\theta_i)$ con una determinata ipotesi θ_{ML} denominata *Ipotesi di Massima Verosimiglianza (Maximum Likelihood, ML).* Per fare questo si ricorre ad un metodo chiamato di *Salita lungo il Gradiente* nel quale si segue la direzione di massima crescita della derivata.

Denotando con w_{ijk} una generica riga della Tabella di Probabilità Condizionata della variabile X_i, essa è data da

$$w_{ijk} = Pr(X_i = x_{ij} | parents(X_i) = u_{ik}) \ ,$$

dove i è l'indice della variabile, j quello del valore assunto in quella riga dalla variabile, e u_{ik} è la configurazione dei valori dei genitori di X_i in quella riga. Si procede quindi al calcolo del gradiente rispetto la generica riga della tabella, applicando inoltre una trasformazione logaritmica che permette di semplificare i passaggi intermedi, qui omessi, secondo la formula:

$$\frac{\delta ln Pr(D|H_i)}{\delta w_{ijk}} = \sum_{d \in D} \frac{Pr(y_{ij}, u_{ik}|d, H_i)}{w_{ijk}} \ ,$$

dove d è il singolo esempio estratto dall'insieme D. Una volta noto l'incremento si procede all'aggiornamento delle righe w_{ijk} applicando la formula

$$w_{ijk} \longleftarrow w_{ijk} + \eta \sum_{d \in D} \frac{Pr(y_{ij}, u_{ik}|d, H_i)}{w_{ijk}} \ ,$$

dove $0 < \eta \leq 1$ è una costante che determina la velocità di apprendimento. Alla fine dell'algoritmo si normalizzano le righe delle Tabelle di Probabilità Condizionata, in modo che siano rispettate le condizioni

$$\begin{cases} \sum_j w_{ijk} = 1 \\ 0 \leq w_{ijk} \leq 1 \end{cases} .$$

Questo algoritmo segue il massimo incremento del gradiente che riesce a calcolare, il che lo rende vulnerabile al problema dei massimi locali.

Algoritmo Expectation-Maximization

L'algoritmo *Expectation-Maximization (EM)* è un metodo iterativo per stimare i parametri di una distribuzione non nota basandosi sui campioni estratti da quella distribuzione. Anche in questo caso si deve agire sui parametri θ per massimizzare la

$$Pr(X_1, X_2, ..., X_n|\theta) = Pr(\mathbf{X}|\theta) .$$

Dato che i dati $\mathbf{X}$ non sono accessibili direttamente, ma solo attraverso le rispettive emissioni $\mathbf{Y}$, questo algoritmo alterna iterativamente una fase di *Expectation*, nella quale si calcolano le probabilità delle proprietà osservabili delle variabili a partire dall'ipotesi corrente θ_t nel calcolo di

$$Pr(\mathbf{Y}|\theta_t) ,$$

ed una fase detta di *Maximization*, dove invece si individua il nuovo set di parametri θ_{t+1} che rendano massima la verosimiglianza precedentemente calcolata

$$\theta_{t+1} = \arg\max_{\theta} Pr(\mathbf{Y}|\theta) .$$

Il metodo ripete l'alternanza dei due passi finché non vengono soddisfatte delle condizioni di convergenza sul set di parametri θ. Del metodo esistono numerose varianti che permettono di applicarlo in più ambiti, come per esempio la possibilità di calcolare in modo approssimato il passo di *Expectation* in modo da imparare più rapidamente reti molto complesse [31].

3.5. Apprendimento della struttura

Quando la struttura della rete non è nota, come può succedere in molti casi reali, si applicano tecniche che permettano di calcolare non solo le Tabelle di Probabilità Condizionata ma anche le relazioni tra le variabili e quindi tra i nodi.

Anche nell'apprendimento strutturale esiste la differenza tra metodi che si basano su dati completi ed incompleti. Questi ultimi adottano tecniche di campionamento

stocastico per prendere decisioni sulla topologia, in maniera analoga a ciò che si fa nell'apprendimento dei parametri, per esempio sfruttando l'algoritmo EM [14].

Ci si concentrerà principalmente sui due principali metodi nell'ambito della disponibilità di dati completi. Entrambi impiegano euristiche per riuscire a trattare il problema, che è *NP-hard*. Il primo approccio è basato sull'*analisi delle dipendenze (constraint-based)* ed il secondo sull'utilizzo di funzioni di *score*.

3.5.1. Analisi delle Dipendenze

Nell'analisi delle dipendenze, detti anche algoritmi *constraint-based*, si utilizzano i dati per cercare dipendenze tra le variabili, aggiungendo un arco nel grafo ogni volta che l'esito di un qualche test di indipendenza condizionale ne indica la necessità. Quando tutti i test sono stati effettuati si ottiene un grafo non orientato ai cui archi verranno successivamente imposti i relativi versi sulla base di apposite regole. Gli algoritmi *constraint-based* forniscono la struttura corretta senza risentire dei massimi locali, contrariamente alle funzioni euristiche di ricerca, se due ipotesi sulla distribuzione di probabilità implicata dai dati sono verificate:

- questa distribuzione di probabilità è esattamente nota, quindi non solo in senso approssimato;
- la distribuzione di probabilità rispetta l'*ipotesi di fedeltà*, proprietà che può essere riassunta con l'impossibilità che esistano nodi condizionatamente indipendenti senza che siano anche d-separati [32].

Nella pratica queste due assunzioni non devono verificarsi necessariamente, poiché è necessario che reggano in senso asintotico, cioè quando una quantità infinita di dati nel dataset di addestramento è disponibile. Questi algoritmi dipendono dalle soglie imposte sui test per la ricerca delle indipendenze condizionali e forniscono reti meno accurate al diminuire della dimensione del dataset, sopratutto quando sono presenti numerosi legami condizionali [35].

3.5.2. Metodi Search and Score

Nell'approccio *Search and Score* si usa invece una funzione di *fitness* ed una strategia di ricerca per esplorare lo spazio delle possibili reti Bayesiane cercando quella che si adatti meglio ai dati. Questo problema è in generale intrattabile, e può essere affrontato efficacemente solo imponendo alcune restrizioni sulla struttura della rete ricercata [13].

