Universidade Federal Fluminense
Escola de Engenharia
Curso de Graduação em Engenharia de Telecomunicações

Letícia Rodrigues de Sousa

Redes Neurais Artificiais Aplicadas à Sistemas de Detecção de Intrusão em Redes de Computadores

Niterói – RJ

2017

Letícia Rodrigues de Sousa

Redes Neurais Artificiais Aplicadas à Sistemas de Detecção de Intrusão em Redes de Computadores

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Telecomunicações da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Engenheiro de Telecomunicações.

Orientador: Prof. Dr. João Marcos Meirelles da Silva

Niterói – RJ

2017

Letícia Rodrigues de Sousa

Redes Neurais Artificiais Aplicadas à Sistemas de Detecção de Intrusão em Redes de Computadores

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Telecomunicações da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Engenheiro de Telecomunicações.

Aprovada em 12 de Julho de 2017.

BANCA EXAMINADORA

Prof. Dr. João Marcos Meirelles da Silva - Orientador
Universidade Federal Fluminese - UFF

Prof. Dr. Murilo Bresciani de Carvalho
Universidade Federal Fluminese - UFF

Prof. Dra. Natália Castro Fernandes
Universidade Federal Fluminense - UFF

Niterói – RJ
2017

.

A figura referente ao arquivo

FichaCatalografica.jpg

fornececido pela Biblioteca

dever aparecer aqui.

.

À minha mãe, que sempre acreditou nas minhas escolhas e ao meu namorado, Aroldo Mascarenhas Neto, que foi meu porto seguro diante das dificuldades deste percurso.

Agradecimentos

Agradeço à minha família que sempre me apoiou e deu suporte aos meus sonhos.

Agradeço aos meus queridos amigos Allex Ornellas, Monick Araujo, Nathalia Soares, Ríguel Ruiz e Marianna Lobosco por me fornecerem suporte emocional em todos os momentos que precisei durante esta jornada.

Agradeço aos meus amigos Gabriel Chaves e Wescley Leal, pela nossa parceria durante a faculdade.

Agradeço ao meu namorado, Aroldo Mascarenhas Neto, por ter ficado sempre ao meu lado, me incentivando e não me deixando desanimar. Muito obrigada por ser meu anjo da guarda e por toda a sua dedicação.

Agradeço ao meu coordenador de estágio na Petrobras, Carlos Frederico Ogliari Arruda Correia, pelos dois anos de amizade e compreensão. Obrigada pela confiança que teve em mim e pelo conhecimento que me foi passado.

Agradeço ao professor Andrés Pablo López Barbero pelos ensinamentos passados durante o período em que trabalhei no Laboratório de Comunicações Ópticas da UFF.

Agradeço a professora Leni Joaquim pelo tempo dedicado a mim durante o tempo em que estagiei no Laboratório de Propagação da UFF.

Agradeço ao meu orientador, professor João Marcos Meirelles da Silva, por todo o conhecimento passado.

Agradeço aos professores Murilo e Natalia Castro Fernandes, por aceitar o convite para compor a banca avaliadora deste trabalho.

Agradeço a todos os professores do Departamento de Engenharia de Telecomunicações pela contribuição à minha formação.

Agradeço a todos os amigos que fiz na UFF por terem tornado este caminho mais leve!

Lista de Figuras

Lista de Tabelas

Sumário

Resumo

O desenvolvimento de softwares que detectam intrusão em redes de computadores é de extrema importância para salvaguardar as informações sensíveis das companhias. Esses softwares podem ser implementados de diversas maneiras e, uma delas, é empregando uma rede neural artificial. O presente trabalho fará uso de um banco de dados que simula o trágefo numa rede real para tal implementação. O banco de dados citado possui informações redundantes e desnecessárias, então diferentes métodos serão utilizados para realizar uma eficiente seleção de atributos nos dados originais. O resultado de cada método será testado na mesma rede neural para que seja possível compará-los.

Palavras-chave: Redes Neurais Aritificiais. Seleção de Atributos. KDD CUP 99.

Abstract

The development of systems that detects intrusion in networks is very important to keep sensitive information safe. Several techniques can be used to implement these systems and, one of them, is using artificial neural networks. In this work, a database that simulates traffic on a real network is going to be used to train and test the net. The major problem when working with very large databases is excessive or unnecessary information, so different methods will be used to perform an efficient feature selection in the original data. The result of each method will be tested at the same neural network to compare them.

Keywords: Artificial Neural Networks. Feature selection. KDD CUP 99.

Capítulo 1

Introdução

No mundo corporativo, além dos produtos e serviços comercializados, há um ativo valioso que as empresas devem sempre proteger: a informação. Com a vasta utilização de computadores para tráfego de dados confidenciais surgiu a necessidade de proteger as redes por onde eles transitam, pois a sua exposição pode gerar perdas financeiras e comprometer as operações e a continuidade dos negócios da companhia.

Atualmente, não é necessário possuir grande conhecimento acerca de um sistema operacional ou protocolo, por exemplo, para explorar suas vulnerabilidades, pois já existem ferramentas automatizadas de exploração de falhas.

A fim de obter informações confidenciais de empresas ou mesmo de pessoas físicas, são feitas tentativas de intrusão a rede onde serão localizados esses dados. Do ponto de vista de gerenciamento da rede, qualquer tipo de acesso não autorizado pode ser considerado um ataque.

Os ataques efetuados às diversas corporações podem ter origens diversas e precisam ser identificados e registrados a fim de se conhecer o seu comportamento e posteriormente encontrar mecanismos que possam mitigar esse problema. Ao mensionar mecanismos que possam impedir ataques à redes de computadores, somos levados a pensar em um conjunto de práticas de segurança da informação que envolvem, além de ações protetivas por parte de pessoas que estejam em posse desses dados relevantes, softwares e hardwares que possam auxiliar na identificação de tais problemas.

Nesse sentido, é necessário dispor de um bom sistema capaz de monitorar a rede para verificar se a mesma permanece segura, identificando e alertando os gestores da rede sobre possíveis incidentes ou mesmo respondendo à atividade suspeita.

Um sistema de detecção de intrusão (intrusion detection system – IDS) é um dispositivo de monitoramento de sistemas capaz de detectar e responder à atividade maliciosas em uma rede de computadores ou mesmo em computadores a partir de dados coletados da rede que está sob análise. O IDS deve reportar a administração situações onde o ataque está em andamento e até mesmo os casos em que a atividade foi mal sucedida.

Os sistemas de detecção de intrusão são caracterizados pela sua metodologia de ação e pela sua implementação, existindo diversas maneiras de se elaborar um IDS. Dentre as diversas maneiras, destacam-se algumas teorias como detecção baseada em perfis, método estatístico e redes neurais artificiais.

1.1 Motivação

A minha motivação vai ser a necessidade dos métodos. No campo de sistemas de detecção de intrusão em redes de computadores, diversos estudos foram realizados, tanto empregando redes neurais artificiais, bem como outras técnicas de aprendizado de máquina.

Os bancos de dados utilizados para as pesquisas são, em sua grande parte, muito grandes e possuem informações redundantes e desnecessárias. É necessário, portanto, que haja um prévio tratamento dos mesmos a fim de melhorar o processamento da rede e aumentar a eficiência do sistema.

É necessário, porém, que se faça uma comparação entre as diferentes técnicas tão utilizadas para este objetivo, a fim de avaliar a contribuição de cada uma para o resultado final.

A motivação para este trabalho é a necessidade de se comparar as técnicas simples de seleção de atributos para a mesma base de dados.

1.2 Objetivos

O presente trabalho tem como objetivo implementar diferentes técnicas de seleção de atributos em um mesmo conjunto de dados, sendo os mesmos referentes ao tráfego em uma rede de computadores, a fim de verificar como uma mesma rede neural artificial responde à diferentes combinações deste conjunto de dados.

Para tal, será utilizado o KDD CUP 99, banco de dados confeccionado com simulações de uma rede de dados real, e técnicas de seleção de atributos, bem como uma rede

neural artificial, presentes no MatLab.

Capítulo 2

Redes Neurais Artificiais

Redes neurais artificiais são técnicas de inteligência artificial que simulam computacionalmente o funcionamento do cérebro humano, a fim de processar dados de maneira inteligente, atuando na tomada de decisão em sistemas. Esta técnica deve ser capaz de obter informações através do processo de aprendizagem e armazenar conhecimento em seus ”neurônios”, que é a unidade inteligente da rede.

Na rede neural artificial, o resultado de todo o processamento envolvido na sinapse do neurônio biológico é representado por um peso. Dentro do neurônio artificial há uma junção somadora que soma todos os sinais de entrada ponderados pelo peso.Feito o procedimento, é aplicada uma função de ativação a esta soma a fim de limitar a saída do neurônio e introduzir uma não linearidade ao sistema.

Inicialmente, a rede é alimentada com informações advindas de um banco de dados, então esses sinais são processados e ponderados pelos pesos atribuidos a cada neurônio para gerar uma saída. Para um processamento eficaz da informação recebida, conectam-se neurônios em diferentes camadas. A primeira camada, que recebe o sinal de entrada, camadas ocultas para processamento e a camada de saída. Para cada problema, haverá um tipo de arquitetura.

As redes neurais podem ser aplicadas a diversos problemas de reconhecimento de padrões. Desta forma, ela é eficaz para utilização em sistemas de detecção de intrusão em redes de computadores, onde é necessário que o sistema faça uma escolha sobre qual o tipo de acesso está sendo efetuado. Neste trabalho, aplicaresmos redes neurais artificiais a sistemas de detecção de intrusão em redes de computadores.

No campo dos sistemas de detecção de intrusão, as redes neurais artificiais são

utilizadas em IDS com metodologia de detecção por anomalias, que identifica desvios de padrões de utilização um de um determinado sistema, o que pode caracterizar um ataque.

2.1 Redes Neurais Artificiais no MatLab

O MatLab apresenta um ToolBox específico para redes neurais artificiais. No programa, é possível encontrar redes neurais capazes de realizar clusterização nos dados, classificando as amostras e separando-as, reconhecimento de padrões e até mesmo trabalhar com redes neurais dinâmicas. Neste trabalho, será utilizado o ToolBox *Neural Net Pattern Recognition* com o objetivo de fazer a rede entender a maneira como estão distrubuidos os dados em classes, percebendo diferentes padrões. Com esta ferramenta é possível fornecer dados de entrada e as classes onde esses dados estão inseridos e, posteriormente, escolher a arquitetura da rede que irá solucionar o problema. Após o treinamento, é possível obter graficamente informações estatísticas sobre o desempenho da atividade.

A rede neural escolhida vem com uma configuração default. É possível acrescentar camadas, alterar a taxa de aprendizagem, alterar a porcentagem de dados utilizada para cada etapa do processo, entre outras coisas. No entando, a rede escolhida foi a rede default, já que a arquitetura da rede não é o foco do projeto no momento.

Capítulo 3

Ataques em Redes de Computadores

O IETF define ataque como um ato intencional pelo qual uma entidade tenta evadir serviços de segurança e violar a política de segurança de um sistema. Ou seja, um assalto real à segurança do sistema que deriva de uma ameaça inteligente. [13]

Existem diferentess maneiras de um atacante realizar um ataque a uma rede de computadores ou sistemas e, ao longo do tempo, é possível observar que as diversas técnicas para este tipo de atividade vem aumentando. Eles tem como objetivo tentar impedir a continuação de um serviço, expor ou roubar dados e alteração de informações do sistema ou dados confidenciais.

Os ataques ocorrem porque as fraquezas da rede são conhecidas ou são conhecidos os mecanismos para afetá-las. Ao passo que são corrigidas suas falhas é natural que ainda seja possível identificar outras brechas e vulnerabilidades ainda não notadas pelos gerenciadores do sistema e então se consiga, com um outro tipo de ataque, afetar o mesmo.

Eles podem ter origem interna ou externa. Para ser classificado como um ataque de origem externa, ele deve se originar fora da rede, por um atacante que tenta acessá-la. É considerado de origem interna quando parte de algum usuário com acesso a rede, que pode fazer uso de seus privilégios para realizar o ataque.

Os ataques podem ser separados em diferentes categorias. As frequentes são Dos, U2R, R2L e Probe, definidos a seguir:

- DoS - Os ataques DoS (*Denial of Service*) tem por objetivo promover uma perda de desempenho de serviços ou sistemas, impossibilitando o seu uso por aqueles que realmente possuem permissão para acessá-los. [2]

  Este tipo de ataque não rouba dados do sistema, mas provoca a sua indisponibilidade

para seus usuários. Apesar de não ocorrer furto de informação, para determinadas corporações, algumas horas fora de operação significa representativas perdas. É também um ataque muito utilizado atualmente como forma de protesto a governos e instituições, fazendo com que suas páginas oficiais saiam do ar por determinado período de tempo.

- R2L - Chamado de ataque de usuário remoto (*Remote to Local attack*), é caracterizado pelo envio de pacotes a uma máquina de uma rede, sem possuir conta na mesma, e, a partir de então são exploradas vulnerabilidades da máquina para ganhar acesso ilegal de usuário local. [10]

- U2R - Um ataque U2R *User to Root* ocorre quando um atacante inicia a exploração do host com uma conta de usuário normal do sistema e consegue explorar vulnerabilidades deste para ganhar acesso de root. [12]

- Probe - Este ataque varre a rede na tentativa de obter informações para burlar controles de segurança. A varredura ocorre através do envio de pacotes à rede, de forma a mapear suas vulnerabilidades. As respostas recebidas da rede são utilizadas pelo atacante para aprender sobre as características da rede e dos sistemas, incluindo a topologia da rede, os hosts ativos e suas respectivas configurações de software, incluindo sistema operacional, software do servidor e versões de aplicativos. [12] Dessa maneira, é possível planear o ataque de forma efetiva.

O banco de dados utilizado neste trabalho lista vinte tipos diferentes de ataques, contidos nas categorias acima citadas, e conexão normal, onde não há registro de ameaças.

Capítulo 4

Revisão Bibliográfica

Muitos estudos vem sendo realizados acerca de sistemas inteligentes de detecção de intrusão em redes utilizando o KDD Cup 99, sendo desenvolvidos a partir de redes neurais artificiais ou mesmo outras abordagens computacionais. Neste capítulo, iremos rever alguns estudos relevantes acerca do tema.

Neha G. Relan *et al.* [3] publicou um estudo onde utilizava algoritmos de árvore de decisão C4.5 e árvore de decisão C4.5 com prunning para treinar e testar o classificador, fazendo previamente uma seleção de atributos. Neste trabalho, outro banco de dados, o NSL_KDD, foi usado em conjunto com 10% do KDD Cup 99.

O algoritmo C4.5 é aplicado para produzir uma árvore de decisão que é utilizada para classificação, fazendo uso do conceito de ganho de informação como critério de divisão. As técnicas com pruning evitam o excesso de ajustes na árvore de decisão, eliminando parte da árvore que não está contribuindo para o classificador, identificando atributos que não adicionem valor ao problema. O algoritmo pode trabalhar tanto com valores discretos quanto com valores contínuos. Ele escolhe os melhores atributos que dividem o conjunto de dados em subconjuntos, então os atributos com maior ganho de informação podem ser utilizados para dividir os dados. Cada nó da folha da árvore de decisão representa uma classe e o processo é interrompido quando o número de instâncias a ser dividido está abaixo de um limiar. Para o estudo citado, foram considerados os 41 atributos do KDD Cup 99 na criação da árvore de decisão com C4.5 e apenas valores discretos deste banco de dados no C4.5 com pruning.

No pré-processamento foi feita uma formatação, limpeza e amostragem dos dados. Depois de treinados, os dados foram testados nos algoritmos. Feita a experiência, os

resultados obtidos utilizando o KDD Cup 99 foram de 92,81% de acurácia no C4.5 e 95,09% no C4.5 com pruning. Já utilizando o banco de dados NSL-KDD, as taxas obtidas foram de 26,39% no C.45 e 98,45% no C4.5 com pruning.

Iftikhar Ahmad *et al.* [4] aplicou redes neurais artificiais em ataques do tipo U2R utilizando algumas amostras do KDD Cup 99 para testar e treinar o sistema, que possui diferentes atributos relacionados a este tipo de ataque, como Loadmodule, Perl e Buffer Overflow.

