MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2016/04.20.20.44-TDI

CLASSIFICAÇÃO AUTOMÁTICA DE SUPERNOVAS USANDO REDES NEURAIS ARTIFICIAIS

Marcelo Módolo

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Lamartine Nogueira Frutuoso Guimarães, e Reinaldo Roberto Rosa, aprovada em 04 de maio de 2016.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3LHG4GE>

INPE
São José dos Campos
2016

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/6921
Fax: (012) 3208-6919
E-mail: pubtc@inpe.br

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21b/2016/04.20.20.44-TDI

CLASSIFICAÇÃO AUTOMÁTICA DE SUPERNOVAS USANDO REDES NEURAIS ARTIFICIAIS

Marcelo Módolo

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Lamartine Nogueira Frutuoso Guimarães, e Reinaldo Roberto Rosa, aprovada em 04 de maio de 2016.

URL do documento original:
<http://urlib.net/8JMKD3MGP3W34P/3LHG4GE>

INPE
São José dos Campos
2016

Dados Internacionais de Catalogação na Publicação (CIP)

Módolo, Marcelo.

M721c Classificação automática de supernovas usando redes neurais artificiais / Marcelo Módolo. – São José dos Campos : INPE, 2016.

xxvi + 204 p. ; (sid.inpe.br/mtc-m21b/2016/04.20.20.44-TDI)

Tese (Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2016.

Orientadores : Drs. Lamartine Nogueira Frutuoso Guimarães, e Reinaldo Roberto Rosa.

1. Classificação automática de supernovas. 2. Análise do espectro de supernovas. 3. Classificação de supernovas a partir da análise do espectro. 4. Redes neurais artificiais. 5. Inteligência computacional. I.Título.

CDU 004.032.26:524.352

Aluno (a): ***Marcelo Módolo***

Título: " CLASSIFICAÇÃO AUTOMÁTICA DE SUPERNOVAS USANDO REDES NEURAIS ARTIFICIAIS".

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de ***Doutor(a)*** em

Computação Aplicada

Dr. José Carlos Becceneri

Presidente / INPE / SJCampos - SP

Dr. Lamartine Nogueira Frutuoso Guimarães

Orientador(a) / IEAv/CTA / SJCampos - SP

Dr. Reinaldo Roberto Rosa

Orientador(a) / INPE / SJCampos - SP

Dr. Stephan Stephany

Membro da Banca / INPE / SJCampos - SP

Dra. Paula Rodrigues Teixeira Coelho

Convidado(a) / IAG/USP / São Paulo - SP

Dr. Elcio Hideiti Shiguemori

Convidado(a) / IEAv/CTA / São José dos Campos - SP

Este trabalho foi aprovado por:

( ) maioria simples

(X) unanimidade

São José dos Campos, 04 de Maio de 2016

“Que a minha coragem não seja petulância e que a minha humildade não seja abjeção”.

Emmanuel

Dedico a meus pais Sidney e Elza e a meus filhos Marina, Mateus e Marcela.

AGRADECIMENTOS

Agradeço, em primeiro lugar, a Deus por me permitir realizar esse sonho e me dar capacidade e força para ir até o fim.

A meus filhos Marina, Mateus e Marcela pela compreensão que sempre tiveram quando não pude lhes dar a atenção e o tempo que mereciam. A meus pais Sidney e Elza pelo apoio incondicional e pela sólida educação que me permitiu chegar até aqui.

Agradeço a meus familiares e amigos que vibraram muito para o sucesso deste projeto. Em particular, a meus irmãos Maristela, Marisol, Marcio, Marilia e Marcilio que sempre me deram força para continuar, e a meu “irmão” Mario Zanotelli que acreditou em mim mais do que eu mesmo. À amiga Magda por ter me incentivado a iniciar o doutorado e ter me apresentado ao professor Lamartine.

Um agradecimento especial ao professor Lamartine pela sua orientação sempre segura, por sua paciência em ouvir minhas dificuldades e por não ter me deixado desistir nos momentos de fraqueza. Ao professor Reinaldo por apontar o caminho e viabilizar o desenvolvimento do projeto.

Ao professor Carlos Santi pelo apoio e pela flexibilização do meu horário de trabalho para dedicar o tempo necessário a este doutorado. Agradeço, em especial, à Marlene, à Sandra e à Cida que fizeram muito mais que seu trabalho para me ajudar a cumprir minhas obrigações na coordenação do curso. A todos os colegas de trabalho da Metodista pela compreensão nas ausências e atrasos em diversas atividades.

Agradeço aos professores da Pós-Graduação em Computação Aplicada com quem muito aprendi nas disciplinas e seminários. Aos colegas de Pós-Graduação pela troca de informações e experiências que ajudaram muito, principalmente, no início difícil com as disciplinas.

Ao IEAv-DCTA, Instituto de Estudos Avançados do Departamento de Ciência e Tecnologia Aeroespacial, por me permitir usar suas instalações e equipamentos no desenvolvimento deste trabalho. Ao INPE, Instituto Nacional de Pesquisas Espaciais, pela oportunidade de aprimorar minha qualificação profissional. À Capes, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, pelo apoio financeiro.

RESUMO

A classificação de supernovas pode ser feita por especialistas humanos a partir da análise visual do seu espectro, mas não é trivial. Apenas alguns astrônomos especialistas são capazes de fazê-lo e com a subjetividade inerente à percepção humana. Os classificadores automáticos existentes não fazem a modelagem usando a forma humana de analisar o espectro para classificar supernovas. Eles somente comparam a similaridade do espectro da supernova recém-descoberta com os espectros de supernovas que já foram classificadas. Este trabalho propõe um método de classificação automática de supernovas baseado em Inteligência Computacional que simula a maneira humana de análise do espectro, mas fazendo uma classificação mais formal e menos propensa a subjetividade da análise humana. O paradigma básico é a forma como os seres humanos fazem a análise, mas o classificador automático utiliza redes neurais artificiais para analisar o espectro e identificar a presença ou ausência de elementos que determinam o tipo supernova. Quatro Redes Neurais Perceptron de Múltiplas Camadas foram construídas. Uma rede neural para identificar cada tipo "clássico" de supernova: Ia, Ib, Ic e II. O classificador foi testado em uma base com 649 espectros de 221 supernovas e os resultados foram muito bons, alcançando 99,2% de acerto na identificação de supernovas do tipo Ia. Isso indica que a classificação realizada por este método pode ser utilizada em situações onde não existe um especialista ou onde seja necessária uma análise automática, sistemática e contínua. A ferramenta desenvolvida neste trabalho foi denominada CIntIa, sigla para Classificador Inteligente de supernovas do tipo Ia.

Palavras-chave: Classificação automática de supernovas. Análise do espectro de supernovas. Classificação de supernovas a partir da análise do espectro. Tipos de supernovas. Redes Neurais Artificiais. Inteligência Computacional.

SUPERNOVAE AUTOMATIC CLASSIFICATION USING ARTIFICIAL NEURAL NETWORKS

ABSTRACT

The supernova classification performed by human experts from visual analysis of its spectrum is not trivial. Only few expert astronomers are able to do so, and only fewer than those attempt to remove the subjectivity inherent to human perception from that analysis. The existing automatic classifiers did not model the human way of analyzing the spectrum to classify supernovas. They only compare the spectrum similarity of newfound supernova with spectra of supernovae already classified. The supernovae classification automatic method proposed here is based on Computational Intelligence, and simulates the human spectrum analysis, making it a more formal classification and less prone to subjectivity of the human itself. The basic paradigm is the way humans perform the analysis. The automatic classifier uses artificial neural networks to analyze the spectrum and identify the presence or absence of elements that determine the supernova type. Four Multilayer Perceptron Neural Network were built. One neural network to identify each "classic" type of supernova: Ia, Ib, Ic and II. The classifier was tested on a database with 649 spectra of 221 different supernovae. The results are very good, reaching 99.2% accuracy in identifying the type Ia supernovae. They indicate that the classification performed by this method can be used in situations, or that have no specialist around, or that require an automatic, systematic and continuous analysis. The tool developed in this work is named CIntIa, from the Portuguese language “Classificador Inteligente de supernovas do tipo Ia”, or Type Ia Supernovas Intelligent Classifier.

Keywords: Supernovae automatic classification. Supernovae spectrum analysis. Supernovae classification from spectrum analysis. Supernovae types. Artificial Neural Network. Computational Intelligence.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE SIGLAS E ABREVIATURAS

ART	*Adaptive Resonance Theory*
BATM	*Bayesian Adaptive Template Matching*
CCAA	*Chinese Center for Antarctic Astronomy*
CfA	*Center for Astrophysics*
CIntIa	Classificador Inteligente de supernovas do tipo Ia
EUA	Estados Unidos da América
FP	*Feature-Richness*
GELATO	*GEneric cLAssification TOol*
GoF	*Goodness-of-Fit*
GOODS	*Great Observatories Origins Deep Survey*
INPE	Instituto Nacional de Pesquisas Espaciais
KDUST	*Kunlun Dark Universe Survey Telescope*
MCP	McCulloch e Pitts
MLP	*Multilayer Perceptron*
M_{Sol}	Massa do SOL
PASSPARTOO	*PAdova Supernova Spectra comPARison TOOl*
RN	Rede Neural
RNA	Redes Neurais Artificiais
SN	Supernova
SN-ABC	*SN Automatic Bayesian Classifier*
SNID	*Supernova Identification*
SNLS	*Supernova Legacy Survey*
SOFT	*Supernova Ontology with Fuzzy Templates*
SUSPECT	*The Online Supernova Spectrum Archive*

SUMÁRIO

Pág.

1 INTRODUÇÃO

Uma supernova tem sua origem quando uma intensa explosão determina o fim da vida de uma estrela. A emissão de luz de uma supernova inicia com uma grande luminosidade que aumenta ainda mais nas duas primeiras semanas, até atingir sua luz máxima. Após, então diminui gradativamente durante aproximadamente um ano (GIUNTI; KIM, 2007). Entre outras possibilidades seu estudo ajuda a entender a expansão do universo.

A explosão que dá origem à supernova pode ser de dois tipos: colapso de núcleo ou termonuclear. A explosão por colapso de núcleo faz parte do ciclo de vida normal de uma estrela que tem massa superior a dez vezes a massa do Sol. A explosão termonuclear ocorre somente se uma estrela com massa dez vezes menor que a massa do Sol está em um sistema binário ou múltiplo. Nesse caso, depois de se tornar uma estrela Anã Branca, ela absolve massa da outra estrela do sistema até sofrer a explosão termonuclear (TORRES, 2010).

Historicamente, a classificação de uma supernova é realizada, considerando características do espectro e propriedades da curva de luz (WHEELER; BENETTI, 2002). Essas características e propriedades dependem da composição do envoltório da estrela que deu origem a supernova, chamada progenitora (TURATTO; BENETTI; PASTORELLO, 2007). Assim, a classificação é feita pela análise da presença ou ausência no espectro dos elementos que compõem a estrela progenitora e pela análise do formato da curva de luz capturada ao longo do tempo (TURATTO, 2003).

Os esquemas de classificação são desenvolvidos desde 1941, quando foram reconhecidos dois tipos de supernovas: tipo I, caracterizado pela ausência de hidrogênio em sua composição; e tipo II, caracterizado pela presença de hidrogênio em sua composição. A partir dessa classificação inicial foram derivados outros tipos de supernovas. Os esquemas de classificação atuais analisam também a presença de hélio e silício no espectro e o formato da

curva de luz para classificar a supernova. Dois desses esquemas, apresentados em Giunti e Kim (2007) e em Turatto (2003), têm pelo menos sete diferentes tipos de supernovas: Ia, Ib, Ic, IIb, IIL, IIP e IIn.

No entanto, apesar da existência de esquemas de classificação, a identificação do tipo da supernova não é um serviço trivial e somente astrônomos especialistas são capazes de fazê-lo. Mas mesmos as classificações feitas por especialistas são consideradas insatisfatórias e baseadas em convicções subjetivas (HARUTYUNYAN, 2008) o que torna a identificação automática do tipo da supernova uma necessidade e, portanto, foco de pesquisas atuais. Mas existem poucos classificadores automáticos desenvolvidos.

Na pesquisa feita para este trabalho foram encontrados seis trabalhos que construiram classificadores automáticos: dois que usam a classificação a partir da análise do espectro e quatro que classificam a partir da análise da curva de luz. As duas pesquisas que resultaram em classificadores automáticos de supernovas a partir da análise do seu espectro foram desenvolvidas nos Estados Unidos da América (EUA) (BLONDIN; TONRY, 2007) e na Itália (HARUTYUNYAN, 2008).

O algoritmo desenvolvido nos EUA foi um trabalho conjunto do *Harvard-Smithsonian Center for Astrophysics* e do *Institute for Astronomy, University of Hawaii* (BLONDIN; TONRY, 2007) implementado em uma ferramenta chamada *Supernova Identification* (SNID) que, além do tipo da supernova, também identifica o *redshift* e a idade da supernova. Ele utiliza técnicas de correlação comumente usadas para a comparação de espectros de galáxias (TONRY; DAVIS, 1979).

Na Itália no *Dipartimento di Astronomia da Università Degli Studi di Padova* algoritmos de classificação de supernovas foram desenvolvidos e implementados na ferramenta *PAdova Supernova Spectra comPARison TOOl* (PASSPARTOO) (HARUTYUNYAN, 2008). O principal algoritmo foi implementado foi o *GEneric cLAssification TOol* (GELATO). A ferramenta

também utiliza medida de similaridade entre espectros para classificar a supernova.

Esses dois trabalhos têm em comum a maneira de definir o tipo da supernova. O classificador faz uma comparação entre o espectro da supernova recém-descoberta e os espectros de supernovas que foram previamente classificados e atribui à supernova recém-descoberta o tipo, cujo espectro tem maior similaridade, ou seja, apenas um espectro determina o tipo dessa supernova. Em ambos os trabalhos não são utilizadas técnicas de Inteligência Computacional, mas técnicas matemáticas convencionais que encontram a similaridade entre os espectros.

Os outros quatro trabalhos encontrados, que fazem a análise da curva de luz, utilizam técnicas de Inteligência Computacional para classificação de supernovas: (POZNANSKI; MAOZ; GAL-YAM, 2007) (RODNEY; TONRY, 2009) (PASCALE, 2011) (KARPENKA; FEROZ; HOBSON, 2013). É importante ressaltar que a classificação pela curva de luz não se enquadra no escopo deste trabalho porque esse classificador foi desenvolvido para utilização em um telescópio do Projeto *Kunlun Dark Universe Survey Telescope* (KDUST) (CHINESE CENTER FOR ANTARCTIC ASTRONOMY, 2010), que precisa conhecer o tipo da supernova logo após seu surgimento para decidir quais informações deverão ser coletadas nas próximas vezes que o telescópio apontar para o mesmo local.

O Projeto KDUST envolve a construção de um observatório astronômico em Kunlun, um dos pontos mais altos do Platô Antártico. O classificador será instalado em um telescópio desse observatório que terá também um controlador automático para manter o telescópio apontado para a supernova enquanto for necessário e, periodicamente, voltar a apontar para a mesma supernova com objetivo de colher informações. O Projeto KDUST tem interesse particular nas supernovas do tipo Ia, pois esse é o único tipo que possibilita

medir as distâncias de luminosidade que ajudam a estudar a expansão acelerada do universo.

A motivação para participação no Projeto KDUST surgiu quando o Instituto Nacional de Pesquisas Espaciais (INPE), tendo a possibilidade de participar do KDUST, propôs alguns projetos de desenvolvimento de ferramentas e análises de dados. Entre eles, um sistema de classificação automática de supernovas, em especial, para identificação de supernovas do tipo Ia.

Nesse contexto, o método de classificação automática de supernovas desenvolvido neste trabalho precisa identificar o tipo da supernova logo depois do seu surgimento, o que somente pode ser feito pela análise do espectro, pois essa classificação deve ser feita em espectros capturados em intervalo de dias próximo da luz máxima, duas semanas antes ou depois. Isso significa que a análise do espectro deve ser feita em espectros capturados até um mês após o surgimento da supernova, pois a partir daí as linhas de alguns elementos ficam difíceis de identificar. No caso específico do hélio, segundo Modjaz; Blondin; *et al.* (2014), suas linhas somente estão visíveis no período entre dez dias antes a quinze dias depois da luminosidade máxima. Por outro lado, a classificação de supernovas pela curva de luz fica inviável nesse contexto porque deve ser feita entre 60 e 90 dias depois do seu surgimento, depois de capturar periodicamente a luminosidade durante esse período para formar a curva de luz necessária a classificação.

Assim, a proposta do trabalho foi criar um método de classificação automática de supernovas, que analisa espectros capturados próximos da luz máxima, mas além disso, esse método foi construído para ser utilizado em lugares onde não existe um astrônomo especialista ou onde seja necessária a classificação automática. Dessa forma, para simular a análise do especialista humano, feita visualmente no espectro em busca da presença ou ausência de determinados elementos, foram utilizados paradigmas de Inteligência Computacional, mais especificamente, Redes Neurais Artificiais.

No início deste trabalho foram analisadas também a possibilidade de utilização dos paradigmas Sistemas Especialistas e Sistemas Nebulosos, mas a dificuldade em obter o conhecimento do especialista para construção da base de conhecimento inviabilizou a utilização dessas técnicas. Por outro lado, a disponibilidade de bancos de espectros rotulados com o tipo da supernova por astrônomos especialistas indicou a viabilidade da utilização de Redes Neurais Artificiais e, em particular, de Redes Neurais Perceptron de Múltiplas Camadas, que são adequadas para resolução de problemas de classificação quando se tem um conjunto rotulado de dados para o treinamento.

A base para a criação deste método de classificação foi simular a análise do especialista humano, mas este trabalho procurou também sistematizar essa análise visando diminuir a subjetividade existente na classificação humana, tornando a análise homogênea. A sistematização é relevante porque pode ajudar a melhoria dos esquemas de classificação, uma vez que, desde o primeiro esquema, subtipos têm sido introduzidos, excluídos e subdivididos (TURATTO, 2003), mas as taxonomias existentes permanecem ambíguas e não exaustivas (HARUTYUNYAN, 2008).

Essa simulação da análise do espectro feita pelo especialista humano é uma das diferenças fundamentais do método desenvolvido neste trabalho com os outros dois métodos pesquisados que analisam o espectro da supernova. Outra diferença é a utilização de redes neurais para identificar o tipo da supernova, enquanto que os outros dois métodos pesquisados apenas medem a similaridade dos espectros. As redes neurais permitem a identificação automática de características comuns do conjunto de espectros usadas na classificação de cada tipo de supernova.

Essa diferença é relevante porque usar características do conjunto de espectros, e não apenas de um deles, minimiza o impacto existente no caso de mudança de classificação de uma determinada supernova. Mudanças que já ocorreram, como no caso de alteração dos tipos de 26 supernovas

apresentados em Modjaz; Blondin; *et al.* (2014). Como os outros dois métodos, que analisam o espectro, consideram apenas o espectro mais similar para determinar o tipo da supernova, podem sofrer um grande impacto no caso da mudança de classificação de uma supernova porque todas as supernovas classificadas a partir daquela mais similar, cujo tipo foi alterado, também precisam ter sua classificação alterada.

O classificador desenvolvido nesse trabalho usa o esquema de classificação de Giunti e Kim (2007), mas os tipos de supernovas identificados se restringem aos tipos chamados “clássicos”: Ia, Ib, Ic e II. Essa restrição se deve a dois aspectos: a análise somente do espectro e a quantidade de espectros disponíveis nos bancos de espectros encontrados. Como o classificador desenvolvido analisa somente o espectro, ele já exclui da sua classificação os subtipos escritos com letras maiúsculas (IIF, IIL e IIP) que são identificados pela análise da curva de luz. Os tipos IIb, IIn e IIpec, apesar de também serem determinados pela análise do espectro, não são identificados individualmente e foram agrupados todos no tipo II, porque a separação nos diversos subtipos poderia não produzir resultados significativos. Isto porque os bancos de espectros encontrados, que foram utilizados para treinamento e teste das redes neurais, contêm poucos espectros do tipo II, inviabilizando a discretização desse conjunto de espectros em subtipos. Considerando os subtipos IIn e IIpec, em particular, em um dos bancos usados neste trabalho não foi encontrado espectro algum com os parâmetros necessários para análise. No outro banco, espectros dos tipos IIb e IIn constituem apenas 1% do total, enquanto espectros do tipo IIpec são 3% do total.

No entanto, apesar do classificador identificar os quatro tipos “clássicos”, seu foco principal é separar as supernovas do tipo Ia dos outros tipos de supernovas, devido ao contexto do seu desenvolvimento estar ligado ao Projeto KDUST. Assim, como esse classificador foi construído para instalação nos telescópios do Projeto KDUST com o objetivo principal de identificar as

supernovas do tipo Ia, ele será chamado a partir daqui de Classificador Inteligente de supernovas do tipo Ia (CIntIa).

A descrição detalhada do CIntIa e dos resultados dos testes, assim como a pesquisa bibliográfica realizada, estão nos outros capítulos deste trabalho. No próximo capítulo estão detalhados os esquemas de classificação de supernovas. O Capítulo 3 explica o funcionamento dos classificadores automáticos pesquisados e o Capítulo 4 mostra um estudo das Redes Neurais Artificiais dirigido para aplicação neste trabalho. O Capítulo 5 descreve o desenvolvimento das quatro versões do CIntIa e o Capítulo 6 mostra os resultados obtidos pelo CIntIa e compara com o desempenho dos classificadores pesquisados. No Capítulo 7 estão as conclusões do trabalho.

2 CLASSIFICAÇÃO DE SUPERNOVAS

O ciclo de vida de uma estrela pode se encerrar de maneiras diferentes. Algumas estrelas se tornam supernovas e outras não, de acordo com sua evolução. Tornar-se supernova representa um fim catastrófico da evolução de uma estrela (WHEELER; BENETTI, 2002). A evolução da estrela depende basicamente da sua massa e de sua participação em um sistema binário ou múltiplo. Caso faça parte de um sistema binário, pode, em algum momento, trocar massa com outra estrela e alterar o comportamento do sistema. No entanto, se for uma estrela isolada, sua evolução depende apenas da sua massa inicial. Segue um resumo do ciclo de vida de uma estrela isolada de acordo com sua massa inicial em relação a massa do Sol (M_{Sol}) (TORRES, 2010):

- Uma estrela com massa inicial menor que 0,8 M_{Sol} se torna gigante vermelha e depois anã branca;

- Uma estrela com massa inicial entre 0,8 M_{Sol} e 10 M_{Sol} se torna gigante vermelha, supergigante, depois nebulosa planetária, quando sua camada exterior de gás é ejetada e ela perde massa para se tornar uma anã branca com massa da ordem de 0,6 M_{Sol};

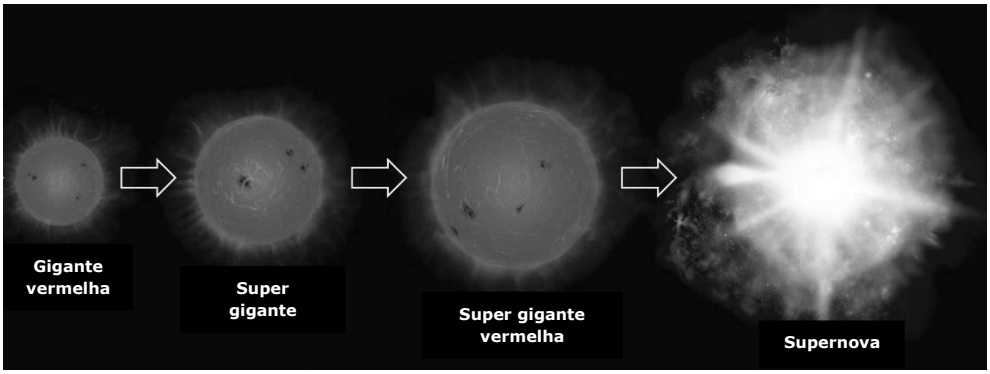
- Uma estrela com massa inicial entre 10 M_{Sol} e 25 M_{Sol} se torna gigante vermelha, supergigante, supernova, quando perde sua massa ejetando os gases exteriores, e terminará como estrela de nêutrons com massas da ordem de 1,4 M_{Sol};

- Uma estrela com massa inicial entre 25 M_{Sol} e 100 M_{Sol} também se torna gigante vermelha, supergigante e supernova, mas termina como um buraco negro com massa da ordem de 6 M_{Sol};

- Uma estrela com massa inicial maior que 100 M_{Sol} perderá sua massa ainda na fase principal, onde consome o hidrogênio do núcleo, e depois evoluirá como uma estrela com massa menor que 100 $M_{Sol.}$

Assim, apenas estrelas com massa superior a 10 M_{Sol} podem dar origem a supernovas se estiverem isoladas. Figura 2.1 mostra a criação de uma supernova a partir da explosão de uma super gigante vermelha.

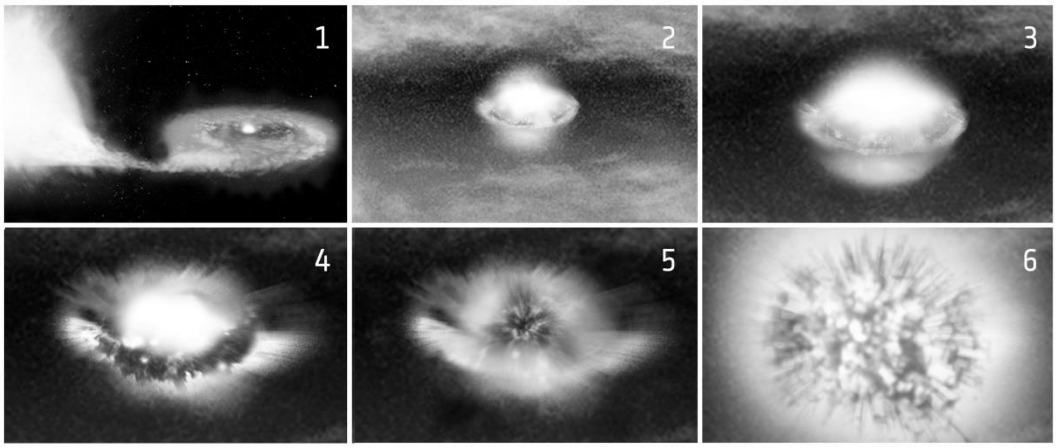
Figura 2.1 - Sequência que origina uma supernova a partir da explosão de uma estrela super gigante vermelha.



Fonte: Adaptado de National Science Foundation (2013)

No entanto, as supernovas também podem se originar a partir da explosão de uma estrela que tem massa inferior a 10 M_{Sol}, desde que ela faça parte de um sistema binário ou múltiplo que tenha uma estrela gigante vermelha. Nesse caso, depois de se tornar uma estrela anã branca, ela absorve massa da gigante vermelha e sofre uma explosão termonuclear que origina uma supernova. Vale ressaltar que sistemas binários ou múltiplos não são exceção, pelo contrário, se estima que mais da metade das estrelas pertencem a sistemas binários ou múltiplos (TORRES, 2010). A Figura 2.2 mostra a sequência de uma supernova originada de uma anã branca.