Come strategie di ricerca si adottano spesso quelle *greedy* per la semplicità di implementazione, tuttavia esistono ampie possibilità per scelte alternative come gli approcci basati sulla programmazione genetica [25], la ricerca *branch and bound* [37], ed altri.

La *Minimal Description Lenght* è un esempio di funzione di *score* basato su concetti della teoria dell'informazione. Con questa politica si cerca di imparare la rete che abbia la minima lunghezza di descrizione nella struttura e nei parametri, cioè nel numero di archi e nel numero di Tabelle di Probabilità Condizionata e delle relative righe. Una rete Bayesiana definisce una distribuzione di probabilità P_{BN} sui dati da essa rappresentata. Usando tale distribuzione si può costruire uno schema di codifica che assegni parole più brevi ai simboli più frequenti in modo da ottenere una rete B la cui descrizione abbia la minima lunghezza.

Denominando con B la rete Bayesiana ottenuta e con $D = \{d_1, d_2, ..., d_N\}$ il dataset di apprendimento, la funzione di score MDL può essere scritta come

$$f_{MDL}(B|D) = \frac{\log N}{2}|B| - \sum_{i=1}^{N} \log P_{BN}(d_i) \ ,$$

dove il primo termine rappresenta la lunghezza della descrizione dei parametri della rete, indicando che vengono usati $\frac{\log N}{2}$ bit per ogni parametro, ed il secondo termine indica il numero di bit che sono necessari per descrivere D basandosi sulla distribuzione di probabilità P_{BN}. Quest'ultimo termine è massimo quando P_{BN} è uguale alla *distribuzione empirica*, cioè la distribuzione ottenuta dalle frequenze relative di ogni evento del dataset D rispetto al totale, favorendo le reti che riescano ad approssimare al meglio i dati originari. Questo parametro da solo non è però sufficiente dacché favorisce reti totalmente connesse.

Una rete totalmente connessa è uno scenario non desiderabile poiché ogni nodo è rappresentato come dipendente dagli altri e quindi si perde il vantaggio delle reti Bayesiane di raffigurare visualmente le indipendenze probabilistiche. Inoltre in queste reti ogni nodo ha un grande numero di genitori, il che implica che la rete sarà composta da un numero esponenziale di parametri, la maggior parte dei quali avrà un'altissima varianza, e l'inferenza su queste reti produrrà previsioni sempre meno affidabili.

Questo problema è noto come *overfitting* o sovradattamento, nel quale la rete sarà perfettamente adattata ai dati di addestramento ma fornirà performance pessime su dati che vi si discostano anche di poco, come quelli di test.

Per bilanciare questa tendenza della rete al sovradattamento entra in gioco il primo termine della funzione, che offre una rappresentazione della complessità della rete, penalizzando quelle molto complesse, in favore delle reti dalla struttura più essenziale.

Capitolo IV

Sistema Bayesiano per il Rilevamento

Il raffinamento delle tecniche di rilevazione delle anomalie ha fatto sì che questo approccio venisse usato con più frequenza in soluzioni commerciali. Benché i *signature-based* IDS siano ancora i preferiti dai produttori software per la loro semplice espandibilità e manutenibilità, sono disponibili e ben diffusi gli IDS che affiancano la rilevazione delle anomalie al metodo basato sulle *signature*.

In questo lavoro di tesi è stato sviluppato un rilevatore Bayesiano basato su un classificatore *naive*, cioè una rete Bayesiana dove si assume che la classe di appartenenza del record influisca su tutte le altre caratteristiche estratte e che queste, nota la classe, siano stocasticamente indipendenti tra loro. Questo rilevatore appartiene alla categoria degli *anomaly-based network* IDS, in quanto si trova a rilevare nelle caratteristiche estratte dai flussi di rete degli scostamenti dagli andamenti tipici di queste *feature* che influenzeranno la probabilità del fatto che un attacco sia in corso o meno. La rete potrà discriminare i comportamenti "ammessi" da quelli "anomali" grazie ad una fase di addestramento nella quale un insieme di record annotati, cioè già classificati, saranno presentati alla rete stessa, così da permettere l'aggiornamento delle Tabelle di Probabilità Condizionata.

Il rilevatore così ottenuto sarà addestrato e testato con l'ausilio di un dataset, che nell'ambito della rilevazione delle intrusioni è una collezione di caratteristiche estratte da tracce di connessioni e pacchetti di rete. Per motivi di efficienza computazionale e precisione nella rilevazione è preferibile non utilizzare tutte le caratteristiche offerte dai dataset, rendendo quindi necessario l'applicazione di tecniche di *feature selection*. In questo lavoro di tesi si è implementato un algoritmo *greedy* per trovare il sottoinsieme ottimale delle caratteristiche che permettano al rilevatore *naive* Bayesiano di effettuare rilevazioni il più precise possibile degli attacchi.

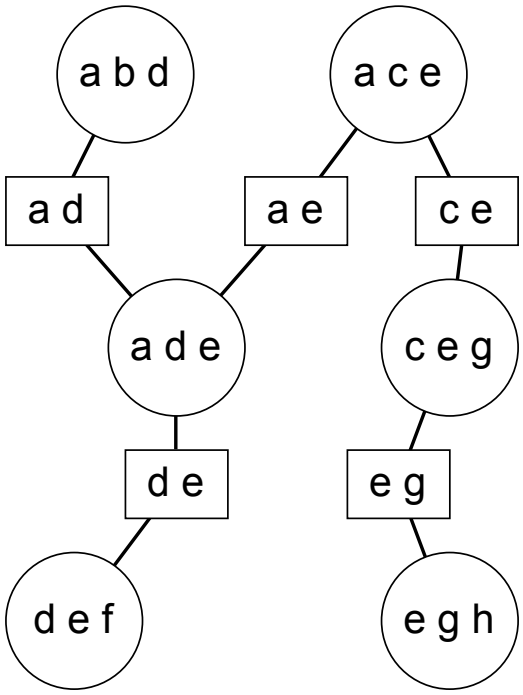

Figura 12: *Esempio di junction tree calcolato su un grafo dotato dell'insieme di nodi* $\Gamma = \{a, b, c, d, e, f, g, h\}$*. Si notino i cluster di variabili, indicati dentro i cerchi, ed i separatori, indicati dentro i rettangoli. I separatori devono contenere le variabili in comune tra due cluster adiacenti.*