O Loadmodule explora vulnerabilidades na maneira como o SUNOS 4.1 carrega os módulos dinamicamente. Estas falhas tornam possível um invasor adquirir privilégios de administrador.

O Perl é um tipo de ataque que explora falhas na maneira como é feita a distribuição de privilégios (root) em implementações do perl em sistemas anteriores.

Já o Buffer Overflow consiste em transbordar buffers de entrada para alterar pontos da memória onde estejam informações importantes.

Foram utilizados 38 dos 41 atributos, já que os outros três apresentavam valores simbólicos. O modelo por ele proposto utiliza uma rede feedfoward, composta por três camadas consecutivas sendo elas a camada de entrada, a camada interna e a camada de saída. Cada camada possui certo número de neurônios com as mesmas características, como função de transferência, taxa de aprendizado etc.

A camada de entrada possuía 38 neurônios devido ao fato de terem sido selecionados 38 atributos do banco de dados. Esta camada pegava as suas entradas do arquivo que contém os dados para treinamento. A camada interna possuía 4 neurônios e a de saída consistia de 2 neurônios, que classificaria os pacotes em normais ou anormais, escrevendo padrões de saída em um arquivo utilizado para análise de ataques.

O sistema foi implementado utilizando NeuroSolutions.

Na fase de treinamento, foi utilizado aprendizado supervisionado com Backpropagation, já que o mesmo converge rapidamente. Após o treinamento, os pesos foram congelados e o desempenho da rede foi avaliado. O sistema mostrou um excelente desempenho no caso de positivos verdadeiros, com taxa de 100% no terino e 95,45 no teste e falsos verdadeiros com taxa de 98,36% no treino e 100% no teste.

Mukhopadhyay *et al.* [5] utilizou uma abordagem de redes neurais artificiais com backpropagation em sistemas de detecção de intrusão. Foram utilizados 46.655 registros

dos 41 atributos banco de dados para treinar e validar a rede neural.

Foram utilizados 6 neurônios na camada de saída pois o tráfego está agrupado em 6 classes de ataques. Após as fases descritas acima, os resultados obtidos foram 73.9% de taxa de sucesso e 26.9% de taxa de erro.

Shreya Dubey *et al.* [6] propôs um método híbrido para sistemas de detecção de intrusão baseado em k-means, naive-bayes e redes neurais back propagation (KBB).

O KDD Cup 99 foi utilizado como entrada do sistema. Inicialmente, foi utilizado o K-means, que é um método de aprendizagem não supervisionado, sendo usado para agrupar os dados de bancos de dados muito grandes. Em seguida, o naive-bayes, um método supervisionado baseado na probabilidade de um evento, foi utilizado. Na etapa de aprendizagem, já com os dados filtrados, foi utilizada uma rede neural com algoritmo backprogapation.

Os testes mostraram que a precisão deste sistema aumenta caso os dados sejam previamente tratados, pois é necessário que as informações cedidas como entrada do sistema sejam significativas. O método estudado ofereceu melhor precisão e taxa de erro que métodos tradicionais, precisando, em contrapartida, de maior capacidade de processamento computacional.

Poojitha.G. *et al.* [7] utilizou 10% do KDD Cup 99 para desenvolver um IDS baseado em uma rede neural Feed Foward treinada com algoritmo backpropagation. A rede neural proposta possui as fases de treinamento e testes, a fim de identificar condições normais ou anormais dos parâmetros dados.

No entanto, antes de treinar a rede, foi feito um pré processamento dos dados a fim de remover atributos redundantes e os atributos não-numéricos foram transformados em atributos numéricos segundo um critério. Na estrutura da rede, os neurônios das camadas internas possuíam tangente hiperbólica como função de ativação e os de saída possuíam uma função de ativação linear. Na camada de entrada haviam 41 neurônios e na camada de saída haviam 4, onde quando todos os neurônios em 0 indicava um acesso normal e um em cada representa um dos quatro ataques.

Os testes mostraram que o sistema proposto obteve uma taxa de precisão de 94,93%, taxa de falso positivo de 0.002% e taxa de falso negativo de 0.7%.

Mohammed A. Ambusaidi *et al.* [8] propôs um sistema de detecção de intrusão utilizando filtros na seleção de atributos. A estrutura do sistema é composta de quatro

fases: coleta de dados, pré processamento dos dados, treinamento e reconhecimento de ataques. A fase de pré processamento de dados é composta pela transferência de dados, onde os valores não numéricos foram transformados em valores numéricos, normalização de dados, onde os dados foram normalizados e seleção de atributos, utilizando variações do algoritmo MFIS.

Na fase de treinamento, foi utilizado o LS-SVM. Como este método apenas lida com classificações binárias e como o KDD Cup 99 possui cinco subconjuntos de atributos, precisou-se de cinco LS-SVM, que se juntam para formar a detecção de intrusão. Cada classificador distingue uma classe da outra. Por exemplo, é possível distinguir normais de anormais, DoS de não DoS, etc.

Para avaliar o desempenho do sistema foram utilizados três conjuntos de dados: KDD Cup 99, NSL_KDD e Kyoto 2006+. A combinação LSSVM-IDS + FMIFS obteve a maior taxa de detecção.

V. Bolon-Canedo *et al.* [9] utilizou 10% do KDD Cup 99 para a fase de treinamento e o banco de dados original na fase de testes. Foi produzido um método que combina discretização e seleção de atributos para obter uma redução no grande número de atributos que o banco de dados contém.

Para a fase de seleção de atributos, os filtros foram utilizados devido ao tamanho do banco de dados, tanto em número de entradas como no de características de cada entrada. Os métodos de filtragem CFS, INTERACT e Consistency-based filter foram testados para verificar se seriam obtidos resultados diferentes.

Os métodos de discretização EMD, EWD e EFD foram escolhidos para o trabalho por serem algoritmos clássicos e o PKID foi escolhido por se tratar de uma nova abordagem que funciona bem em grandes conjuntos de dados. Além destes, o BL também foi escolhido.

Antes de aplicar discretização e seleção de atributos, foi realizada uma análise a fim de verificar como os métodos de discretização se comportam como são combinados com métodos de classificação. O objetivo era demonstrar que os mesmos tem influência nos erros de classificação. A partir de análises, foi verificado que, para o KDD Cup 99, os melhores resultados são obtidos com o algoritmo C4.5. No trabalho foram utilizadas C4.5 e Naive Bayes, com parâmetro Kernel sintonizado.

Os métodos de filtros foram combinados com os de discretização para avaliar os

resultados. Em termos de discretização, quando usando PKID ou EMD, os filtros selecionaram menos recursos do que quando se utiliza EWD, EFD ou BL. Cada combinação leva a um subconjunto diferente de recursos, assim um classificador poderia obter diferentes resultados de desempenho. Então, não é possível determinar qual combinação de filtro e método de discretização é a melhor.

Então, foram combinados os métodos de discretização com os filtros e os algoritmos de classificação.

A partir dos resultados, o EMD parece ser o melhor método de discretização para C4,5. A melhor taxa de positivos verdadeiros foi obtida com a combinação EMD +INT + NB, mas o melhor resultado na tabela é conseguido com EMD + Cons + C45, uma vez que esta combinação tem o melhor erro e taxa de positivos verdadeiros.

Capítulo 5

Implementação

5.1 O KDD CUP 99

Para a realização deste trabalho, utilizou-se 10% do KDD CUP 99,que contém 42 colunas representando os atributos, e 311.029 linhas representando as características de cada atributo.

O banco de dados supracitado foi confeccionado por Stolfo *et al.* [11] em conjunto com a conferência KDD, para uma tarefa de aprendizagem de classificadores, com informações coletadas em nove semanas de tráfego de rede. Foi criado um conjunto de dados com uma grande variedade de intrusões que simulam uma LAN típica da Força Aérea dos Estados Unidos, sendo atacada por diferentes maneiras. Nele, as conexões possuiem diversas características expressas nos atributos e por fim são classificadas como conexão normal ou ataque. Caso seja uma atividade maliciosa, a mesma é categorizada entre 20 tipos de ataque.

Na tabela Tabela 5.2 está listado cada atributo, excluindo-se a última coluna, pois a mesma representa o tipo de acesso:

	Atributo	Descrição
1	duration	Duração da conexão
2	protocol type	Tipo de protocolo (tcp, udp, etc)
3	service	Serviço de rede no destino
4	flag	Status normal ou de erro na conexão
5	src_bytes	Bytes da origem para o destino

6	dst_bytes	Bytes do destino para a origem
7	land	1 se a conexão é para o mesmo host e 0 caso contrário
8	wrong_fragment	Fragmentos errados
9	urgent	Diz respeito a pacotes urgentes
10	hot	Indicadores
11	num_failed_logins	Tentativas inválidas de login
12	logged in	Loguin efetuado
13	num_compromised	Condições comprometidas
14	root_shell	Obtenção do root
15	su_attempted	tentativa su_root
16	num_root	Acessos root
17	num_file_creations	Operações de criação de arquivo
18	num_shells	Comandos do shell
19	num_access_files	Operações em arquivos de controle de acesso
20	num_outbound_cmds	Comandos de saída em uma sessão de ftp
21	is_host_login	Loguin pertence a lista ”hot”
22	s_guest_login	Loguin é um ”guest”
23	count	Conexões com o mesmo host que a atual
24	srv_count	Conexões com o mesmo serviço que a atual
25	serror_rate	Conexões que possuem erros ”SYN”
26	srv_serror_rate	Conexões que possuem erros ”SYN”
27	rerror_rate	Conexões que possuem erros ”REJ”
28	srv_rerror_rate	Conexões que possuem erros ”REJ”
29	same$_srv_rate$	Conexões com o mesmo serviço
30	diff_srv_rate	Conexões com serviços diferentes
31	srv_diff_host_rate	Conexões com hosts diferentes
32	dst_host_count	count para host de destino
33	dst_host_srv_count	srv_count para host de destino
34	dst_host_same_srv_rate	same_srv_rate para host de destino
35	dst_host_diff_srv_rate	diff_srv_rate para host de destino
36	dst_host_same_src_port_rate	same_src_port_rate para host de destino

37	dst_host_srv_diff_host_rate	srv_diff_host_rate para host de destino
38	dst_host_serror_rate	serror_rate para host de destino
39	dst_host_srv_serror_rate	srv_serror_rate para host de destino
40	dst_host_rerror_rate	rerror_rate para host de destino
41	dst_host_srv_rerror_rate	Contagem srv_rerror_rate para host de destino

Tabela 5.1: Lista de Atributos do KDD CUP 99.

Como mensionado, a última coluna representa o tipo de acesso: acesso normal ou intrusão. As características de intrusão são divididas nas classes de ataques citadas anteriormente, no capítulo [inserir referencia aqui] deste documento. A tabela [inserir referencia] mostra os 20 diferentes tipos de ataques contidos no banco de dados supracitados divididos em suas categorias:

DoS	Back, Land, Pod, Neptune, Teardrop, Smurf.
U2R	Perl Rootkit, Buffer-overflow, Load module.
Probe	Nmap, Satan, Ipsweep, Portsweep.
R2L	Guess-passwd, Physpy, Imap, Multihop, ftp-write, Warezclient, Warezmater.

Tabela 5.2: Tipos de ataques divididos em classes no KDD CUP 99.

É um banco de dados muito utilizado para trabalhos em redes neurais artificiais por conter um número significativo de dados.

Apesar de muito completo, o KDD CUP 99 apresenta uma série de deficiências que dificultam a sua utilização em IDSs e outras aplicações práticas, pois possui muitas informações redundantes e duplicadas, fazendo surgir efeitos indesejáveis no classificador. Ao alimentar a rede neural com parâmetros redundantes ou que não agregam informação, a fase de treinamento é prejudicada, diminuindo a capacidade da mesma de aprender. Além disso, existem problemas de processamento de grandes bancos de dados em desktops comuns, pois a complexidade matemática é aumentada. Devido a isto, é necessário que se faça previamente uma seleção de atributos.

5.2 Seleção de Atributos

A seleção de atributos é de primordial importancia em qualquer projeto de reconhecimento de padrões. Dado um conjunto inicial de dados, a intenção é representa-los a partir de dados mais compactos e significativos.

Visto que um banco de dados possui informações gerais, temos conjuntos de atributos que servem para uma aplicação e outros que não são necessariamente úteis, enquanto que para outro tipo de aplicação, os atributos considerados proveitosos podem mudar. Por isso, essa etapa é importante para personalizar o banco de dados.

Como o desenvolvimento se dará no ambiente do MatLab, é necessário adequar o tamanho do banco de dados e o tipo de variáveis que nele estão contidas para que os algoritmos pudessem ser executados. Para isso, foi preciso alterar as informações na forma de string para números correspondentes. Algumas colunas de atributos continham valores que não adicionavam informação, sendo todos eles iguais. Desta forma, tais atributos foram eliminados antes mesmo da execução dos algoritmos, para que não fosse oferecida nenhuma informação vazia aos algoritmos de treinamento.

Para que o MatLab pudesse processar o banco de dados, foi necessário reduzi-lo devido ao seu grande tamanho. Para verificar se o processo de redução do tamanho iria alterar significativamente as propriedades estatísticas dos dados, foram calculados a média e a variância de cada coluna de atributos no banco original. Linhas do banco de dados foram excluídas aleatoriamente, com a ajuda de um programa em VBA, no Excel. Ao final do processo, sobraram 66.721 linhas com média e variância próximas ao banco de dados inicial, o que sugere que, mesmo após a exclusão de uma grande quantidade de linhas, a informação contida no banco de dados estará bem representada pelas amostras selecionadas.

Este número, 66.721, não foi escolhido previamente. Foram feitos testes com números maiores de linhas, porém sempre com problema na execução por ter excedido a memória.Então, ao verificar que 66.721 era uma quantidade viável, a redução da quantidade de linhas foi encerrada.

A última coluna corresponde aos tipos de acesso, então, ela não deve ser utilizada na seleção de atributos, sendo eliminada nesta fase. Feito isto, sobraram 33 atributos.

Para este trabalho, foram utilizados 9 atributos do banco de dados original. Esse número foi escolhido com base em estudos anteriores, que apontam ser este um número

bom para se trabalhar com redes neurais.

Serão utilizados quatro métodos diferente para seleção de atributos: Principal Component Analysis, Factor Analysis, Singular Value Decomposition e Nonnegative Matrix Factorization. As funções para implementação de tais técnicas com o MatLab foram *pca*, *factoran*, *svd* e *nnmf*, recpectivamente.

O PCA (análise dos componentes principais) é uma das técnicas mais utilizadas em reconhecimento de padrões. método de transformação de variáveis de forma a torna-las mais significativas a partir de combinações lineares das variáveis originais. As componentes principais são selecionadas de modo que a maior parte da energia contida no conjunto de vetores originais esteja concentrada no conjunto de vetores projetados nas direções das componentes principais.

5.2.1 Principal Component Analysis

O Principal Component Analysis (PCA) é uma das técnicas mais utilizadas em reconhecimento de padrões. É método de transformação de variáveis de forma a torná-las mais significativas a partir de combinações lineares das variáveis originais. Os componentes principais são selecionados de modo que a maior parte da energia contida no conjunto de vetores originais esteja concentrada no conjunto de vetores projetados nas direções dos componentes principais.

A partir de uma matriz onde as colunas representam características de determinadas amostras, sendo essas representadas nas linhas desta matriz, a representação da interdependência entre os seus dados é representada pela matriz de covariância. A análise dos componentes principais transforma esta representação em uma onde tem-se variáveis não correlacionadas e em ordem de variância. As amostras podem então ser comparadas usando os coeficientes que apresentam maior variância.

No MatLab, a função *pca* retorna os coeficientes dos componentes principais da matriz de dados selecionada. Os coeficientes são retornados em uma outra matriz, onde em cada coluna contém os coeficientes de um componente principal em ordem decrescente de variância de componente.

Para extrair um novo banco de dados através desse processo, foram localizados os 9 maiores valores de coeficientes fornecidos na primeira coluna da matriz retornada. No KDD CUP 99, selecionamos nas colunas correspondentes as posições dos coeficientes,

separando-as. Esse novo conjunto formou o banco de dados para ser utilizado posteriormente na rede neural.