Figura 2.2 - Sequência que origina uma supernova a partir da explosão de uma estrela anã branca em um sistema binário com uma estrela gigante vermelha.



Fonte: Choi (2014)

A supernova que teve sua origem em um sistema binário a partir da explosão termonuclear de uma anã branca é uma supernova do tipo Ia. Todos outros tipos de supernovas têm origem a partir da explosão de uma única estrela super gigante vermelha por colapso de núcleo.

Além da diferença do tipo de estrela que deu origem a supernova, diferentes tipos de supernovas podem existir de acordo com o tamanho do envoltório de hidrogênio que pode ser muito diferente de estrela para estrela, mesmo quando elas têm massa inicial muito próximas, o que faz com que os efeitos da explosão da estrela tenham uma variação muito grande. Esses tipos podem ser determinados a partir da análise do espectro luminoso e da curva de luz formada com o decorrer do tempo (WHEELER; BENETTI, 2002).

A classificação vem sendo desenvolvida desde 1941 quando foi reconhecida a existência de dois tipos de supernovas: tipo I que se caracteriza pela ausência de hidrogênio em sua composição; e tipo II com presença de hidrogênio (TURATTO, 2003). A partir dessa classificação inicial foram derivados subtipos de supernovas e, atualmente, ainda são sugeridos novos subtipos a partir de novas descobertas, como o tipo Iax (FOLEY; CHALLIS, *et al.*, 2013). Mas nem

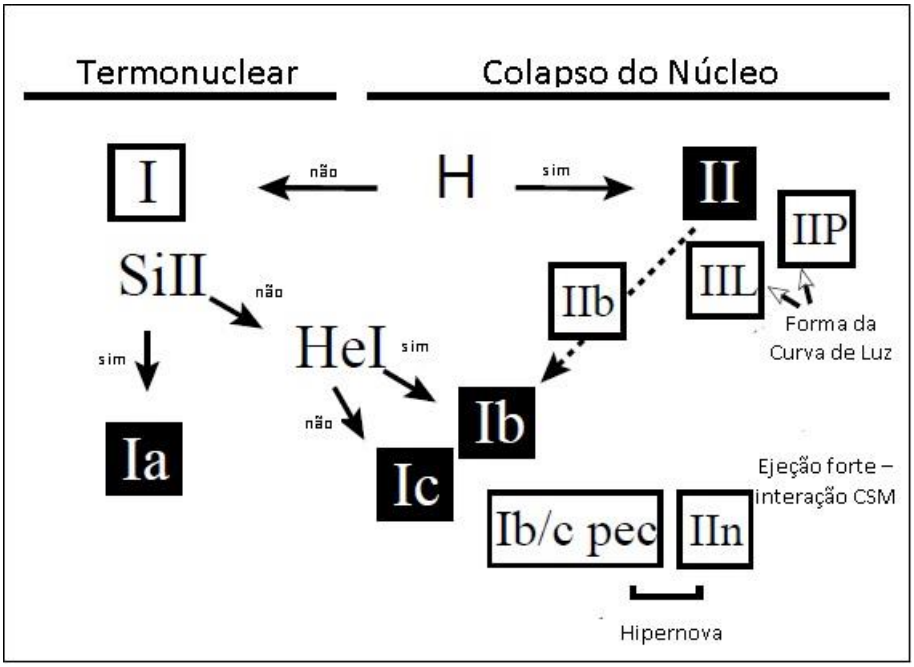
todos são incorporados imediatamente aos esquemas de classificação. Os principais esquemas de classificação atuais consideram pelo menos sete diferentes tipos de supernovas: Ia, Ib, Ic, IIb, IIL, IIP e IIn (GIUNTI; KIM, 2007) (TURATTO, 2003).

As seções 2.1 e 2.2 mostram, respectivamente, os esquemas de classificação apresentados por Turatto (2003) e Giunti e Kin (2007). A seção 2.3 contém algumas considerações sobre esses dois esquemas.

2.1. Esquema de classificação de Turatto

No esquema de classificação proposto por Turatto (2003), assim como nos outros esquemas, as supernovas do tipo Ia estão associadas com a explosão termonuclear de anãs brancas e os outros tipos de supernovas estão associados com o colapso do núcleo de estrelas massivas. A Figura 2.3 mostra o esquema de classificação de supernovas proposto em Turatto (2003).

Figura 2.3 - Esquema de Classificação de Supernovas de Turatto



Fonte: Adaptado de Turatto (2003)

Segundo esse esquema a classificação também leva em consideração a presença ou ausência de hidrogênio, de hélio e de silício. A presença ou não desses elementos permite determinar os tipos “clássicos” de supernovas: Ia, Ib, Ic e II.

- Tipo Ia: ausência de hidrogênio e presença de silício
- Tipo Ib: ausência de hidrogênio e silício e presença de hélio
- Tipo Ic: ausência de hidrogênio, silício e hélio
- Tipo II: presença de hidrogênio

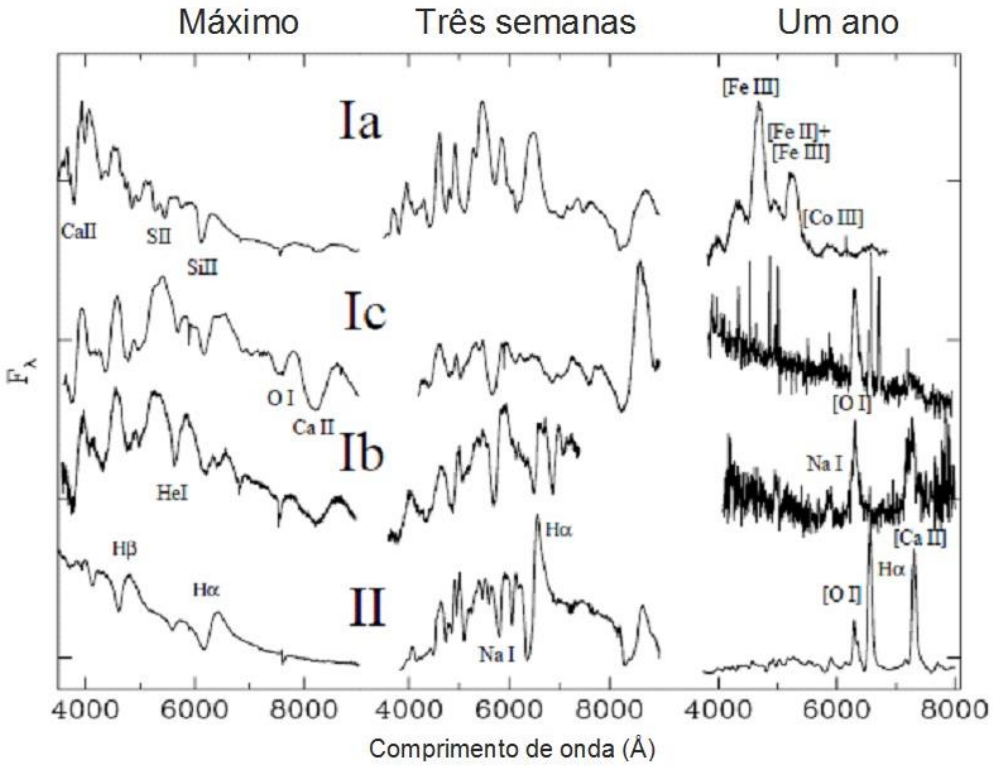
Além desses tipos “clássicos” o esquema de Turatto tem também os subtipos IIL e IIP que não dependem exclusivamente da presença ou ausência de elementos para sua determinação, mas também da análise da curva de luz.

Assim, os subtipos determinados por propriedades espectrais são escritos em letras minúsculas e os subtipos determinados pelas propriedades da curva de luz são escritos em letra maiúscula. Essa diferenciação na escrita, usada em todos os esquemas de classificação, mostra que a identificação dos subtipos escritos em letras maiúsculas somente pode ser feita pela análise da curva de luz depois de decorridos alguns meses do surgimento da supernova e que a identificação dos subtipos escritos em letras minúsculas poder ser feita pela análise do seu espectro, logo após seu surgimento, próximo da luminosidade máxima, quando as linhas dos elementos H, SI e He estão presentes no espectro. Com o decorrer do tempo fica cada vez mais difícil identificar essas linhas, conforme exemplificado na Figura 2.4.

A Figura 2.4 mostra exemplos de espectros de supernovas que foram capturados em três momentos diferentes: na luminosidade máxima, três semanas depois e um ano depois da luminosidade máxima. Os espectros capturados na luminosidade máxima mostram a presença ou ausência dos elementos H, He e SI nos diferentes tipos de supernova, enquanto que nos

outros espectros, capturados três semanas e um ano após a luminosidade máxima, não é possível verificar a presença de todos esses elementos, tornando inviável a utilização dos espectros para identificação do tipo de supernova.

Figura 2.4 - Exemplos de espectros de supernovas na luminosidade máxima, três semanas depois e um ano depois.



Fonte: Adaptado de Turatto (2003)

Segundo Turratto (2003), os diferentes tipos de supernova têm também outras características que são resumidas a seguir.

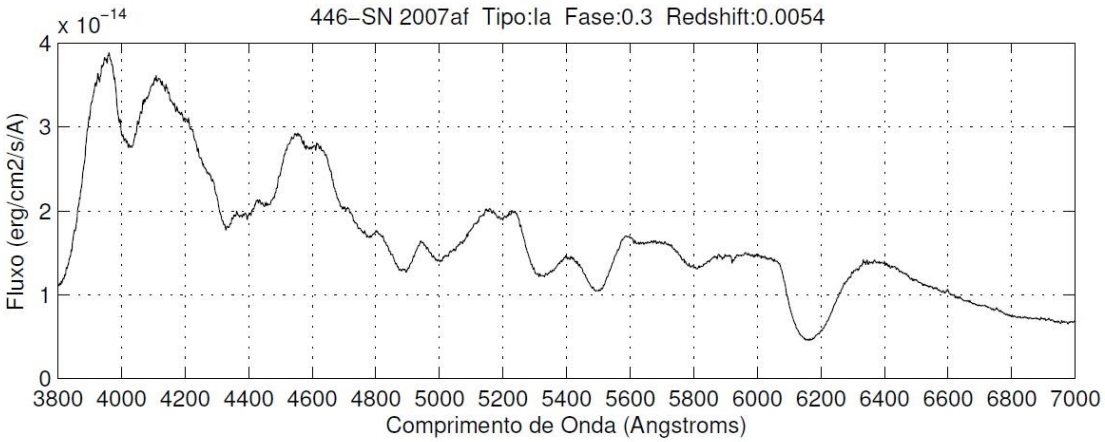
2.1.1. Tipo Ia

Esse tipo de supernova tem uma importância muito grande devido a seu papel em determinar a geometria do universo, pois tem luminosidade alta e dispersão de luminosidade relativamente pequena, a partir do brilho máximo.

Supernovas do tipo Ia foram descobertas em todos os tipos de galáxias e não são associados com os braços de espirais tão fortemente quanto os outros tipos de supernovas. Seus espectros são caracterizados por linhas de elementos, tais como, cálcio, oxigênio, silício e enxofre na luminosidade máxima e pela ausência de hidrogênio durante o tempo todo. Com o passar do tempo, cresce a contribuição da linha de ferro e passados vários meses o espetro é dominado pelas linhas de ferro. O comportamento global homogêneo tanto do espectro quanto da curva de luz levou a um consenso de que são associados à explosão termonuclear de uma anã branca.

A curva de luz da supernova tipo Ia se caracteriza por um pico secundário depois de 20 a 30 dias do pico máximo. A análise do conjunto homogêneo de dados óticos levou à descoberta de uma correlação entre o pico da luminosidade e a forma da curva de luz, que mostra uma taxa de declínio mais lenta em objetos mais brilhantes. Essa correlação torna a supernova tipo Ia um indicador útil para distâncias cosmológicas e muito importante para o estudo da expansão do universo. A Figura 2.5 mostra um espectro da SN 2007af do tipo Ia, capturado no dia da luz máxima.

Figura 2.5 - Espectro da SN 2007af do tipo Ia



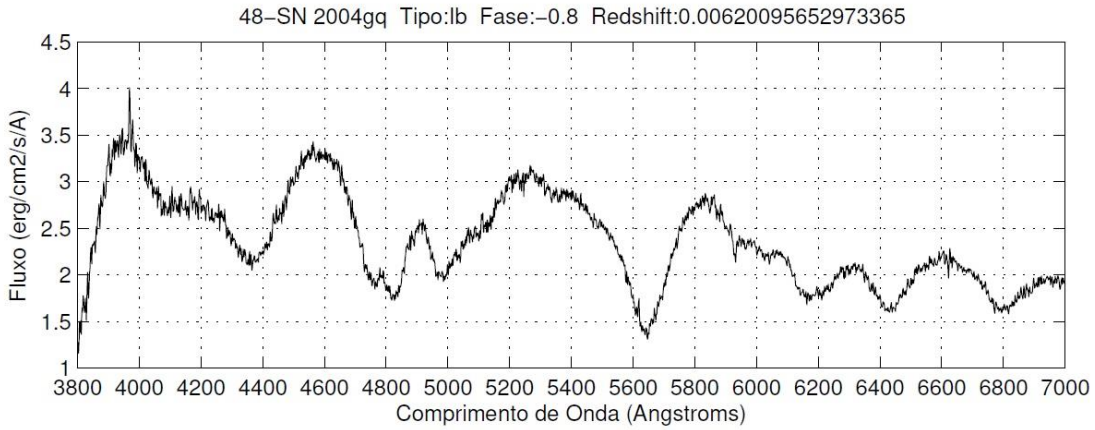
Fonte: Adaptado de Harvard-Smithsonian Center for Astrophysics (2010)

2.1.2. Tipo Ib e Ic

As supernovas do tipo Ib e Ic aparecem somente em galáxias do tipo espiral e estão associadas a estrelas massivas (estrelas com mais de dez vezes a massa do Sol), possivelmente mais massivas que as estrelas progenitoras das supernovas do tipo II. Elas apresentam emissão de rádio relativamente forte, com índices espectrais íngremes.

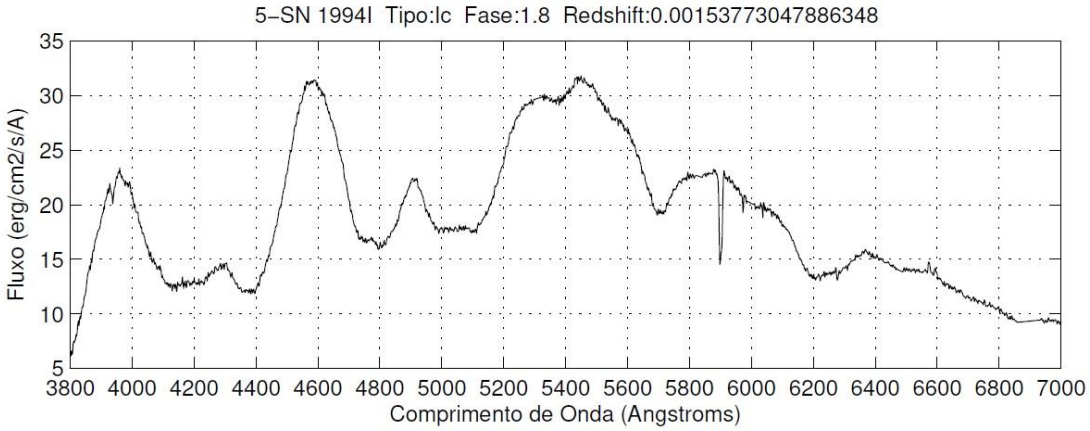
As características das supernovas do tipo Ib são a ausência de linhas de hidrogênio e silício e a presença de hélio. As supernovas que não tem presença de hélio são do tipo Ic. Além dessa diferença entre os tipos Ib e Ic, as linhas de oxigênio são relativamente mais fortes no tipo Ic que no tipo Ib e as linhas de emissão das nebulosas são mais largas. Em geral, o tipo Ib se mostra mais homogêneo que o tipo Ic. A Figura 2.6 mostra um espectro da SN 2004gq do tipo Ib, capturado no dia anterior à luz máxima, e a Figura 2.7 apresenta o espectro da SN 1994I do tipo Ic, capturado um dia depois da luz máxima.

Figura 2.6 - Espectro da SN 2004gq do tipo Ib



Fonte: Adaptado de Harvard-Smithsonian Center for Astrophysics (2010)

Figura 2.7 - Espectro da SN 1994I do tipo Ic



Fonte: Adaptado de Harvard-Smithsonian Center for Astrophysics (2010)

2.1.3. Tipo II

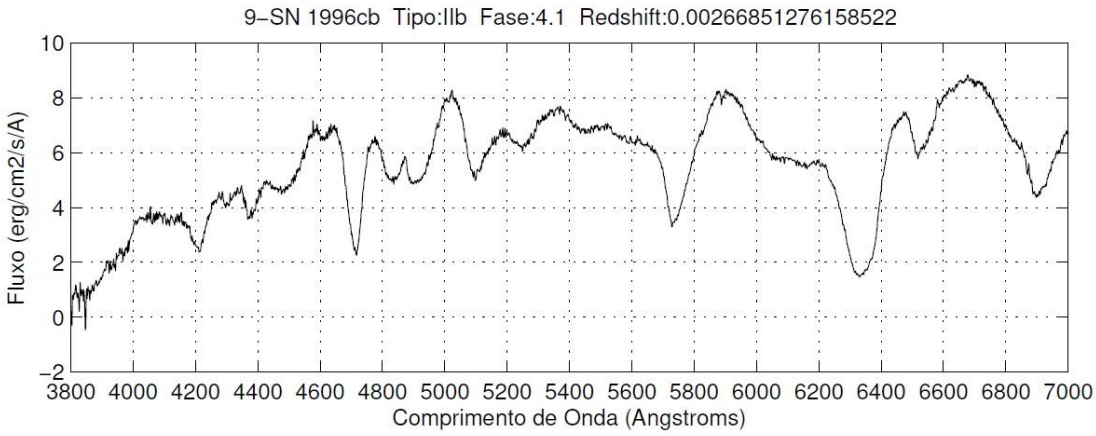
As supernovas do tipo II são caracterizadas pela presença de hidrogênio no seu espectro. Elas não aparecem em galáxias velhas, são fortemente associadas com regiões de recente formação de estrelas e surgem a partir do colapso do núcleo de estrelas massivas.

Esse tipo de supernova mostra uma grande variedade de propriedades em suas curvas de luz e em seus espetros. Existem quatro subtipos de supernova do tipo II: IIb, IIP, IIL e IIn. Existem, ainda, objetos peculiares que não se enquadram em qualquer desses tipos.

2.1.3.1. Tipo IIb

As supernovas do tipo IIb no início tem espectros semelhantes as supernovas do tipo II, mas com o passar do tempo seu espectro fica semelhante as supernovas dos tipos Ib e Ic. Essa transformação do tipo II para o tipo Ib ou Ic constitui a ligação perdida entre as supernovas que mantém o envoltório (tipo II) e as supernovas que se despojam do envoltório (tipo I). A Figura 2.8 mostra um espectro da SN 1996cd do tipo IIn, capturado quatro dias depois de luz máxima.

Figura 2.8 - Espectro da SN 1996cb do tipo IIb



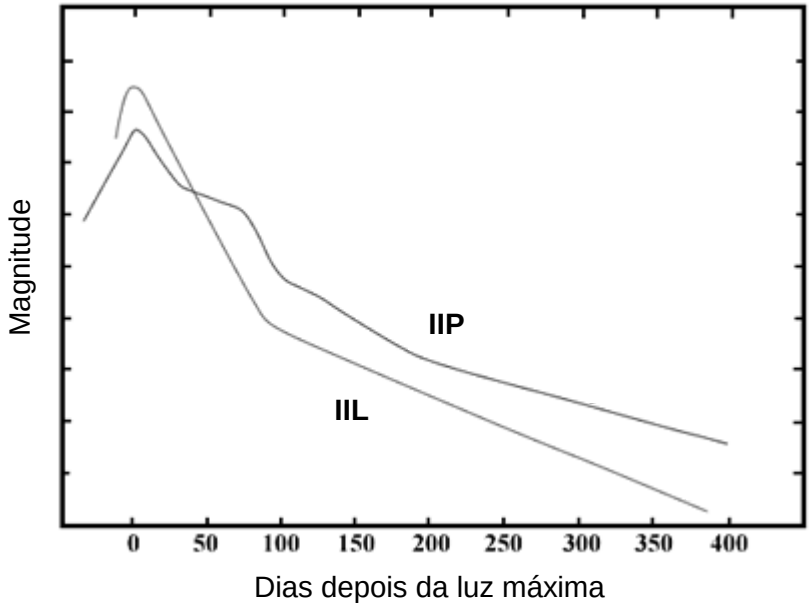
Fonte: Adaptado de Harvard-Smithsonian Center for Astrophysics (2010)

2.1.3.2. Tipo IIP e IIL

No tipo IIP a luminosidade deixa de declinar rapidamente depois do máximo, formando um platô com dois a três meses de duração. No tipo IIL a luminosidade é linear e tem um declínio ininterrupto, provavelmente devido a um menor envoltório de massa. No entanto, as supernovas não são claramente separáveis nesses dois tipos, pois existem vários casos intermediários com pequenos platôs, que não permitem identificar uma grande diferença espectral entre os dois tipos. A Figura 2.9 mostra exemplos de curvas de luz de supernovas dos tipos IIL e IIP.

As progenitoras do tipo IIL têm envoltórios menores que aqueles do tipo IIP, provavelmente devido à perda de massa durante sua evolução. Um cenário geral tem sido proposto no qual a evolução do envoltório comum em sistemas binários maciços com diferentes raios de massa e as separações dos componentes pode levar a vários graus de diminuição da massa do envoltório. De acordo com esse cenário a sequência de tipos IIP – IIL – IIb – Ib – Ic no esquema de Turatto está ordenada de acordo com a diminuição de massa do envoltório.

Figura 2.9 - Exemplos de curvas de luz de supernovas dos tipos IIL e IIP.



Fonte: Adaptado de Swinburne University of Technology (2015)

2.1.3.3. Tipo IIn

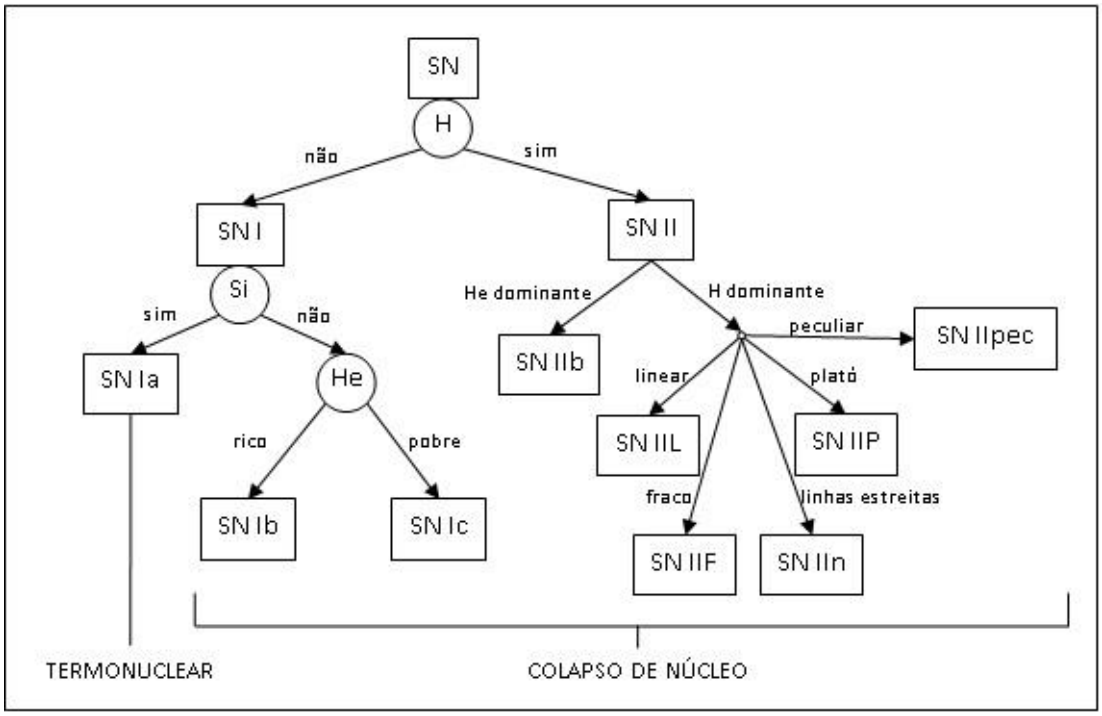
A letra “n” da denominação IIn vem da palavra “*narrow*” (estreita), devido as linhas de emissão estreitas. Os espectros desse tipo de supernova tem evolução lenta.

2.2. Esquema de classificação de Giunti e Kin

O esquema de classificação de supernovas de Giunti e Kin (2007), mostrado na Figura 2.10, contempla a maioria dos tipos existentes no esquema de Turatto (2003) e acrescenta dois outros tipos: IIF e IIpec.

Esse esquema de classificação também considera que as supernovas do tipo Ia estão associadas com a explosão termonuclear de estrelas anãs brancas e os outros tipos de supernovas estão associados com o colapso do núcleo de estrelas massivas.

Figura 2.10 - Esquema de Classificação de Supernovas de Giunti e Kin



Fonte: Adaptado de Giunti e Kim (2007)

O esquema também propõe a classificação a partir da análise do espectro levando em consideração a presença ou ausência de hidrogênio, de hélio e de silício para determinar os tipos “clássicos” de supernovas.

Ainda segundo esse esquema de classificação, as supernovas do tipo IIb tem o He predominando sobre o H. Entre os subtipos do tipo II em que o H é predominante sobre o He estão o tipo IIn, no qual o espectro mostra emissão de linhas estreitas, e o tipo IIpec que tem características peculiares que não se enquadram nos outros tipos.

O esquema também tem subtipos de supernovas do tipo II que somente são diferenciadas pelo formato da curva de luz: o tipo IIL tem uma luminosidade quase linear no tempo; o tipo IIP tem um platô na sua variação temporal de luminosidade no tempo; e o tipo IIF no qual a supernova é considerada fraca.

Os critérios de classificação usados e os subtipos definidos para as supernovas do tipo II têm algumas diferenças em relação àqueles mostrados na classificação de Turatto, apesar de considerarem também o formato da curva de luz para definição de alguns tipos.

2.3. Considerações sobre os esquemas de classificação

A utilização desses esquemas de classificação de supernovas para identificar o tipo da supernova não é trivial e somente alguns poucos astrônomos especialistas são capazes de fazê-lo. Turatto, Benetti e Pastorello (2007) afirmam que a classificação de algumas supernovas não é óbvia e às vezes pode ser subjetiva e Harutyunyan (2008) considera a classificação feita por especialistas humanos insatisfatória e baseada em convicções subjetivas. Essa subjetividade se deve à análise ser feita visualmente no espectro, o que pode levar a classificações diferentes de uma mesma supernova, pois não existem parâmetros exatos. Essa diferença de classificação pode ser vista em Modjaz; Blondin; *et al.* (2014) que propõe correções na classificação de 26 supernovas.