4.1. Apprendimento della rete Bayesiana

Un classificatore Bayesiano utilizza un insieme di caratteristiche estratte dai dati grezzi per condurre calcoli inferenziali in modo da rispondere alle *query* probabilistiche che gli vengono poste. Queste caratteristiche dovrebbero essere in numero minore possibile, così da rendere le operazioni di calcolo più rapide dato che l'inferenza Bayesiana è un problema che può diventare intrattabile in strutture di rete poco favorevoli. Nel presente lavoro di tesi è stato adottato un rilevatore *naive* Bayesiano, nel quale le caratteristiche dipendono solo dalla classe, il che permette di semplificare le operazioni di calcolo. Tuttavia l'ottimizzazione resta un obiettivo desiderabile ed in quest'ottica si è scelto di non adottare tutte le possibili caratteristiche, ma di ricercarne il sottoinsieme migliore in termini di correlazione con la classe, utilizzando una euristica per la ricerca nello spazio delle *feature*. Questa scelta è giustificata anche dall'osservazione del fatto che non tutte le caratteristiche sono significative per l'identificazione della classe di appartenenza di un record, dato che alcune di esse si trovano ad essere maggiormente correlate ad altre *feature* piuttosto che alla classe stessa.

Gli algoritmi di ricerca basati su euristiche *best-first* o *greedy* cercano una soluzione

effettuando ad ogni passo la scelta localmente ottima, sperando di giungere ad un massimo globale. Questo non sempre succede poiché se l'algoritmo incontra un massimo locale non sarà più in grado di allontanarsene, rinunciando a muoversi in altre regioni dello spazio delle soluzioni nonostante la possibilità di precludersi il raggiungimento della soluzione ottima. Tuttavia le soluzioni trovate sono spesso accettabili approssimazioni della soluzione migliore e sopratutto vengono calcolate in tempi più brevi, rendendo questi algoritmi indispensabili quando si ha a che fare con problemi computazionalmente intrattabili per la dimensione del loro dominio.

L'algoritmo che è stato usato in questo lavoro cerca i gruppi di caratteristiche che diano risultati migliori in termini di precisione nella rilevazione degli attacchi, iniziando con la scelta della singola *feature* capace di garantire i migliori risultati ed intraprendendo successivamente un ciclo di confronti con tutte le altre caratteristiche cercando una seconda caratteristica da affiancare alla prima in modo che la coppia delle due offra rilevazioni più attendibili e così via. La ricerca viene fermata quando alla fine di un ciclo di confronti si ha che nessuna delle caratteristiche rimanenti riesca a migliorare le prestazioni del gruppo già stabilito nelle iterazioni precedenti.

Nella ricerca delle *feature* più significative per la rilevazione delle intrusioni con un rilevatore *naive* Bayesiano sarà necessario confrontare risultati di rilevazioni basate su set diversi di *feature*. Avendo n caratteristiche una ricerca esaustiva richiederebbe un numero di confronti dell'ordine di $O(n^2)$. Per ogni confronto bisogna inoltre considerare il tempo necessario alla costruzione e interrogazione della rete Bayesiana che dipende dal numero di caratteristiche scelte. Per ogni set di *feature* viene infatti costruita, compilata (per permettere l'immissione delle evidenze e l'utilizzo del *junction tree*) ed interrogata una rete Bayesiana e non è possibile ignorare il carico computazionale aggiunto da queste operazioni.

La costruzione della rete consta di operazioni lineari, a differenza della compilazione. In quest'ultimo passaggio andrebbero infatti applicati algoritmi molto onerosi computazionalmente, in quanto il problema della ricerca del *junction tree* ottimale è *NP-completo*; tuttavia le euristiche utilizzate per aggirare questo problema permettono di ottenere un risultato, seppur non necessariamente ottimo, in tempo polinomiale. L'inferenza sulla rete è invece generalmente esponenziale nell'ampiezza delle tabelle di probabilità associate ai *cluster* di nodi. Nel caso specifico di un classificatore *naive* Bayesiano, data la semplice topologia di rete, possiamo assumere che l'onere computazionale T delle operazioni inferenziali sia lineare con il numero di nodi di evidenza. Possiamo quindi calcolare una approssimazione della complessità temporale con la formula

$$\sum_{i=1}^{n}(n-i+1)iT \leq nT\sum_{i=1}^{n}(n-i+1) = nT\sum_{i=1}^{n} i = Tn\frac{n(n+1)}{2} \approx O(n^3)T \ ,$$

dove il termine iT rappresenta il tempo impiegato dalla rete per l'inferenza quando sono presenti i nodi, mentre il termine $(n - i + 1)$ indica il fatto che alla prima iterazione si effettuano n test e si sceglie la caratteristiche migliore, alla seconda iterazione si effettuano $n - 1$ test per verificare le prestazioni della coppia composta dalla *feature* precedentemente scelta e a turno ognuna delle rimanenti, e così via.

Una complessità temporale cubica nel numero di caratteristiche è computazionalmente pesante ma con l'applicazione di una euristica è possibile abbattere i tempi di calcolo. Si è scelto di modificare la ricerca classica includendo una euristica *best-first*, in modo tale che per ogni sequenza di test si scelga di adottare la caratteristica che faccia aumentare maggiormente le performance di rilevazione. Se alla fine di una iterazione di test si ha che nessuna delle caratteristiche provate ottiene un aumento delle prestazioni, nessuna di esse verrà scelta per entrare a far parte del gruppo per continuare i test. La composizione del gruppo corrente e le sue prestazioni saranno in quel caso salvate altrove per confrontarle con gli altri gruppi che verranno creati a partire da caratteristiche iniziali diverse.