5.2.2 Factor Analysis

A função *factoran* calcula a máxima estimativa de verossimilhança da matriz em um modelo de análise fatorial com fatores comuns. O princípio da máxima verossimilhaça é utilizado para se obter estimadores. O estimador de máxima verossimilhança é o valor que maximiza a probabilidade dos dados apresentados.

A fim de obter um novo conjunto de dados, foram selecionados na primeira coluna os nove menores estimadores de verossimilhança. Esses dados indicam os nove atributos menos parecidos, digamos, com os demais. Esse conjunto foi reservado para ser utilizado posteriormente na rede neural.

5.2.3 Singular Value Decomposition

O Singular Value Decomposition é um dos algorítmos mais eficientes da álgebra linear, sendo muito utilizado para redução de dimensionalidade em reconhecimento de padrões e outras aplicações, como recuperação de dados.

Para uma matriz X_{lxn}, existem matrizes U_{lxl} e V_{nxn} que satisfazem

$$X = U \begin{bmatrix} \Lambda^{1/2} & O \\ O & 0 \end{bmatrix} V^H$$

ou

$$Y = \begin{bmatrix} \Lambda^{1/2} & O \\ O & 0 \end{bmatrix} = U^H X V$$

Sendo $\Lambda^{1/2}$ a matriz diagonal das raízes dos autovalores da matriz $X^H X$ e O uma matriz de elementos nulos.

Ou seja, existem matrizes U e V que transformam X na estrutura representada por Y. Portanto, sendo u_i e v_i os elementos das matrizes U e V, respectivamente, nota-se que:

$$X = \sum_{i=0}^{r-1} \sqrt{\lambda_i} u_i v_i^H$$

onde λ_i são os autovalores de $X^H X$. [1]

O algorítmo *svd* do MatLab decompõe a matriz fornecida a ele da seguinte maneira: dada uma matriz A, a mesma é decomposta em novas matrizes U, S e V, onde $A = USV'$. Do exposto anterior, percebemos que a matriz S é a matriz que carrega os autovalores de A.

Para aplicar esta ferramenta à seleção de atributos, escolheu-se os nove maiores da matriz U projetando-os na matriz S, que contém os autovalores de X em ordem decrescente, formando o novo banco de dados.

5.2.4 Nonnegative Matrix Factorization

Em alguns casos como, por exemplo, ao se trabalar com dados que envolvam pixels e valores probabilísticos, estes não podem estar representados por valores negativos, para que as leis físicas não sejam contrariadas. Aqui, esuda-se um método que, dada uma matriz X_{lxn}, esta mesma é decomposta em matrizes H e W, onde

$$X \approx WH$$

e os elementos das matrizes W e H são não nulos. [1]

A função *nnmf (A, k)* fatora a matriz *Anxm* em fatores não negativos escolhidos de forma a minimizar o valor médio quadrático residual entre A e WH. Esta fatoração não é exata, pois WH tem grau inferior a A. O valor k deve ser menor ou igual ao número de colunas de A.

Para manter os valores em não negativos, utilizamos este método. Selecionamos $k = 9$ para obter uma matriz W_{nx9} e H_{9xH}. Para o valor mais forte em cada linha de H, este está relacionado ao melhor peso para a coluna correspondente de W. Portanto, para selecionar um novo conjunto de dados, selecionamos o maior valor em cada linha de H, multiplicando-o para cada coluna correspondente em W.

5.3 Inserindo os dados na rede neural

Esta seção será dividida em duas atividades: na primeira, classificamos os parâmetros de entrada em acesso normal ou intrusão. Na segunda, dividiremos a intrusão nos tipos de ataque que o banco de dados cobre, ou seja, U2R, R2L, DoS e Probe.

Então, a última coluna do banco de dados será o nosso *target*, dividido de duas maneiras diferentes. A primeira contém as classes *normal* e *intrusão* e a segunda contém as classes *normal*, *DoS*, *U2R*, *R2L* e *Probe*.

Essas classes devem ser representadas de forma binária para ser entregue á rede neural.

Para ambas as atividades a rede neural utilizada foi a mesma.

Para que pudesse ser feita uma comparação entre os métodos de seleção de atributos escolhidos, os conjuntos de dados gerados a partir de casa método foi intruduzido como dados iniciais de uma mesma rede neural.

Foi utilizada uma rede neural simples no MatLab, com os parâmetros expostos na Tabela 5.3:

Rede neural MatLab	patternnet
Número de camadas ocultas	10
Parâmetros para treinamento	70%
Parâmetros para treinamento	15%
Parâmetros para treinamento	15%
Medidor de performance	MSE

Tabela 5.3: Parâmetros da rede neural utilizada.

O arranjo esquemático da arquitetura da rede utilizada para uma rede com duas classes é mostrada na figura Figura 5.2 e em uma rede com cinco classes na Figura **??**.

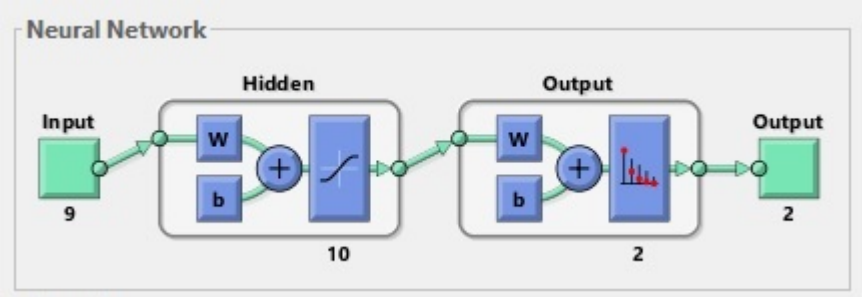

Figura 5.1: Arquitetura da Rede Neural Utilizada para Duas Classes.

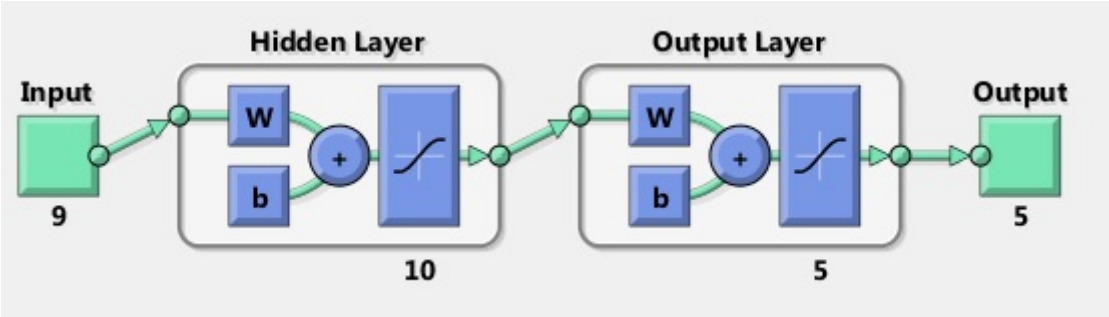

Figura 5.2: Arquitetura da Rede Neural Utilizada para Cinco Classes.

A diferença entre as redes é apenas o número de saídas, visto que esta quantidade está relacionada ao número de classes em que desejamos agrupar os dados.

Capítulo 6

Resultados

6.1 Classificação: acesso normal x invasão

Aqui iremos exibir os resultados para os algorítmos quando separamos o tipo de acesso em apenas duas classes.

6.1.1 Pincipal Component Analysis

O primeiro teste foi feito com o novo banco de dados extraído utilizando o método PCA, com o algoritmo do MatLab *pca*.

A Figura 6.1 mostra um resumo de como ocorreu o treinamento com os dados. O treinamento parou ao atingir 6 iterações na validacão, o que ocorreu no período 151.

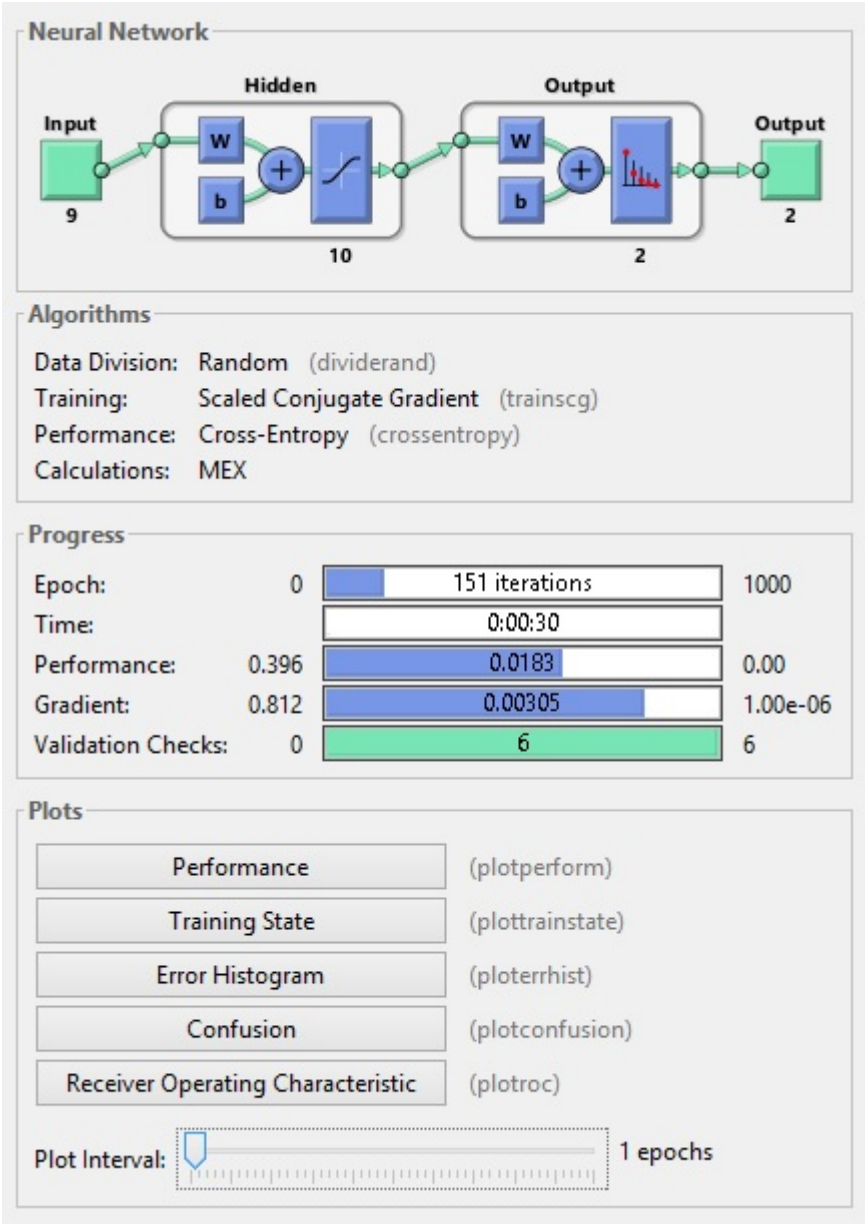

Figura 6.1: Informações gerais de treinamento para PCA com duas classes.

A fim de analisar os erros de teste, treinamento e validação, é apresentada a Figura 6.2.

O melhor desempenho para a validação ocorreu no período 145.

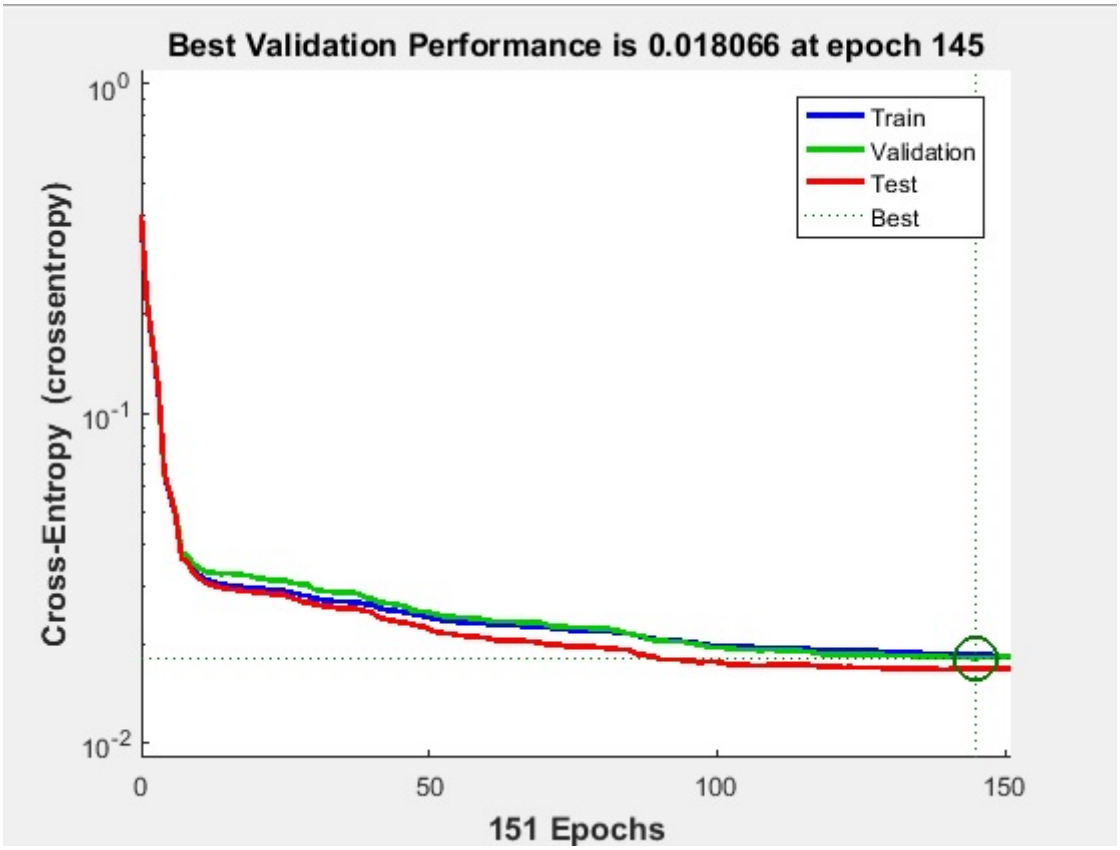

Figura 6.2: Performance do PCA com duas classes.

Na Figura 6.3 é mostrada a matriz de confusão para o PCA utilizando duas classes.

Nesta matriz, as célula na diagonal informam a porcentagem de casos em que houve correta classificação dos dados em determinada classe. As céculas fora da diagonal mostram os casos de classificação incorreta. Na célula azul, é mostrada a porcentagem total dos dados classificados corretamente, em verde, e a porcentagem dos casos classificados de maneira errada, em vermelho.

Na fase de treinamento, o resultado obtido foi de 99.2% de acertos e 0.8% de erros.

Para a validação, obtivemos um percentual de 99.3% de classificações corretas e 0.7% de classificações incorretas.

Já para a fase de teste, foi observado 99.4% de acertos e 0.6% de erros.

Portando, este método apresentou um total de 99.3% de classificações corretas e 0.7% de classificações incorretas. Este resultado mostra um reconhecimento muito bom das classes.