Buscando melhorar a eficiência na classificação de supernovas, novas técnicas foram desenvolvidas (POZNANSKI; GAL-YAM, *et al.*, 2002) (GAL-YAM; POZNANSKI, *et al.*, 2004) (HOWELL; SULLIVAN, *et al.*, 2005) (SULLIVAN; HOWELL, *et al.*, 2006) (KUZNETSOVA; CONNOLLY, 2007). Além disso, a automatização na classificação das supernovas tornou-se foco de pesquisas, pois surge como uma forma de minimizar essa subjetividade e permitir a classificação mesmo em locais onde não se tem um astrônomo especialista. Dentre essas pesquisas foram encontrados seis trabalhos que resultaram em classificadores automáticos de supernovas e que são descritos no próximo capítulo.

3 CLASSIFICADORES AUTOMÁTICOS PESQUISADOS

Os classificadores automáticos de supernovas, que foram encontrados na pesquisa feita neste trabalho, podem ser divididos em dois tipos: os que analisam o espectro e os que analisam a curva de luz, pois nenhum deles faz as duas análises.

Uma questão importante a se considerar para decidir qual tipo de análise utilizar é o período em relação ao surgimento da supernova no qual se pode fazer a análise. O espectro pode (e deve) ser analisado em período próximo da luz máxima, que ocorre cerca de duas semanas após o surgimento da supernova, ou seja, a classificação pela análise do espectro pode ser feita em espectros capturados desde o surgimento da supernova até cerca de 30 dias depois do surgimento. Enquanto a classificação pela análise da curva de luz somente pode ser feita cerca de 60 a 90 dias depois do surgimento da supernova, quando seu formato permite diferenciar as características de cada tipo de supernova. Assim, apenas a classificação pela análise do espectro pode ser feita logo após a captura do primeiro espectro, permitindo saber rapidamente se o tipo da supernova interessa ou não para captura de mais informações.

Na pesquisa feita aqui foram encontrados seis trabalhos que desenvolveram e implementaram classificadores automáticos de supernovas. Quatro desses trabalhos fazem a classificação a partir da análise da curva de luz e dois classificam a partir da análise do espectro. Os classificadores que analisam a curva de luz apresentam resultados que permitem calcular o percentual de acerto na classificação e são usados para comparação com os resultados do CIntIa. Os classificadores que analisam o espectro não apresentam resultados que permitam a comparação.

A próxima seção descreve brevemente os classificadores que analisam a curva de luz porque foram utilizados apenas na comparação dos resultados. A seção

3.2 descreve em detalhes os classificadores que analisam o espectro, pois algumas de suas características foram utilizadas na construção do CIntIa.

3.1. Classificadores automáticos por análise da curva de luz

Os quatro trabalhos pesquisados que desenvolveram classificadores automáticos que analisam a curva de luz da supernova são: um trabalho desenvolvido em conjunto entre a *School of Physics and Astronomy, Tel-Aviv University em Israel e o Astronomy Department, California Institute of Technology em Pasadena* nos EUA (POZNANSKI; MAOZ; GAL-YAM, 2007); um trabalho desenvolvido no *Institute for Astronomy da University of Hawaii em Honolulu* (RODNEY; TONRY, 2009); uma dissertação de mestrado desenvolvida no *Dipartimento di Astronomia da Università Degli Studi di Padova* na Itália (PASCALE, 2011); e um trabalho desenvolvido em conjunto pelo *Department of Physics of the Stockholm University* na Suécia e pelo *Astrophysics Group of Cavendish Laboratory* of the University of Cambridge nos EUA (KARPENKA; FEROZ; HOBSON, 2013).

Todos esses trabalhos desenvolveram e testaram classificadores automáticos, publicando seus resultados que permitem calcular o percentual de acerto na classificação dos tipos de supernovas. No entanto, nenhum dos quatro trabalhos identificou todos os tipos de supernovas, restringindo a classificação aos tipos chamados “clássicos” e, mesmo assim, agrupando esses tipos em apenas duas ou três classes. Os quatro trabalhos utilizaram paradigmas de Inteligência Computacional para criação do seu método de classificação. Os métodos desenvolvidos por cada trabalho estão descritos brevemente nas próximas seções.

3.1.1. Classificador SN Automatic Bayesian Classifier

O classificador desenvolvido em conjunto pela *Tel-Aviv University e o pelo California Institute of Technology* (POZNANSKI; MAOZ; GAL-YAM, 2007) apresenta um método chamado *SN Automatic Bayesian Classifier (SN-ABC)*

que foi desenvolvido usando metodologias probabilísticas e Bayesianas para estudar a viabilidade de uma classificação automática probabilística de supernovas com base em sua curva de luz. A classificação apenas separa as supernovas em duas classes de acordo com sua origem: aquelas originadas de uma explosão termonuclear (tipo Ia) e aquelas originadas de um colapso de núcleo (outros tipos).

A abordagem geral para a classificação da supernova se baseia na técnica de ajuste do modelo Bayesiano. A partir do conhecimento prévio do *redshift* da supernova candidata, é ajustado um conjunto de distribuições de energia espectral do modelo para a fotometria observada da supernova. Em seguida, é maximizada a probabilidade para encontrar as características da supernova e derivar a evidência de que tipo é a supernova.

3.1.2. Classificadores Bayesiano e Nebuloso

O trabalho do *Institute for Astronomy da University of Hawaii* (RODNEY; TONRY, 2009) apresenta dois métodos de classificação. O primeiro é uma melhoria do programa *Bayesian Adaptive Template Matching* (BATM) que usa uma abordagem Bayesiana e o segundo é o *Supernova Ontology with Fuzzy Templates* (SOFT) que é baseado em Teoria dos Conjuntos Nebulosos. Ambos os métodos fazem a separação das supernovas em apenas dois tipos quanto a sua origem: originadas de explosão termonuclear (tipo Ia) e originadas de explosão por colapso do núcleo (outros tipos).

3.1.3. Classificadores Discriminação Logística e LogitBoost

O classificador desenvolvido na *Università Degli Studi di Padova* (PASCALE, 2011) utiliza um conjunto de curvas de luz de supernovas para treinar e testar

dois algoritmos de aprendizado de máquina: Discriminação Logística Linear e LogitBoost.

Foram feitos dois treinamentos separados cujos resultados também são mostrados separadamente. Primeiro os algoritmos foram treinados para classificar as curvas de luz em três classes: SN Ibc, SN II e SN Ia. Vale notar que os tipos Ib e Ic foram agrupados no tipo Ibc. Depois, foram treinados para identificar apenas duas classes, separando o que são supernovas do tipo Ia e as que não são tipo Ia: SN Ia e SN não-Ia.

O algoritmo Discriminação Logística opera por busca de melhores coeficientes da linha, separando duas classes em cada um dos subespaços de duas dimensões e produzindo três hiperplanos que tem a possibilidade de individualizar cada uma das três classes.

O algoritmo LogitBoost é baseado nos algoritmos tradicionais de *boost* que são classificadores muito simples chamados classificadores fracos porque realizam a classificação somente um pouco melhor que a classificação aleatória, ou seja, os algoritmos de *boost* tem a habilidade de classificar as curvas de luz parametrizadas com um sucesso um pouco maior que um classificador aleatório.

A ideia é que diferentes esquemas de aprendizagem podem ter seus resultados combinados e chegar a um resultado eficaz. O aprendizado em um algoritmo de *boost* é organizado de maneira iterativa, onde, a cada iteração um algoritmo fraco opera sobre um conjunto de dados, que é uma versão modificada do conjunto original. No algoritmo LogitBoost a classificação feita por cada classificador fraco contribui para a definição da melhor linha que separa as classes em cada um dos subespaços de duas dimensões.

3.1.4. Classificador com Redes Neurais Artificiais

O classificador desenvolvido pelo *Department of Physics of the Stockholm University*, Suécia, e pelo *Astrophysics Group of Cavendish Laboratory*, EUA, (KARPENKA; FEROZ; HOBSON, 2013) usa redes neurais artificiais para separar supernovas em duas classes, aquelas que são do tipo Ia e as supernovas que não são do tipo Ia (não-Ia). A classificação é feita por uma Rede Neural Perceptron de Múltiplas Camadas, cuja entrada é um vetor de características da curva de luz e a saída é a probabilidade dessa supernova ser do tipo Ia.

3.2. Classificadores automáticos por análise do espectro

Os dois trabalhos encontrados que criaram e implementaram métodos para determinação automática do tipo da supernova a partir análise do espectro são: o trabalho desenvolvido em conjunto pelo *Harvard-Smithsonian Center for Astrophysics* em Cambridge, EUA e pelo *Institute for Astronomy, University of Hawaii* em Honolulu, EUA (BLONDIN; TONRY, 2007); e o trabalho desenvolvido na Itália para uma tese de doutorado no *Dipartimento di Astronomia da Università Degli Studi di Padova* (HARUTYUNYAN, 2008).

Os dois trabalhos apresentam seus métodos de classificação e analisam a classificação feita em algumas supernovas, mas não mostram resultados que permitam calcular o percentual de acerto na identificação dos tipos de supernovas.

O primeiro trabalho desenvolveu o Classificador *Supernova Identification* (SNID) (BLONDIN; TONRY, 2007) e o segundo trabalho resultou no desenvolvimento do Classificador *GEneric cLAssification TOol* (GELATO) (HARUTYUNYAN, 2008). A descrição desses trabalhos é mostrada nas próximas seções.

3.2.1. Classificador Supernova Identification (SNID)

O classificador automático chamado *Supernova Identification* (SNID) (BLONDIN; TONRY, 2007), determina o tipo, o *redshift* e a idade da supernova usando seu espectro. O algoritmo é baseado na comparação de um espectro de entrada com os espectros armazenados, usando técnicas de correlação que já foram usadas para comparação de espectros de galáxias.

Os autores afirmam que uma classificação segura do tipo da supernova ainda é um desafio, principalmente, devido à inadequação do atual sistema de classificação que é puramente empírico. Afirmam também que o esquema de classificação é mais complicado ainda devido a subclasse chamada “peculiar” que pode estar associada a qualquer um dos tipos de supernovas: Ia, Ib, Ic e II.

3.2.1.1. Técnicas de correlação

Na técnica de correlação utilizada, um espectro de supernova com *redshift* z_s é correlacionado com um espectro armazenado de *redshift* zero. Quando é conhecido o tipo da supernova, essa correlação resulta no *redshift* da supernova. Quando o *redshift* da supernova é conhecido, a correlação consegue medir a similaridade entre os espectros.

3.2.1.2. Base de dados

A base de dados usada consiste de 879 espectros de 65 supernovas, sendo 322 espectros do tipo Ia, 19 espectros do tipo Ib/c e 353 espectros do tipo II. Os espectros foram retirados dos arquivos públicos SUSPECT: *The Online Supernova Spectrum Archive* (HOGAN; PARRENT; FELDT, 2010) e do *Center for Astrophysics* (CfA) *Supernova Data Archive* (HARVARD-SMITHSONIAN CENTER FOR ASTROPHYSICS, 2010).

3.2.1.3. Determinação do tipo da supernova

A ferramenta SNID foi desenvolvida para determinar os *redshift* de supernovas, mas também pode ser usada para determinar seu tipo, quando o *redshift* é conhecido. A ferramenta foi testada para determinação do tipo de apenas cinco supernovas.

3.2.2. Classificador Generic Classification Tool (GELATO)

O classificador automático chamado *GEneric cLAssification TOol* (GELATO) (HARUTYUNYAN, 2008) usa um banco de espectros de supernovas disponibilizado pelo grupo de supernovas de Padova (*Asiago Supernova Archive*) (BARBON, 2010) que coletou uma grande quantidade de supernovas descobertas desde os anos 90. Essa coleção contém 2708 espectros de 388 supernovas de todos os tipos.

Foram desenvolvidos softwares que realizam a comparação automática de espectros resultando na ferramenta chamada *Padova Supernova Spectra comPARison TOOl* (PASSPARTOO) (HARUTYUNYAN, 2008). Essa ferramenta trabalha com diferentes algoritmos, mas todos realizam a comparação automática de um espectro de entrada com um conjunto de espetros de supernovas já classificados, buscando encontrar o espectro com maior similaridade com aquele de entrada. Entre esses algoritmos, o mais usado é o GELATO, descrito detalhadamente na próxima seção. Os outros dois algoritmos são descritos brevemente em seguida.

3.2.2.1. Ferramenta GELATO

A ferramenta *GEneric cLAssification TOol* (GELATO) (HARUTYUNYAN, 2008) realiza uma comparação detalhada de um dado espectro de entrada com um grande número de espectros armazenados. Ele busca economizar tempo do

conhecimento de um especialista humano, tornando o processo repetitivo e servindo de guia para quem não é especialista.

A ferramenta divide cada espectro em um determinado número de caixas (*bins*) espectrais. Tanto o espectro de entrada quanto os espectros armazenados são divididos em 11 caixas, cujo tamanho de cada uma varia de 250 a 1100 angstroms. A comparação entre dois espectros é feita caixa a caixa, de maneira que a distorção da distribuição de energia espectral causada pelo *redshift* é reduzida quando comparada com a distorção causada pela comparação do espectro inteiro. Assim, essa abordagem tem a vantagem de minimizar os efeitos do *redshift* no espectro da supernova.

Um único conjunto de caixas fixas é adotado para comparação dos espectros de todos os tipos de supernovas, mas a possibilidade de considerar caixas variáveis é uma melhoria que deve ser feita em uma nova versão da ferramenta.

As próximas seções mostram como GELATO faz o pré-processamento, determina o espectro mais parecido e classifica as supernovas.

3.2.2.2. Pré-processamento

O processamento feito na comparação dos espectros é relativamente simples, mas um pré-processamento é necessário para preparação dos espectros. Os passos seguidos no pré-processamento são:

- **Correção de *redshift***: as velocidades radiais das respectivas supernovas são obtidas do Catálogo de Supernovas Asiago (BARBON, 2010).

- **Suavização de vagão (*boxcar*)**: são adotadas caixas de 70 angstroms para reduzir os ruídos, removendo aqueles de alta-frequência e ao mesmo tempo preservando as características espectrais.

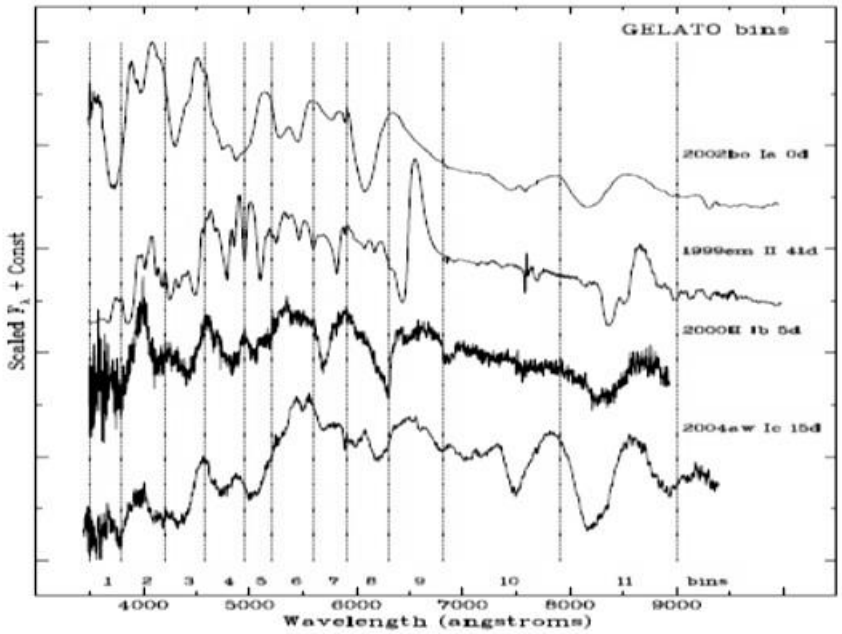
- **Reamostragem**: é feita uma interpolação linear com os pontos fixados a cada oito angstroms com objetivo de tornar o número de pontos de dados igual para todos os espectros.

- **Divisão em caixas (*bins*)**: divide o espectro em 11 caixas usando os valores predefinidos, conforme segue.

    - Caixa 1: 3504 a 3792 angstroms
    - Caixa 2: 3800 a 4192 angstroms
    - Caixa 3: 4200 a 4576 angstroms
    - Caixa 4: 4584 a 4936 angstroms
    - Caixa 5: 4944 a 5192 angstroms
    - Caixa 6: 5200 a 5592 angstroms
    - Caixa 7: 5600 a 5896 angstroms
    - Caixa 8: 5904 a 6296 angstroms
    - Caixa 9: 6304 a 6800 angstroms
    - Caixa 10: 6808 a 7904 angstroms
    - Caixa 11: 7912 a 9000 angstroms

Exemplo de espectros com suas onze caixas demarcadas são mostrados na Figura 3.1.

Figura 3.1 - As caixas de GELATO mostradas em exemplos de espectros de supernovas de diferentes tipos.



Fonte: Harutyunyan (2008)

Para cada espectro pré-processado é criado um arquivo que contém a correção de *redshift*, a suavização e as caixas espectrais. Esse arquivo é submetido ao processamento para determinação do espectro mais parecido.

3.2.2.3. Determinação do espectro mais parecido

Quando GELATO é executado e recebe um espectro de entrada, ele faz o pré-processamento desse espectro da mesma maneira que fez com os espetros armazenados. Depois, o processamento feito para determinar o espectro mais parecido consiste em calcular, para cada uma das N caixas do espectro, a distância relativa (δj) entre a caixa j do espectro de entrada e a caixa correspondente do espectro armazenado. Essa operação é feita para todos os espectros armazenados usando a equação:

$$\delta_j = \frac{1}{n. < f >_j} \sum_{i=1}^{n} |f_i - F_i^{norm}| \quad (3.1)$$

Sendo que:

- δ_j: distância relativa entre a caixa j do espectro de entrada e a caixa correspondente do espectro i
- n: número de espectros armazenados
- $<f>_j$: valor médio do fluxo na caixa j
- f_i: espectro de entrada do comprimento de onda λ_i na caixa j
- F_i: espectro armazenado i
- F_i^{norm}: é o fluxo Fi dimensionado para o fluxo do espectro de entrada da caixa j

Para determinar o espectro mais similar é necessário calcular a média das distâncias relativas (Δ) do espectro de entrada para o espectro armazenado. Isso é feito usando a equação:

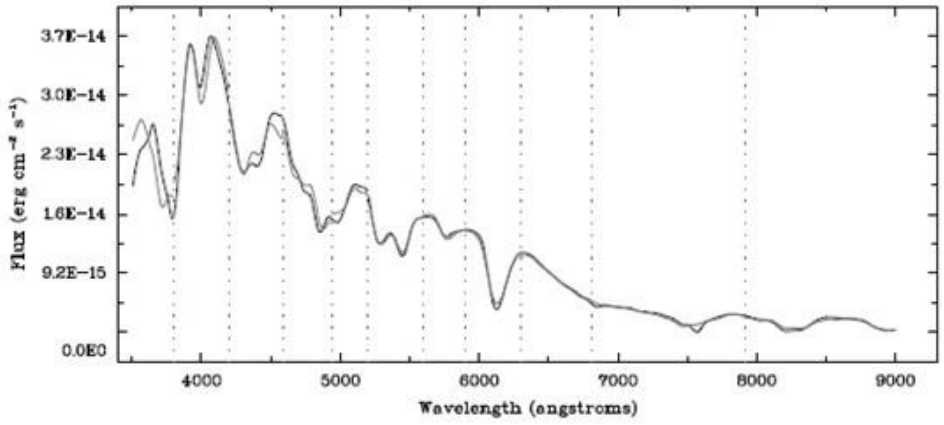
$$\Delta = \frac{1}{N} \sum_{j=1}^{n} \delta_j \quad (3.2)$$

Sendo que:

- Δ: média das distâncias relativas das caixas do espectro de entrada para o espectro armazenado
- N: número de caixas do espectro de entrada
- δj: distância relativa entre a caixa j do espectro de entrada e a caixa correspondente do espectro armazenado

O espectro armazenado mais similar ao espectro de entrada será aquele com menor valor da média das distâncias relativas (Δ). A Figura 3.2 mostra um exemplo de comparação entre dois espectros de diferentes supernovas.

Figura 3.2 - Exemplo de comparação de similaridade dos espectros das supernovas 1996X e 1994D.



Fonte: Harutyunyan (2008)

3.2.2.4. Qualidade do ajuste e classificação de supernovas

Apesar de definir a similaridade entre os espectros pela minimização da distância relativa, essa medida não permite definir a qualidade do ajuste entre os espectros. O valor da distância relativa de um espectro A pode ser menor que de outro espectro B para um determinado espectro de entrada, mas o ajuste entre os espectros pode ser melhor para B do que para A, dependendo da intensidade das linhas dos espectros.

A confiança é menor para determinação de similaridade entre espectros com linhas fracas e continuas, do que para espectros com linhas intensas. Ou seja, se pode avaliar melhor a similaridade entre espectros na presença de linhas espectrais, o que faz a classificação de espectros ser melhor quando eles contêm linhas intensas.

Em geral, os valores da distância relativa computados para casos de espetros com características espectrais fracas e contínuas é sistematicamente menor que aqueles de espectros que tem muitas características espectrais fortes. Para corrigir isso, foi calibrado o valor da distância relativa usando coeficientes que indicam se o espectro é mais ou menos rico em características. Esse parâmetro que define a riqueza de características (FP - *Feature-Richness*) é calculado pela equação

$$FP = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{n}\sum_{i=1}^{n}\frac{|f_i - F_i^{flat}|}{< f >_j} \tag{3.3}$$

Sendo que:

- FP: riqueza de características (*Feature-Richness*)
- N: número de caixas
- n: número de pontos da caixa j
- $<f>_j$: valor médio do fluxo na caixa j
- f_i: comprimento da onda λ_i na caixa j
- F_i: espectro da caixa i
- F_i^{flat}: melhor linha reta de ajuste para o espectro na caixa i

Para cada ajuste de espectro é definido o valor da qualidade do ajuste (GoF – *Goodness-of-Fit*) usando a equação

$$Gof = \left(\frac{\Delta}{FP}\right)^{-1} \tag{3.4}$$

Sendo que:

- GoF: qualidade do ajuste (*Goodness-of-Fit*)

- Δ: média das distâncias relativas das caixas do espectro de entrada para o espectro armazenado

- FP: riqueza de características (*Feature-Richness*)

A qualidade do ajuste permite uma estimativa numérica da qualidade do ajuste, que pode ser usada para comparar o ajuste de diferentes supernovas.

Os valores da qualidade do ajuste foram testados comparando espectros de diferentes conjuntos de supernovas. Os resultados foram comparados com a classificação independente de três astrônomos e mostraram que quando a qualidade do ajuste é maior ou igual a 1,4, a qualidade do ajuste é satisfatória e a determinação do tipo de supernova é segura. Quando a qualidade do ajuste é maior ou igual a 1 e menor que 1,4, pode resultar em ajustes bons ou razoáveis com detecção correta do tipo de supernova ou ajustes pobres com identificação incorreta do tipo de supernova. Se a qualidade do ajuste é menor que 1 a supernova é peculiar e não tem casamento de padrões com qualquer supernova armazenada.

Os autores afirmam que essa versão de GELATO não apresentou inconsistências apesar de necessitar de refinamento para estimar a qualidade do ajuste e estabelecer níveis de confiança com base nos valores da qualidade do ajuste. Afirmam também que GELATO é usado rotineiramente pelos membros do grupo de estudos de supernovas de Padova para classificar supernovas, mostrando-se como uma boa ferramenta. No entanto, não foram divulgados resultados que permitam calcular os percentuais de acerto da ferramenta.

3.2.3. Considerações sobre classificadores pesquisados

A ferramenta SNID (BLONDIN; TONRY, 2007) foi projetada para descobrir o *redshift* de uma supernova, mas também pode ser usada para descobrir o tipo

de supernova. A ferramenta foi testada em poucos exemplos na tarefa de determinação do tipo. A ferramenta GELATO (HARUTYUNYAN, 2008) foi construída com o objetivo de determinar o tipo da supernova e, segundo os autores, foi testada em uma quantidade significativa de supernovas, apesar de não ser mostrado o seu percentual de acerto na classificação. Alguns dos passos do pré-processamento realizado por ela foram utilizados neste trabalho. Esses dois classificadores foram utilizados posteriormente em conjunto no Programa de Classificação de Supernovas Asiago que analisou 221 supernovas nos dois primeiros anos do projeto (TOMASELLA; BENETTI, *et al.*, 2014), mas também não são apresentados resultados que permitam calcular os percentuais de acerto dos classificadores.

Os dois classificadores apresentados não utilizam paradigmas de Inteligência Computacional, mas métodos matemáticos convencionais que usam correlações para encontrar a similaridade entre os espectros. Ambos os trabalhos fazem a análise do espectro para medir a similaridade do espectro recém-descoberto com os espectros do banco já previamente classificados. Eles classificam a supernova recém-descoberta com o tipo daquela previamente classificada cujo espectro é mais similar. A classificação da supernova recém-descoberta é feita com base em apenas um espectro do banco, considerado o mais similar. Vale ressaltar também, que não consideram os critérios usados pelos especialistas, simplesmente analisam a similaridade com espectros já classificados.

Assim, partindo da constatação que a classificação feita por esses classificadores considera as características de apenas um espectro, que é o mais similar, e considerando também que as taxonomias existentes para classificação de supernovas são insatisfatórias, por serem ambíguas e não exaustivas (HARUTYUNYAN, 2008), pode-se afirmar que a classificação com base em único espectro é uma fragilidade dessas ferramentas. A alteração da classificação de uma supernova do banco de espectros pode fazer com que a classificação de várias supernovas se torne incorreta, caso os espectros

dessas supernovas tenham usado como base para classificação a supernova que teve o tipo alterado. Vale ressaltar que a mudança na classificação de supernovas é uma questão pertinente e atual, que já aconteceu recentemente como pode ser visto em Modjaz; Blondin; *et al.* (2014) que propôs correções na classificação existente para 26 supernovas de diversos tipos.

O método de classificação desenvolvido neste trabalho usa Redes Neurais Artificiais para analisar e identificar padrões de um conjunto de espectros, minimizando os erros que podem haver na alteração dos tipos de algumas supernovas. O próximo capítulo apresenta um estudo das Redes Neurais Artificiais, dirigido para a aplicação deste classificador.

4 REDES NEURAIS ARTIFICIAIS

Redes Neurais Artificiais (RNAs) são modelos computacionais baseados na estrutura e no funcionamento do cérebro biológico que, apesar de não rodar programas, controla o comportamento do indivíduo, sendo responsável por seu pensamento, percepção, cognição e emoção (BRAGA; CARVALHO; LUDEMIR, 2000). O fundamento das RNAs está baseado na hipótese de que a atividade mental consiste basicamente na atividade eletroquímica dos neurônios (RUSSEL; NORVIG, 2013).