Per quanto riguarda le *feature* iniziali è stato scelto di selezionarne una sola per volta con un ciclo esterno all'iterazione di test e, a partire da quella, svolgere tutti i confronti con le altre caratteristiche. Questa scelta è giustificata dal fatto che le *feature* non sono scorrelate tra loro ed una particolare caratteristica iniziale può cambiare le performance e le *feature* scelte successivamente per far parte dell'insieme durante le iterazioni di test.

Le performance tra iterazioni vengono confrontate per stabilire quale set di caratteristiche abbia una maggiore precisione nella rilevazione; questo confronto è effettuato tramite i parametri di sensitività e specificità misurati per ogni gruppo di *feature*. Più in dettaglio, si era inizialmente posto che una caratteristica venisse aggiunta al gruppo già costituito se entrambi i parametri fossero stati strettamente maggiori di quelli misurati prima dell'aggiunta della *feature* ma negli esperimenti si è notato che un'alta sensitività è generalmente più difficoltosa da raggiungere, al contrario di ciò che avviene per la specificità. Dato che uno scenario del genere implica un numero maggiore di falsi negativi, cioè di attacchi non rilevati, è stato scelto di permettere una tolleranza fino al 20% nei confronti effettuati sulla specificità. Questo significa che durante una iterazione di test è possibile accogliere una *feature* che apporti una specificità minore fino al 20% rispetto ai parametri ottenuti dalla rilevazione effettuata senza questa caratteristica ma che al contempo fornisca una sensitività strettamente maggiore.

Grazie a questa modifica si sono ottenuti risultati che mostrano una diminuzione dei falsi negativi a discapito di un aumento dei falsi positivi, tuttavia si è ritenuto che un numero maggiore di falsi allarmi fosse un compromesso accettabile in cambio dell'ottenimento di rilevazioni più efficaci.

Algoritmo 1 Algoritmo *greedy* per la ricerca dell'insieme di caratteristiche che forniscano rilevazioni migliori

1: $added \leftarrow$ true
2: $max\_performance \leftarrow 0$ ▷ la variabile max_performance conterrà la performance migliore di un ciclo di test
3: $temp\_feature \leftarrow$ null
4: **for** i **from** 1 **to** $num\_features$ **do**
5: $Group \leftarrow feature[i]$ ▷ sceglie la prima *feature* del gruppo
6: $group\_performance \leftarrow$ naiveBayesTest($feature[i]$)
7: **while** $added =$ true **do** ▷ se nessuna *feature* aumenta le performance di rilevazione, termina l'iterazione di test
8: $added =$ false
9: **for each** $feature\ f \notin Group$ **do**
10: $temp\_group \leftarrow Group + f$
11: $temp\_performance \leftarrow$ naiveBayesTest($temp\_group$)
12: **if** $temp\_performance > max\_performance$ **then**
13: $max\_performance \leftarrow temp\_performance$
14: $temp\_feature \leftarrow f$
15: **end if**
16: **end for each**
17: **if** $max\_performance > group\_performance$ **then** ▷ se una delle *feature* testate ha aumentato la precisione della rilevazione, sarà aggiunta al gruppo già stabilito
18: $group\_performance \leftarrow max\_performance$
19: $Group \leftarrow Group + temp\_feature$
20: $added \leftarrow true$
21: **end if**
22: **end while**
23: save($Group, group\_performance$) ▷ salvataggio della associazione tra gruppo e performance
24: **end for**

4.2. Strumenti software

Il rilevatore Bayesiano è stato implementato in Java. Ci si è avvalsi del framework Netica [29] che permette la creazione, l'addestramento ed il testing di reti Bayesiane. Il train set è stato convertito in un formato leggibile da Netica e dato in input alla rete perché si addestrasse secondo un semplice approccio frequentistico, reso possibile dal fatto che il dataset non ha record con valori mancanti o rumorosi.

4.2.1. Ambiente di sviluppo Java

Come ambiente di sviluppo per il codice Java è stato usato Eclipse. Eclipse è una piattaforma di sviluppo *open-source* realizzata dalla Eclipse Foundation. Questo ambiente di sviluppo è ottimizzato per la programmazione Java ma è possibile estenderlo ad altri linguaggi tramite gli appositi plugin [11].

4.2.2. Framework per Reti Bayesiane

Netica, sviluppata dalla Norsys Software Corporation, è un'applicazione multipiattaforma che permette di costruire una rete Bayesiana e di utilizzarla per l'inferenza. Tra le sue funzionalità c'è la possibilità di leggere i dati per l'addestramento da file o database e di poter visualizzare graficamente la rete ed i suoi parametri.

Oltre all'applicazione vera e propria sono disponibili delle API in vari linguaggi, tra i quali Java. Le funzionalità fornite da queste classi sono state integrate nel codice sviluppato in modo da ridurre i tempi di sviluppo e sfruttare le potenzialità di questo framework per l'inferenza Bayesiana.

4.2.3. Framework per il data mining

Dopo l'esecuzione dell'algoritmo si è scelto di confrontare i risultati ottenuti con quelli suggeriti da un framework per il *machine learning*. Weka è un framework sotto licenza GNU GPL che mette a disposizione algoritmi e strumenti per il *data mining*, scritto in Java e sviluppato dall'Università di Waikato.

4.3. Valutazione sperimentale

Nell'ambito delle funzioni di classificazione binaria definiamo alcuni parametri utili per il confronto di risultati di classificazioni diverse. Questi parametri sono:

- veri positivi o *True Positive (TP)* le istanze del problema correttamente identificate come casi positivi;
- falsi positivi, *False Positive (FP)* o Errori di Tipo I le istanze del problema erroneamente identificate come casi positivi;

- veri negativi o *True Negative (TN)* le istanze del problema correttamente identificate come casi negativi;
- falsi negativi, *False Negative (FN)* o Errori di Tipo II le istanze del problema erroneamente identificate come casi negativi.