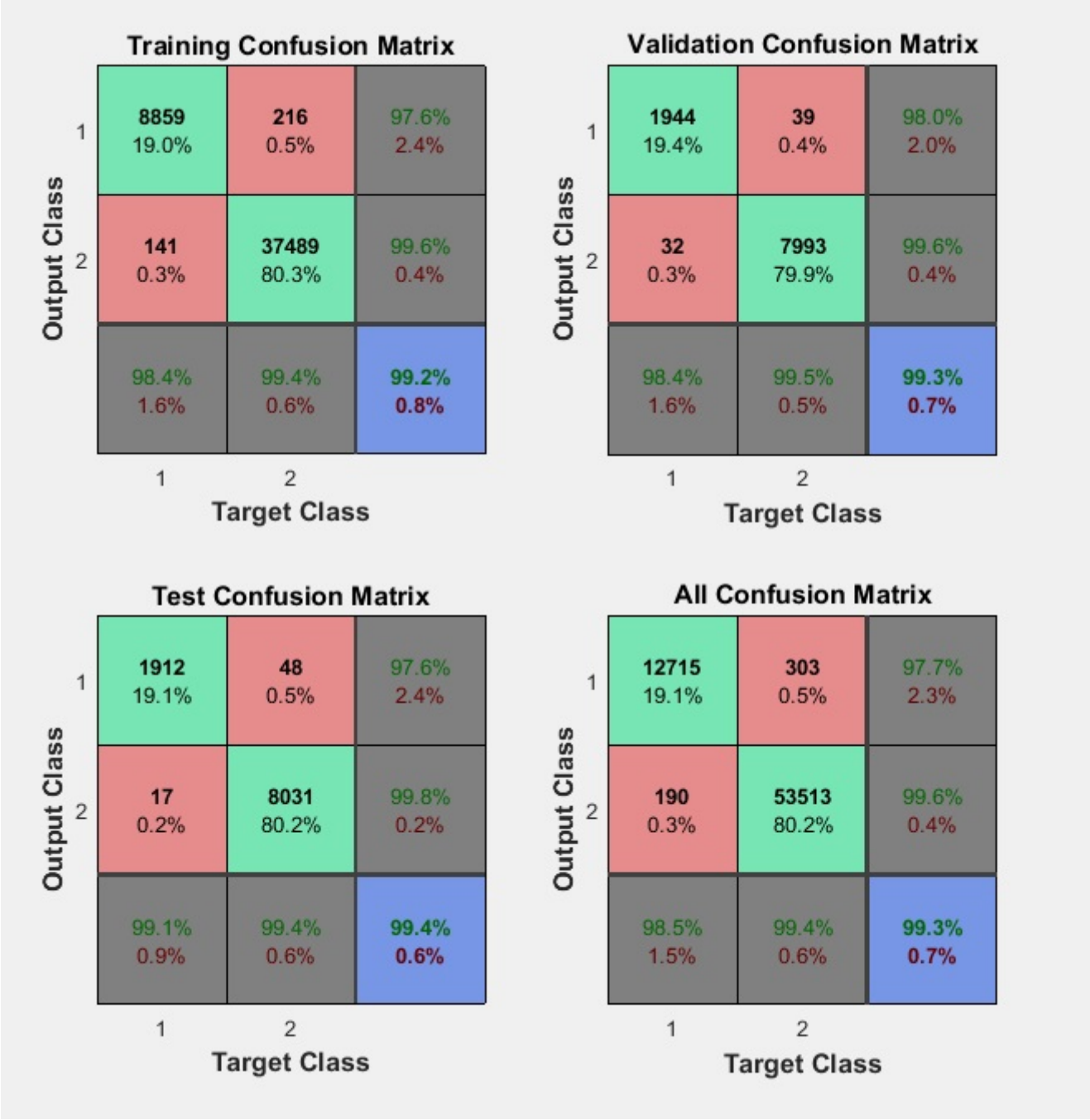

Figura 6.3: Matriz de confusão para o PCA com duas classes.

6.1.2 Factor Analysis

Agora iremos fazer a análise dos resultados para o método de seleção de atributos utilizando o *factoran*, com duas classes.

A Figura 6.4 apresenta a visão de geral do treinamento do banco de dados extraído com o *factoran* com duas classes. É possível observar que o treinamento parou quando as iterações atingiram o período 68.

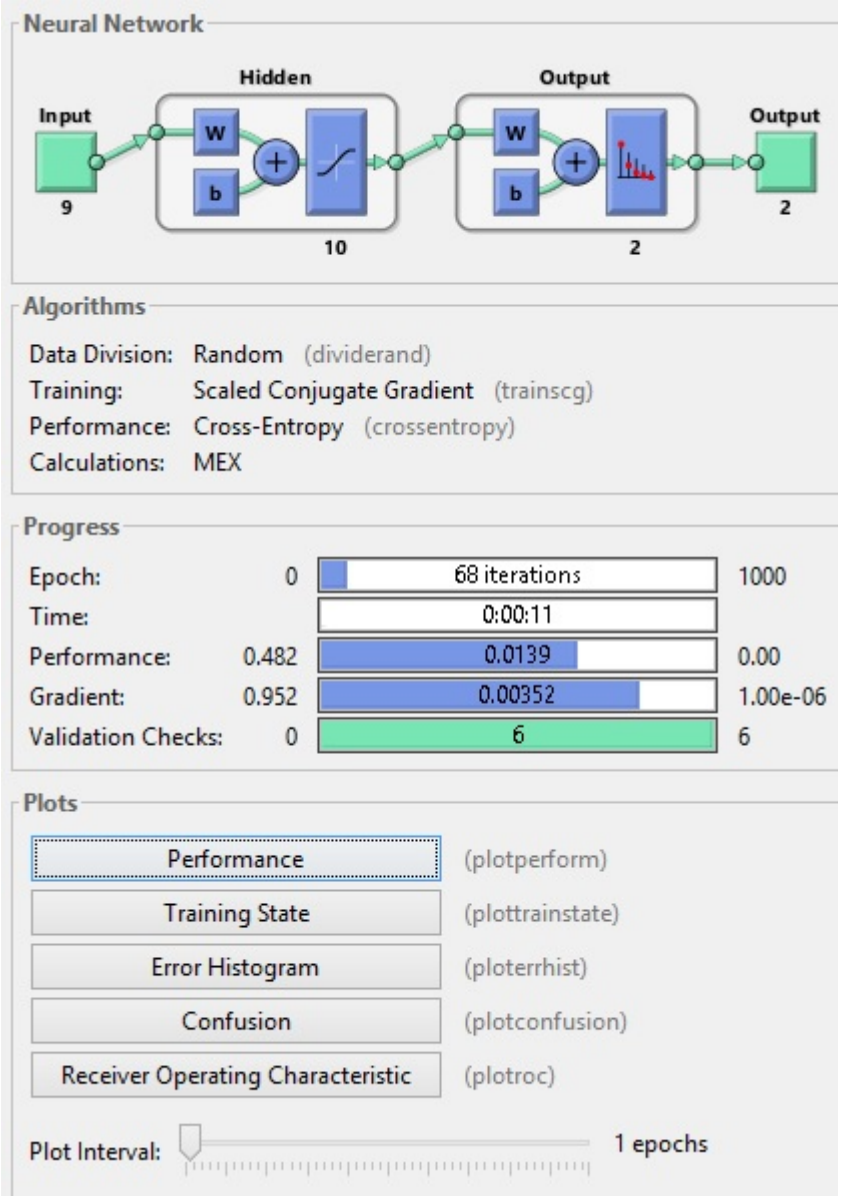

Figura 6.4: Informações gerais de treinamento para Factoran com duas classes.

Ao analisar a performance, a Figura 6.5 elucida que a mehor performance ocorreu no período 68 da rede neural, no exato momento onde o treinamento parou.

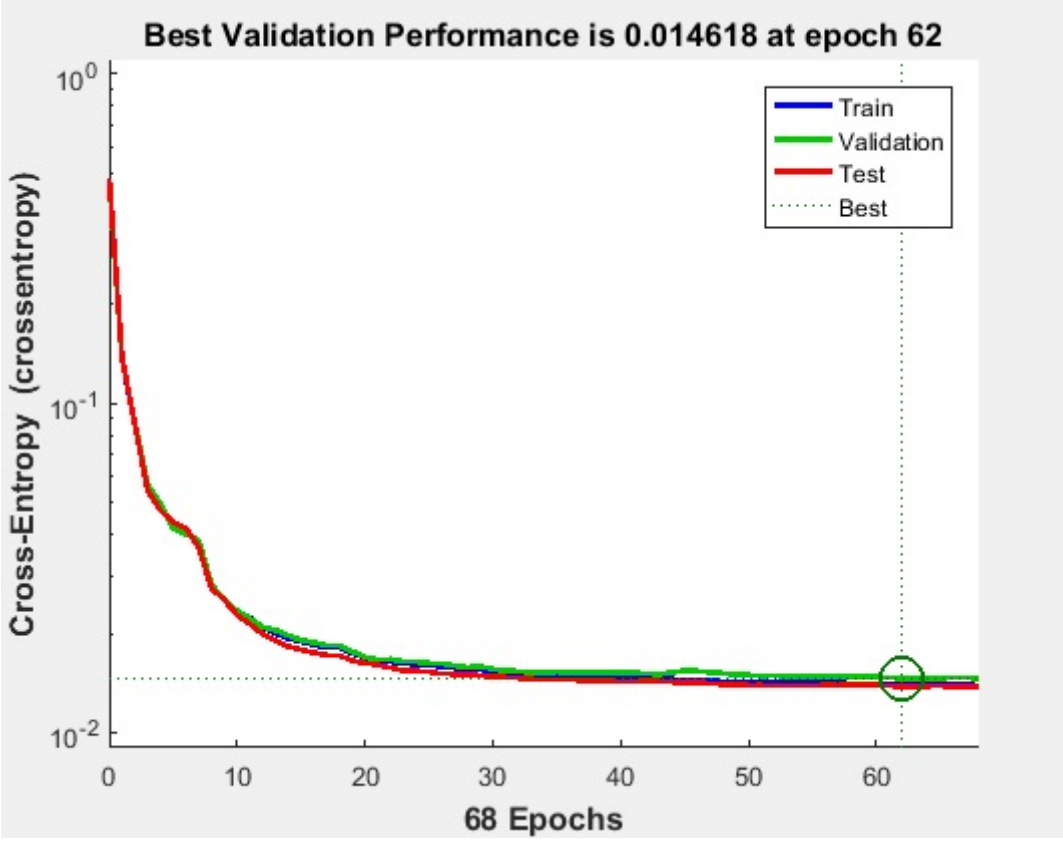

Figura 6.5: Performance do factoran com duas classes.

Ao analisar a matriz de confusão na Figura 6.6, observa-se que, na fase de treinamento, o resultado obtido foi de 99.2% de acertos e 0.8% de erros.

Para a validação, obtivemos um percentual de 99.3% de classificações corretas e 0.7% de classificações incorretas.

Já para a fase de teste, foi observado 99.2% de acertos e 0.8% de erros.

Portando, este método apresentou um total de 99.2% de classificações corretas e 0.8% de classificações incorretas. Esses resultados também são muito bons.

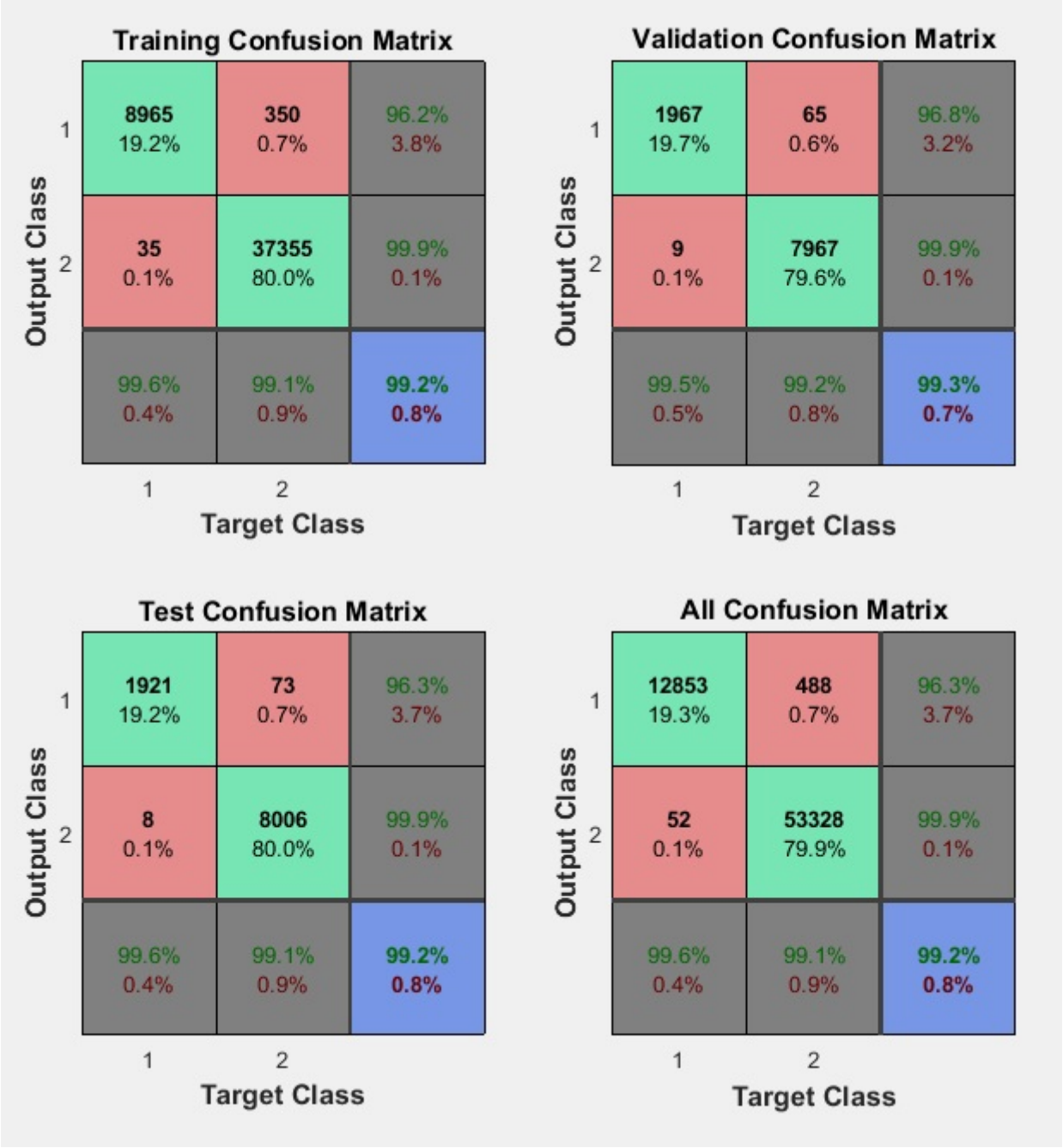

Figura 6.6: Matriz de confusão para o factoran com duas classes.

6.1.3 Singular Value Decomposition

Agora iremos fazer a análise dos resultados para o método de seleção de atributos utilizando o *SVD*, com duas classes.