Assim, as RNAs se baseiam na estrutura fisiológica do cérebro e tentam reproduzir o comportamento básico e a dinâmica das funções das redes biológicas. Mas as RNAs diferem bastante das redes biológicas do ponto de vista físico. As similaridades são basicamente que os dois sistemas usam computação paralela e distribuída, se comunicam por meio de conexões sinápticas, possuem detectores de características, redundância e modularização das conexões. No entanto, apesar de poucas similaridades, as características comuns permitem às RNAs reproduzir várias funções atribuídas exclusivamente a inteligência humana (BRAGA; CARVALHO; LUDEMIR, 2000).

O sucesso na utilização das RNAs para resolução de diversos problemas se deve a algumas características de sua metodologia usada na resolução desses problemas, tais como esses apresentados em Bittencourt (2006):

- Ter capacidade de aprender através de exemplos, generalizando o aprendizado para reconhecer novas instâncias similares;
- Apresentar bom desempenho em tarefas onde o conhecimento do especialista não está disponível;
- Ser adequada a problemas onde não existe conhecimento a respeito dos modelos matemáticos dos domínios de aplicação;

- Possuir elevada imunidade ao ruído, sendo capaz de manter o desempenho em presença de algumas informações falsas ou ausentes;
- Apresentar a possibilidade de simulação de raciocínio “a priori” e impreciso.

Essas características tornam as RNAs adequadas a um grande número de tarefas. Entre elas destacam-se essas apresentadas em Luger (2004):

- Classificação: separa as entradas em categorias ou grupos;
- Reconhecimento de padrões: identifica padrões complexos nas entradas;
- Evocação de memória: inclui memória de endereçamento por conteúdo;
- Predição: determina tendências com base nas entradas;
- Otimização: minimiza ou maximiza uma função;
- Filtragem de ruídos: separa o sinal do ruído de fundo.

O processamento da informação nas RNAs é feito de maneira paralela e distribuída por uma grande quantidade de unidades “simples” de processamento, chamadas neurônios, conectadas entre si (REZENDE, 2003), mas as operações são independentes de qualquer unidade de processamento isolada e o resultado é uma função do conjunto das unidades.

A “inteligência” das RNAs não está baseada na representação simbólica do raciocínio humano. Seu comportamento inteligente vem das interações entre neurônios, assim, o conhecimento não está representado explicitamente em determinado local, mas emerge da rede. O conhecimento é adquirido através de um processo de aprendizagem e armazenado pela força de coesão entre os neurônios, conhecida como peso sináptico (HAYKIN, 2001).

A primeira pesquisa sobre RNAs, que teve como objetivo criar um modelo artificial para o neurônio biológico, aconteceu em 1943, quando o psiquiatra e neuroanatomista Warren McCulloch e o matemático Walter Pitts criaram um modelo artificial de um neurônio, conhecido como neurônio MCP, e apresentaram suas capacidades computacionais. A partir daí surgiram novas pesquisas, mas um novo marco aconteceu somente em 1958 quando Frank Rosenblatt apresentou seu novo modelo, o *Perceptron*, que com sinapses ajustáveis poderiam ser treinados para classificar certos padrões. Mas esse modelo não era capaz de resolver algumas tarefas simples e poucas pesquisas foram desenvolvidas, até que em 1982 John Hopfield mostrou as propriedades associativas das RNAs, o que marcou a retomada das pesquisas. Em 1986 com a criação do algoritmo de treinamento por retropropagação (*back-propagation*), por Rumelhart, Campbell e Borguetti, as RNAs se tornaram capazes de resolver problemas considerados “difíceis”. A partir daí aconteceu uma explosão de interesse nas RNAs resultando na criação de novos modelos com novos algoritmos de aprendizagem e novas arquiteturas de RNAs (BRAGA; CARVALHO; LUDEMIR, 2000).

Esse capítulo contém um levantamento bibliográfico sobre as RNAs. A próxima seção mostra as características e o funcionamento do neurônio biológico e do neurônio artificial. A Seção 4.2 apresenta os processos de aprendizagens das RNAs. A Seção 4.3 mostra algumas arquiteturas das RNAs e a Seção 4.4 descreve alguns modelos de RNAs.

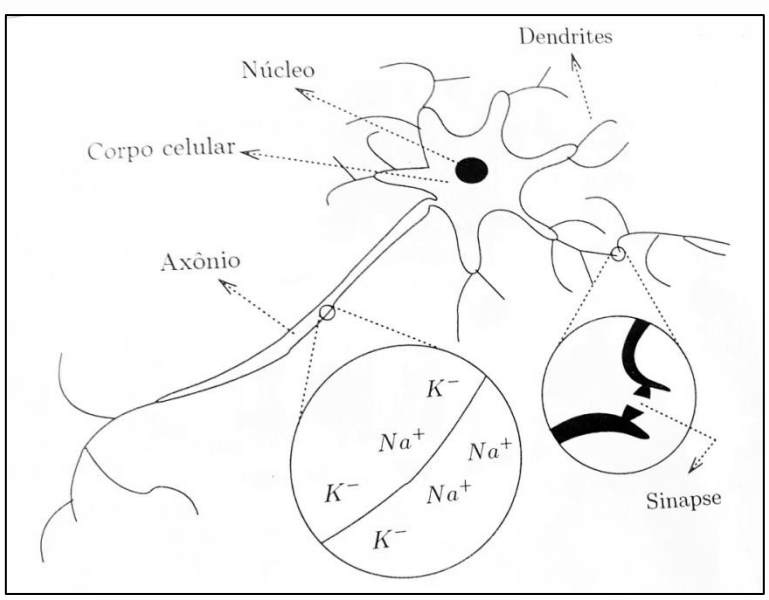
4.1. Neurônio biológico e neurônio artificial

A criação de modelos para o neurônio biológico foi o alvo das primeiras pesquisas sobre RNAs. Como o neurônio é a unidade básica de processamento, a sua modelagem em neurônios artificiais foi o ponto de partida para criação de modelos mais complexos. A estrutura e o funcionamento dos neurônios biológico e artificial são mostrados a seguir.

4.1.1. Neurônio biológico

As RNAs buscam modelar a estrutura do neurônio biológico e simular o seu comportamento e relacionamento com outros neurônios. Como toda modelagem, o foco é na representação das principais partes que são usadas no processamento. Nesse caso, essas partes são: corpo celular, dendritos, axônio e sinapses. A Figura 4.1 mostra essas partes principais de um neurônio biológico.

Figura 4.1 - Neurônio biológico.



Fonte: Bittencourt (2006)

Cada parte, mostrada na Figura 4.1, tem uma função específica no processamento e na troca de informações feitas entre neurônios:

- **Dendritos**: recebem as informações de outras células
- **Corpo Celular**: processa a informação
- **Axônio**: conduz a informação até a extremidade
- **Sinapses**: enviam a informação para as outras células

O funcionamento do neurônio biológico é bastante simples. Ele mantém-se em estado de repouso até que recebe um sinal e sofre pequenas oscilações. Caso o sinal seja excitatório ele envia um sinal aos próximos neurônios e retorna ao estado de repouso. Caso o sinal seja inibitório ele apenas retorna ao estado de repouso sem nada enviar. Esse comportamento foi simulado no neurônio artificial.

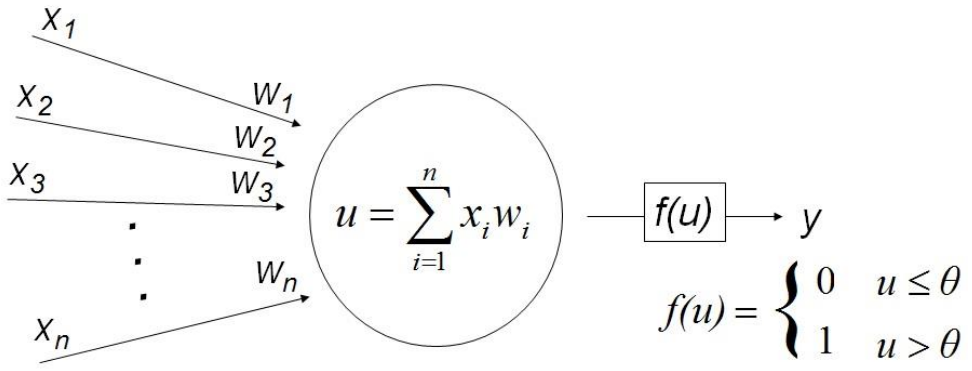
4.1.2. Neurônio artificial

O primeiro neurônio artificial foi um modelo matemático, proposto pelos pesquisadores McCulloch e Pitts como uma simplificação do neurônio biológico, para simular o comportamento dos neurônios (BRAGA; CARVALHO; LUDEMIR, 2000). Esse modelo ficou conhecido como neurônio MCP.

4.1.2.1. Neurônio MCP

A Figura 4.2 mostra uma representação gráfica do neurônio MCP. Ele tem diversas entradas (dendritos) ponderadas por pesos (comportamento das sinapses) e um único canal (axônio) condutor da saída binária, que indica potenciais de ação ou repouso, até as diversas extremidades que distribuem essa mesma saída aos diversos neurônios que estejam ligadas.

Figura 4.2 - Neurônio MCP.



Fonte: Adaptada de Rezende (2003)

Sendo que:

- x_i: entrada *i*
- w_i: peso *i*
- *u*: saída do combinador linear
- *f(u)*: função de ativação
- *y*: saída do neurônio
- θ: limiar de ativação

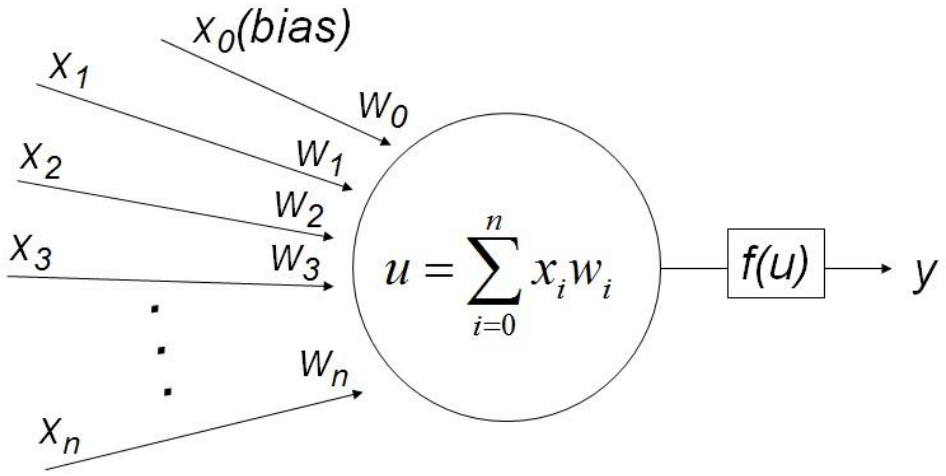
No neurônio MCP, mostrado na Figura 4.2, a saída linear (u) é o resultado do combinador linear que calcula o produto escalar (ou produto interno) das entradas (x_i) e dos pesos (w_i). Os pesos (w_i) não são ajustáveis, ou seja, são fixos e não podem ser modificados. A função de ativação (f(u)) é linear e ativada se acima de um limiar (θ) (REZENDE, 2003).

A partir do neurônio MCP foram propostos outros modelos de neurônio que permitem a saída de outros valores além de zero ou um. O neurônio não-linear é um desses modelos.

4.1.2.2. Neurônio não-linear

O neurônio não-linear mais utilizado atualmente pela comunidade de redes neurais tem sua representação gráfica mostrada na Figura 4.3. Em comparação com o neurônio MCP, as principais mudanças feitas no neurônio não-linear são os pesos (w_i) que são ajustáveis, a função de ativação (f(u)) que pode ser não linear e o termo *bias* que é acrescido ao produto escalar das entradas (x_i) pelos pesos (w_i) .

Figura 4.3 - Neurônio Não-linear.



Fonte: Adaptada de Haykin (2001)

Sendo que:

- x_i: entrada i
- w_i: peso i
- u: saída do combinador linear
- f(u): função de ativação
- y: saída do neurônio
- *Bias*: entrada externa que tem seu valor sempre fixo

O termo *bias* é um valor positivo ou negativo que é aplicado externamente, ou seja, não faz parte das entradas recebidas pela rede ou de outros neurônios. Seu valor aumenta ou diminui a entrada líquida da função de ativação, de acordo com seu valor positivo ou negativo, respectivamente. O termo *bias* permite deslocar o hiperplano da origem no eixo das abscissas e, com sua utilização, o limiar não precisa se alterado durante o treinamento, apenas os pesos (HAYKIN, 2001).

As características do neurônio não-linear permitem usar diferentes tipos de funções de ativação, facilitando a adequação da saída do neurônio ao tipo do problema. A Tabela 4.1 mostra quatro tipos de funções de ativação que podem ser utilizadas no neurônio não-linear: limiar, linear por partes, sigmoide logística e tangente hiperbólica (QUILES, 2004).

Tabela 4.1 - Tipos de Funções de Ativação.

Tipo de Função	Função
Limiar	$f(u) = \begin{cases} 1 \; se \; u \geq 0 \\ 0 \; se \; u < 0 \end{cases}$
Linear por Partes	$f(u) = \begin{cases} 1 \; se \; u \; \geq \frac{1}{2} \\ u \; se - \frac{1}{2} < u < \frac{1}{2} \\ 0 \; se \; u \; \leq -\frac{1}{2} \end{cases}$
Sigmoide Logística	$f(u) = (1 + \exp(-au))^{-1}$
Tangente Hiperbólica	$f(u) = \tanh(u)$

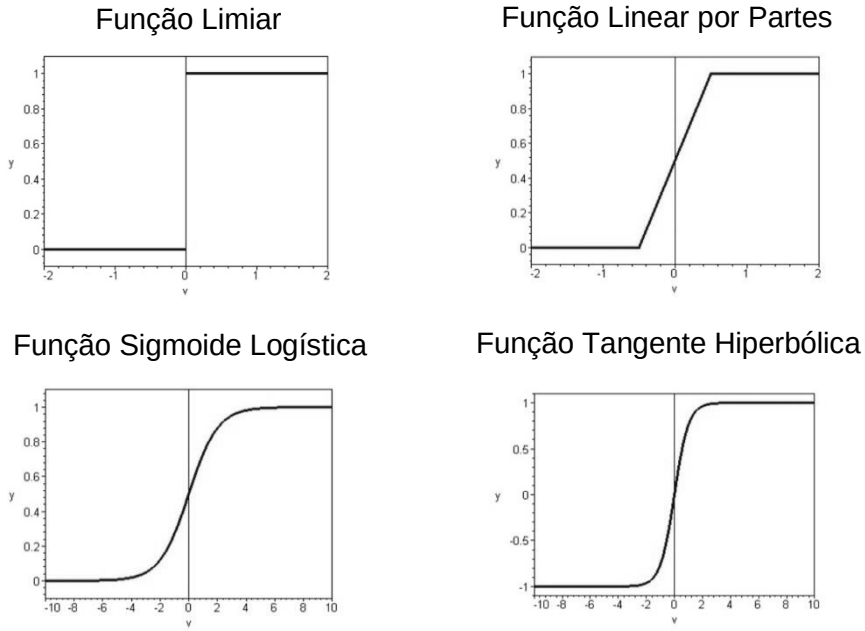
Fonte: Adaptada de Quiles (2004)

Sendo que:

- *u*: saída do combinador linear
- *f(u)*: função de ativação
- *α*: parâmetro de inclinação
- exp: exponencial (e)
- tanh: tangente hiperbólica

A Figura 4.4 ilustra graficamente as quatro funções de ativação mostradas na Tabela 4.1.

Figura 4.4 - Representação Gráfica dos Tipos de Funções de Ativação.



Fonte: Quiles (2004)

A função limiar, é a mesma usada pelo neurônio MCP, que também pode ser utilizada nesse tipo de neurônio. A função linear por partes é uma variação da função limiar que cresce linearmente e permite, além da saída dos valores zero e um, a saída de valores reais no intervalo de zero a um.

A função sigmoide logística também tem como saída valores reais no intervalo entre zero e um. O parâmetro α permite que se obtenham diferentes inclinações. Ela é a função mais utilizada atualmente na construção de RNAs porque exibe um balanceamento adequado entre o comportamento linear e o comportamento não-linear. A função tangente hiperbólica é usada quando se necessita que a saída de função esteja no intervalo entre -1 e +1, assumindo uma forma antissimétrica em relação a origem. Assim, ela permite que a função assuma valores negativos, o que traz benefícios analíticos (HAYKIN, 2001). A escolha da função de ativação é uma definição importante no processo de aprendizagem das RNAs.

4.2. Processo de aprendizagem

A habilidade de aprender é uma capacidade primordial das RNAs. A aprendizagem acontece partir de um processo interativo de ajustes aplicados sobre seus pesos sinápticos (HAYKIN, 2001). Como aprender é um conceito usado em várias áreas, não existe uma definição de aprendizagem que seja consenso. Mas, no contexto das RNAs pode-se definir aprendizagem como:

> Aprendizagem é um processo pelo qual os parâmetros livres de uma rede neural são adaptados através de um processo de estimulação pelo ambiente no qual a rede está inserida. O tipo de aprendizagem é determinado pela maneira pela qual a modificação dos parâmetros ocorre *(HAYKIN, 2001)*.

Essa definição mostra que uma RNA interage com o ambiente e que existem diferentes tipos de aprendizagem, logo, pode-se perceber que não existe um único algoritmo de aprendizagem para todas as RNAs. Basicamente, as diferenças entre os tipos de aprendizagem dizem respeito a maneira como é feito o ajuste do peso sináptico de um neurônio. Essas diferenças resultam em cinco diferentes regras e dois paradigmas de aprendizagem.

4.2.1. Regras de aprendizagem

As regras de aprendizagem diferem entre si pela maneira como é feito o ajuste do peso sináptico e pela maneira como a RNA se relaciona com seu ambiente.

Segundo Haykin (2001) existem cinco regras básicas de aprendizagem para as RNAs. (1) Aprendizagem por correção de erro que se fundamenta na filtragem ótima. (2) Aprendizagem baseada em memória que explicitamente memoriza os dados de entrada. (3) Aprendizagem *hebbiana* e (4) aprendizagem competitiva que são inspiradas em considerações neurobiológicas. (5) Aprendizagem de Boltzmann cuja base vem da mecânica estatística.

4.2.1.1. Aprendizagem por correção de erro

A aprendizagem por correção de erro pressupõe que se conhece a saída desejada para RNA para que o erro possa ser calculado. Assim, considerando um neurônio *j* da camada de saída, o erro (*e*) desse neurônio será calculado subtraindo a saída desse neurônio (*y*) da saída esperada (t), conforme segue

$$e_j = t_j - y_j \tag{4.1}$$

Sendo que:

- e_j: erro do neurônio j
- t_j: saída esperada para o neurônio j
- y_i: saída encontrada para o neurônio j

A correção do erro é um processo interativo de ajuste dos pesos sinápticos, de forma que, quando o erro (e) é negativo, ou seja, quando a saída encontrada (y) é maior que a saída esperada (t), o peso (w) é decrementado. Quando o erro é positivo, ou seja, quando a saída encontrada (y) é menor que a saída esperada (t), o peso (w) é incrementado. Considerando *n* um passo desse processo interativo, a atualização do peso (w) de um neurônio *j*, cuja entrada *i* resultou no erro (e), é dada pela equação

$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n) \tag{4.2}$$

Sendo que:

- w_{ij}: peso do neurônio j para a entrada i
- n: número do passo no processo de ajuste dos pesos
- Δw_{ij}: variação do peso para o erro do neurônio j com a entrada i

A regra de aprendizagem, normalmente referida como regra delta, usada para calcular a variação do peso (Δw) considera, além dos valores da entrada (x) e do erro (e), o valor da taxa de aprendizagem (η), que é uma constante positiva do conjunto dos números reais no intervalo entre 0 e 1, usada para ajustar a proporção do erro (e) que será usada para cálculo do peso (w). O cálculo da variação do peso (Δw) é

$$\Delta w_{ij} = \eta x_i e_j \quad (4.3)$$

Sendo que:

- Δw_{ij}: variação do peso para o erro do neurônio j com a entrada i
- η: taxa de aprendizagem
- x_i: entrada i
- e_j: erro do neurônio j

A taxa de aprendizagem tem uma influência muito grande para que seja alcançada a estabilidade e convergência do processo de aprendizagem interativo.

A aprendizagem por correção de erro é utilizada nas RNAs Perceptron e também nas RNAs Perceptron de Múltiplas Camadas.

4.2.1.2. Aprendizagem baseada em memória

Na aprendizagem baseada em memória a maioria dos exemplos de entrada-saída deve ser armazenada explicitamente em uma grande memória que é representada por um conjunto de pares ordenados (x_i, d_i) formado pelo vetor de entrada (x_i) com a resposta desejada (d_i) representada por um valor escalar. A classificação de um novo vetor de entrada (x_{teste}) é feita através de uma busca na vizinhança local dessa entrada.

Os algoritmos de treinamento baseados em memória se diferenciam entre si de acordo com a variação de dois ingredientes principais:

- O critério utilizado para definir a vizinhança local do novo vetor de entrada (x_{teste});
- A regra de aprendizagem aplicada aos exemplos de entrada da vizinhança local do novo vetor de entrada (x_{teste}).

Um importante classificador baseado em memória são as Redes de Função de Base Radial (HAYKIN, 2001).

4.2.1.3. Aprendizagem Hebbiana

A aprendizagem hebbiana tem sua base na força da sinapse em função da sincronia dos neurônios que estão em ambos os lados dessa sinapse. Essa aprendizagem é chamada hebbiana porque o neuropsicólogo Donald Heeb propôs em 1949 um postulado como uma base da aprendizagem associativa. O postulado foi proposto no contexto neurobiológico, mas sua aplicação em RNAs pode ser descrito como uma regra em duas partes:

1. A força de uma sinapse é aumentada se os dois neurônios que formam essa sinapse são ativados simultaneamente;
2. A força de uma sinapse é diminuída se os dois neurônios que formam essa sinapse são ativados assincronamente.

A forma mais simples de aprendizagem hebbiana é

$$\Delta w_{ij} = \eta y_j x_i \qquad (4.4)$$

Sendo que:

- Δw_{ij}: variação do peso para o erro do neurônio j com a entrada i
- η: taxa de aprendizagem

- x_i: sinal pré-sináptico
- y_i: sinal pós-sináptico

A aprendizagem hebbiana é utilizada nas RNAs que usam o Modelo de Hopfield.

4.2.1.4. Aprendizagem competitiva

Na aprendizagem competitiva os neurônios competem entre si e o vencedor ganha o direito de disparar enquanto os outros permanecem em repouso. A regra dessa aprendizagem é formada por três elementos básicos:

- Um conjunto de neurônios que respondem diferentemente aos padrões de entrada devido apenas a alguns pesos sinápticos distribuídos aleatoriamente que os diferenciam;
- Um limite que é imposto sobre a força de cada neurônio;
- Um mecanismo que permite existir somente um neurônio vencedor.

A regra de aprendizagem competitiva padrão é

$$\Delta w_{ij} = \begin{cases} \eta(x_i - w_{ij}) & se\ j\ vencer \\ 0 & se\ j\ perder \end{cases} \quad (4.5)$$

Sendo que:

- Δw_{ij}: variação do peso para o erro do neurônio j com a entrada i
- w_{ij}: peso para o erro do neurônio j com a entrada i
- η: taxa de aprendizagem
- x_i: entrada i

Essa regra de aprendizagem é utilizada nas Redes de Kohonen.

4.2.1.5. Aprendizagem de Boltzmann

A aprendizagem de Boltzmann é um algoritmo estocástico derivado da mecânica estatística. Os neurônios operam de maneira binária e sempre estão em um estado ligado, representado por +1, ou desligado, representado por -1. Os neurônios formam a chamada Máquina de Boltzmann que tem uma função de energia cujo valor é determinado pelos estados individuais dos neurônios.

Existem dois grupos funcionais de neurônios: os visíveis que formam uma interface entre a rede e o ambiente; e os ocultos que operam livremente. A máquina tem dois modos de operação:

- Condição presa: os neurônios visíveis estão todos presos a estados determinados pelo ambiente;
- Condição de operação livre: todos os neurônios estão livres.

A regra de aprendizagem de Boltzmann é

$$\Delta w_{ij} = (\rho_{ij}^{+} - \rho_{ij}^{-}), i \neq j \quad (4.6)$$

Sendo que:

- Δw_{ij}: variação do peso para o erro do neurônio j com a entrada i
- w_{ij}: peso para o erro do neurônio j com a entrada i
- η: taxa de aprendizagem
- ρ_{ij}^{+}: correlação dos estados dos neurônios i e j na condição presa
- ρ_{ij}^{-}: correlação dos estados dos neurônios i e j na condição livre

As correlações entre os neurônios assumem valores no intervalo entre -1 e +1.

As regras de aprendizagem estão associadas a um dos paradigmas de aprendizagem: supervisionada ou não-supervisionada.

4.2.2. Paradigmas de aprendizagem

Existem dois paradigmas de aprendizagem que se diferenciam de acordo com as características do conjunto de exemplos usados no treinamento. Quando cada exemplo apresenta uma saída esperada a aprendizagem é dita supervisionada e quando não existe uma saída esperada a aprendizagem é chamada não-supervisionada (BRAGA; CARVALHO; LUDEMIR, 2000).

4.2.2.1. Aprendizagem supervisionada

Na aprendizagem supervisionada existe um supervisor externo que fornece a saída desejada para cada entrada da rede. O objetivo é ajustar os parâmetros da rede de maneira que, para cada entrada, a saída da rede seja aquela fornecida. Cada saída da rede tem seu erro calculado que é usado pelo algoritmo de treinamento para ajuste dos pesos sinápticos visando minimizar esse erro.

Esse paradigma de aprendizagem é capaz de realizar tarefas como classificação de padrões e aproximação de funções. A regra de aprendizagem por correção de erro é um exemplo desse paradigma. O modelo de RNAS Perceptron e, também, o Perceptron de Múltiplas Camadas se enquadram nesse paradigma.

4.2.2.2. Aprendizagem não-supervisionada

A aprendizagem não-supervisionada funciona sem um supervisor que forneça a saída esperada ou acompanhe o processo de aprendizado. O conjunto de exemplos usados no treinamento contêm apenas os padrões de entrada. O algoritmo de treinamento busca criar novos grupos automaticamente

estabelecendo uma harmonia com as regularidades estatísticas das entradas e formando novas representações internas. O algoritmo somente consegue encontrar os padrões ou características no conjunto de exemplos de entrada se houver uma redundância nos dados de entrada.

A aprendizagem competitiva e a aprendizagem hebbiana são regras que exemplificam esse paradigma. O modelo de Hopfield e o Mapa de Kohonen usam esse paradigma em seus algoritmos de aprendizagem.