Su queste entità si basano due misure statistiche particolarmente significative: la sensitività e la specificità. La sensitività (detta anche tasso dei veri positivi, *true positive rate*) misura la quantità di veri positivi che sono stati classificati correttamente secondo la formula

$$\textit{Sensitività} = \frac{TP}{TP + FN} \ .$$

La specificità (nota anche come tasso dei veri negativi, *true negative rate*) indica la proporzione di veri negativi correttamente identificati, seguendo la legge

$$\textit{Specificità} = \frac{TN}{TN + FP} \ .$$

Un classificatore perfetto dovrebbe fornire il 100% di sensitività (identificando *tutti* i veri positivi) e specificità (identificando *solo* i veri positivi).

4.3.1. Caratteristiche del Dataset

Per le operazioni di training e testing è stato scelto il dataset NSL-KDD. Esso è stato costruito a partire dal dataset KDDCup'99 [23], che venne formato proprio per essere usato nell'ambito della rilevazione delle intrusioni dalla collaborazione tra il *Defense Advanced Research Projects Agency (DARPA)*, l'*Air Force Research Laboratory (AFRL)* ed il *Massachusetts Institute of Technology (MIT) Lincoln Laboratory*, e rimane uno dei pochi dataset pubblici disponibili per questo settore.

I record del dataset sono stati costituiti analizzando il traffico della rete simulata (sia nel background che negli attacchi) nel MIT *Lincoln Laboratory* durante il programma *DARPA99 IDS Valutation Program*, che è risultato in 7 settimane di traffico di rete, principalmente in formato *tcpdump*. Da questi dati grezzi sono stati raccolti circa 5 milioni di connessioni nel training set, formato a partire dalle prime 5 settimane, e circa 2 milioni di record nelle restanti 2 settimane per il test set. Per ogni record sono state registrate 41 caratteristiche e la classe (connessione legittima o il tipo di attacco).

Le caratteristiche possono essere raccolte in tre gruppi:

1. caratteristiche di base, quelle che è possibile estrarre da una singola connessione TCP o da pacchetti IP, descritte nella Tabella 1;
2. caratteristiche del contenuto, estratte per rilevare quegli attacchi che vengono perpetrati tramite servizi di alto livello, le cui tracce vanno cercate nella porzione dati dei pacchetti, descritte nella Tabella 2;

3. caratteristiche del traffico, descritte nelle Tabelle 3 e 4, che vengono calcolate sulle connessioni che rientrano in due finestre, la prima costituita dalle connessioni che hanno avuto luogo negli ultimi 2 secondi e la seconda dalle ultime 100 connessioni. Queste caratteristiche sono suddivise in ulteriori due gruppi: quelle calcolate sulle connessioni dirette verso lo *stesso host* della connessione attualmente attiva, e quelle dirette verso lo *stesso servizio* della connessione attuale.

Tabella 1: *Caratteristiche di base estratte dalle connessioni del KDDCup'99 dataset*

Nome	**Descrizione**	**Tipo**
duration	durata in secondi della connessione	intero
protocol_type	tipo del protocollo (TCP, UDP, ICMP)	nominale
service	servizio offerto alla destinazione (HTTP, telnet, ecc.)	nominale
src_bytes	numero di bytes di dati dalla sorgente alla destinazione	intero
dst_bytes	numero di bytes di dati dalla destinazione alla sorgente	intero
flag	status della connessione	nominale
land	vale 1 se la connessione è da/verso lo stesso host/porta	binario
wrong_fragment	numero di pacchetti "errati"	intero
urgent	numero di pacchetti urgenti	intero

Tabella 2: *Caratteristiche relative al contenuto delle connessioni estratte dal KDDCup'99 dataset*

Nome	**Descrizione**	**Tipo**
hot	numero di accessi a cartelle di sistema e creazioni ed esecuzioni di programmi	intero
num_failed_logins	numero di tentativi falliti di login	intero
logged_in	vale 1 se un tentativo di login è andato a buon fine, altrimenti 0	binario
num_compromised	numero di volte in cui non è stato trovato un file/percorso e comandi di salto	intero
root_shell	vale 1 se viene ottenuta la shell di root, altrimenti 0	binario
su_attempted	vale 1 se viene usato il comando *'su root'*, altrimenti 0	binario
num_root	numero di accessi come utente root	intero
num_file_creations	numero di file creati	intero
num_shells	numero di prompt di comandi invocate	intero
num_access_files	numero di operazioni su file per il controllo degli accessi	intero
num_outbound_cmds	numero di comandi in uscita in una sessione FTP	intero
is_hot_login	vale 1 se il login effettuato è nella lista "hot"	binario
is_guest_login	vale 1 se il login effettuato è di un utente "guest"	binario

Tabella 3: *Caratteristiche relative al traffico estratte dalle connessioni del KDDCup'99 dataset da una finestra composta dalle connessioni avute luogo negli ultimi 2 secondi*

Nome	**Descrizione**	**Tipo**
count	numero di connessioni allo stesso host	intero
serror_rate[1]	percentuale di connessioni che hanno errori "SYN"	reale
rerror_rate[1]	percentuale di connessioni che hanno errori "REJ"	reale
same_srv_rate[1]	percentuale di connessioni allo stesso servizio	reale
diff_srv_rate[1]	percentuale di connessioni ad un servizio differente	reale
srv_count[1]	numero di connessioni allo stesso servizio	intero
srv_serror_rate[2]	percentuale di connessioni che hanno errori "SYN"	reale
srv_rerror_rate[2]	percentuale di connessioni che hanno errori "REJ"	reale
srv_diff_host_rate[2]	percentuale di connessioni ad host differenti	reale

Tabella 4: *Caratteristiche relative al traffico estratte dalle connessioni del KDDCup'99 dataset da una finestra composta dalle ultime 100 connessioni*

Nome	**Descrizione**	**Tipo**
dst_host_count	numero di connessioni allo stesso host	intero
dst_host_srv_count	numero di connessioni allo stesso servizio	intero
dst_host_same_srv_rate	percentuale di connessioni allo stesso servizio	reale
dst_host_diff_srv_rate	percentuale di connessioni ad un servizio differente	reale
dst_host_same_src_port_rate	percentuale di connessioni in cui le porte di origine e destinazione sono le stesse	reale
dst_host_srv_diff_host_rate[2]	percentuale di connessioni ad host differenti	continuo
dst_host_serror_rate[1]	percentuale di connessioni che hanno errori "SYN"	reale
dst_host_srv_serror_rate[2]	percentuale di connessioni che hanno errori "SYN"	reale
dst_host_rerror_rate[1]	percentuale di connessioni che hanno errori "REJ"	reale
dst_host_srv_rerror_rate[2]	percentuale di connessioni che hanno errori "REJ"	reale