A Figura 6.7 apresenta a visão de geral do treinamento do banco de dados extraído com o *SVD* com duas classes. É possível observar que o treinamento parou com 97 períodos, tendo a sua melgor performance no período 66, como é mostrado na Figura 6.8

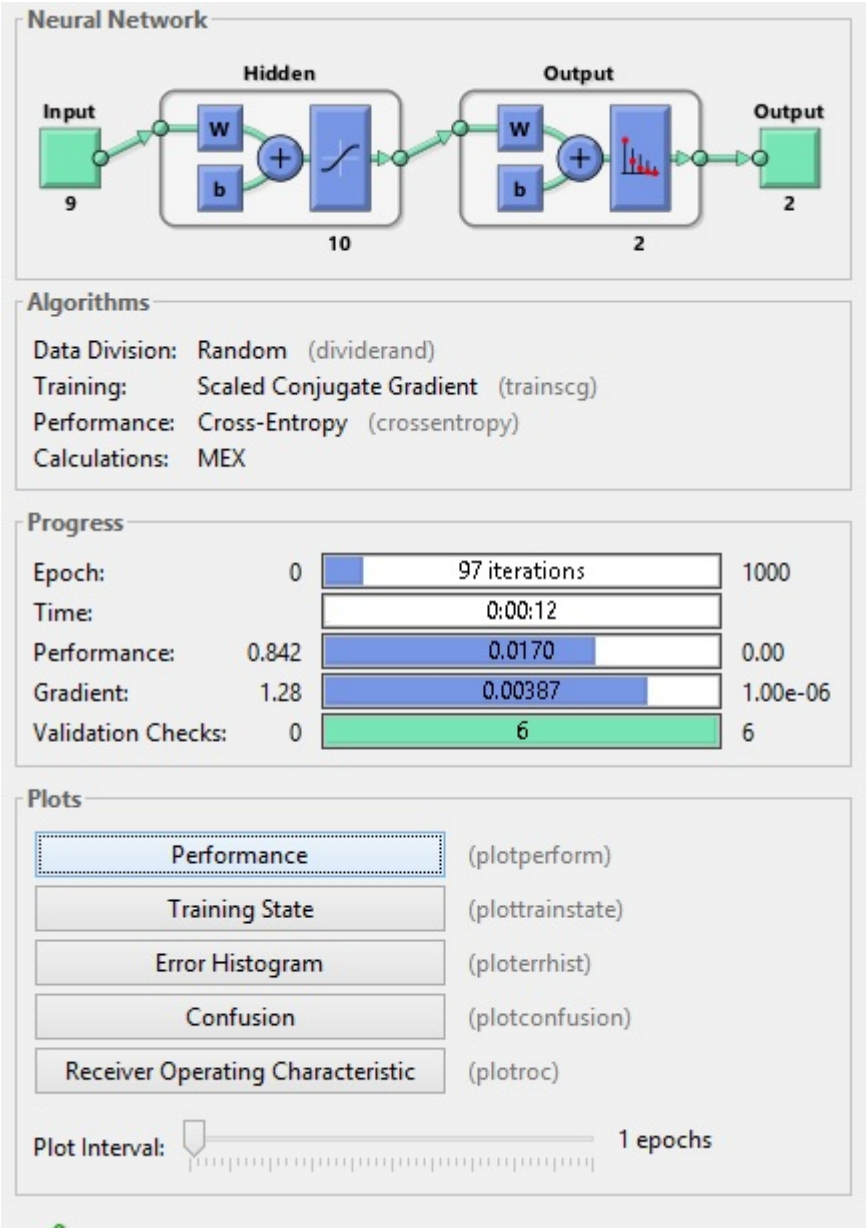

Figura 6.7: Informações gerais de treinamento para SVD com duas classes.

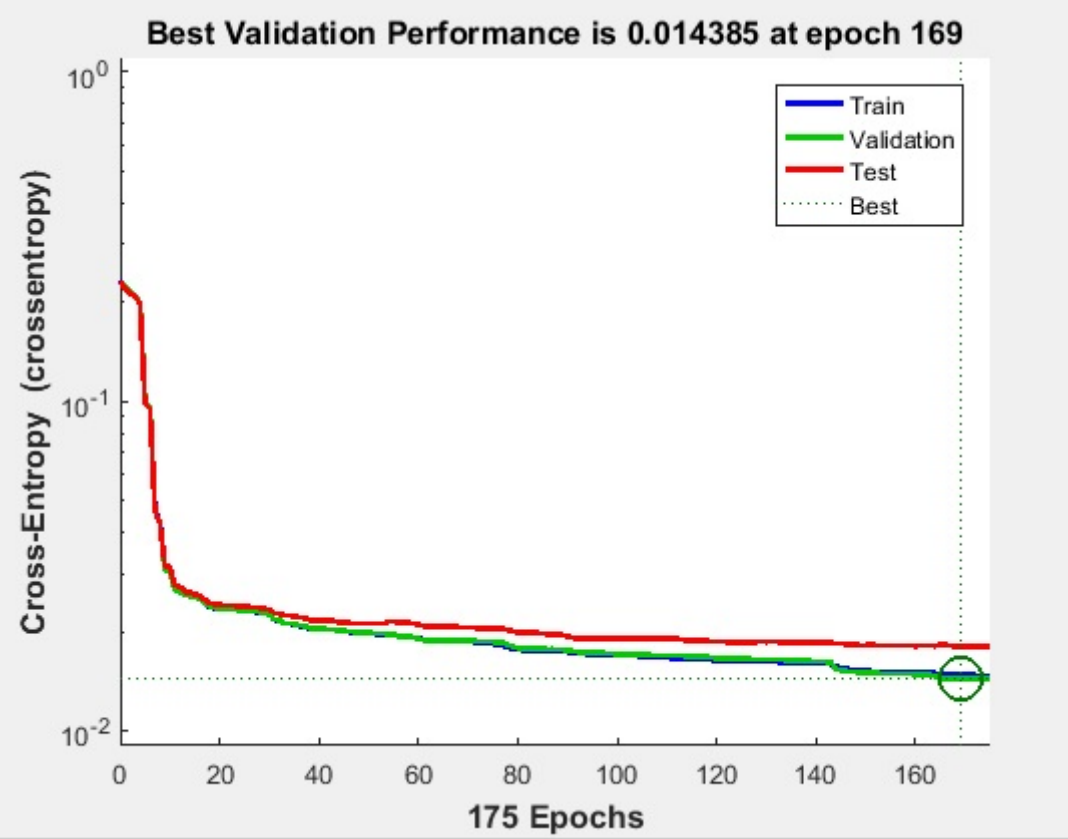

Figura 6.8: Performance do SVD com duas classes.

Ao analisar a matriz de confusão na Figura 6.9, observa-se que, na fase de treinamento, o resultado obtido foi de 99.1% de acertos e 0.9% de erros.

Para a validação, obtivemos um percentual de 99.0% de classificações corretas e 1.0% de classificações incorretas.

Já para a fase de teste, foi observado 98.9% de acertos e 1.1% de erros.

Portando, este método apresentou um total de 99.0% de classificações corretas e 1.0% de classificações incorretas. Apesar de ser um resultado não tão animador quanto os outros dois anteriores, ainda é possível observar uma excelente acurácia neste algorítmo.

As células que não estão na diagonal mostram uma excelente classificação deste método, pois nelas é possível perceber que pouco foi classificado de maneira errada.

Figura 6.9: Matriz de confusão para o SVD com duas classes.

6.1.4 Nonnegative Matrix Factorization

Por fim, será feita a análise do desempenho do algoritmo *nnmf* para duas classes.

A Figura 6.10 apresenta a visão de geral do treinamento do banco de dados extraído com o *nnmf*.

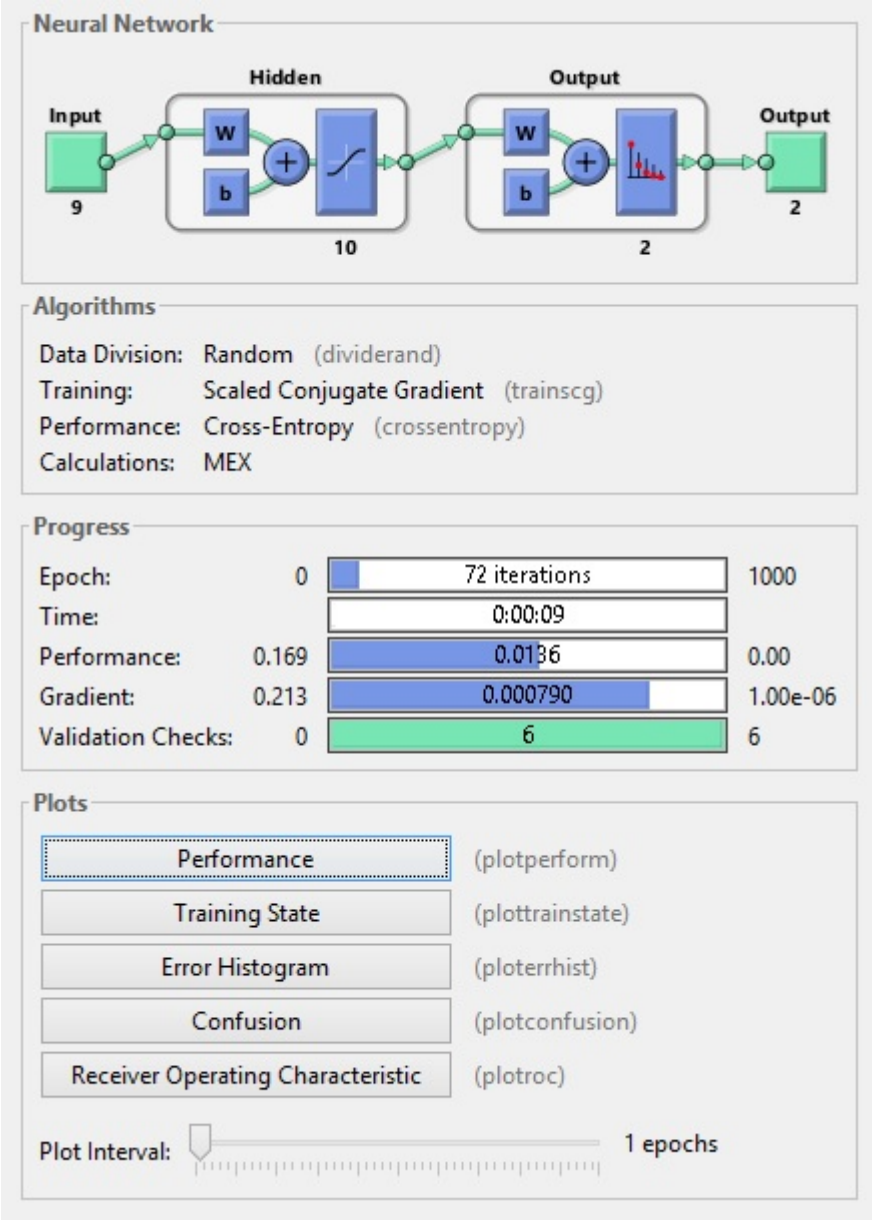

Figura 6.10: Informações gerais de treinamento para nnmf com duas classes.

Observa-se que o treinamento teve fim ao atingir 72 períodos.

A Figura 6.11 elucida que a melhor performance ocorreu com 66 períodos de iteração da rede neural.

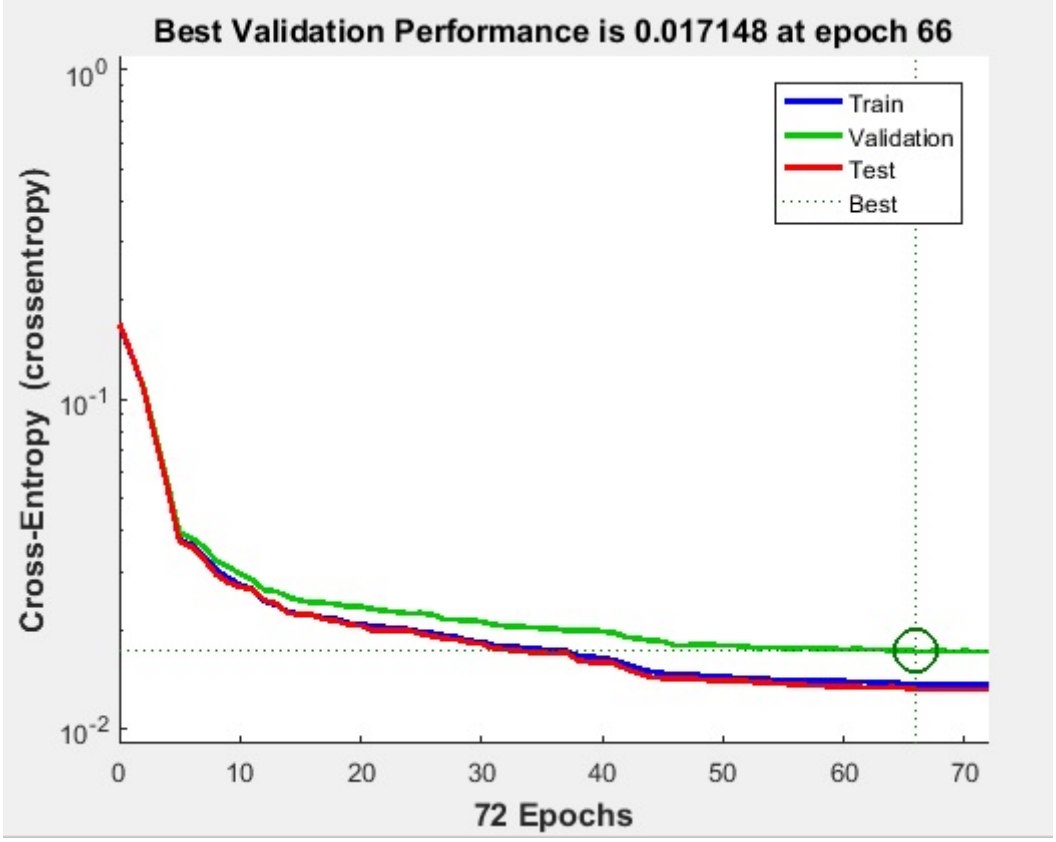

Figura 6.11: Performance do nnmf com duas classes.

Aatriz de confusão na Figura 6.12, mostra que, na fase de treinamento, o resultado obtido foi de 99.2% de acertos e 0.8% de erros.

Para a validação, obtivemos um percentual de 99.0% de classificações corretas e 1.0% de classificações incorretas.

Já para a fase de teste, foi observado 99.2% de acertos e 0.8% de erros.

Portando, este método apresentou um total de 99.2% de classificações corretas e 0.8% de classificações incorretas.

Figura 6.12: Matriz de confusão para o nnmf com duas classes.

Além disso, pode-se perceber que as saídas foram quase perfeitas ao observar nas células em cor rosa o quão pouco de dados foi mal classificado.

6.1.5 Comparação dos métodos trabalahndo com duas classes

Iremos aqui comparar os métodos utilizados para seleção de atributos quanto à sua eficiência na classificação dos dados em conexão normal ou intrusão.

A matriz de confusão mostra seu desempenho quanto à classificação em quatro casos, sendo eles treinamento, validação, teste e total. Iremos comparar o desempenho dos algorítimos quanto ao resultado total, a fim de mostrar um parecer mais geral possível.

O gráfico da Figura 6.13 mostra como se comportou cada algorítmo:

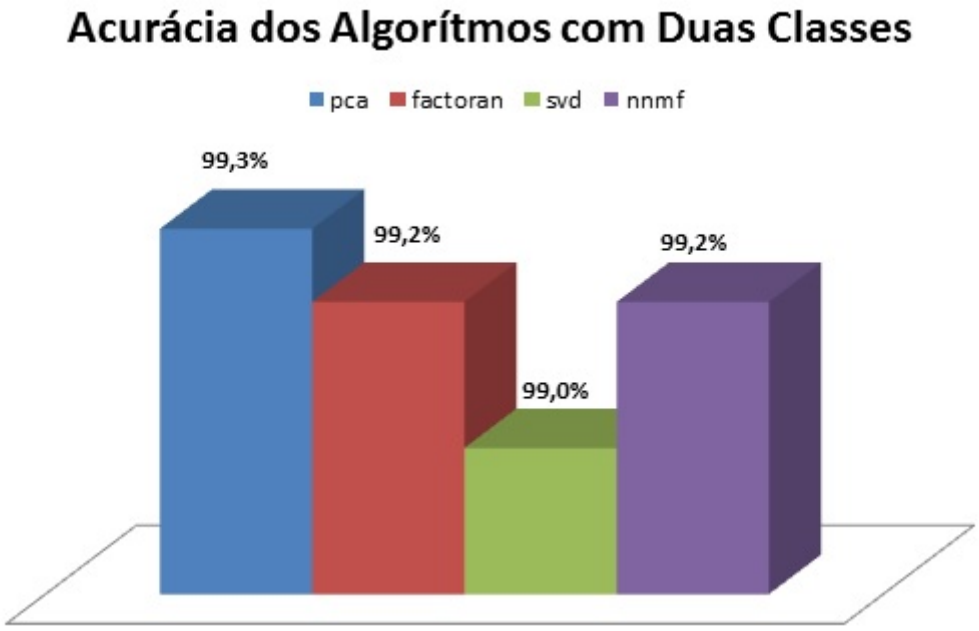

Figura 6.13: Gráfico comparativo da acurácia dos algorítimos para 2 classes.

É possível verificar que o melhor desempenho ocorreu para o algorítmo *pca*, com 99,3% de acurácia.

O pior desempenho foi observado no *svd*, com 90% de acurácia. Ainda assim, este é um resultado muito bom e este método pode ser utilizado sem problemas em aplicações como a exposta neste trabalho.