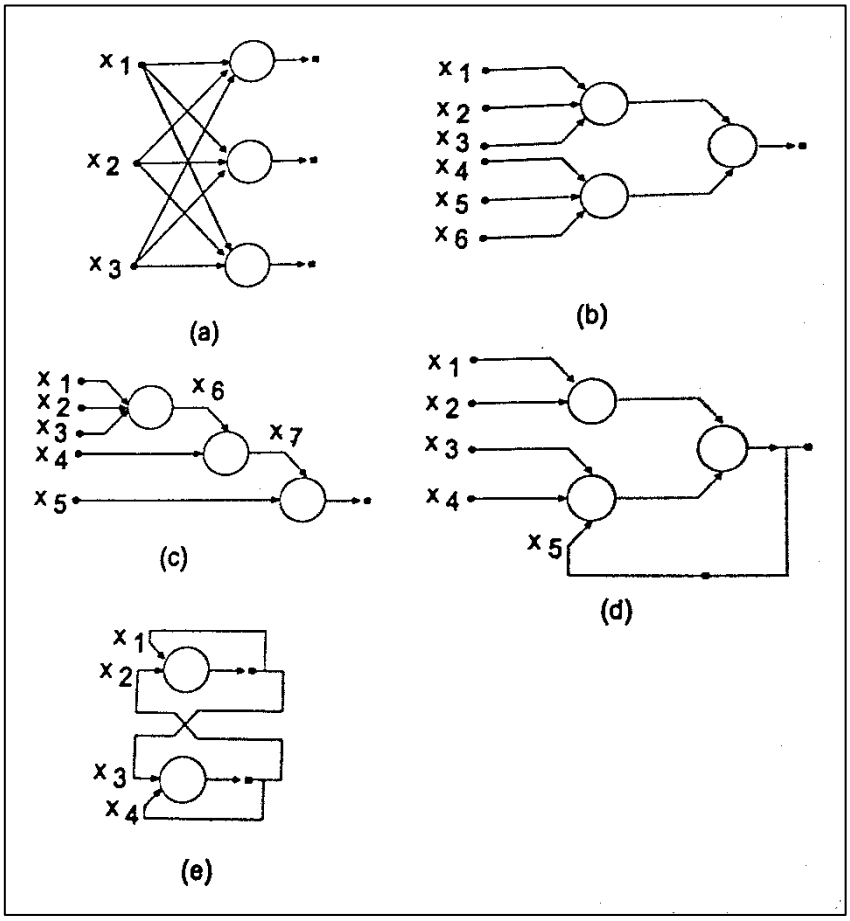
O processo de aprendizagem da RNA está diretamente relacionado com a sua arquitetura.

4.3. Arquiteturas de redes neurais

A arquitetura que será usada na RNA depende do tipo de problema que será tratado. Um problema linearmente separável, por exemplo, pode ser resolvido por uma Rede Neural *Perceptron*. Assim como, um problema que envolve processamento temporal deve ser resolvido usando Redes Neurais Recorrentes.

Os principais parâmetros que fazem parte da definição da arquitetura da RNA são: número de camadas, número de neurônios em cada camada, tipo de conexão entre os neurônios e topologia da rede (BRAGA; CARVALHO; LUDEMIR, 2000). A Figura 4.5 mostra alguns exemplos de arquiteturas das RNAs que são definidas de acordo com esses parâmetros e discutidas na sequência.

Figura 4.5 - Representação Gráfica de exemplos das Arquiteturas das RNAs.



Fonte: Braga; Carvalho e Ludemir (2000)

4.3.1. Quanto ao número de camadas

O parâmetro número de camadas divide as redes neurais em dois tipos: redes de camada única e rede de múltiplas camadas. As redes de camada única têm somente um neurônio entre qualquer entrada e qualquer saída da rede. São exemplos as imagens ‘a’ e ‘e’ da Figura 4.5.

As redes de múltiplas camadas têm pelo menos dois neurônios entre alguma entrada e alguma saída da rede, ou seja, tem pelo menos uma camada intermediária (ou escondida) entre a entrada e a saída da rede. São exemplos as imagens ‘b’, ‘c’ e ‘d’ da Figura 4.5.

4.3.2. Quanto ao tipo de conexões

As conexões de uma rede neural podem ser de dois tipos: acíclica ou cíclica. Na rede acíclica a saída de um neurônio que está na *i*-ésima camada não pode servir de entrada para um neurônio que está em uma camada de índice menor ou igual a *i* em relação a entrada. As imagens ‘a’, ‘b’ e ‘c’ da Figura 4.5 são exemplos de rede acíclica.

Na rede cíclica a saída de um neurônio que está na *i*-ésima camada é usada como entrada para um neurônio que está em uma camada de índice menor ou igual a *i*. As imagens ‘d’ e ‘e’ da Figura 4.5 são exemplos de rede cíclica. A imagem ‘d’ é considerada um caso particular de rede cíclica porque se comporta como autômato reconhecedor onde a única saída final ligada às entradas fornece o comportamento de autômato. A imagem ‘e’ também é um caso particular de rede cíclica, chamada auto associativa porque associam o padrão de entrada com ele mesmo, e são úteis para recuperação ou “regeneração” de um padrão de entrada.

4.3.3. Quanto a conectividade

As redes neurais podem ser fracamente (ou parcialmente) conectadas como nas imagens ‘b’, ‘c’ e ‘d’ da Figura 4.5 ou completamente conectadas como nas imagens ‘a’ e ‘e’ da Figura 4.5. Em uma rede completamente conectada cada entrada será processada por todos os neurónios da rede. Caso isso não ocorra a rede é fracamente conectada.

4.4. Modelos de redes neurais

Vários modelos de RNAs foram desenvolvidos com variações no processo de aprendizagem e na arquitetura da rede. Cada modelo de aprendizagem tem

seu algoritmo de treinamento, de acordo com as regras e paradigmas de aprendizagem adotados, e também adota uma determinada arquitetura de rede, de acordo com o número de camadas, o tipo de conexão entre os neurônios e a topologia da rede.

Os modelos clássicos Perceptron e Perceptron de Múltiplas Camadas (MLP) adotam a aprendizagem supervisionada por correção de erro. Eles se diferenciam um do outro pelo número de camadas, sendo que as redes neurais Perceptron têm apenas uma camada, enquanto as redes neurais MLP têm mais de uma camada. Ambos são adequados a problemas de classificação, mas as RNAs Perceptron resolvem somente problemas linearmente separáveis.

Entre os diversos outros modelos, destacam-se os modelos de memória matricial como o modelo não linear de Willshaw, o modelo linear de Kohonen e Anderson e o modelo recorrente de Hopfield. Entre os modelos que tem capacidade de auto-organização podem ser citadas as redes de Kohonen e as redes ART (*Adaptive Resonance Theory*) (BRAGA; CARVALHO; LUDEMIR, 2000).

O modelo não-linear de Willshaw utiliza a regra de aprendizagem hebbiana com uma única camada de neurônios com função de ativação não-lineares e conexões sinápticas binárias. O modelo linear também usa a aprendizagem hebbiana e tem uma única camada de neurônios com conexões sinápticas binárias, mas sua função de ativação é linear. O modelo recorrente de Hopfield usa função de ativação não-linear e tem também uma única camada de neurônios, mas difere dos outros dois modelos porque as saídas estão ligadas às entradas por um atraso de tempo, o que dá ao modelo características temporais em que a resposta da rede depende sempre do seu estado anterior. Os três modelos usam o paradigma não-supervisionado de aprendizagem e são adequados para problemas de evocação de memória.

Os modelos que possuem capacidade de organização também usam o paradigma não-supervisionado de aprendizagem, mas aplicam a regra de aprendizagem competitiva. A rede de Kohonen contém duas camadas, sendo que a primeira camada é responsável pela aquisição dos padrões e a segunda camada, conhecida como Mapa de Kohonen, permite a representação de dados N-dimensionais em um espaço M-dimensional, sendo que M < N. As redes ART também têm duas camadas, a primeira é a camada de entrada e a segunda, denominada camada de ressonância, é a camada de armazenamento. Ela tem um algoritmo de aprendizagem “plástico” para se adaptar a novos padrões de entrada indefinidamente. Esses dois modelos são adequados para problemas de reconhecimento de padrões, categorização de dados e predição.

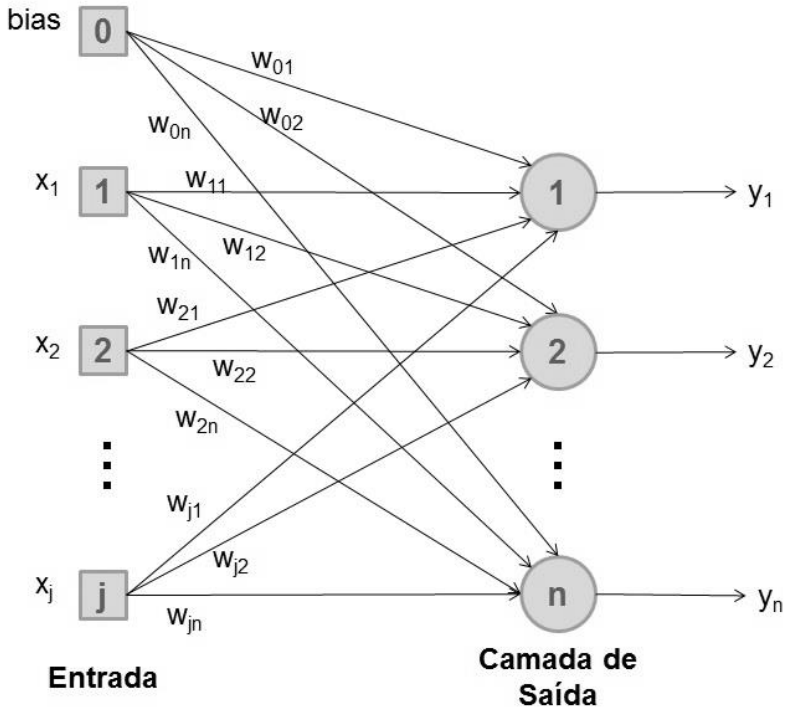
A classificação de supernovas, que é o foco deste trabalho, tem um conjunto de exemplos com a saída esperada, logo pode ser resolvido utilizando a regra de aprendizagem por correção de erros do paradigma supervisionado. Assim, RNAs Perceptron e MLP são adequadas a esse problema e são detalhadas no restante deste capítulo.

4.4.1. Redes Neurais Perceptron

As redes neurais Perceptron utilizam a regra de aprendizagem por correção de erro do paradigma supervisionado e sempre convergem para padrões linearmente separáveis. A Figura 4.6 mostra um exemplo de uma rede neural Perceptron que tem j entradas e n neurônios na sua única camada.

A arquitetura da rede neural Perceptron tem apenas uma camada que é acíclica e totalmente conectada, e usa a função de ativação limiar que é uma função linear. Seus valores de entrada e saída são binários. A regra de propagação é dada pelo produto interno das entradas pelos pesos com adição do termo bias (HAYKIN, 2001).

Figura 4.6 - Rede Neural Perceptron.



Fonte: Produção do autor.

O algoritmo de aprendizagem da rede perceptron se baseia no comportamento do neurônio que não tem variação de peso se a saída estiver correta, mas que incrementa o peso quanto a saída da rede é menor que a saída esperada e decrementa o peso quando a saída da rede e maior que a saída esperada. A variação do peso da entrada i para o neurônio j é calculada aplicando a taxa de aprendizagem e o valor do erro j sobre a entrada i, conforme segue

$$\Delta w_{ij} = \eta x_i e_j \tag{4.7}$$

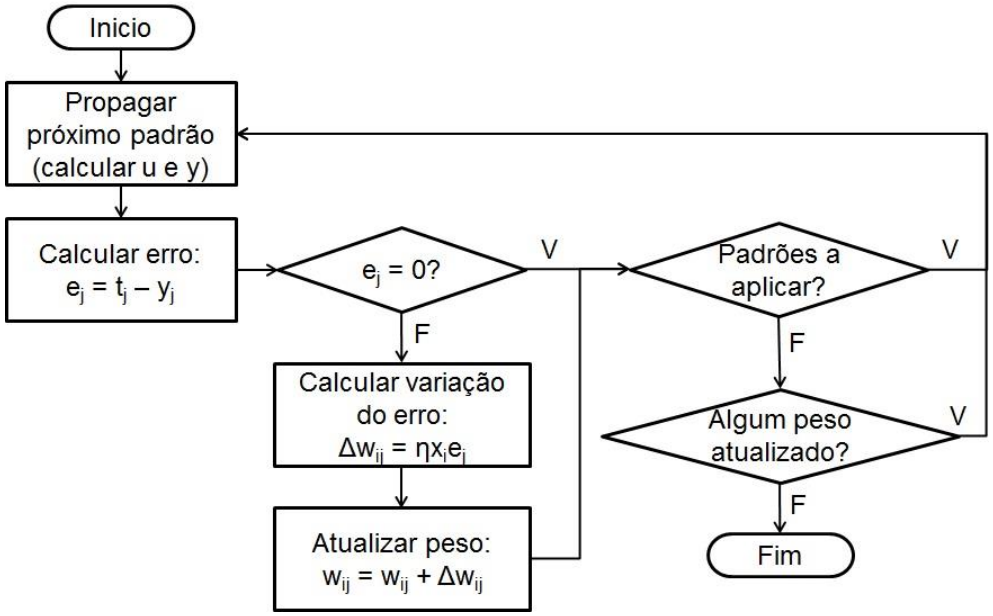
Sendo que:

- Δw_{ij}: variação do peso para o erro do neurônio j com a entrada i
- η: taxa de aprendizagem
- x_i: entrada i

- e_j: erro do neurônio j

A Figura 4.7 mostra o fluxograma do algoritmo de treinamento de uma rede neural Perceptron.

Figura 4.7 - Fluxograma do algoritmo de treinamento da Rede Neural Perceptron.



Fonte: Produção do autor.

Os passos do algoritmo de treinamento da Figura 4.7 são:

1. Iniciar os pesos com valores randômicos e pequenos ou com pesos iguais a zero;

2. Aplicar próximo padrão de entrada com sua saída desejada (t_j) e verificar a saída da rede (y_j);

3. Calcular o erro de saída dos neurônios: $e_j = t_j - y_j$;

4. Se erro diferente de zero:

    - Calcular a variação do peso: $\Delta w_{ij} = \eta x_i e_j$;

- Atualizar todos os pesos: $w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n)$;

5. Se ainda tem padrões a aplicar, volta ao passo 2;

6. Se algum padrão apresentou erro, voltar ao passo 2 com primeiro padrão.

A restrição das RNAs Perceptron de resolver somente problemas linearmente separáveis foi resolvida com as RNAs Perceptron de Múltiplas Camadas.

4.4.2. Redes Neurais Perceptron de Múltiplas Camadas

As redes neurais Perceptron de Múltiplas Camadas (MLP) são, como o próprio nome já diz, redes Perceptron com mais de uma camada. As redes neurais MLP também utilizam a regra de aprendizagem por correção de erro do paradigma supervisionado. Sua arquitetura tem pelo menos duas camadas que são acíclicas e totalmente conectadas. A regra de propagação é dada pelo produto interno das entradas ponderadas pelos pesos com adição do termo bias, mas deve considerar que a saída da camada anterior é a entrada da camada atual (HAYKIN, 2001). A regra de propagação é

$$u_j = \sum_{i=0}^{n} x_i w_{ij} \tag{4.8}$$

Sendo que:

- u_j: saída do combinador linear do neurônio j
- x_i: entrada i
- w_{ij}: peso para o erro do neurônio j com a entrada i

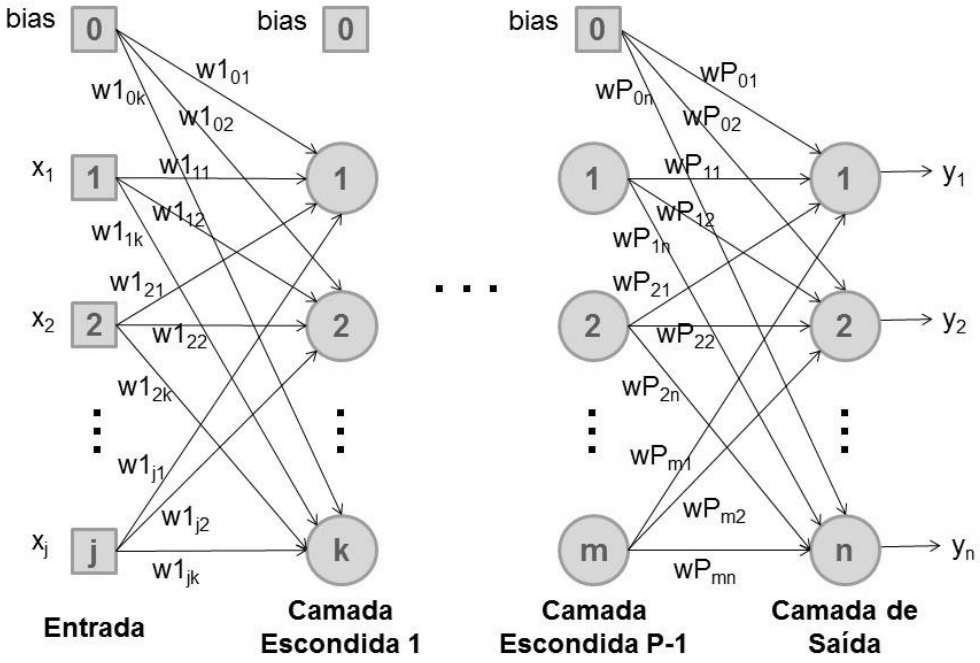
As redes neurais MLP geralmente usam a função de ativação não-linear sigmoide logística e seus valores de entrada e saída podem ser contínuos. A função sigmoide logística é

$$y_j = F(u_j) = \frac{1}{1+\exp(-\alpha u_i)} \tag{4.9}$$

Sendo que:

- y_j: saída do neurônio j
- $F(u_j)$: função de ativação do neurônio j
- u_i: saída do neurônio i
- exp: função exponencial, onde exp(x) = e^x (e ≡ 2,718281828)
- α: parâmetro de inclinação

Figura 4.8 - Rede Neural Perceptron de Múltiplas Camadas.



Fonte: Produção do autor.

Um exemplo da arquitetura de uma rede neural MLP é mostrada na Figura 4.8. Ela tem j entradas, P camadas e diferentes quantidades de neurônios em cada

camada, sendo, por exemplo, k neurônios na camada 1 e m neurônios na camada P-1 e n neurônios na camada de saída.

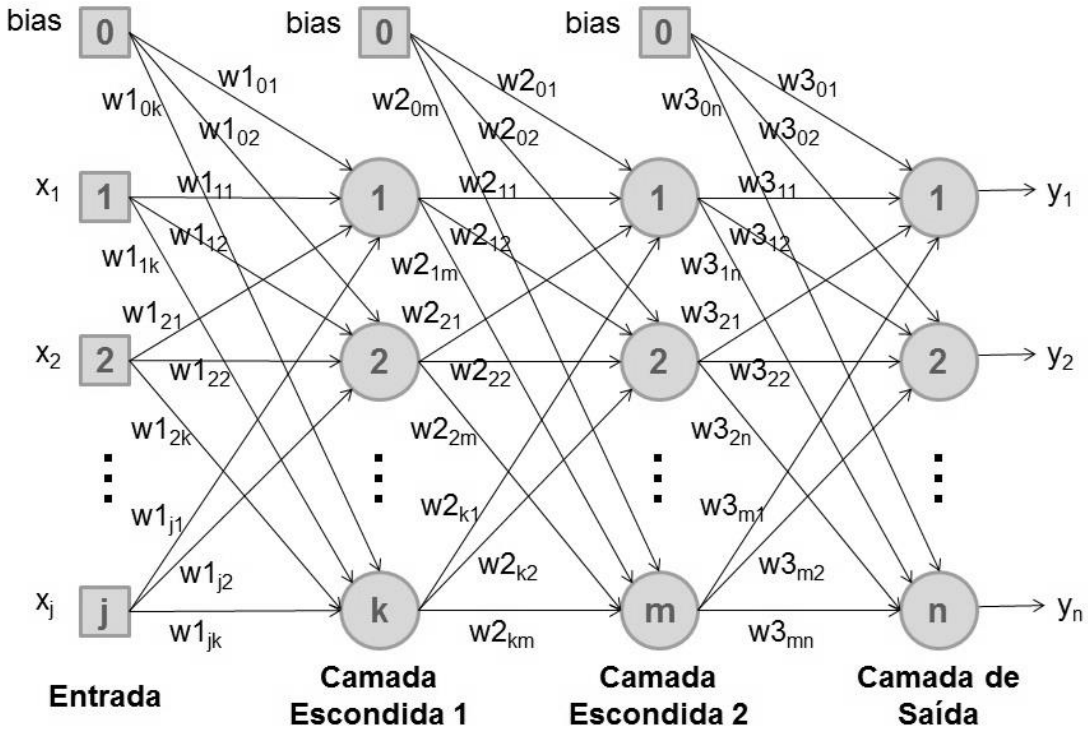
Como o número de camadas e o número de neurônios por camada pode ser variável, é necessário que esses parâmetros sejam ajustados da melhor maneira possível para que a rede neural resolva o problema proposto com o menor processamento possível. O número de neurônios da camada de saída depende da quantidade de saídas na solução buscada, mas a definição do número de camadas escondidas e de quantos neurônios terá cada camada é um problema empírico em que o especialista deve testar várias configurações para achar aquela mais adequada. Existem algumas tentativas de automatizar esse processo como o trabalho de Anochi (2015) que criou algoritmos para que as redes se autoconfigurassem, mas a maior parte dos trabalhos ainda usa o método manual que depende do especialista.

Devido à saída das camadas intermediárias não terem um valor esperado para servir como supervisor na correção dos erros, o grande desafio dessa rede foi encontrar um algoritmo de aprendizado para atualizar os pesos das camadas intermediárias. Esse problema foi resolvido pelo algoritmo de aprendizagem por retropropagação (*back-propagation*), cuja ideia central é que os erros da camada de saída sejam retro propagados para as camadas intermediárias. O algoritmo de retropropagação pode ser definido em duas fases:

- Propagação: as entradas se propagam pela rede até a camada de saída
- Retorno: os erros retornam atualizando os pesos desde a camada de saída até a primeira camada escondida

A Figura 4.9 mostra um exemplo de uma rede neural MLP com duas camadas escondidas.

Figura 4.9 - Rede Neural Perceptron de Múltiplas Camadas com duas camadas escondidas.



Fonte: Produção do autor.

A Figura 4.10 ilustra um exemplo de aplicação do algoritmo de retropropagação para um padrão do conjunto de treinamento. A aplicação do algoritmo para todos os padrões do treinamento é chamada de Época.

Essas fases de propagação e retropropagação se repetem para todos os padrões de treinamento até que o erro seja menor que um valor estipulado. A atualização dos pesos é feita de maneira semelhante àquela feita na rede neural Perceptron. O cálculo da variação dos pesos é

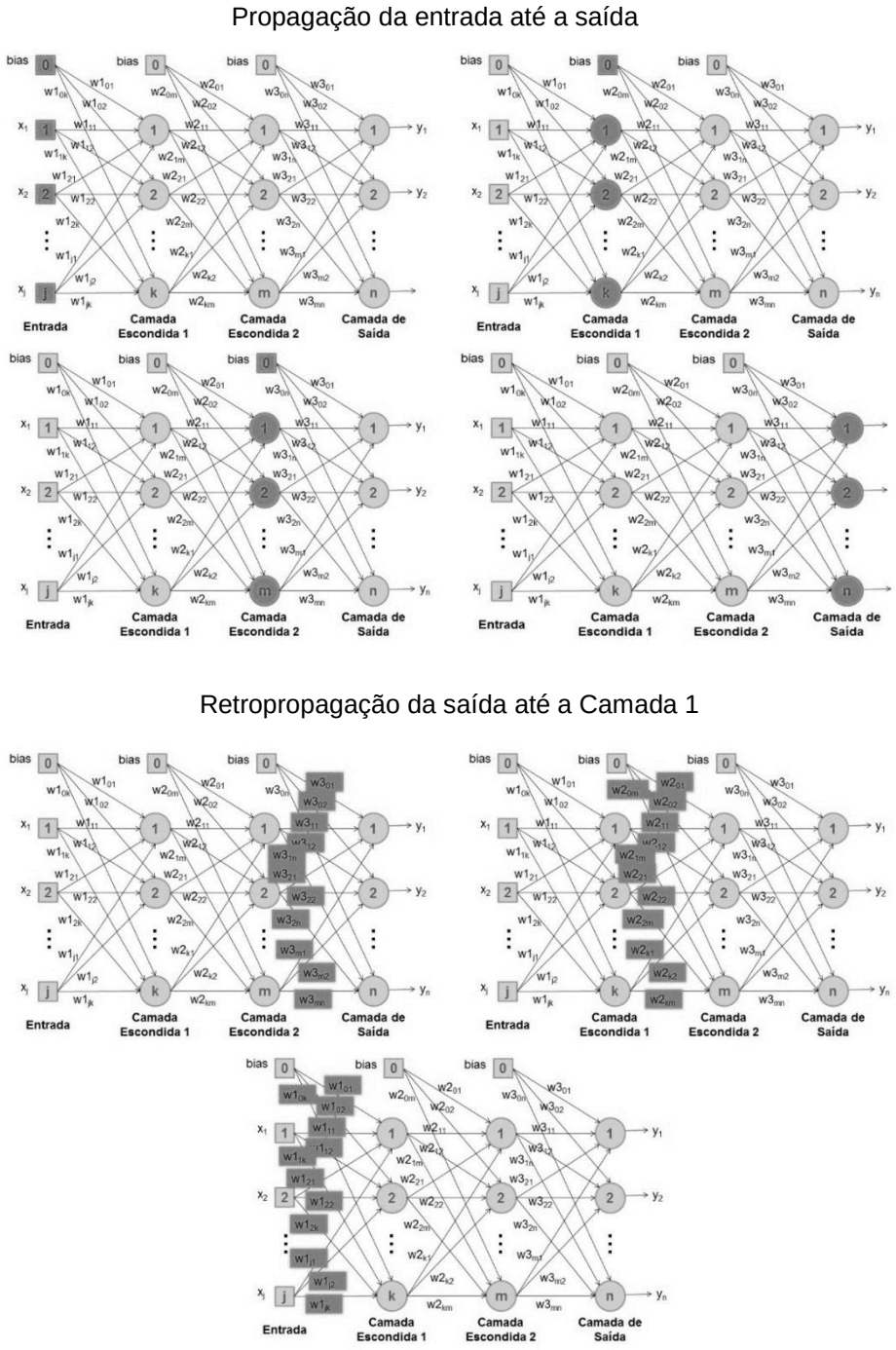
$$w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n) \qquad (4.10)$$

Sendo que:

- w_{ij}: peso do neurônio j para a entrada i
- n: número do passo no processo de ajuste dos pesos

- Δw_{ij}: variação do peso para o erro do neurônio j com a entrada i

Figura 4.10 - Exemplo de funcionamento do processo de aprendizagem em uma Rede Neural MLP com duas camadas escondidas.