I tipi di attacco che sono stati simulati nel training set sono 24, che ricadono in quattro categorie:

1. *Denial of Service* (DoS), nel quale si cerca di rendere insufficiente la memoria o la capacità di calcolo impedendo di servire gli utenti legittimi, negando loro l'accesso alla risorsa;

2. *User to Root* (U2R), che consiste nel tentativo di un utente con credenziali limitate di accedere a privilegi di amministratore;

[1]Caratteristica che si riferisce solo alle connessioni verso gli stessi host
[2]Caratteristica che si riferisce solo alle connessioni verso gli stessi servizi

3. *Remote to Local* (R2L), dove un utente remoto cerca di sfruttare qualche vulnerabilità del sistema per acquisire l'accesso locale alla risorsa, ottenendo di autenticarsi come un utente legittimo del sistema;

4. *Probing*, cioè il tentativo di analizzare qualche caratteristica della risorsa o della rete per scoprire se offre qualche vulnerabilità che è possibile sfruttare.

Per aumentare il realismo della simulazione nel test set sono stati inclusi in più 14 attacchi diversi, così da avere una distribuzione di probabilità degli attacchi profondamente diversa che nel training set.

Purtroppo però il KDDCup'99 non è un dataset ottimale, poiché presenta un grande numero di record ripetuti sia del train set che del test set, come si può verificare dai dati raccolti nelle Tabelle 5 e 6. Questo fa sì che l'apprendimento degli algoritmi venga squilibrato per favorire la rilevazione dei record più frequenti, penalizzando gli attacchi presenti in numero minore di volte ma che possono essere anche più pericolosi e facendo in modo che anche la fase di test venga falsata, finendo per fornire risultati troppo ottimistici a causa della rilevazione ripetuta di record già incontrati.

Tabella 5: *Statistiche sui dati ridondanti nel KDDCup'99 train set*

	Records Originali	**Records Distinti**	**Rapporto di riduzione**
Attacchi	3.925.650	262.178	$93,32\%$
Non attacchi	972.781	812.814	$16,44\%$
Totale	4.898.431	1.074.992	$78,05\%$

Tabella 6: *Statistiche sui dati ridondanti nel KDDCup'99 test set*

	Records Originali	**Records Distinti**	**Rapporto di riduzione**
Attacchi	250.436	29.378	$88,26\%$
Non attacchi	60.591	47.911	$20,92\%$
Totale	311.027	77.289	$75,15\%$

Per superare i problemi del KDDCup'99 è stato costruito il dataset NSL-KDD [38], che ne costituisce una versione ridotta e priva di duplicati. La mancanza di duplicati elimina la distorsione introdotta nel dataset originale, rendendo l'apprendimento ed il testing dipendenti solo dalla bontà del rilevatore e non dalla frequenza del record.

Il nuovo dataset comprende le stesse feature del precedente: 41 caratteristiche per ogni connessione più la classe di appartenenza, aggiungendo in più un valore di difficoltà costituito dal numero di rilevatori usati durante la sua costruzione che ha classificato

correttamente il record, un concetto introdotto nel dataset NSL-KDD dove 21 rilevatori sono stati addestrati su porzioni diverse del train set del KDDCup'99 ed usati per etichettare l'intero dataset, ottenendo così 21 possibili classi per ogni record e indicando il *livello di difficoltà* come il numero di etichette correttamente assegnate.

Infine un ulteriore vantaggio di questo dataset è la compattezza, il che lo rende facilmente trasportabile ed accorcia i tempi di apprendimento e testing. Consta nella sua totalità di meno di 150.000 record, un numero non elevato, e costituisce un sottoinsieme casuale del KDDCup'99 privato di duplicati. Le statistiche di questo campione sono mostrate nelle Tabelle 7 e 8, dove sono presentate raccolte per livello di difficoltà.

Tabella 7: *Statistiche sul campione scelto dal KDDCup'99 train set*

Livello di difficoltà	Records Distinti	Percentuale	Records Selezionati
0-5	407	0,04	407
6-10	768	0,07	767
11-15	6.525	0,61	6.485
16-20	58.995	5,49	55.757
21	1.008.297	93,80	62.557
Totale	1.074.992	100,00	125.973

Tabella 8: *Statistiche sul campione scelto dal KDDCup'99 test set*

Livello di difficoltà	Records Distinti	Percentuale	Records Selezionati
0-5	589	0,76	585
6-10	847	1,10	838
11-15	3.540	4,58	3.378
16-20	7.845	10,15	7.049
21	64.468	83,41	10.694
Totale	77.289	100,00	22.544

4.3.2. Risultati Sperimentali

L'euristica adottata ha ottenuto risultati interessanti sia dal punto di vista della complessità effettiva dell'algoritmo di ricercad sia dal punto di vista delle rilevazioni.

Il metodo adottato ha abbattuto la complessità temporale dell'algoritmo di ricerca esaustiva, che era dell'ordine di $O(n^3)$, fermando le iterazioni di test in media dopo meno di una decina di tentativi per ogni *feature* iniziale, portando quindi la complessità ad un più trattabile $O(n)$. I risultati di rilevazione sono stati generalmente buoni. Ci sono stati casi in cui, come era prevedibile che accadesse, si sono verificate esecuzioni a

partire da alcune *feature* che hanno raggiunto subito dei massimi locali, nei quali una delle due caratteristiche restava a valori modesti mentre l'altra raggiungeva percentuali troppo elevate per poter essere superata dalle iterazioni successive, impedendo quindi all'altro parametro il raggiungimento di risultati migliori. Altre esecuzioni hanno invece raggiunto dei buoni tassi di rilevazione, si sono infatti ottenuti valori di specificità e sensitività dell'ordine dell'80%.