6.2 Classificação: acesso normal x tipo de invasão

Agora iremos analisar os resultados dos métodos utilizados quando fazemos uma divisão nos ataques. Como citado anteriormente, além de conexão normal, temos conexões consideradas ataques do tipo DoS, U2R, R2L e Probe. Então, iremos aqui analisar os resultados para essas cinco classes.

Aqui os ataques serão representados por seu número de classe. O ataque Dos é representado pela classe 2, o U2R pela classe 3, Probe pela classe 4 e R2L pela classe 5.

6.2.1 Principal Component Analysis

Seguindo a ordem da avaliação anterior, iremos começar pelo *pca* que utiliza como método o *Principal Component Analysis*.

A Figura 6.14 fornece uma ideia geral do tratamento da rede neural quando trabalhamos com este algorítmo. Foi atingido o número de 322 períodos.

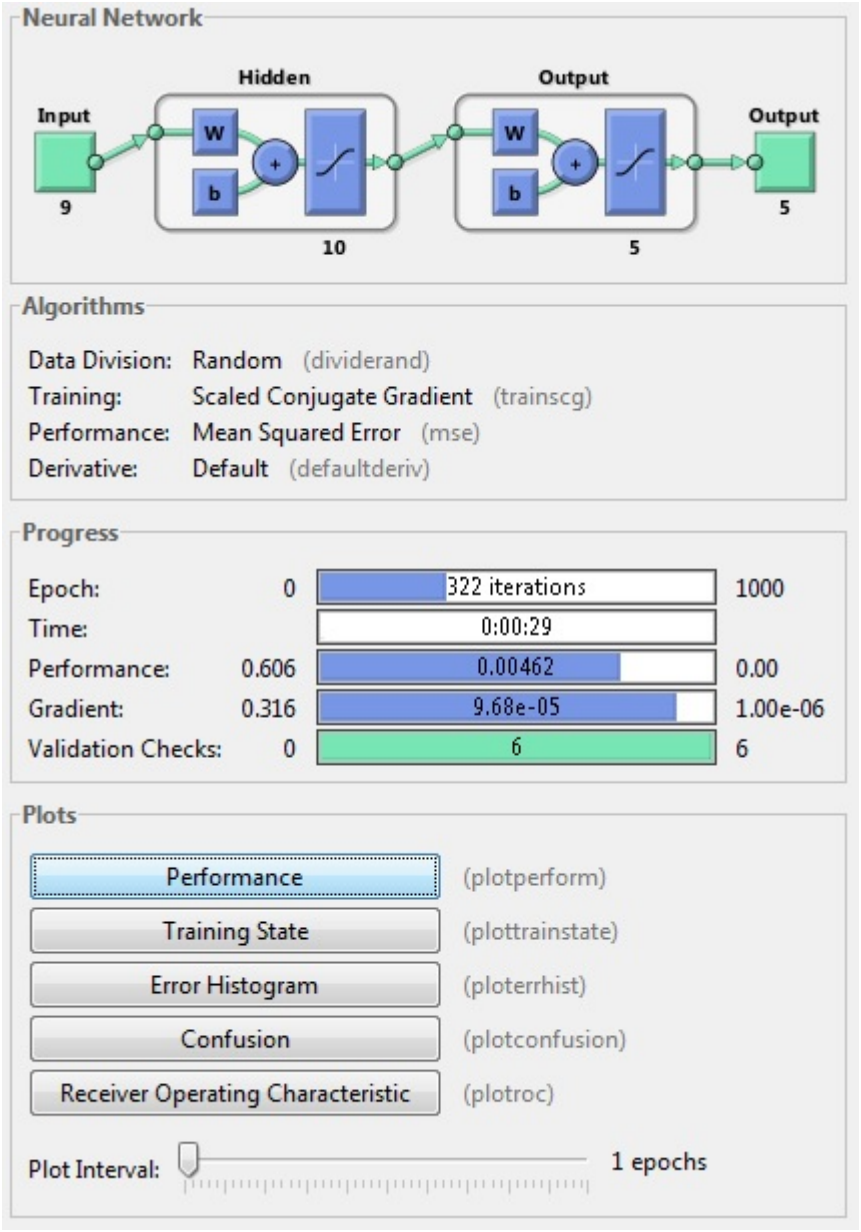

Figura 6.14: Informações gerais de treinamento para PCA com cinco classes.

A melhor performance ocorreu no período 316, como exibido na Figura **??**.

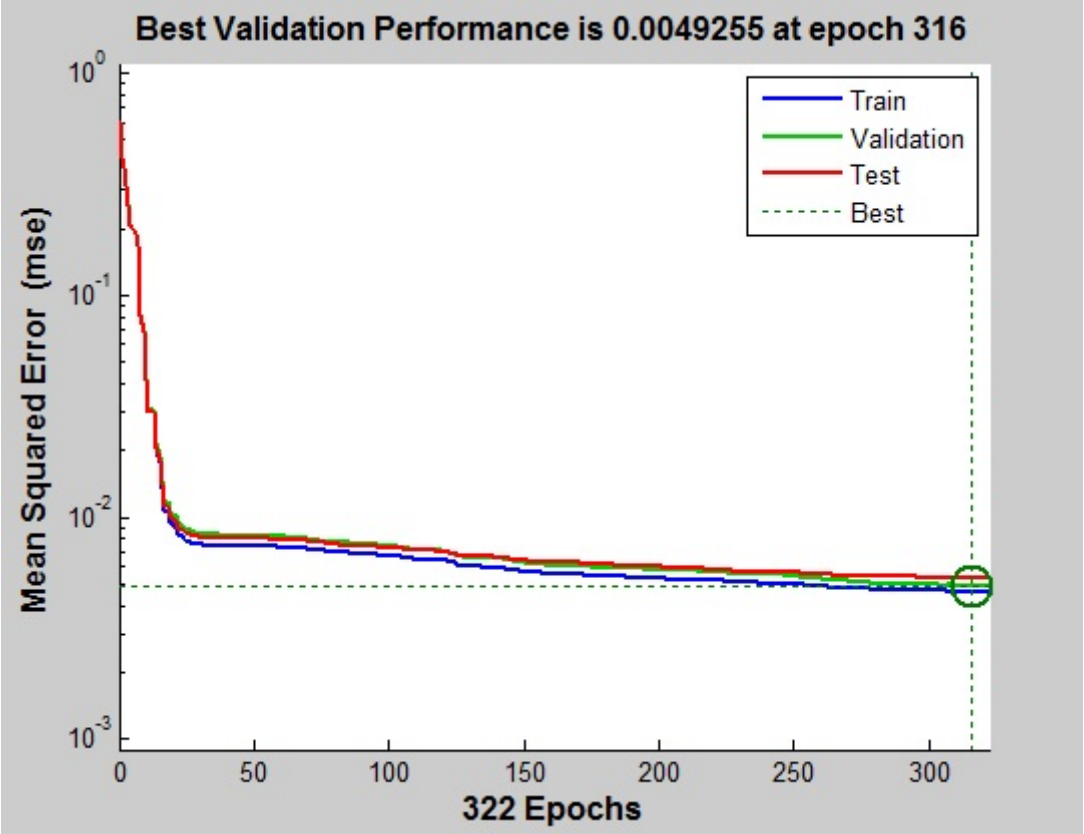

Figura 6.15: Performance do PCA com cinco classes.

Investigando a matriz de confusão deste caso, na Figura 6.16, percebemos uma precisão de 98,8% na fase de treinamento, 98,7% na fase de validação e 98,6% na fase de teste. O total geral de precisão foi de 98,8% de classificações bem sucedidas e 1,2% de classificações mal sucedidas.

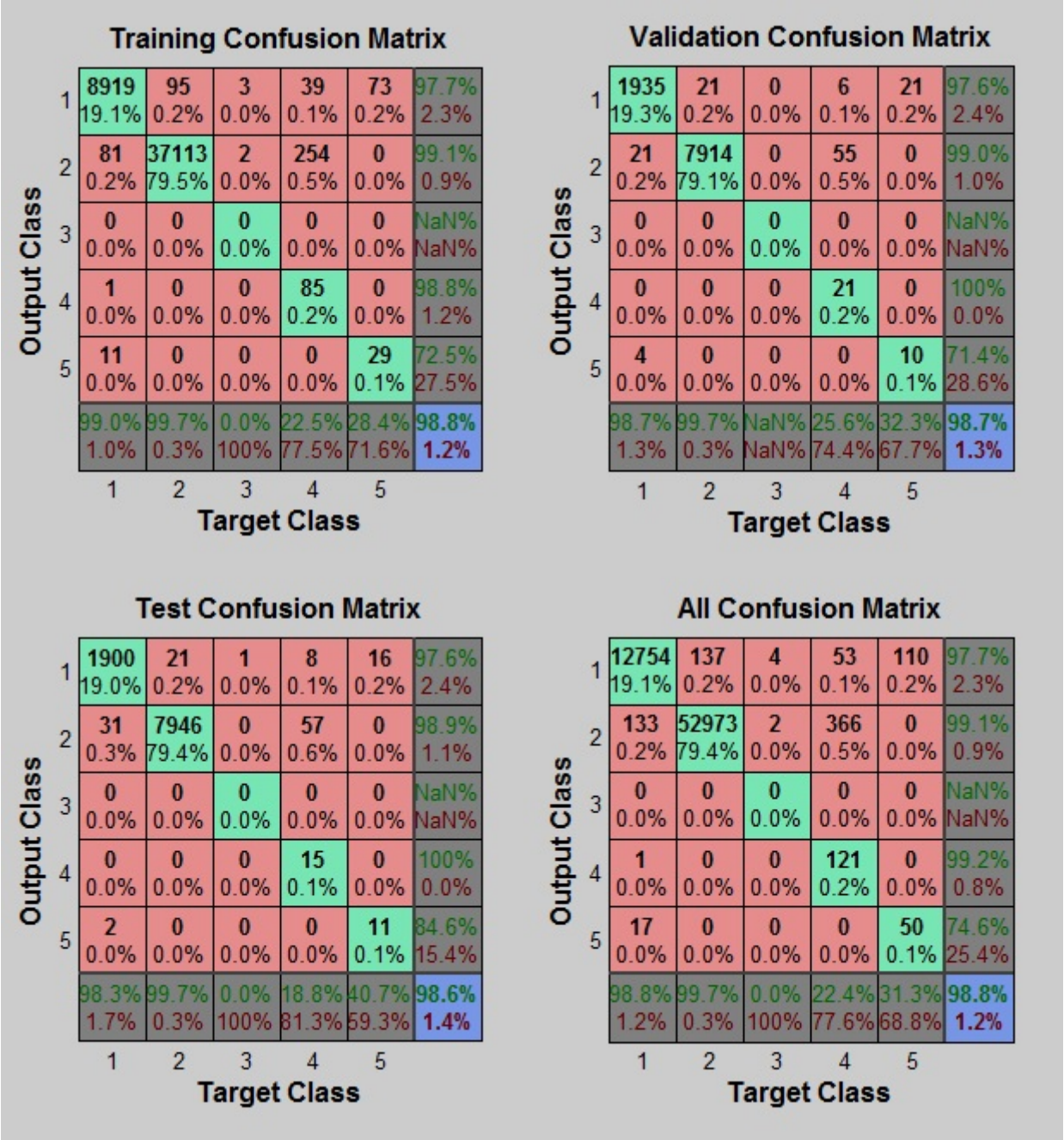

Figura 6.16: Matriz de confusão do PCA com cinco classes.

As células coloridas com rosa mostram uma classificação muito boa, sendo a pior delas 2,4% de erro de classe na fase de teste para a classe 2, que representa o ataque DoS.

6.2.2 Factor Analysis

A Figura 6.17 fornece uma ideia geral do tratamento da rede neural quando trabalhamos com este algorítmo. Foi atingido o número de 64 períodos.

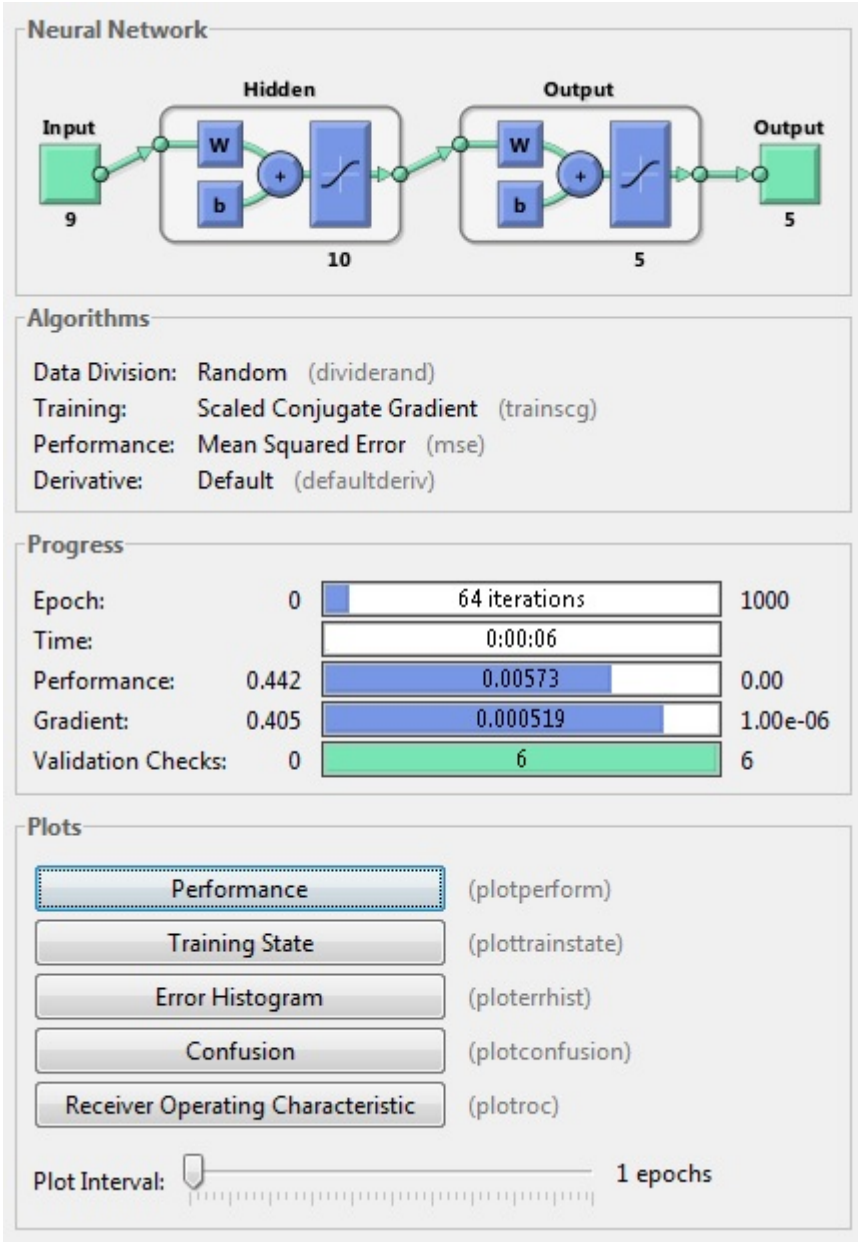

Figura 6.17: Informações gerais de treinamento para factoran com cinco classes.

Sua melhor performance está relacionada ao período 58, como mostra a Figura **??**.