Fonte: Produção do autor.

A equação da atualização dos pesos é

$$\Delta w_{ij} = \eta x_i \delta_j \quad (4.11)$$

Sendo que:

- Δw_{ij}: variação do peso para o erro do neurônio j com a entrada i
- η: taxa de aprendizagem
- x_i: entrada i
- δ_j: erro do neurônio j

Além da taxa de aprendizagem que tem uma influência muito grande para que seja alcançada a estabilidade e convergência do processo de aprendizagem, o termo momentum também pode ser adicionado ao processo de aprendizagem para acelerar o processo de treinamento, reduzindo o perigo de instabilidade. A atualização dos pesos com a utilização do termo momentum (α) fica assim

$$\Delta w_{ij} = \eta x_i \delta_j + \alpha[w_{ij}(n) - w_{ij}(n-1)] \quad (4.12)$$

Sendo que:

- Δw_{ij}: variação do peso para o erro do neurônio j com a entrada i
- η: taxa de aprendizagem
- x_i: entrada i
- δ_j: erro do neurônio j
- α: termo momentum
- w_{ij}: peso do neurônio j para a entrada i
- n: número do passo no processo de ajuste dos pesos

O termo momentum torna o processo de aprendizagem mais estável, além de evitar possíveis mínimos locais e diminuir as oscilações observadas quando a rede está próxima de um mínimo local.

Os erros da camada de saída precisam ser propagados para as camadas escondidas, assim, os erros são calculados multiplicando pela derivada da função de ativação a diferença entre a saída esperada e a saída da rede. O cálculo dos erros da camada de saída é

$$\delta_j = (t_j - y_j)F'(u_j) \tag{4.13}$$

Sendo que:

- δ_j: erro do neurônio j
- t_j: valor de saída esperado do neurônio j
- y_j: saída do neurônio j
- $F'(u_j)$: derivada da função de ativação de j

O cálculo do erro das camadas escondidas depende do erro da próxima camada e dos pesos que foram atualizados para essa camada. O cálculo dos erros das camadas escondidas é

$$\delta_j = F'(u_j)\sum_{k=1}^{n}\delta_k w_{jk} \tag{4.14}$$

Sendo que:

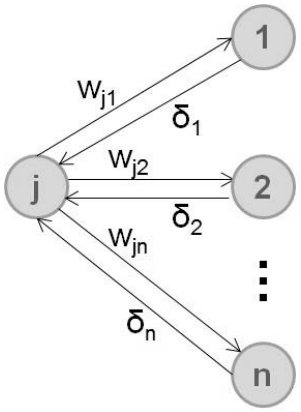
- δ_j: erro do neurônio j da camada atual
- $F'(u_j)$: derivada da função de ativação de j
- δ_k: erro do neurônio k da próxima camada

- w_{jk}: peso do neurônio j para o próximo neurônio k

- n: número de neurônios da próxima camada

A Figura 4.11 ilustra um exemplo do cálculo do erro para um neurônio j da camada escondida.

Figura 4.11 - Exemplo dos erros e pesos usados no cálculo do erro de um neurônio da camada escondida no algoritmo de retropropagação em uma Rede Neural MLP



Fonte: Produção do autor.

O cálculo da derivada da função de ativação é

$$F'(u_j) = \frac{\exp(-u_i)}{[1+\exp(-u_i)]^2} \quad (4.15)$$

Sendo que:

- $F(u_j)$: função de ativação do neurônio j

- u_i: saída do neurônio i

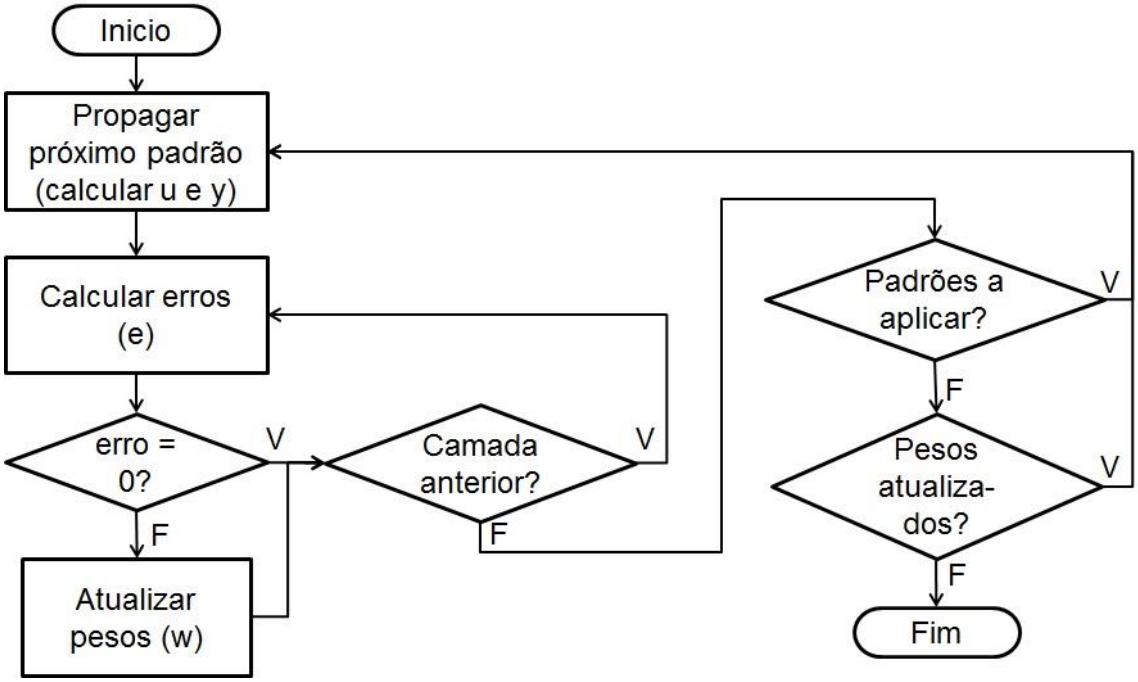
- exp: função exponencial, onde $\exp(x) = e^x$ ($e \equiv 2{,}718281828$)

O algoritmo de treinamento tem muitos passos semelhantes ao algoritmo da rede neural Perceptron, mas é preciso considerar que o processo deve ser repetido para cada uma das camadas. Os passos do algoritmo de treinamento da rede neural MLP são:

1. Iniciar os pesos com valores randômicos e pequenos ($|w| < 0,1$) ou com pesos iguais a zero;
2. Aplicar próximo padrão de entrada com sua saída desejada;
3. Calcular as saídas dos neurônios iniciando pela primeira camada escondida até a camada de saída;
4. Calcular o erro de cada neurônio da camada;
5. Se erro for maior que tolerância, atualizar todos os pesos;
6. Repetir os passos 4 e 5 iniciando pela camada anterior a camada de saída até a primeira camada escondida;
7. Se ainda tem padrões a aplicar, volta ao passo 2;
8. Se o erro de algum dos padrões é maior que a tolerância, voltar ao passo 2 com primeiro padrão.

A Figura 4.12 mostra o fluxograma do algoritmo de treinamento de uma rede neural MLP.

Figura 4.12 - Fluxograma do algoritmo de treinamento de uma Rede Neural MLP.



Fonte: Produção do autor.

A rede neural Perceptron de Múltiplas Camadas foi utilizada para identificação dos tipos de supernovas a partir do espectro, feita pelo CIntIa. Os parâmetros das RNAs usadas no classificador foram todos ajustados empiricamente com base na literatura consultada. O desenvolvimento do classificador, incluindo as RNAs, é apresentado no próximo capítulo.

5 DESENVOLVIMENTO DO CLASSIFICADOR INTELIGENTE DE SUPERNOVAS DO TIPO Ia (CIntIa)

O classificador automático desenvolvido neste trabalho foi denominado CIntIa, que é uma abreviatura de Classificador Inteligente de supernovas do tipo Ia. A criação deste nome se deve ao desenvolvimento do classificador estar ligado ao Projeto *Kunlun Dark Universe Survey Telescope* (KDUST) (CHINESE CENTER FOR ANTARCTIC ASTRONOMY, 2010) que tem interesse particular na observação de supernovas do tipo Ia para o estudo da expansão acelerada do universo.

O Projeto KDUST tem como uma de suas principais missões científicas investigar o fenômeno da expansão cósmica acelerada. Ele usará, entre outras técnicas, a medida de distâncias de luminosidade de supernovas do tipo Ia que podem fornecer restrições aos parâmetros da equação de estado da energia escura (ZHAO; ZHAN, *et al.*, 2011). As supernovas do tipo Ia tem características especiais que permitem medir as distâncias de luminosidade por ser o único tipo que tem sua origem na explosão termonuclear da estrela progenitora, enquanto todos os outros tipos têm sua origem em uma explosão por colapso do núcleo (TURATTO, 2003).

Fisicamente, o Projeto consiste na criação de um observatório astronômico em Kunlun, um dos pontos mais altos do Platô Antártico. As metas do projeto incluem a busca de planetas com características iguais às da Terra que estejam em órbita ao redor de outras estrelas, o estudo da origem do universo e da sua composição, e a observação do momento no qual as primeiras estrelas se formaram no universo (CHINESE CENTER FOR ANTARCTIC ASTRONOMY, 2010).

Os telescópios poderão capturar espectros de objetos com comprimentos de ondas de 3800 a 10000 angstroms e de 9500 a 25000 angstroms. Eles serão instalados para observações recorrentes de curvas de luz e observações

seletivas de espectros de supernovas, em especial espectros de supernovas tipo Ia.

Vale ressaltar que o estudo de supernovas tipo Ia é foco de muitas pesquisas, desde que foi descoberta sua importância no estudo da expansão acelerada do universo, entre elas, (RIESS; PRESS; KIRSHNER, 1996) (HILLEBRANDT; NIEMEYER, 2000) (BRANDT; TOJEIRO, *et al.*, 2010) (MILNE; BROWN, *et al.*, 2013) (KOBAYASHI; KEN'ICHI; HACHISU, 2015), inclusive no Brasil (LAGO, 2011). Comprovam a importância desses estudos, os trabalhos que resultaram no Prêmio Nobel em Física de 2011 para os pesquisadores Saul Perlmutter, Brian P. Schmidt e Adam G. Riess pela descoberta da expansão acelerada do universo através da observação de supernovas distantes do tipo Ia: (PERLMUTTER; ALDERING, *et al.*, 1999) e (RIESS; STROLGER, *et al.*, 2004).

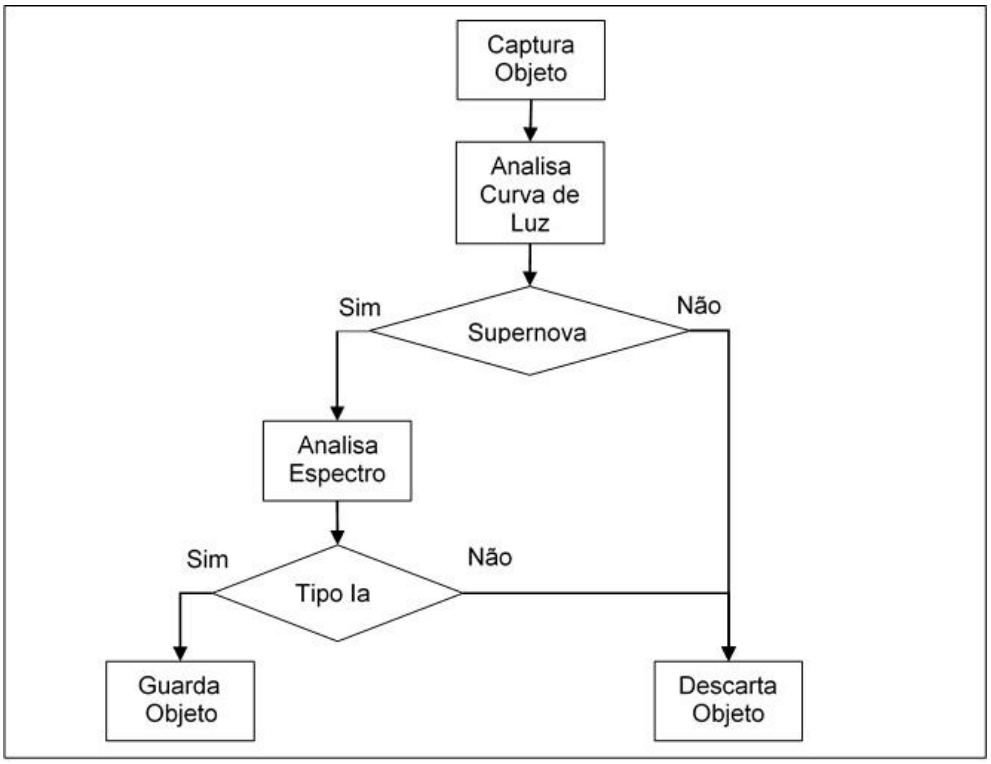
O Projeto KDUST tem a participação de grupos de astrônomos da China, Estados Unidos da América (EUA) e Austrália. O projeto é liderado pelo *Chinese Center for Antarctic Astronomy* (CCAA) e tem também a participação de pesquisadores do *National Astronomical Observatory of China, do Purple Mountain Observatory, do Nanjing Institute of Astronomical Optics Technology, do Polar Research Institute of China e do Institute of High Energy Physics* (CHINESE CENTER FOR ANTARCTIC ASTRONOMY, 2010).

A participação do Instituto Nacional de Pesquisas Espaciais (INPE) no Projeto KDUST surgiu como uma possibilidade através do desenvolvimento de ferramentas e análises de dados. Entre essas ferramentas propostas para o KDUST estava um sistema automático de classificação de supernovas que motivou o desenvolvimento do CIntIa.

O CIntIa deve ser instalado em um telescópio do KDUST, juntamente com um controlador automático que permite a manipulação do telescópio depois de detectada a supernova. O controlador deve manter o telescópio apontado para a supernova enquanto for necessário para captura de informações e,

periodicamente, voltar a apontar para essa supernova com objetivo de colher mais informações. A detecção automática da supernova será feita comparando a luminosidade da estrela em diferentes dias (BERNAT, 1986) (THIEBAUT; BOËR; ROQUES, 2002) (POST, 2015) e a identificação do tipo de supernova será feita pelo CIntIa a partir da análise do seu espectro. Esse sistema deve ser implementado nos telescópios do KDUST para identificar e estudar, em particular, supernovas do tipo Ia. A Figura 5.1 mostra um esquema proposto para o funcionamento de um telescópio quando um objeto é capturado.

Figura 5.1 - Esquema proposto para o funcionamento de um telescópio do Projeto KDUST quando um objeto é capturado.



Fonte: Produção do autor.

O desenvolvimento do controlador automático do telescópio e do sistema de identificação da supernova não faz parte do escopo deste trabalho, mas esse contexto é importante porque define o tipo de análise que deve ser feita. Como

o método de classificação automática de supernovas desenvolvido nesse trabalho precisa identificar o tipo da supernova logo depois da sua descoberta, não pode esperar 60 a 90 dias para fazer a análise da curva de luz. Assim, o classificador deve fazer a análise do espectro da supernova logo que seja capturado.

Portanto, utilizando a análise do espectro, a classificação feita neste trabalho já exclui a identificação dos tipos de supernovas escritos em letras maiúsculas (IIF, IIL e IIP), que são identificados pela análise da curva de luz. Mas, além da exclusão desses tipos, o CIntIa não identifica separadamente os tipos IIb, IIn e IIpec devido a existência de poucos espectros de supernovas desses tipos disponíveis nos dois bancos de espectros que foram utilizados neste trabalho, ou seja, agrupa esses tipos, identificando apenas o tipo II. Um dos bancos tem 1% de espectros dos tipos IIb e IIn e 3% de espectros do tipo IIpec. O outro banco tem 2% de espectros do tipo IIb e não tem espectros dos tipos IIn e IIpec. Logo, fazer a identificação separadamente desses subtipos pode levar a produção de resultados pouco significativos. Assim, a classificação de supernovas feita por esse método usa o esquema de classificação de supernovas de Giunti e Kim (2007), mostrado na Figura 2.10, mas identifica apenas os tipos chamados “clássicos”: Ia, Ib, Ic e II.

Os bancos de espectros utilizados no desenvolvimento e teste do CIntIa estão disponíveis na Web e tem os tipos das supernovas já identificados por astrônomos especialistas. O primeiro banco de espectros utilizado foi SUSPECT: The *Online Supernova Spectrum Archive*, hospedado no *Department of Physics and Astronomy da The University of Oklahoma* (HOGAN; PARRENT; FELDT, 2010). O segundo foi obtido no site da *Harvard-Smithsonian Center for Astrophysics* (CfA) (HARVARD-SMITHSONIAN CENTER FOR ASTROPHYSICS, 2010). Esses bancos são detalhados na próxima seção.

O CIntIa foi desenvolvido usando Redes Neurais Perceptron de Múltiplas Camadas (MLP), pois os dados já rotulados que foram obtidos nos bancos de espectros são propícios à sua utilização para problemas de classificação. Também porque redes neurais MLP têm um baixo custo computacional e, logo, consomem pouca energia, o que é relevante para seu uso nos telescópios do Projeto KDUST devido a sua localização no Platô Antártico. Além disso, a implementação das redes neurais MLP pode ser feita em hardware, caso necessário.

Todas as redes neurais MLP implementadas neste trabalho usaram os valores tradicionais para seus parâmetros. Esses valores foram atribuídos no início do treinamento e não houve necessidade de mudança devido aos treinamentos terem convergido rapidamente em todas as configurações testadas. Assim a função de ativação utilizada foi a Sigmoide Logística e os valores dos outros parâmetros usados em todas as redes neurais foram:

- Taxa de Aprendizagem = 0,5
- Momentum = 0,3
- Bias = 1
- Erro tolerado = 0,001

Foram desenvolvidas quatro versões do CIntIa e, apesar de três delas não terem obtido os resultados esperados, são descritas aqui em sua ordem cronológica como forma de mostrar o caminho que foi percorrido para chegar à versão atual. Apesar de todas as versões terem sido desenvolvidas usando redes neurais MLP, cada uma das versões do CIntIa analisa diferentes partes do espectro e utiliza diferentes quantidades de RNAs.

Quanto às partes do espectro, a 1ª versão usa todo o espectro da supernova para treinamento da rede neural e a 2ª versão usa a divisão em caixas de Harutyunyan (2008). Na 3ª versão foram usados intervalos próximos às linhas

dos elementos hidrogênio, hélio e silício. A 4ª versão, que é a versão atual, analisa os mesmos intervalos do espectro que são examinados pelo especialista humano na sua análise visual.

Quanto ao número de redes neurais, a 1ª versão usa uma única rede neural para identificar todos os tipos de supernova. A 2ª e a 3ª versões usam uma rede neural para identificar a presença ou ausência de cada um dos elementos que diferenciam os tipos de supernovas: hidrogênio, silício e hélio. A versão atual (4ª versão) usa uma rede neural para cada tipo "clássico" de supernova, ou seja, cada espectro é analisado por todas as redes neurais, onde cada rede informa se a supernova é ou não de um determinado tipo.

Detalhes do desenvolvimento do CIntIa são mostrados nas próximas seções deste capítulo, mas os resultados dos testes realizados são mostrados somente no próximo capítulo. A seção 5.1 apresenta os bancos de espectros utilizados no treinamento e teste das redes neurais e descreve como foi feita a seleção de espectros. A seção 5.2 apresenta o pré-processamento realizado. A seção 5.3 descreve o desenvolvimento das três primeiras versões do CIntIa e a seção 5.4 descreve o desenvolvimento da 4ª versão, que é a versão atual do CIntIa.

5.1. Bancos de espectros

Na pesquisa por bancos de espectros públicos para utilização no desenvolvimento do CIntIa foram encontrados dois bancos disponíveis na Web: o SUSPECT: The *Online Supernova Spectrum Archive* (HOGAN; PARRENT; FELDT, 2010) e o banco obtido no site do *Harvard-Smithsonian Center for Astrophysics* (CfA) (HARVARD-SMITHSONIAN CENTER FOR ASTROPHYSICS, 2010). Ambos os bancos são formados por espectros capturados em diferentes observatórios e por diferentes instrumentos. Todos os seus espectros já estão rotulados, ou seja, foram todos classificados em um dos tipos de supernovas por astrônomos especialistas.

Esses dois bancos de espectros foram utilizados no desenvolvimento de diferentes versões do CIntIa por não terem sido obtidos ao mesmo tempo. No desenvolvimento das três primeiras versões somente o banco SUSPECT estava disponível, pois as informações que permitiram obter o banco CfA ficaram disponíveis somente em 2014 (MODJAZ; BLONDIN, *et al.*, 2014), quando já tinham sido desenvolvidas essas três primeiras versões. O banco de espectros CfA foi utilizado na versão atual do CIntIa porque ele tem mais espectros disponíveis e mais informações sobre os espectros, como, por exemplo, a dispersão espectral. Vale ressaltar também, que os dois bancos de espectros foram escolhidos devido à qualidade das suas informações e facilidade de obtenção do conjunto de espectros, mas, além deles, espectros de outros bancos podem ser obtidos individualmente (YARON; GAL-YAM, 2012).

Para a utilização desses bancos foi necessário fazer uma seleção dos espectros com determinadas características que é descrita nas próximas duas seções para cada um dos bancos de espectros.

5.1.1. Banco de espectros SUSPECT

O banco de espectros de supernovas SUSPECT: The Online Supernova Spectrum Archive está hospedado no *Department of Physics and Astronomy da The University of Oklahoma* e tem um total de 1741 espectros de 185 supernovas capturados no período de 1989 a 2006 (HOGAN; PARRENT; FELDT, 2010).

Para utilização no CIntIa os espetros precisam seguir algumas restrições quanto ao comprimento de onda e período em relação a luz máxima para que seja possível identificar as linhas dos elementos que permitem a identificação dos tipos “clássicos” de supernovas. Entre os espectros disponíveis nesse banco foram selecionados 331 espectros de diferentes tipos de supernovas seguindo restrições:

- Comprimento de onda: de 3800 a 6800 angstroms;
- Período em relação à luz máxima: de -14 a +14 dias.

O conjunto de 331 espectros selecionados foi subdividido em dois, ficando 80% dos espectros para treinamento e 20% dos espectros para testes das redes neurais. Assim, foram utilizados 265 espectros para treinamento e 66 espectros para testes. A divisão dos conjuntos seguiu também a restrição: espectros de cada supernova tem que estar todos em um só conjunto.

A Tabela 5.1 mostra os tipos dos 331 espectros selecionados. A Tabela 5.2 mostra os tipos dos 265 espectros usados no treinamento e a Tabela 5.3 mostra os tipos dos 66 espectros usados para teste.

Tabela 5.1 - Tipos de todos os espectros do banco SUSPECT selecionados para Treinamento e Teste das Redes Neurais

Tipo	**Nº Supernovas**	**Nº Espectros**
Ia	28	244
Ib	5	11
Ic	9	42
II	14	34
TOTAL	**56**	**331**

Fonte: Produção do autor.

Tabela 5.2 - Tipos de todos os espectros do banco SUSPECT selecionados para Treinamento das Redes Neurais

Tipo	**Nº Supernovas**	**Nº Espectros**
Ia	18	195
Ib	4	9
Ic	6	34
II	9	27
TOTAL	**37**	**265**

Fonte: Produção do autor.

Tabela 5.3 - Tipos de todos os espectros do banco SUSPECT selecionados para Teste das Redes Neurais

Tipo	**Nº Supernovas**	**Nº Espectros**
Ia	10	49
Ib	1	2
Ic	3	8
II	5	7
TOTAL	**19**	**66**

Fonte: Produção do autor.

Todos os espectros selecionados para treinamento e teste foram submetidos a um mesmo pré-processamento.

5.1.2. Banco de espectros do CfA

O banco de espectros disponibilizado pelo The CfA *Supernova Group*[1] do Centro de Astrofísica *Harvard-Smithsonian da Universidade de Harvard* é composto por 2603 espectros de 462 supernovas do tipo Ia obtidos no período de 1993 a 2008 (BLONDIN; MATHESON, *et al.*, 2012) e por 645 espectros de 73 supernovas dos tipos Ib, Ic e II obtidos no período de 2004 a 2009 (MODJAZ; BLONDIN, *et al.*, 2014).

A seleção dos espectros usados para treinamento e teste considerou um intervalo menor de dias com o objetivo de selecionar os espectros mais próximos da luz máxima. Os critérios de seleção de espectros usados nesse banco de espectros foram:

- Comprimento de onda: de 3800 a 7400 angstroms
- Dias desde a luz máxima: de -3 a +7 dias
- Dispersão espectral: menor ou igual a 1,5 angstroms por pixel

Usando esses critérios foram selecionados 559 espectros de 192 supernovas do tipo Ia e 90 espectros de 29 supernovas dos outros tipos mostrados na Tabela 5.4.

Os espectros selecionados também foram divididos em dois conjuntos: Treinamento (80%) e Teste (20%). O conjunto de Treinamento, com o objetivo de fazer a validação no treinamento da Rede Neural, foi dividido em dois subconjuntos: Estimação (80%) e Validação (20%) (HAYKIN, 2001). A divisão dos conjuntos também seguiu a restrição: espectros de cada supernova tem que estar todos em um só conjunto.

[1] Esta pesquisa fez uso do CfA Supernova Archive, que é financiado em parte pela National Science Foundation através AST concessão 0.907.903.

Tabela 5.4 - Tipos de todos os espectros do banco CfA selecionados para Treinamento e Teste das Redes Neurais

Tipo	**Nº Supernovas**	**Nº Espectros**
Ia	192	559
Ib	12	33
Ic	12	44
II	5	13
TOTAL	**221**	**649**

Fonte: Produção do autor.

Foram selecionados 531 espectros (80% do TOTAL) de 178 supernovas para Treinamento, cujos tipos são mostrados na Tabela 5.5.

Tabela 5.5 - Tipo dos espectros do banco CfA selecionados para Treinamento das Redes Neurais

Tipo	**Nº Supernovas**	**Nº Espectros**
Ia	154	453
Ib	10	28
Ic	10	38
II	4	12
TOTAL	**178**	**531**

Fonte: Produção do autor.

Os 531 espectros selecionados para treinamento foram divididos em estimação e validação. Foram selecionados 451 espectros (80% do Treinamento) de 144 supernovas para Estimação, cujos tipos são mostrados na Tabela 5.6. Foram selecionados 80 espectros (20% do Treinamento) de 34 supernovas para Validação, cujos tipos são mostrados na Tabela 5.7.

Tabela 5.6 - Tipo dos espectros do banco CfA selecionados para Estimação do Treinamento das Redes Neurais

Tipo	**Nº Supernovas**	**Nº Espectros**
Ia	124	379
Ib	8	24
Ic	8	36
II	4	12
TOTAL	**144**	**451**

Fonte: Produção do autor.