Nella Tabella 9 sono indicati i migliori risultati ottenuti. La prima colonna contiene la *feature* iniziale a partire dalla quale è stata condotta la ricerca, nella seconda colonna sono indicate le caratteristiche scelte dall'algoritmo e nelle ultime due i valori di sensitività e specificità rilevati per quel gruppo di *feature*. Si noti che il primo risultato è costituito da due parametri molto simili, mentre gli altri presentano una differenza di almeno dieci punti percentuali tra un valore e l'altro con quello di sensitività generalmente inferiore. Si noti inoltre che il secondo ed il quarto risultato offrono le maggiori sensitività, con il secondo che ha un rapporto quasi ottimo di falsi negativi su negativi, laddove il quarto sacrifica quest'ultimo aspetto in cambio di una sensitività leggermente maggiore.

Sono state infine scelte delle caratteristiche tramite l'algoritmo *RankSearch*, eseguito sul framework *Weka (Waikato Environment for Knowledge Analysis)*[39], per confrontare la qualità delle rilevazioni effettuate tramite le caratteristiche ottenute sperimentalmente con quelle suggerite. L'algoritmo *RankSearch* ha complessità lineare nel numero dei parametri ed opera a partire da un insieme di *feature* di base vuoto, procedendo con l'aggiunta di caratteristiche e valutando insiemi di dimensione crescente di *feature* finché una funzione obiettivo (la funzione classificatrice) non restituisce risultati stabili. La valutazione è effettuata considerando l'abilità predittiva di ogni caratteristica ed il grado di ridondanza con le altre, favorendo quindi le *feature* significative per le predizioni ma poco correlate tra loro. In Tabella 10 sono riportati i due set di *feature* suggeriti da questo algoritmo, il primo insieme ottenuto inizializzando l'algoritmo con ognuna delle tre caratteristiche indicate ed il secondo insieme a partire dalla quarta *feature*, e le prestazioni del gruppo di caratteristiche sulla rete Bayesiana. Si noti come i risultati ottenuti effettuando la rilevazione con questi insiemi di caratteristiche siano paragonabili a quelli ottenuti tramite l'algoritmo *greedy* proposto che ha fornito anche valori di sensitività nettamente maggiori.

Tabella 9: *Migliori risultati sperimentali ottenuti*

Feature iniziale	**Gruppo ottimale trovato**	**Sensitività**	**Specificità**
protocol_type	protocol_type wrong_fragment urgent is_host_login srv_count dst_host_diff_srv_rate	81,26%	85,73%
service	protocol_type service hot num_compromised is_guest_login rerror_rate srv_rerror_rate dst_host_diff_srv_rate	83,12%	93,38%
dst_bytes	dst_bytes urgent num_failed_logins num_compromised root_shell num_root num_shells num_access_files is_host_login is_guest_login	73,65%	92,20%
srv_rerror_rate	hot logged_in root_shell rerror_rate srv_rerror_rate	85,37%	75.45%

Tabella 10: *Risultati ottenuti usando le caratteristiche suggerite dal framework Weka*

Feature iniziale	Gruppo ottimale suggerito	Sensitività	Specificità
protocol_type, service, dst_bytes,	service flag src_bytes dst_bytes logged_in serror_rate srv_serror_rate same_srv_rate diff_srv_rate dst_host_srv_diff_host_rate dst_host_serror_rate dst_host_srv_serror_rate	70,83%	96,56%
srv_rerror_rate	protocol_type service flag src_bytes dst_bytes wrong_fragment logged_in num_compromised su_attempted num_root num_access_files count serror_rate srv_serror_rate rerror_rate srv_rerror_rate same_srv_rate diff_srv_rate srv_diff_host_rate dst_host_count dst_host_srv_count dst_host_same_srv_rate dst_host_diff_srv_rate dst_host_same_src_port_rate dst_host_srv_diff_host_rate dst_host_serror_rate dst_host_srv_serror_rate dst_host_srv_rerror_rate	68,21%	96,66%

Capitolo V

Conclusioni

In questo lavoro di tesi è stato affrontato il problema della rilevazione delle intrusioni nelle reti di calcolatori. Si è deciso di adottare una rete *naive* Bayesiana per eseguire tecniche di inferenza statistica allo scopo di calcolare la probabilità di una intrusione a partire da caratteristiche estratte dai flussi di dati che scorrono nella rete da proteggere.

Il sistema proposto è stato addestrato e testato utilizzando il dataset pubblico NSL-KDD, costruito per essere usato nell'ambito dell'*intrusion detection* a partire dal dataset precedente KDDCup'99.

Per la selezione delle caratteristiche da utilizzare nei calcoli inferenziali si è implementato un algoritmo di ricerca approssimata basato su una euristica di tipo *best-first*. Le performance dell'algoritmo di ricerca sono state migliorate dall'euristica adottata, permettendo di abbattere la complessità computazionale rendendola dell'ordine di $O(n)$ invece dell'originale $O(n^3)$ della ricerca esaustiva. Questo algoritmo ha trovato possibili insiemi di *feature* per la rilevazione delle intrusioni che sono stati usati dal rilevatore *naive* Bayesiano per effettuare i test inferenziali con il dataset NSL-KDD.

I risultati ottenuti dal rilevatore sono stati paragonati in termini di sensitività e specificità a quelli ottenuti sulla base delle caratteristiche suggerite dall'algoritmo *RankSearch* del *framework* Weka e si sono dimostrati paragonabili se non superiori, realizzando valori dell'ordine dell'80-90% di accuratezza della rilevazione.

Possibili sviluppi futuri possono essere l'utilizzo di euristiche differenti, come per esempio quelle basate sugli algoritmi genetici multiobiettivo che sono in grado di esplorare in maniera efficiente e rapida l'intero spazio delle soluzioni allo scopo di massimizzare il più possibile entrambi i parametri di ottimizzazione, cioè sensitività e specificità, ottenendo un set di soluzioni ottime in senso Paretiano che sono dette *non dominate* ovvero sono tali per cui non esiste nessun'altra soluzione che sia migliore contemporaneamente per tutti gli obiettivi valutati tramite un'apposita funzione di ottimizzazione; il luogo di queste soluzioni è detto *fronte di Pareto*. Una soluzione può

far parte del fronte di Pareto anche se non ne domina nessun'altra, a patto che essa non sia dominata da altre soluzioni. Adottando tali accorgimenti, le soluzioni ottenute consentirebbero l'esecuzione dell'algoritmo con una taratura a grana fine permettendo di enfatizzare uno dei parametri obiettivo in base alle politiche di sicurezza desiderate.