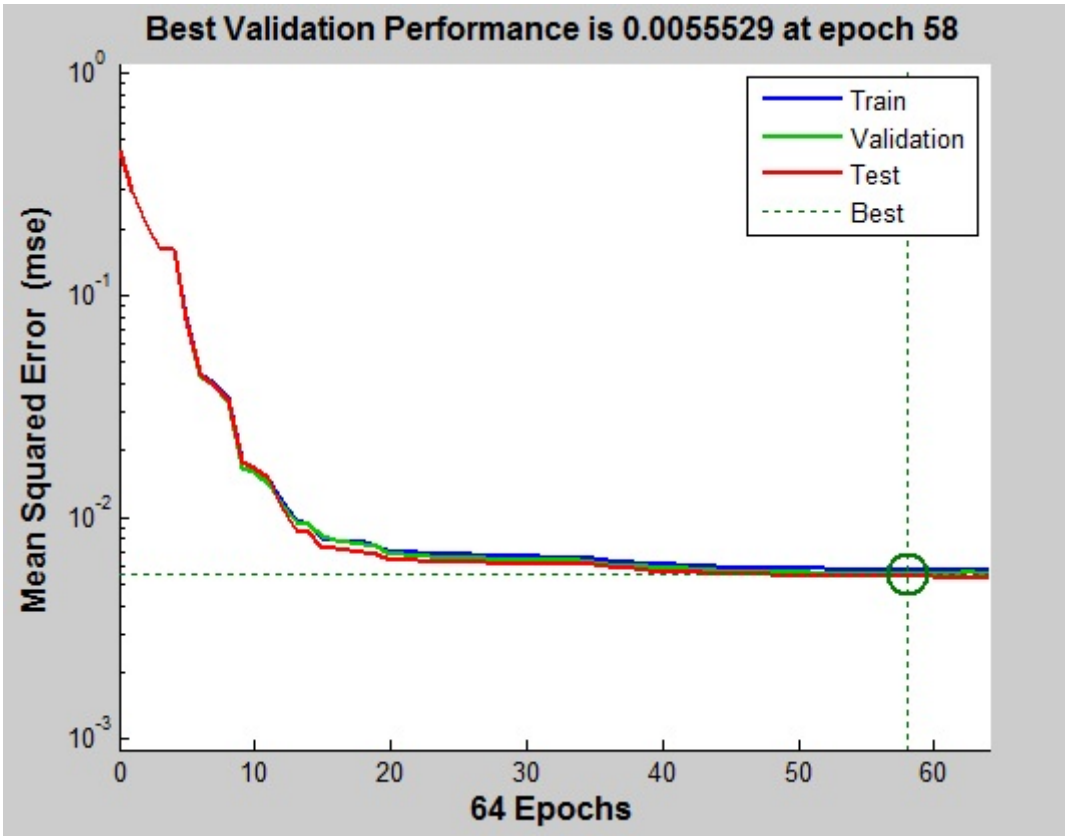

Figura 6.18: Performance do factoran com cinco classes.

A matriz de confusão na Figura 6.19, mostra que o *factoran* obteve acurácia de 98,4% na fase de treinamento, 98,4% na fase de validação e 98,5% na fase de teste. O total geral de acurácia foi de 98,4% de classificações bem sucedidas.

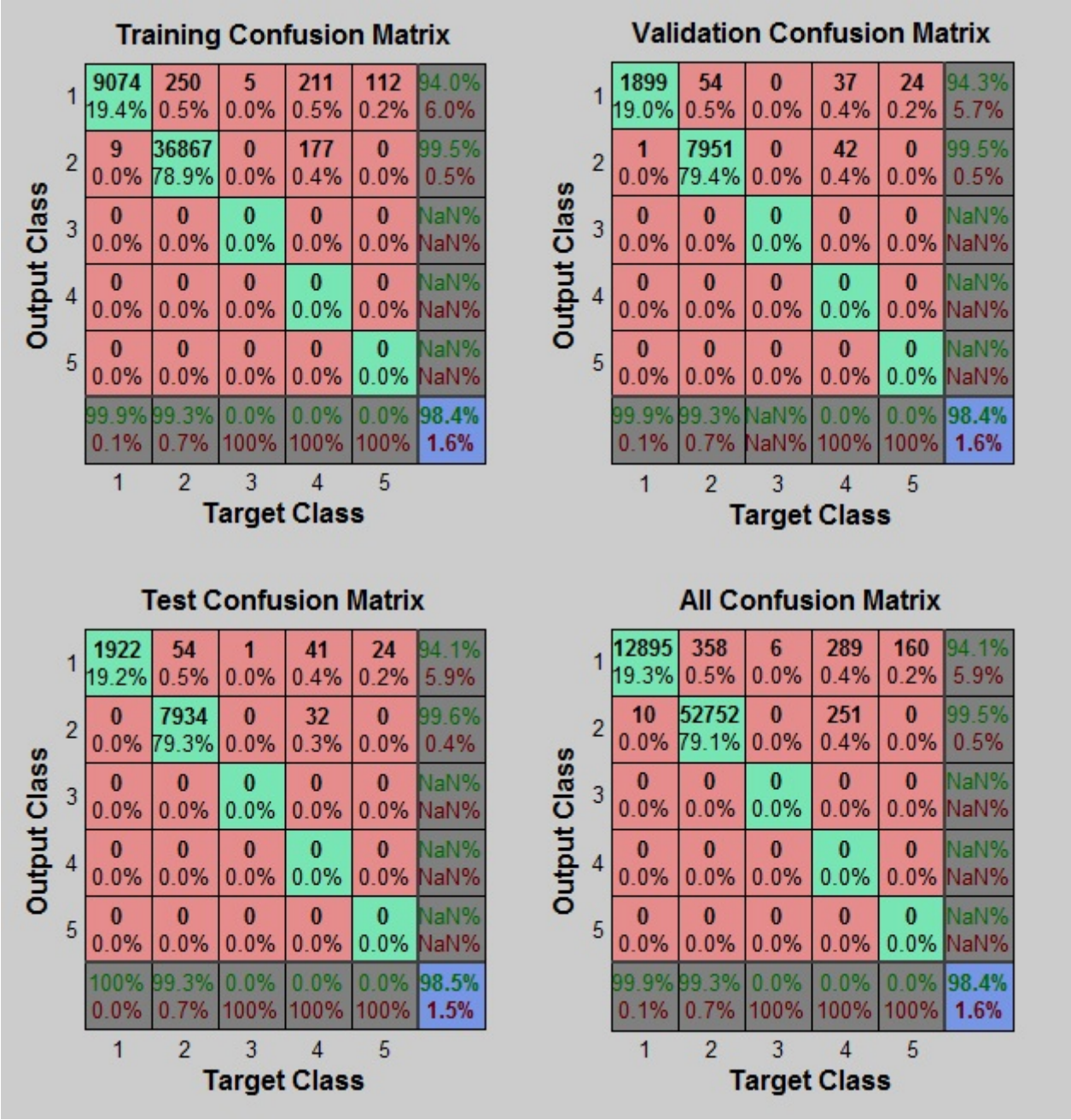

Figura 6.19: Matriz de confusão do factoran com cinco classes.

Pode-se observar excelente classificação dos dados.

6.2.3 Singular Value Decomposition

Para ter uma ideia geral do algorítmo *svd*, mostra-se a Figura 6.20.

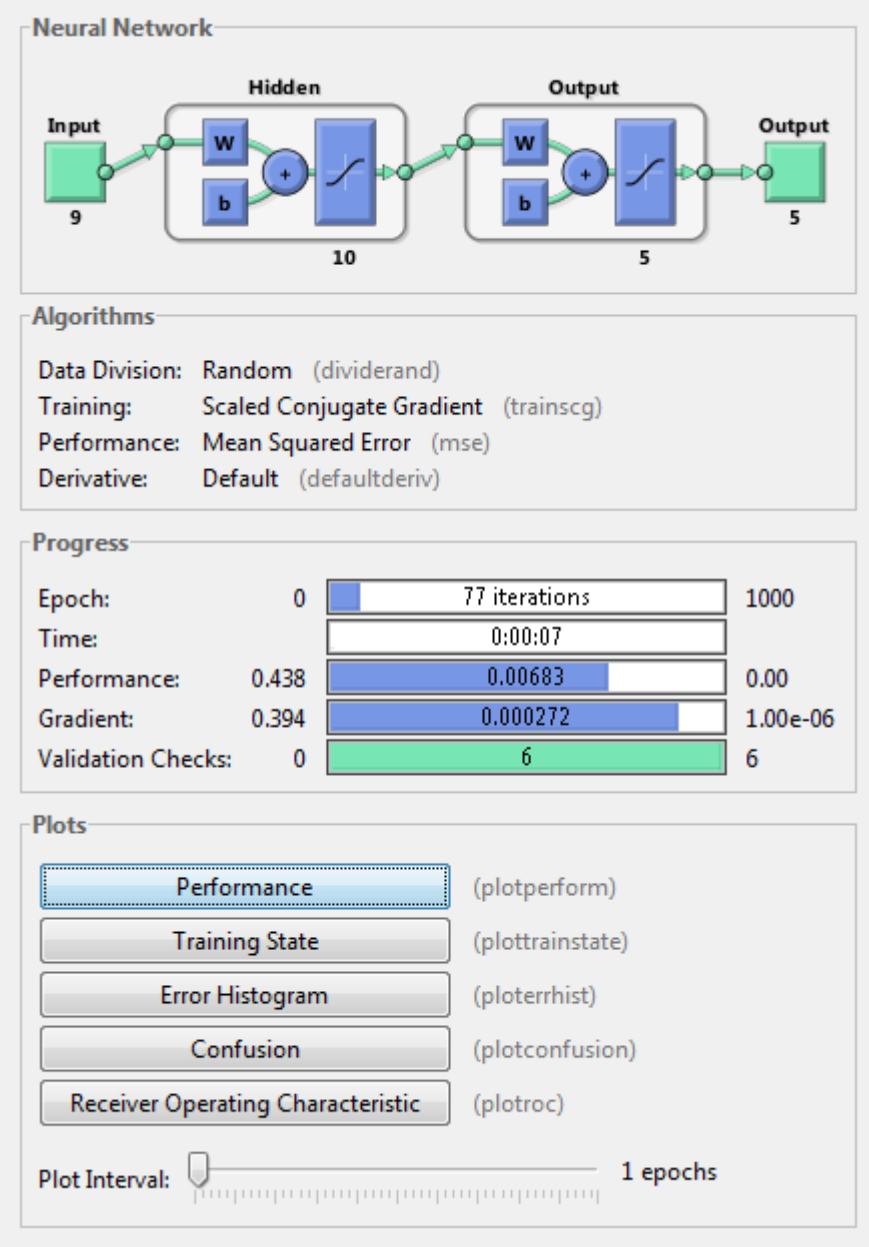

Figura 6.20: Informações gerais de treinamento para svd com cinco classes.

O treinamento ocorreu até que atingisse 77 períodos.

Sua melhor performance ocorreu no período 71, como mostra a Figura **??**.

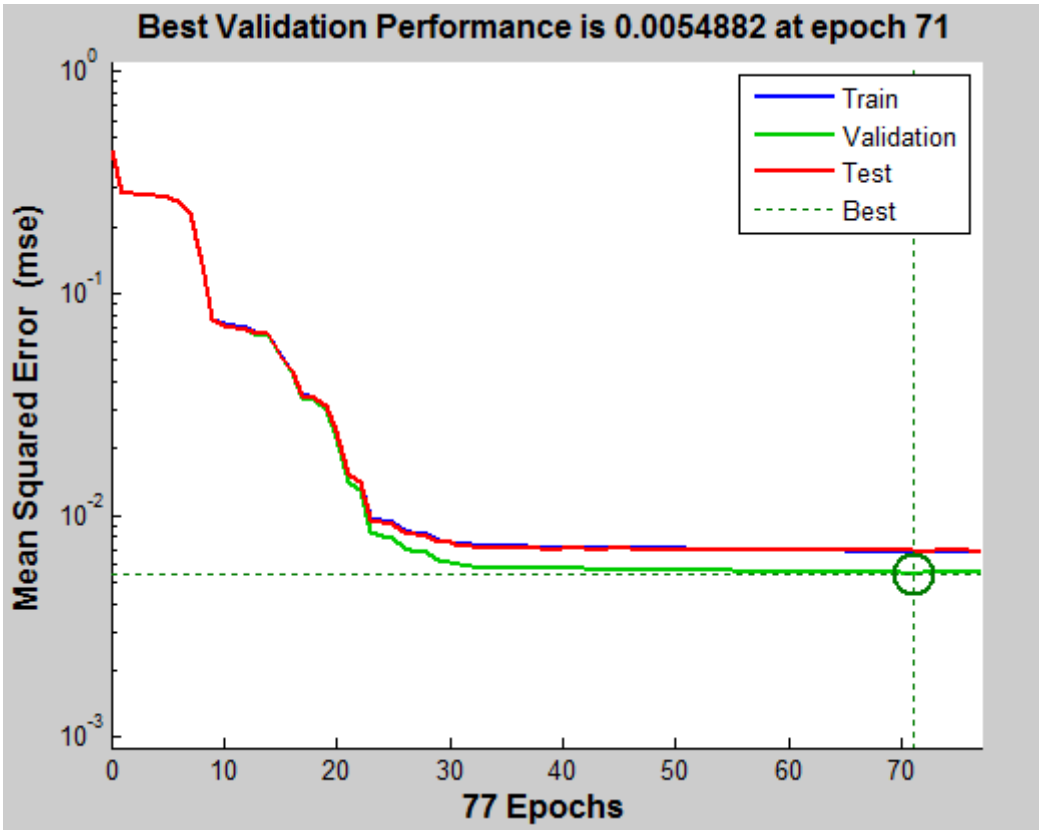

Figura 6.21: Performance do svd com cinco classes.

O algorítmo em questão Figura 6.22, apresentou acurácia de 98,2% na fase de treinamento, 98,6% na fase de validação e 98,2% na fase de teste. O total geral de acurácia foi de 98,2% de classificações bem sucedidas e 1,8% de classificações mal sucedidas.

Figura 6.22: Matriz de confusão do svd com cinco classes.

6.2.4 Nonnegative Matrix Facotization

O resultado geral do nnmf é mostrado na Figura 6.23.

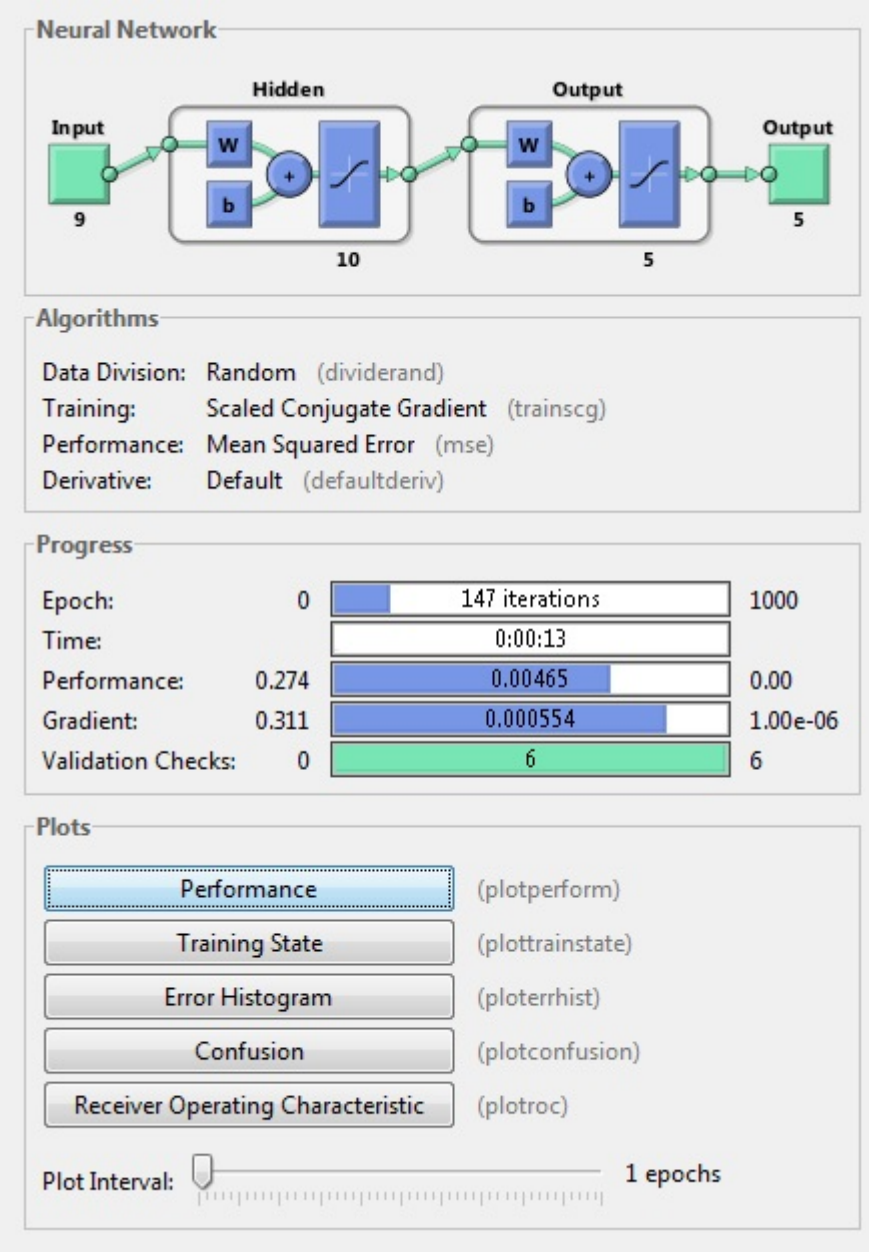

Figura 6.23: Informações gerais de treinamento para nnmf com cinco classes.