Tabela 5.7 - Tipo dos espectros do banco CfA selecionados para Validação do Treinamento das Redes Neurais

Tipo	**Nº Supernovas**	**Nº Espectros**
Ia	30	74
Ib	2	4
Ic	2	2
II	0	0
TOTAL	**34**	**80**

Fonte: Produção do autor.

Como pode ser observado na Tabela 5.7, devido a pequena quantidade de supernovas do tipo II, o conjunto de validação não contou com supernovas do tipo II.

Para o conjunto de Teste foram selecionados 118 espectros (20% do TOTAL) de 43 supernovas, cujos tipos são mostrados na Tabela 5.8.

Tabela 5.8 - Tipo dos espectros do banco CfA selecionados para teste das Redes Neurais

Tipo	**Nº Supernovas**	**Nº Espectros**
Ia	38	106
Ib	2	5
Ic	2	6
II	1	1
TOTAL	**43**	**118**

Fonte: Produção do autor.

Todos os espectros selecionados para Treinamento e Teste foram submetidos a um mesmo pré-processamento.

5.2. Pré-processamento

O pré-processamento realizado foi o mesmo para os dois bancos de espectros utilizados na construção e teste das diferentes versões do CIntIa. Foi realizado o pré-processamento nos espectros seguindo os passos e os parâmetros usados em Harutyunyan (2008), com exceção da divisão em caixas. Seguem as operações que foram realizadas no pré-processamento:

5.2.1. Fazer a correção de redshift (z)

O redshift de cada supernova estava disponível na documentação dos bancos de espectros. A correção foi feita para cada ponto do espectro pela equação:

$$\lambda_0 = \lambda/(z+1) \quad (5.1)$$

Sendo que:

- λ: comprimento de onda observada
- λ_0: comprimento de onda se o objeto estivesse em repouso
- z: *redshift*

5.2.2. Suavizar o gráfico com parâmetro de 70 angstroms

A suavização foi feita usando a função *smooth do* MATLAB, conforme segue:

$$Z = smooth(Y, SPAN) \quad (5.2)$$

Sendo que:

- Z: vetor resultante da suavização
- Y: vetor com os valores do fluxo para cada comprimento de onda
- SPAN: número de pontos usado para computar cada elemento de Z

O método usado nessa função foi o padrão do MATLAB que é o *Moving Average* e o valor do parâmetro *SPAN* foi de 70 angstroms.

5.2.3. Interpolar os espectros a cada oito angstroms

A interpolação feita usou a função *interp1* do MATLAB que tem a seguinte sintaxe:

$$Vq = interp1(X, V, Xq, METHOD) \quad (5.3)$$

Sendo que:

- Vq: vetor resultante da interpolação
- X: vetor com os valores de comprimento de onda do espectro
- V: vetor com os valores do fluxo para cada comprimento de onda
- Xq: vetor com os novos valores de comprimento de onda do espectro
- *METHOD*: método utilizado neste trabalho foi a interpolação cúbica *('cubic'*).

5.2.4. Normalizar para vetor de magnitude um

A normalização foi feita utilizando as funções *max* e *min* do MATLAB que, respectivamente, encontram o valor máximo e o valor mínimo de um vetor. A equação usada na normalização foi:

$$Vn = (\mathrm{V} - \min(\mathrm{V})) \,/\, (\max(\mathrm{V}) - \min(\mathrm{V})) \quad (5.4)$$

Sendo que:

- Vn: vetor resultante da normalização
- V: vetor com os valores do fluxo para cada comprimento de onda

5.2.5. Automatização do pré-processamento

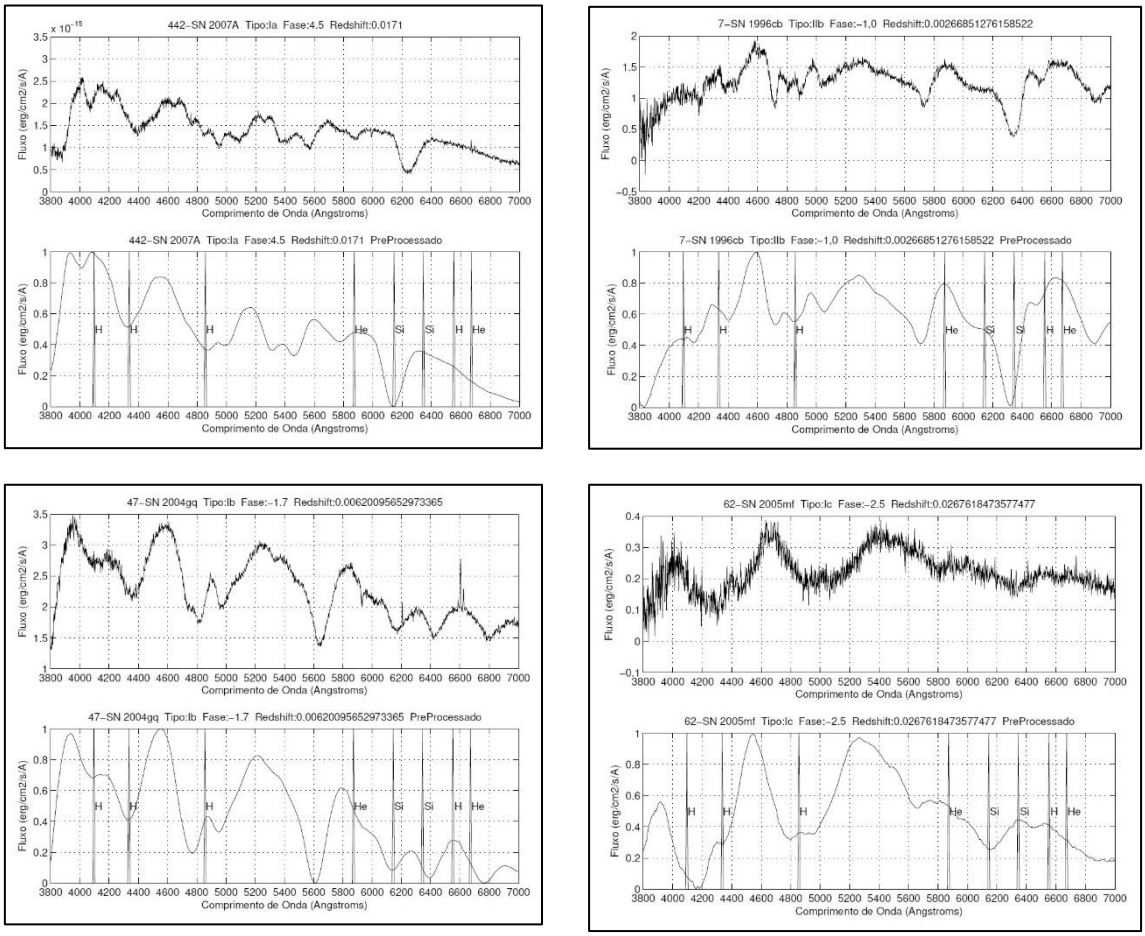
Além da divisão dos espectros nos conjuntos de treinamento, validação e teste, e da realização desses quatro passos no pré-processamento, também foi feita a preparação necessária para que os arquivos com os dados estivessem no formato adequado para servirem de entrada às redes neurais.

Todos os passos do pré-processamento foram automatizados em programas desenvolvidos usando o MATLAB. Os quatro primeiros passos foram implementados no programa “CorrigeRedshiftSuavizaInterpolaNormaliza.m” cujo código está disponível no Apêndice A. o programa permite que os parâmetros do pré-processamento possam ser escolhidos, facilitando os testes para diferentes valores de configuração de suavização e interpolação. Também permite escolher o intervalo em angstrom para os espectros que serão pré-processados.

O programa tem como primeira entrada um arquivo “.txt” que contém as seguintes informações sobre cada espectro que será pré-processado: nome da supernova, data do espectro, comprimento de onda mínimo e máximo do espectro, tipo, fase e redshift da supernova. Esse arquivo tem também o nome do arquivo que deve ser lido para se obter os valores de comprimento de onda (em angstrom) e fluxo (erg/cm2/s/A) do espectro.

A segunda entrada do programa está em um laço que lê os valores de comprimento de onda e fluxo de cada espectro. Nos dados de cada arquivo lido são feitos os quatro passos do pré-processamento e são gerados dois arquivos: um arquivo “.csv” com os valores pré-processados e um arquivo “.ps” onde são plotados os gráficos dos espectros para posterior análise visual. O arquivo “.csv” tem os dados de um espectro por linha que é o formato utilizado como entrada para as redes neurais depois de alguns ajustes. A Figura 5.2 mostra exemplos de quatro espectros antes e depois do pré-processamento.

Figura 5.2 – Exemplos de quatro espectros antes e depois do pré-processamento



Fonte: Produção do autor.

O arquivo “.csv” resultante do pré-processamento foi usado na rede neural da 1ª versão do CIntIa que analisava todo o espectro, mas, como nas outras versões as redes neurais analisavam apenas partes do espectro, foram criados mais programas para selecionar as partes do espectro usadas. A seleção para a 2ª versão do CIntIa foi feita manualmente, mas para a 3ª e 4ª versão foram desenvolvidos no MATLAB, respectivamente os programas “CriaCSVcomLinhasElementos.m” e “CriaCSVcomIntervalosDoEspectro.m”, que também estão no Apêndice A.

Os dois programas têm as mesmas entradas e saídas, somente são diferentes na maneira como é feita a seleção das colunas dos dados. Ambos recebem como entrada o arquivo ".csv" resultante do pré-processamento feito pelo programa "CorrigeRedshiftSuavizaInterpolaNormaliza.m" e selecionam as colunas de acordo com os parâmetros da versão do CIntIa. Depois de selecionar as colunas os programas dividem os espectros nos conjuntos treinamento, validação e teste, seguindo os percentuais definidos para cada conjunto e a regra de que os espectros de uma supernova têm que estar todos no mesmo conjunto. A saída dos programas são três arquivos ".csv", cada um com seu conjunto de espectros para treinamento, validação ou teste.

Esses arquivos ".csv" são utilizados como entrada das redes neurais nas diferentes versões do CIntIa que foram desenvolvidas. A descrição de cada uma das versões está nas próximas seções.

5.3. Versões anteriores do CIntIa

Antes da versão atual do CIntIa foram desenvolvidas outras três versões que usaram o mesmo banco de espectros de supernovas que foi submetido ao mesmo pré-processamento. O banco de espectros usado foi o SUSPECT: *The Online Supernova Spectrum Archive* (HOGAN; PARRENT; FELDT, 2010), do qual foram selecionados 331 espectros sendo que 265 espectros foram usados no treinamento e 66 espectros foram usados nos testes das redes neurais. Todas as versões utilizaram Redes Neurais Perceptron de Múltiplas Camadas (MLP) para classificação das supernovas.

As descrições das três primeiras versões do CIntIa são mostradas nas próximas seções na sua sequência temporal porque as observações feitas nos testes de determinada versão foram usadas para tentar melhorar os resultados da próxima versão. No entanto, apesar desses resultados dos testes feitos em cada versão terem sido usados como motivação para a próxima versão, eles

são mostrados somente no próximo capítulo, pois o foco deste capítulo é descrever o classificador.

5.3.1. 1ª versão do CIntIa

Nessa versão foi criada uma única rede neural MLP para identificar todos os tipos “clássicos” de supernovas, analisando todo o espectro no intervalo de 3800 a 6800 angstrons. Como esse intervalo foi interpolado a cada oito angstroms, a rede neural teve 376 entradas. Como a saída é a classificação em um dos quatro tipos clássicos, a camada de saída da rede neural tem dois neurônios com saída binária que combinados resultam em uma das quatro classes possíveis: 00 – tipo Ia; 01 – tipo Ib; 10 – tipo Ic; e 11 – tipo II.

A partir dessa primeira versão foi observado que algumas partes do espectro não eram utilizadas para classificar os tipos de supernovas e foi construída uma nova versão para analisar apenas partes relevantes do espectro.

5.3.2. 2ª versão do CIntIa

Essa versão do CIntIa usa a proposta de divisão do espectro da supernova nas onze caixas de Harutyunyan (2008) que têm tamanhos diferentes de acordo com os picos ou vales do espectro que são observados para indicar a presença ou ausência dos elementos usados para classificação da supernova. A Tabela 5.9 mostra o intervalo e as características do espectro de cada uma das onze caixas.

Nessa versão foram utilizadas apenas algumas caixas em cada etapa da classificação de acordo com o elemento que se está buscando determinar a presença ou ausência. Assim, as caixas utilizadas para a detecção de um determinado elemento são aquelas que contêm as linhas desse elemento.

Tabela 5.9 - Caixas com os intervalos e as características espectrais das supernovas

Caixas	**Intervalo**	**Características espectrais**		
	Å	**Ia**	**Ib/c**	**II**
1	3504 - 3792	Ca II	Ca II	Ca II
2	3800 - 4192	Si II, Ca II	Ca II	Ca II, H_{δ}
3	4200 - 4576	Mg II, Fe II	Fe II	Mg II, Fe II, H_{γ}
4	4584 - 4936	Fe II	Fe II	Fe II, H_{β}
5	4944 - 5192	Fe II	Fe II	Fe II
6	5200 - 5592	S II	S II, O I	S II
7	5600 - 5896	Si II, Na I	Na I, He I	Si II, Na I
8	5904 - 6296	Si II	He I	Si II
9	6304 - 6800	Fe II	Si II, He I	O I, H_{α}
10	6808 - 7904	O I	O I	O I
11	7912 - 9000	Ca II	Ca II	Ca II

Fonte: Adaptada de Harutyunyan (2008).

O espectro foi dividido em apenas oito caixas (de 2 a 9), sendo que as caixas 1, 10 e 11 não são usadas, pois não contém os elementos necessários para a classificação e o intervalo dos espectros usados não contempla os valores extremos dessas caixas que estão abaixo de 3800 (caixa 1) e acima de 6800 angstroms (caixas 10 e 11).

Foram criadas três redes neurais MLP para verificar a presença dos elementos hidrogênio, hélio e silício que determinam os tipos das supernovas, sendo que cada rede neural busca verificar a presença de um elemento diferente.

A utilização das redes neurais para análise de cada espectro deve ser feita na mesma sequência usada no esquema da Figura 2.10 (GIUNTI; KIM, 2007) para identificação dos tipos. A identificação do elemento seguinte será feita somente sobre o conjunto de espectros que foram selecionados pela rede neural anterior. A ordem de utilização das redes neurais pelo CIntIa deve ser:

- 1ª Rede Neural: hidrogênio
- 2ª Rede Neural: silício
- 3ª Rede Neural: hélio

Assim, a 1ª rede neural (H) identifica que o espectro é do tipo II em caso de presença de hidrogênio e seleciona o espectro para verificação da 2ª rede neural em caso de ausência de hidrogênio. A 2ª rede neural será usada somente no espectro que foi identificado com ausência de hidrogênio pela 1ª rede neural. Caso a 2ª rede neural verifique a presença de silício, classifica a supernova como tipo Ia, caso verifique sua ausência seleciona o espectro para verificação da 3ª rede neural. A 3ª rede neural somente será usada no espectro que foi identificado com ausência de hidrogênio pela 1ª rede neural e com ausência de silício pela 2ª rede neural. Caso a 3ª rede neural verifique que o espectro é rico em hélio, classifica a supernova como tipo Ib, caso verifique que o espectro é pobre em hélio, classifica a supernova como tipo Ic.

As caixas que são utilizadas por cada rede neural variam de acordo com o elemento que se deseja identificar. Um determinado subgrupo de caixas é usado para cada elemento, conforme a linha desse elemento esteja ou não presente nessa caixa. Seguem as caixas usadas por cada rede neural:

- 1ª Rede Neural (H): caixas 2, 3, 4 e 9;
- 2ª Rede Neural (Si): caixas 8 e 9;
- 3ª Rede Neural (He): caixas 2, 7, 8 e 9;

A 1ª rede neural, construída para separar as supernovas do tipo I das supernovas do tipo II utilizando as caixas 2, 3, 4 e 9, tem 206 entradas, devido aos intervalos de cada caixa e a interpolação a cada oito angstroms. Foram usados 265 espectros para treinamento e 66 espectros para teste dessa rede neural.

A 2ª rede neural foi construída para separar as supernovas do tipo Ia das supernovas do tipo Ib e Ic utilizando as caixas 8 e 9. Ela tem 113 entradas correspondentes aos intervalos dessas caixas. Foram usados 238 espectros para treinamento e 59 espectros para teste porque os espectros das supernovas do tipo II foram excluídos dos conjuntos de treinamento e teste, uma vez que essa rede analisa apenas supernovas do tipo I para separar o tipo Ia dos tipos Ib e Ic.

A 3ª rede neural foi construída para separar as supernovas do tipo Ib das supernova do tipo Ic utilizando as caixas 2, 7, 8 e 9. Ela tem 202 entradas correspondentes aos intervalos dessas caixas. Devido a exclusão dos espectros das supernovas Ia dos conjuntos de treinamento e teste, somente foram usados 43 espectros para treinamento e 10 espectros para testes dessa rede neural.

Apesar do número de entradas ser diferente para cada rede neural, a camada de saída tem sempre um neurônio em todas elas, pois sua saída é binária, informando se o elemento está presente (1) ou não está presente (0) naquele espectro.

A partir da análise dessa segunda versão surgiu a ideia de ser mais preciso na identificação do elemento e a próxima versão do CIntIa considera um intervalo próximo das linhas de cada elemento.

5.3.3. 3ª versão do **CIntIa**

Nessa versão foi criada uma rede neural MLP para cada elemento a ser identificado usando o intervalo do espectro que contém a linha de elemento no centro. Esse intervalo foi testado para alguns tamanhos diferentes, mas aquele que obteve o melhor resultado foi de -100 a +100 angstroms a partir da linha do elemento no espectro. Para identificação das linhas dos elementos foram utilizados os seguintes valores.

- Hidrogênio: série de Balmer mostrada na Tabela 5.10

Tabela 5.10 - Série de Balmer

Cor	**Nome**	**λ (em angstroms)**
Vermelho	Hα	6562,8
Verde	Hβ	4861,3
Azul	Hγ	4340,5
Violeta	Hδ	4101,7

Fonte: Eisberg e Resnick (1979)

- Silício:
  - Pico em 6150 angstroms
  - Vale em 6355 angstroms

- Hélio:
  - He I em 5876 angstroms
  - He I em 6678 angstroms

Da mesma forma que foi feito na 2ª versão, foram criadas três redes neurais MLP para verificar a presença dos elementos hidrogênio, hélio e silício que determinam os tipos das supernovas de acordo com o esquema da Figura 2.10 (GIUNTI; KIM, 2007). Também como na versão anterior a saída é binária, informando se o elemento está presente (1) ou não está presente (0) naquele espectro.

A 1ª rede neural foi construída para separar as supernovas do tipo I das supernovas do tipo II, a partir da identificação do hidrogênio, utilizando as quatro linhas da série de Balmer. Ela tem 104 entradas, devido aos intervalos de cada linha e a interpolação a cada oito angstroms. Foram usados 265 espectros para treinamento e 66 espectros para teste das redes neurais.

A 2ª rede neural foi criada para separar as supernovas do tipo Ia das supernovas dos tipos Ib e Ic, a partir da identificação do silício, utilizando as linhas de pico e vale. Ela tem 52 entradas, devido ao silício ter somente duas linhas nos espectros. Devido a essa rede fazer a separação de subtipos das supernovas do tipo I, os espectros do tipo II foram excluídos. Foram utilizados 238 espectros para treinamento e 59 espectros para teste.

A 3ª rede neural foi criada para separar as supernovas do tipo Ib das supernovas dos tipos Ic, a partir da identificação das linhas de hélio. Ela tem 52 entradas, correspondentes aos intervalos das duas linhas do hélio. Devido a essa rede fazer a separação das supernovas do tipo Ib das supernovas do tipo Ic, os espectros dos outros tipos de supernovas foram excluídos. Foram utilizados 43 espectros para treinamento e 10 espectros para teste.

As observações feitas nessa 3ª versão do CIntIa levaram construção da próxima versão com os intervalos dos elementos mais próximos possível dos intervalos usados pelo especialista humano. Assim, as estratégias utilizadas nessas três versões do CIntIa serviram como base para construção da sua versão atual.

5.4. Versão atual do CIntIa

As configurações usadas nas outras versões do CIntIa mostraram que usar somente uma rede neural para identificar todos os tipos não é uma boa estratégia, então foi usada nessa versão a ideia das duas últimas versões de construir várias redes neurais, uma para cada verificação necessária. No entanto, a utilização de várias redes neurais em sequência não se mostrou uma boa ideia na 2ª e 3ª versões, assim surgiu a ideia de criar uma rede neural independente para identificar diretamente cada um dos tipos “clássicos”.

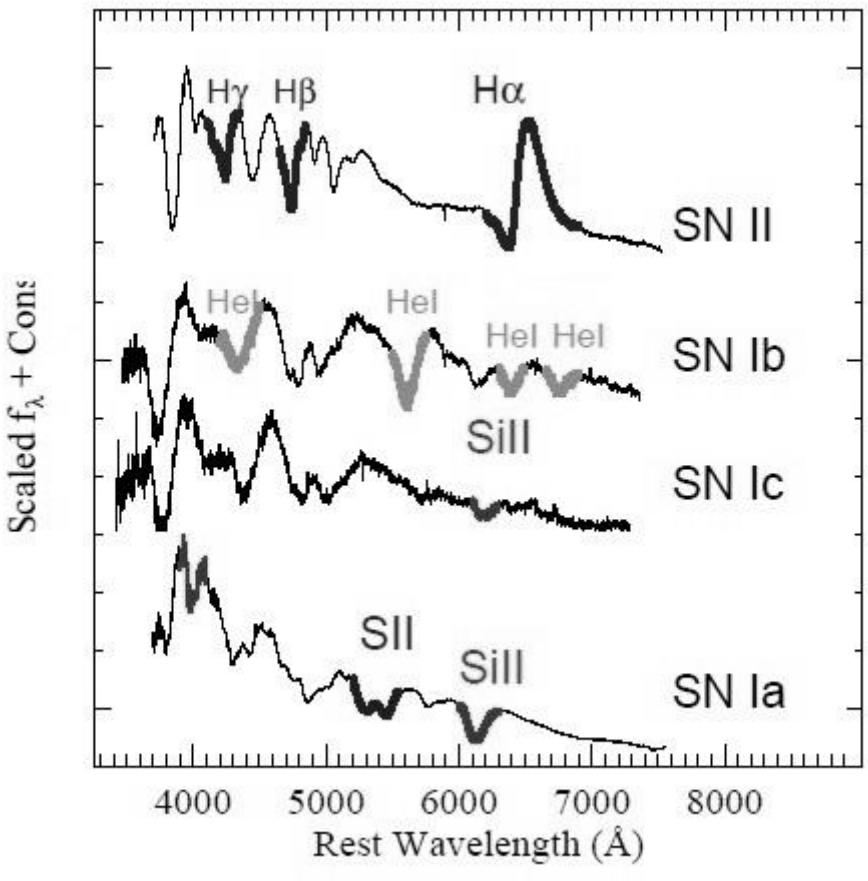
A definição de quais intervalos usar para identificação de cada um desses tipos de supernova considerou que os resultados não foram muito bons devido a definição muito rigorosa dos limites dos intervalos usados nas 11 caixas ou nas linhas de elementos. Assim, surgiu a ideia de utilizar sem um limite muito rigoroso a mesma área analisada pelo especialista humano na sua análise visual, definindo os intervalos a serem utilizados a partir das marcações feitas pelos especialistas nos espectros.

A Figura 5.3 mostra um exemplo dos intervalos assinalados pelo especialista humano na identificação dos elementos para classificação de supernovas (MODJAZ; BLONDIN, *et al.*, 2014).

As marcações do especialista humano da Figura 5.3 serviram como base para a criação dos intervalos, mas também foi feita uma análise visual dos gráficos plotados durante o pré-processamento dos espectros. Assim, os intervalos definidos para utilização na versão atual do CIntIa são:

- Ia: 5000 a 6500 angstroms;
- Ib: 5500 a 7000 angstroms;
- Ic: 5500 a 6500 angstroms;
- II: 4.000 a 5000 e 6000 a 7.000 angstroms.

Figura 5.3 - Exemplo dos intervalos que o especialista humano usa para identificar os elementos do espectro.



Fonte: Adaptado de Modjaz e Blondin *et al.* (2014)

Nessa versão um determinado conjunto de intervalos de cada espectro é analisado para classificar o espectro como pertencente um dos tipos “clássicos” de supernova: Ia, Ib, Ic e II. O número de entradas de cada uma das quatro redes neurais é diferente, pois depende dos intervalos do espectro utilizados que são diferentes para cada rede neural. No entanto, como a saída das quatro redes neurais informa se o espectro é ou não de um dos tipos “clássicos”, a camada de saída é formada por apenas um neurônio.

O banco de espectros utilizado nessa versão do CIntIa foi obtido no site do Centro de Astrofísica *Harvard-Smithsonian (CfA)* do qual foram selecionados 649 espectros.

Foi construída uma Rede Neural Perceptron de Múltiplas Camadas (MLP) para determinação de cada tipo de supernova. O espectro é analisado por cada rede neural para determinar se pertence ou não ao tipo de supernova identificado por essa rede neural.

Assim, foram construídas quatro redes neurais MLP para identificação dos tipos “clássicos” de supernovas, sendo uma rede para cada um desses tipos: Ia, Ib, Ic e II. As quantidades de padrões usados em todas as redes neurais foram: 451 para treinamento, 80 para validação e 118 para teste.

As redes neurais foram implementadas em um sistema escrito na linguagem C que foi desenvolvido especialmente para este trabalho. O programa principal desse sistema tem como entradas: um arquivo “.txt” com as configurações da rede neural, como número de camadas e tamanho das camadas, que é usado para treinamento e teste; e dois arquivos “.csv”, resultantes do pré-processamento feito no MATLAB, com as informações de cada espectro por linha, um com o conjunto de treinamento e outro arquivo com o conjunto de teste.

O programa permite que sejam ajustados parâmetros da rede neural como: taxa de aprendizagem, momentum, erro tolerado, bias e quantidade máxima de épocas. Um menu permite escolher entre: treinar a rede neural, ler um treinamento já feito, validar ou testar a rede neural. São criados arquivos resultantes do treinamento feito que servem como entrada para validação e teste. A saída dos testes são arquivos com informações sobre os acertos e erros da rede neural. O código desse programa está disponível no Apêndice B.

No próximo capítulo são mostrados os resultados obtidos nos testes com as quatro versões do CIntIa, o desempenho dos classificadores pesquisados e as comparações dos resultados do CIntIa com esses classificadores.

6 RESULTADOS DOS CLASSIFICADORES

Foram desenvolvidas quatro versões do CIntIa em busca de resultados significativos, mas os resultados dos métodos utilizados nas três primeiras versões foram considerados insuficientes para os objetivos deste trabalho. A quarta versão, chamada aqui de versão atual, apresentou resultados considerados suficientes.

Os resultados da versão atual do CIntIa foram comparados com os resultados das três versões anteriores para uma análise do aumento do percentual de acerto na classificação de cada versão. O cálculo do percentual de acerto, em todos os classificadores, foi feito dividindo o número de espectros que tiveram seu tipo identificado corretamente pelo número total de espectros usados no teste.