Bibliografia

[1] Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy (Vol. 99). Technical report.

[2] Bosworth, S., Kabay, M. E. (Eds.). (2002). Computer security handbook. John Wiley and Sons.

[3] The Bro Network Security Monitor. Disponibile online all'indirizzo http://www.bro.org/index.html

[4] Cannady, J. (2000). Applying CMAC-based online learning to intrusion detection. In Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on (Vol. 5, pp. 405-410). IEEE.

[5] Chebrolu, S., Abraham, A., Thomas, J. P. (2005). Feature deduction and ensemble design of intrusion detection systems. Computers and Security, 24(4), 295-307.

[6] Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. Artificial intelligence, 42(2), 393-405.

[7] Debar, H., Dacier, M., Wespi, A. (1999). Towards a taxonomy of intrusion-detection systems. Computer Networks, 31(8), 805-822.

[8] Depren, O., Topallar, M., Anarim, E., Ciliz, M. K. (2005). An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. Expert systems with Applications, 29(4), 713-722.

[9] Di Pietro, R., Mancini, L. V. (2008). Intrusion detection systems. Springer.

[10] Durgin, N. A., Zhang, P. (2005). Profile-based adaptive anomaly detection for network security. Sandia National Laboratories Technical Report, SAND2005-7293.

[11] Eclipse. Disponibile all'indirizzo http://www.eclipse.org/

[12] Fisk, M., Varghese, G. (2002). Applying fast string matching to intrusion detection. Los Alamos National Lab NM.

[13] Friedman, N., Geiger, D., Goldszmidt, M. (1997). Bayesian network classifiers. Machine learning, 29(2-3), 131-163.

[14] Friedman, N. (1998, July). The Bayesian structural EM algorithm. In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (pp. 129-138). Morgan Kaufmann Publishers Inc..

[15] Fung, R. M., Chang, K. C. (1989). Weighing and Integrating Evidence for Stochastic Simulation in Bayesian Networks. In UAI (pp. 209-220).

[16] Girardin, L. (1999, April). An Eye on Network Intruder-Administrator Shootouts. In Workshop on Intrusion Detection and Network Monitoring (pp. 19-28).

[17] Gunes Kayacik, H., Nur Zincir-Heywood, A., Heywood, M. I. (2007). A hierarchical SOM-based intrusion detection system. Engineering Applications of Artificial Intelligence, 20(4), 439-451.

[18] Gomez, J., Dasgupta, D. (2002, June). Evolving fuzzy classifiers for intrusion detection. In Proceedings of the 2002 IEEE Workshop on Information Assurance (Vol. 6, No. 3, pp. 321-323). New York: IEEE Computer Press.

[19] Guo, H., Hsu, W. (2002, July). A survey of algorithms for real-time Bayesian network inference. In AAAI/KDD/UAI02 Joint Workshop on Real-Time Decision Support and Diagnosis Systems. Edmonton, Canada.

[20] Henrion, M. (1988). Propagation of uncertainty by probabilistic logic sampling in Bayes' networks. In Uncertainty in Artificial Intelligence (Vol. 2, pp. 149-164).

[21] Huang, C., Darwiche, A. (1994). Inference in belief networks: A procedural guide. International journal of approximate reasoning, 11(1), 158.

[22] Jolliffe, I. (2005). Principal component analysis. John Wiley and Sons, Ltd.

[23] KDD Cup 1999. Disponibile online all'indirizzo http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[24] Kumar, S., Spafford, E. H. (1994). A pattern matching model for misuse intrusion detection.

[25] Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R. H., Kuijpers, C. M. H. (1996). Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 18(9), 912-926.

[26] Liao, Y., Vemuri, V. R., Pasos, A. (2007). Adaptive anomaly detection with evolving connectionist systems. Journal of Network and Computer Applications, 30(1), 60-80.

[27] Lin, Y., Druzdzel, M. (1999). Stochastic sampling and search in belief updating algorithms for very large Bayesian networks. In working notes of the AAAI Spring Symposium on search techniques for problem solving under uncertainty and incomplete information (pp. 77-82).

[28] Luger, G., Maccabe, A., Servilla, M. (1990). The architecture of a network-level intrusion detection system. Department of Computer Science, College of Engineering, University of New Mexico.

[29] Netica-J Java API, della Norsys Software Corporation. Disponibile all'indirizzo http://www.norsys.com/

[30] Pearl, J. (1988). Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann.

[31] Russell, S., Norvig, P. (2009). Artificial Intelligence: A Modern Approach. 3rd Edition. Prentice-Hall

[32] Scheines, R. (1997). An introduction to causal inference.

[33] Smith, L. I. (2002). A tutorial on principal components analysis. Cornell University, USA, 51, 52.

[34] Snort®. Disponibile all'indirizzo http://www.snort.org/

[35] Steck, H. (2001). Constraint-based structural learning in Bayesian networks using finite data sets (Doctoral dissertation, Technische Universität München, Universitätsbibliothek).

[36] Suricata. Disponibile all'indirizzo http://suricata-ids.org/

[37] Suzuki, J. (1996). Learning Bayesian belief networks based on the MDL principle: an efficient algorithm using the branch and bound technique.

[38] Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009.

[39] Snort®. Disponibile all'indirizzo http://www.cs.waikato.ac.nz/ml/weka/

[40] Wu, S. X., Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems: A review. Applied Soft Computing, 10(1), 1-35.

[41] Zhang, C., Jiang, J., Kamel, M. (2003). Comparison of BPL and RBF network in intrusion detection system. In Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing (pp. 466-470). Springer Berlin Heidelberg.

[42] Zhang, H. (2004). The optimality of naive Bayes. A A, 1(2), 3.