O treinamento ocorreu até que atingisse 147 períodos.

A Figura 6.24 mostra o gráfico da sua performance. A melhor performance ocorreu no período 141.

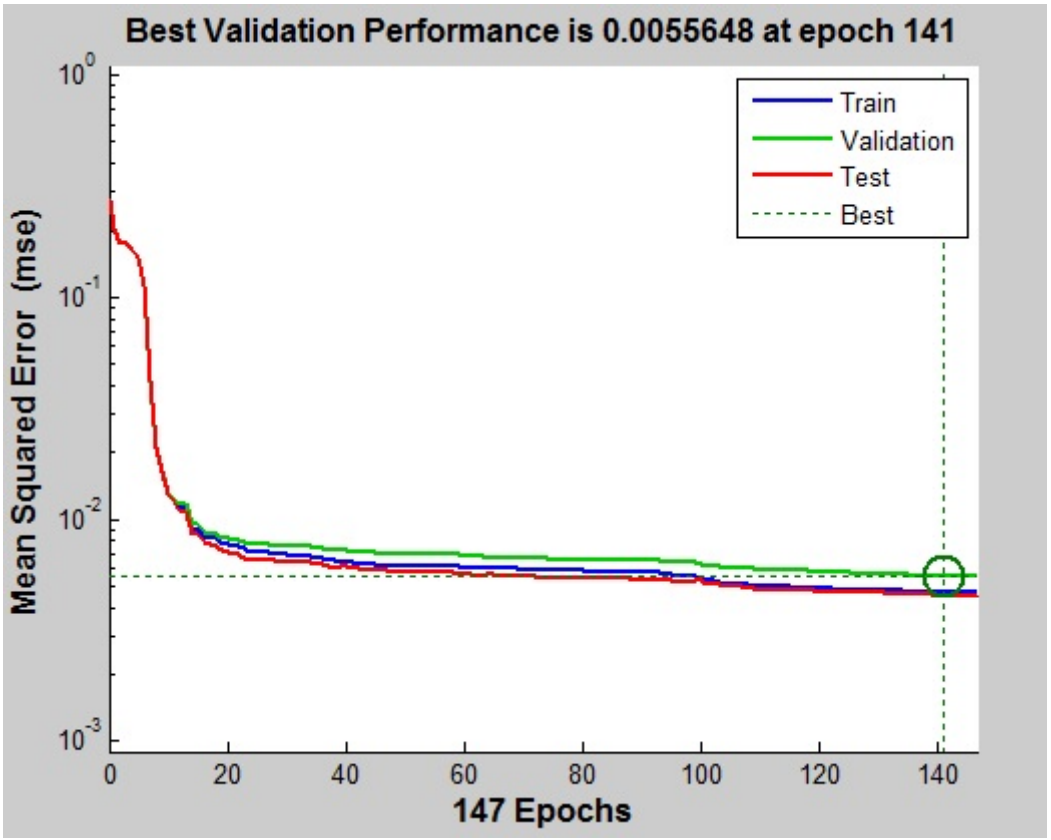

Figura 6.24: Performance do nnmf com cinco classes.

Na Figura 6.25, vemos que o *nnmf* apresentou acurácia de 98,5% na fase de treinamento, 98,2% na fase de validação e 98,6% na fase de teste. O total geral de acurácia foi de 98,5% de classificações bem sucedidas.

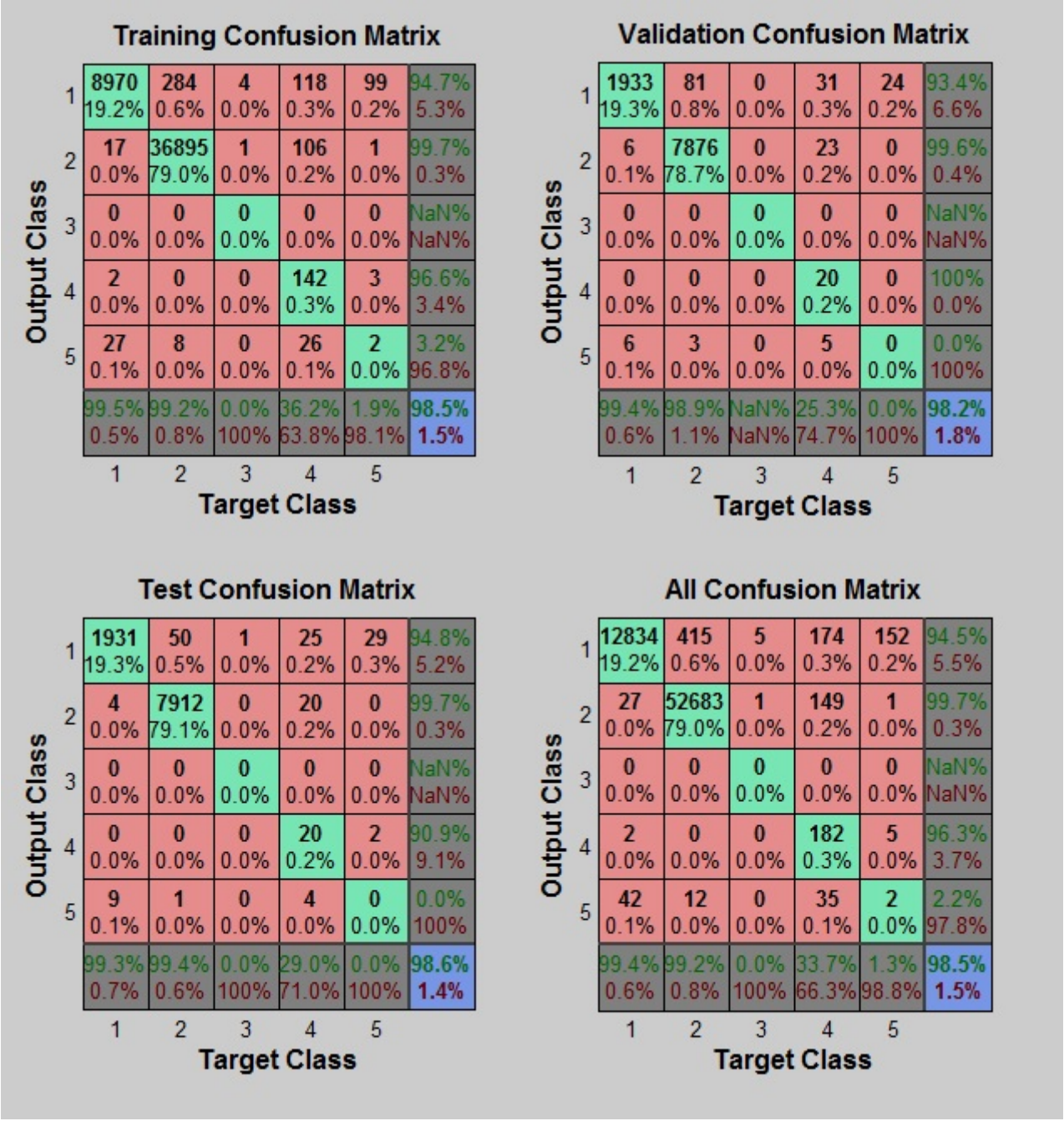

Figura 6.25: Matriz de confusão do nnmf com cinco classes.

6.2.5 Comparação dos métodos trabalhando com cinco classes

Adotando este tipo de classificação, é possível verificar que algumas classes tiveram pouca ou nenhuma representatividade perante as demais. Isso se dá pelo número desproporcional de amostras das diferentes classes. O banco de dados utilizado não é homogêneo em relação as classes, ou seja, algumas apresentam exemplares em quantidade extremamente maior que outras. Isso faz com que o classificador não seja capaz de reconhecer as classes menos

favorecidas com tanta precisão e, por vezes, tal classificador pode fazer a fusão entre duas classes que o mesmo julga semelhantes.

Apesar do desempenho em relação a classificações corretas ter caído em relação ao tipo de classificação anterior, o *pca* continua sendo o melhor método de seleção de atributos para este caso, como exposto na Figura 6.26.

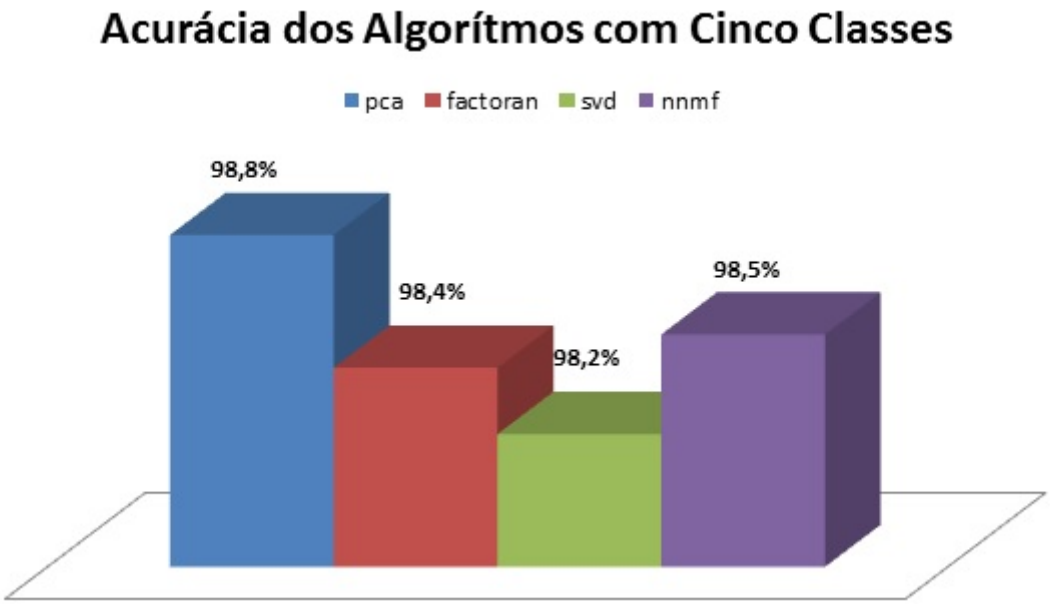

Figura 6.26: Gráfico comparativo da precisão dos algorítimos para 5 classes.

No entando, se antes os algorítmos *factoran* e *nnmf* estavam empatados neste tipo de comparação, agora é possível observar uma sutil diferença entre os mesmos.

O *svd* continua tendo um pior desempenho, contudo, o *Singular Value Decomposition* ainda pode ser considerado um bom método.

Ao se adicionar mais classes é comum obter uma queda no funcionamento da rede, já que agora ela necessita trabalhar com um maior número de opções para tomar decisões.

Capítulo 7

Conclusão

A partir do exposto, podemos perceber que apesar de todos os algorítimos terem apresentado bons resultados, o que obteve melhor desempenho em ambos os casos foi o pca, que utiliza o método Principal Component Analysis, como mostrado nos gráficos expostos nas Figuras 6.13 e 6.26.

Esse resultado era esperado, visto que o método que mais se destacou é o mais utilizado para fins como este.

Os outros métodos apresentados também obtiveram um excelente resultado, com diferenças mínimas de precisão. Isso se deve ao fato de, incialmente, já ter sido feita uma pré seleção de atributos, retirando do banco de dados atributos que não agregavam informações, deixando o mesmo pouco vulnerável ao método utilizado. Isto significa que as informações contidas no KDD CUP 99 são capazes de formar bons subconjuntos de dados para testes como o que foi feito neste trabalho.

O fato de o classificador ter ignorado algumas classes ao se dividir o banco de dados em cinco classes mostra que poderia ter sido feito incialmente o balanço de amostras para cada classe. Este processo não foi feito no presente trabalho a fim de se tentar preservar o máximo possível os dados originais.

Capítulo 8

Sugestões para trabalhos futuros

Com base no trabalho desenvolvido, podem ser identificadas vertentes para trabalhos futuros.

A rede utilizada no MatLab foi uma rede padrão que o programa oferece. É interessante fazer alterações na rede, combinando diversas outras características a fim de observar se os resultados dos métodos testados se diferenciam de forma mais acentuada.

Pode-se não restringir o ambiente de trabalho ao MatLab, verificando maneiras de implementar o que foi feito de outras maneiras, como, por exemplo, programando em Python. Desta forma, outros métodos de seleção que aqui não foram abordados, como, por exemplo, o *Idependent Component Analysis*, o ICA, podem ser testados e comparados.

Caso haja necessidade de fazer a classificação em cinco classes, ou seja, separar o classificador em acesso normal e tipo de ataque, é interessante deixar os dados com número de amostras próximos para que não ocorra falha na classificação, garantindo que nenhuma classe pequena será dissolvida entre as maiores.

Referências Bibliográficas

[1] KOUTROMBAS, Sergio Theodoridis, Konstantinos Koutroumbas. *Pattern Recognition, 4a. edição*, Academic Press, 2009.

[2] CARVALHO, Luciano Gonçalves. *Segurança de Redes*, Editora Ciência Moderna, 2005.

[3] NEHA G. RELAN, DHARMARAJ R. PATIL. *Applying Neural Network to U2R Attacks*, 2010 IEEE Symposium on Industrial Electronics and Applications, 2010.

[4] IFTIKHAR AHMAD, AZWEEN B ABDULLAH, ABDULLAH S ALGHAMDI. *Implementation of Network Intrusion Detection System using Variant of Decision Tree Algorithm*, 2015 International Conference on Nascent Technologies in the Engineering Field, 2015.

[5] I MUKHOPADHYAY, M CHAKRABORTY, S CHAKRABARTI, T CHATTERJEE. *Back Propagation Neural Network Approach to Intrusion Detection System*, 2011 International Conference on Recent Trends in Information Systems, 2011.

[6] SHREYA DUBEY, JIGYASU DUBEY. *KBB:A Hybrid Method for Intrusion Detection*, 2015 IEEE International Conference on Computer, Communication and Control, 2015.

[7] POOJITHA G., NAVEEN KUMAR K., JAYARAMI REDDY P. *Intrusion Detection using Artificial Neural Network*, 2010 Second International conference on Computing, Communication and Networking Technologies, 2010.

[8] MOHAMMED A. AMBUSAIDI, XIANGJIAN HE, PRIYADARSI NANDA, ZHIYUAN TAN. *Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm*, IEEE Transactions on Computers, VOL. 65, NO. 10, 2016.

[9] V. BOLON-CANEDO, N. SANCHEZ-MAROFIO, A. ALONSO-BETANZOS. *A Combination of Discretization and Filter Methods for Improving Classification Performance in KDD Cup 99 Dataset*, Proceedings of International Joint Conference on Neural Networks, Atlanta, Georgia, USA, 2009.

[10] ALEX L. RAMOS, CÍCERO N. DOS SANTOS. *Combinando Algoritmos de Classificação para Detecção de Intrusão em Redes de Computadores*, Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, Brasil, 2011.

[11] SALVATORE J. STOLFO, WEI FAN, WENKE LEE, ANDREAS PRODROMIDIS, PHILIP K. CHAN. *Cost-based Modeling for Fraud and Intrusion Detection: Results from the JAM Project*, Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX '00), Hilton Head. Janeiro de 2000

[12] LÍLIA DE SÁ DILVA, *Uma metodologia para detecção de ataques no tráfego de redes baseada em redes neurais*, 2007. 256. Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada - INPE, São José dos Campos

[13] INTERNET ENGINEERING TASK FORCE, *Internet Security Glossary, Version 2*, agosto de 2007