Também foi feita uma comparação dos resultados da versão atual do CIntIa com o desempenho dos classificadores pesquisados. No entanto, a comparação com os dois classificadores pesquisados, que identificam o tipo da supernova pela análise do espectro (SNID e GELATO), não foi possível porque esses trabalhos não mostraram valores que permitam o cálculo do percentual de acertos ou de outras medidas obtidas na classificação da supernova.

Os quatro trabalhos encontrados, que fazem a análise da curva de luz, divulgaram números do seu desempenho e foram usados para comparação com a versão atual do CIntIa. Nessa comparação foram utilizadas outras medidas além do percentual de acerto.

Vale ressaltar que existem diferenças de período da análise e de objetivos nas classificações feitas pela análise do espectro ou pela análise da curva de luz. Enquanto a primeira deve ser feita logo no surgimento da supernova e pode ajudar a decidir quais informações devem ser coletadas na sequência, a segunda somente pode ser feita depois de cerca de 60 a 90 dias do surgimento da supernova e é importante para classificar aquelas supernovas que não

tiveram espectros capturados. Mas apesar dessas diferenças, foi entendido que os resultados das classificações podem ser comparados em termos de quantidade de acertos e erros na identificação das supernovas. Vale destacar também que a comparação foi possível porque os classificadores pesquisados, apesar de analisarem a curva de luz, identificam os tipos “clássicos” de supernovas: Ia, Ib, Ic e II.

Os resultados das quatro versões do CIntIa e o desempenho dos classificadores pesquisados, assim como as comparações dos resultados, são mostrados neste capítulo. Os resultados das três primeiras versões são mostrados na próxima seção. Os resultados da versão atual do CIntIa e a comparação com as versões anteriores são apresentados na seção 6.2. O desempenho dos classificadores que fazem a classificação pela curva de luz e a comparação desse desempenho com a versão atual do CIntIa são mostrados na seção 6.3.

6.1. Resultados das três primeiras versões do CIntIa

Os espectros selecionados para os testes das três primeiras versões do CIntIa foram do banco de espectros SUSPECT, estão no intervalo de 3.800 a 6.800 angstroms e foram capturados no intervalo de 14 dias antes até 14 dias depois da luz máxima.

As três versões foram submetidas a testes para diferentes configurações quanto ao número de camadas e à quantidade de neurônios nas camadas. A variação do número de camadas se restringiu a uma ou duas camadas. O número de neurônios em cada camada variou entre 5 a 30 na primeira camada e entre 3 a 10 na segunda camada. Os resultados dos testes das três primeiras versões e as considerações feitas sobre os resultados são mostrados nas próximas seções.

6.1.1. Resultados da 1ª versão

A única rede neural MLP construída para a 1ª versão do CIntIa tem 376 entradas, foi treinada em 265 padrões e testada em 66 padrões. As configurações da rede neural usadas nos testes são apresentadas na Tabela 6.1, juntamente com o percentual de acerto de cada uma delas.

Tabela 6.1 - Configurações da única rede neural MPL da 1ª versão do CIntIa

Camada 1	Camada 2	Nº de Erros	% Acertos
5	-	17	74,2%
10	-	17	74,2%
15	-	17	74,2%
20	-	17	74,2%
25	-	24	63,6%
30	-	16	75,8%
10	3	18	72,7%
10	10	18	72,7%
20	4	18	72,7%
30	6	18	72,7%

Fonte: Produção do autor.

A rede neural que obteve o melhor resultado tem somente uma camada escondida com 30 neurônios. Ela apresentou 16 erros para os 66 padrões de teste que resultam em 75,8% de acerto na identificação dos quatro tipos clássicos de supernovas.

Analisando os erros por tipo de supernova, foram identificados erradamente: 11 espectros do tipo Ia, um espectro do tipo Ib e quatro espectros do tipo Ic. Os espectros do tipo II foram todos identificados corretamente.

6.1.2. Resultados da 2ª versão

A 2ª versão do CIntIa divide o espectro usando as caixas de Harutyunyan e tem três redes neurais que verificam, em sequência, a presença ou ausência dos elementos hidrogênio, silício e hélio. As configurações testadas para a 1ª rede neural são apresentadas na Tabela 6.2.

Tabela 6.2 - Configurações da 1ª rede neural MPL da 2ª versão do CIntIa

Camada 1	Camada 2	Nº de Erros	% Acertos
5	-	4	93,9%
10	-	4	93,9%
15	-	4	93,9%
20	-	4	93,9%
25	-	4	93,9%
30	-	5	92,4%
10	3	9	86,4%
10	10	9	86,4%
20	4	9	86,4%
30	6	8	87,9%

Fonte: Produção do autor.

Várias redes neurais obtiveram resultados iguais que foram os melhores na identificação do hidrogênio, como pode ser visto na Tabela 6.2. Aquela que teve menor quantidade de neurônios foi a rede com somente uma camada escondida de cinco neurônios.

Entre as configurações testadas na 2ª rede neural, que são mostradas na Tabela 6.3, o melhor resultado foi obtido por várias redes neurais com diferentes configurações. Entre elas a rede neural com menor número de neurônios tem uma única camada escondida de cinco neurônios.

Tabela 6.3 - Configurações da 2ª rede neural MPL da 2ª versão do CIntIa

Camada 1	Camada 2	Nº de Erros	% Acertos
5	-	12	79,6%
10	-	12	79,6%
15	-	12	79,6%
20	-	12	79,6%
25	-	12	79,6%
30	-	12	79,6%
10	3	12	79,6%
10	10	12	79,6%
20	4	12	79,6%
30	6	17	71,2%

Fonte: Produção do autor.

A Tabela 6.4 mostra as configurações testadas na 3ª rede neural e aquela que obteve o melhor resultado com menor número de neurônios também foi a rede neural com uma camada escondida de cinco neurônios.

Tabela 6.4 - Configurações da 3ª rede neural MPL da 2ª versão do CIntIa

Camada 1	Camada 2	Nº de Erros	% Acertos
5	-	5	50,0%
10	-	5	50,0%
15	-	5	50,0%
20	-	5	50,0%
25	-	5	50,0%
30	-	5	50,0%
10	3	5	50,0%
10	10	5	50,0%
20	4	6	40,0%
30	6	5	50,0%

Fonte: Produção do autor.

Os melhores resultados obtidos pela 2ª versão do CIntIa são resumidos a seguir para cada uma das três redes neurais usadas nos testes. A 1ª rede neural, que verifica a presença do hidrogênio, teve quatro erros para os 66 padrões de teste. Isso significa que essa versão foi capaz de separar as supernovas do tipo I das supernovas do tipo II com 93,9% de acerto. A 2ª rede neural, que verifica a presença do silício, teve 12 erros para os 59 padrões de teste. Isso significa que essa versão do CIntIa foi capaz de separar as supernovas do tipo Ia das supernovas dos tipos Ib e Ic com 79,7% de acerto. Já a 3ª rede neural, que verifica a presença do hélio, teve cinco erros para os 10 padrões de teste. Isso significa que essa rede neural foi capaz de separar as supernovas do tipo Ib das supernovas dos tipos Ic com 50,0% de acerto.

6.1.3. Resultados da 3ª versão

A 3ª versão do CIntIa divide o espectro usando um intervalo com as linhas dos elementos no centro e, como a versão anterior, tem três redes neurais que verificam, em sequência, a presença ou ausência dos elementos hidrogênio, silício e hélio.

A Tabela 6.5 mostra as configurações testadas para a 1ª rede neural da 3ª versão do CIntIa. Algumas obtiveram o mesmo resultado, mas aquela que obteve o melhor resultado na identificação de hidrogênio com menos neurônios tem somente uma camada escondida com cinco neurônios.

Tabela 6.5 - Configurações da 1ª rede neural MPL da 3ª versão do CIntIa

Camada 1	Camada 2	Nº de Erros	% Acertos
5	-	2	97,0%
10	-	2	97,0%
15	-	3	95,5%
20	-	2	97,0%
25	-	2	97,0%
30	-	2	97,0%
10	3	2	97,0%
10	10	2	97,0%
20	4	4	93,9%
30	6	4	93,9%

Fonte: Produção do autor.

A 2ª rede neural tem suas configurações testadas apresentadas na Tabela 6.6. A rede neural que obteve o melhor resultado com menos neurônio na identificação de silício tinha somente uma camada escondida com 15 neurônios.

Tabela 6.6 - Configurações da 2ª rede neural MPL da 3ª versão do CIntIa

Camada 1	Camada 2	Nº de Erros	% Acertos
5	-	9	84,7%
10	-	9	84,7%
15	-	8	86,4%
20	-	13	78,0%
25	-	8	86,4%
30	-	8	86,4%
10	3	8	86,4%
10	10	18	69,5%
20	4	16	72,9%
30	6	16	72,9%

Fonte: Produção do autor.

As configurações da 3ª rede neural são mostradas na Tabela 6.7. Pode ser observado que todas redes neurais obtiveram o mesmo resultado, sendo que a que obteve o melhor resultado na identificação de silício com menos neurônios tinha somente uma camada escondida com cinco neurônios.

Tabela 6.7 - Configurações da 3ª rede neural MPL da 3ª versão do CIntIa

Camada 1	Camada 2	Nº de Erros	% Acertos
5	-	4	60,0%
10	-	4	60,0%
15	-	4	60,0%
20	-	4	60,0%
25	-	4	60,0%
30	-	4	60,0%
10	3	4	60,0%
10	10	4	60,0%
20	4	4	60,0%
30	6	4	60,0%

Fonte: Produção do autor.

Os melhores resultados obtidos pela 3ª versão do CIntIa são resumidos a seguir para cada uma das redes neurais. A 1ª rede neural, construída para identificar o hidrogênio, teve dois erros para os 66 padrões de teste, ou seja, consegue separar as supernovas do tipo I das supernovas do tipo II com 97,0% de acerto. A 2ª rede neural, construída para identificar o silício, teve 8 erros para os 59 padrões de teste. Assim, as supernovas do tipo Ia foram separadas das supernovas dos tipos Ib e Ic com 86,4% de acerto. A 3ª rede neural, construída para identificar o hélio, teve 4 erros para os 10 padrões de teste, o que significa que a separação das supernovas do tipo Ib das supernovas do tipo Ic foi feita com 60,0% de acerto.

6.1.4. Considerações sobre os resultados das três primeiras versões do CIntIa

Analisando os testes feitos com as três primeiras versões do CIntIa, pode-se verificar que o percentual de acerto da 1ª versão (75,8%) mostrou não ser uma boa estratégia o uso de todo o espectro e a classificação de todos os tipos de supernovas ao mesmo tempo, principalmente, considerando que a maioria dos erros foi com supernovas do tipo Ia, que são importantes para estudo da expansão do universo.

Na 2ª versão do CIntIa a divisão em caixas se mostrou uma estratégia interessante que obteve um bom resultado (93,9%) na separação dos tipos I e II, mas um resultado não tão bom (79,7%) para separação das supernovas do tipo Ia dos outros tipos. A separação dos tipos Ib e Ic teve um resultado ruim (50,0%), mas é preciso considerar que o número baixo de espectros usados no treinamento (43) e teste (10) em comparação com as outras duas redes neurais.

A utilização das linhas que identificam presença ou ausência de cada elemento no espectro, feita na 3ª versão do CIntIa, se mostrou uma estratégia um pouco melhor que a divisão em caixas. Os resultados obtidos por cada nova versão do CIntIa foram melhores que a versão anterior, mas mantiveram as mesmas proporções. A separação dos tipos I e II teve o melhor resultado (97,0%) que a versão anterior. A separação do tipo Ia dos tipos Ib e Ic teve um resultado não tão bom (86,4%), mas melhor que a rede neural da versão anterior. A separação dos tipos Ib e Ic teve um resultado razoável (60,0%), mas também melhor que a versão anterior.

6.2. Resultados da versão atual do CIntIa

Os espectros selecionados para os testes da versão atual do CIntIa foram do banco de espectros CfA, estão no intervalo de 3.800 a 7400 angstroms e foram capturados no intervalo de três dias antes até sete dias depois da luz máxima. Cada espectro foi dividido com base nos intervalos usados pelos especialistas humanos.

Essa versão usa quatro redes neurais independentes na classificação, sendo que, cada rede neural verifica se o espectro é ou não é de um dos quatro tipos "clássicos" de supernova: Ia, Ib, Ic e II. Nos testes foram usados 118 padrões que foram submetidos a cada uma das quatro redes neurais.

Na versão atual também foram construídas diferentes configurações quanto ao número de camadas e ao número de neurônios por camada para cada uma das quatro redes neurais MLP. Foram feitos testes nas quatro redes neurais com variação de uma ou duas camadas intermediárias e com o número de neurônios variando de 10 a 40 na primeira camada e de 3 a 10 na segunda camada.

Para essa versão, além do percentual de acerto na identificação do tipo da supernova, mostrado também nas versões anteriores, são apresentadas outras medidas dos resultados obtidos: o coeficiente Kappa e os índices de Acurácia, Precisão e Recall.

6.2.1. Medidas usadas na comparação dos resultados

As medidas apresentadas nesta seção são utilizadas para comparação dos resultados da versão atual do CIntIa com o desempenho dos classificadores pesquisados. A Acurácia mede a proximidade da classificação feita com a classe real do objeto. A Precisão mede o grau de variação do conjunto de classificações feitas, de maneira que, quanto maior a precisão, menor e a variabilidade na classificação. O Recall mede a porcentagem de objetos

classificados como sendo daquela classe em relação a todos os objetos que são realmente da classe.

O coeficiente Kappa mede o grau de concordância em relação ao acaso, ou seja, o quanto a concordância está além do esperado para o acaso. Ele mede a concordância entre dois ou mais avaliadores ou entre dois ou mais métodos de classificação. A concordância varia de zero a um, é diretamente proporcional ao valor de Kappa e significa concordância total quando seu valor for igual a um (LANDIS; KOCH, 1977).

Essas medidas podem ser obtidas a partir da análise da matriz de confusão (ou matriz de classificação) que apresenta para cada classe o número de classificações corretas versus o número de classificações feitas pelo classificador. A diagonal da matriz contém o número de acertos para cada classe e a matriz de confusão ideal apresenta todos valores iguais a zero fora da diagonal principal (REZENDE, 2003). A Tabela 6.8 mostra um exemplo de uma matriz de confusão construída para duas classes, onde os objetos são classificados como sendo ou não de um determinado tipo X.

Tabela 6.8 - Matriz de Confusão para duas classes.

Matriz de Confusão		**Classificador**	
		Classificados como Tipo X	Classificados como Não Tipo X
Classe real	Tipo X	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não Tipo X	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Fonte: Adaptada de Rezende (2003).

Os valores que devem ser apresentados na matriz da confusão da Tabela 6.8 são obtidos da seguinte maneira:

- Verdadeiro Positivo (VP): número de objetos classificados corretamente como sendo do Tipo X;
- Falso Positivo (FP): número de objetos classificados erradamente como sendo do Tipo X;
- Falso Negativo (FN): número de objetos classificados erradamente como NÃO sendo do Tipo X;
- Verdadeiro Negativo (VN): número de objetos classificados corretamente como NÃO sendo do Tipo X.

A partir da matriz de confusão a Acurácia é calculada pela equação:

$$Acurácia = \frac{VP + VN}{VP + FP + FN + VN} \quad (6.1)$$

O cálculo da Precisão é feito pela equação:

$$Precisão = \frac{VP}{VP + FP} \quad (6.2)$$

O Recall é calculado pela equação:

$$Recall = \frac{VP}{VP + FN} \quad (6.3)$$

O cálculo do coeficiente de Kappa é feito pela equação (LANDIS; KOCH, 1977):

$$Kappa = \frac{\pi_o - \pi_e}{1 - \pi_e} \quad (6.4)$$

Sendo que:

- π_o: concordância observada

- π_e: concordância esperada

No caso deste trabalho, o coeficiente Kappa mede a concordância entre a classificação feita pelo especialista humano e a classificação feita por cada classificador automático. A força da concordância pode ser obtida a partir do valor de Kappa (LANDIS; KOCH, 1977), conforme mostrado na Tabela 6.9.

Tabela 6.9 - Relação entre o valor de Kappa e a força da concordância.

Valor de Kappa	**Força da Concordância**
< 0,00	Pobre
0,00 a 0,19	Leve
0,20 a 0,39	Razoável
0,40 a 0,59	Moderada
0,60 a 0,79	Substancial
0,80 a 1,00	Quase perfeita

Fonte: Adaptada de Landis e Koch (1977).

Os valores dessas medidas para cada uma das quatro redes neurais da versão atual do CIntIa são mostrados nas próximas seções.

6.2.2. 1ª Rede Neural – Tipo Ia (RN Ia)

A 1ª rede neural verifica se o espectro é ou não é de uma supernova do tipo Ia. Foram feitos testes com diferentes configurações da rede neural e os resultados de cada configuração são mostrados na Tabela 6.10.

Tabela 6.10 - Configurações da 1ª rede neural MPL da versão atual do CIntIa

Camada 1	Camada 2	Nº de Erros	% Acertos
10	-	1	99,2%
20	-	1	99,2%
25	-	1	99,2%
40	-	1	99,2%
10	3	1	99,2%
10	10	1	99,2%
20	4	1	99,2%
40	8	1	99,2%

Fonte: Produção do autor.

Todas as configurações da rede neural obtiveram o mesmo percentual de acerto, assim, foi selecionada a rede neural com menos neurônios que tem única camada intermediária com 10 neurônios e precisou de 676 épocas para realizar o treinamento. O resultado dessa rede neural foi apenas um erro, resultando em 99,2% de acerto. A Tabela 6.11 mostra a matriz de confusão da 1ª rede neural.

Tabela 6.11 - Matriz de Confusão da 1ª rede neural MPL da versão atual do CIntIa

	CIntIa		
Classe	**Tipo**	Ia	NãoIa
	Ia	106	0
	NãoIa	1	11

Fonte: Produção do autor.

Podemos observar na Tabela 6.11 que o erro foi um falso positivo que identificou como sendo do tipo Ia um espectro de outro tipo. O coeficiente Kappa da matriz de confusão é 0,95, o que é um ótimo valor, pois indica que a concordância foi quase perfeita, segundo a Tabela 6.9. Outros valores, que também podem ser calculados a partir da matriz de confusão para comparação com os outros classificadores pesquisados, são: Acurácia = 0,99; Precisão = 0,99 e Recall = 1,00.

6.2.3. 2ª Rede Neural – Tipo Ib (RN Ib)

Os resultados dos testes feitos para as configurações da 2ª rede neural, que verifica se o espectro é ou não é de uma supernova do tipo Ib, são mostrados na Tabela 6.12.

Tabela 6.12 - Configurações da 2ª rede neural MPL da versão atual do CIntIa

Camada 1	Camada 2	Nº de Erros	% Acertos
10	-	4	96,6%
20	-	4	96,6%
25	-	4	96,6%
40	-	4	96,6%
10	3	4	96,6%
10	10	4	96,6%
20	4	4	96,6%
40	8	4	96,6%

Fonte: Produção do autor.

Essa rede neural também teve todas as configurações com o mesmo percentual de acerto e também foi selecionada a rede neural com uma única camada intermediária com 10 neurônios. Ela precisou de 1092 épocas para realizar o treinamento. Essa rede neural teve quatro erros que resultam em 96,6% de acerto. A Tabela 6.13 mostra a matriz de confusão da 2ª rede neural.

Tabela 6.13 - Matriz de Confusão da 2ª rede neural MPL da versão atual do CIntIa

	CIntIa		
Classe	**Tipo**	Ib	NãoIb
	Ib	1	4
	NãoIb	0	113

Fonte: Produção do autor.

Os erros mostrados na Tabela 6.13 foram todos falso negativo, pois identificou como não sendo do tipo Ib os espectros que são desse tipo. O coeficiente Kappa da matriz de confusão é 0,32, que é um valor muito baixo, indicando uma concordância razoável, segundo a Tabela 6.9.

6.2.4. 3ª Rede Neural – Tipo Ic (RN Ic)

Os resultados de cada configuração da 3ª rede neural, que verifica se o espectro é ou não de uma supernova do tipo Ic, são mostrados na Tabela 6.14. Seis das configurações testadas para a rede neural obtiveram o mesmo percentual de acerto. Assim, foi selecionada a rede neural com menos neurônios que tem única camada intermediária com 25 neurônios e precisou de 631 épocas para realizar o treinamento.

Tabela 6.14 - Configurações da 3ª rede neural MPL da versão atual do CIntIa

Camada 1	Camada 2	Nº de Erros	% Acertos
10	-	4	96,6%
20	-	4	96,6%
25	-	3	97,5%
40	-	3	97,5%
10	3	3	97,5%
10	10	3	97,5%
20	4	3	97,5%
40	8	3	97,5%

Fonte: Produção do autor.

Essas quatro redes neurais que apresentaram o melhor resultado tiveram três erros que resultam em 97,5% de acerto. A Tabela 6.15 mostra a matriz de confusão da 3ª rede neural.

Tabela 6.15 - Matriz de Confusão da 3ª rede neural MPL da versão atual do CIntIa

	CIntIa		
Classe	**Tipo**	Ic	NãoIc
	Ic	3	3
	NãoIc	0	112

Fonte: Produção do autor.

Os erros mostrados na Tabela 6.15 também foram todos falso negativo, identificando como não sendo do tipo Ic os espectros que são desse tipo. O coeficiente Kappa da matriz de confusão é 0,65, que é um valor bom, pois indica uma concordância substancial, segundo a Tabela 6.9.

6.2.5. 4ª Rede Neural – Tipo II (RN II)

A rede neural que verifica se o espectro é do tipo II tem suas configurações testadas mostradas na Tabela 6.16.

Tabela 6.16 - Configurações da 4ª rede neural MPL da versão atual do CIntIa

Camada 1	Camada 2	Nº de Erros	% Acertos
10	-	2	98,3%
20	-	2	98,3%
25	-	2	98,3%
40	-	2	98,3%
10	3	2	98,3%
10	10	2	98,3%
20	4	2	98,3%
40	8	2	98,3%

Fonte: Produção do autor.

Todas as configurações dessa rede neural também obtiveram o mesmo percentual de acerto e também foi selecionada a rede neural com menos neurônios que tem única camada intermediária com 10 neurônios. Ela precisou de 558 épocas para realizar o treinamento. A rede neural teve dois erros, ou

seja, 98,3% de acerto. A Tabela 6.17 mostra a matriz de confusão da 4ª rede neural.

Tabela 6.17 - Matriz de Confusão da 4ª rede neural MPL da versão atual do CIntIa

	CIntIa		
Classe	**Tipo**	II	NãoII
	II	1	0
	NãoII	2	115

Fonte: Produção do autor.

Os erros mostrados na Tabela 6.17 foram todos falso positivo, pois identificaram como sendo do tipo II os espectros que são de outro tipo. O coeficiente Kappa da matriz de confusão é 0,49, que é um valor razoável, indicando uma concordância moderada, segundo a Tabela 6.9.

Os resultados obtidos pela versão atual do CIntIa são considerados significativos, principalmente na classificação das supernovas do tipo Ia que são o foco do Projeto KDUST.

6.2.6. Resumo dos resultados das quatro redes neurais

Cada um dos erros das quatro redes neurais da versão atual do CIntIa é mostrado na Tabela 6.18, identificando o resultado esperado e o resultado obtido para cada espectro que foi identificado erradamente.

Para cada rede neural são mostrados o número de erros e o percentual de acertos. Para cada espectro que foi identificado erradamente são mostrados o número do padrão no teste feito (padrão), o tipo da supernova que foi identificado pelo especialista humano (tipo), o nome da supernova (nome), a

distância da luz máxima em número de dias que o espectro foi capturado (fase) e a classificação da supernova feita pela rede neural (classe RN). A classe RN indica se o espectro foi classificado como daquele tipo (Tipo_Xx) ou classificado como não sendo daquele tipo (Tipo_NãoXx).

Tabela 6.18 - Resumo dos resultados das quatro Redes Neurais MLP (RN) para os 118 padrões de teste

Rede Neural	**Nº de Erros**	**% Acertos**	**Padrão**	**Tipo**	**Nome**	**Fase**	**Classe RN**
RN Ia	1	99,2%	110	Ic	SN 2004fe	1,9	Tipo_Ia
RN Ib	4	96,6%	115	Ib	SN 2009er	-2,7	Tipo_NaoIb
			116	Ib	SN 2009er	-1,7	Tipo_NaoIb
			117	Ib	SN 2009er	0,3	Tipo_NaoIb
			118	Ib	SN 2009er	2,0	Tipo_NaoIb
RN Ic	3	97,5%	107	Ic	SN 2004fe	-1,1	Tipo_NaoIc
			108	Ic	SN 2004fe	-0,1	Tipo_NaoIc
			113	Ic	SN 2007bg	3,4	Tipo_NaoIc
RN II	2	98,3%	113	Ic	SN 2007bg	3,4	Tipo_II
			116	Ib	SN 2009er	-1,7	Tipo_II

Fonte: Produção do autor.

Como pode ser observado na Tabela 6.18 as redes neurais que identificam supernovas dos tipos Ia e II (RN Ia e RN II) somente obtiveram erros de falso positivo, ou seja, o espectro foi identificado erradamente como sendo do tipo Ia ou II. Isso significa que os espectros de todas as supernovas dos tipos Ia e II foram identificadas corretamente. Enquanto as redes neurais que identificam supernovas dos tipos Ib e Ic (RN Ib e RN Ic) obtiveram somente erros de falso negativo, ou seja, os espectros dos tipos Ib e Ic não foram identificados como sendo desses tipos.

Na Tabela 6.18, a supernova 2004fe é do tipo Ic, mas não foi identificada como tal pela RN Ic, e também foi identificada incorretamente pela RN Ia. A supernova 2007bg é do tipo Ic e não foi identificada pela RN Ic, mas foi identificada incorretamente pela RN II. A supernova 2009er é do tipo Ib e não foi identificada corretamente pela RN Ib, mas foi identificada erradamente pela RN II. Esses três casos mostram que de 59 supernovas usadas para teste, somente a identificação de três supernovas apresentaram erros considerando as quatro redes neurais: SN 2004fe (tipo Ic), SN 2007bg (tipo Ic) e SN 2009er (tipo Ib). Na próxima seção é feita a análise dos resultados de cada uma dessas três supernovas.

6.2.7. Análise dos resultados

A comparação dos valores do coeficiente Kappa das quatro redes neurais mostra que a 1ª rede neural, que identifica supernovas do tipo Ia, tem o melhor resultado (0,95), enquanto a 2ª rede neural, que identifica supernovas do tipo Ib, teve o pior resultado (0,32).

Os testes da versão atual do CIntIa foram feitos em 118 espectros de 43 supernovas e apenas oito espectros de três supernovas apresentaram erros na identificação do tipo. Todas as supernovas do tipo Ia e II foram identificadas corretamente. Os erros de identificação ocorreram somente nas supernovas do

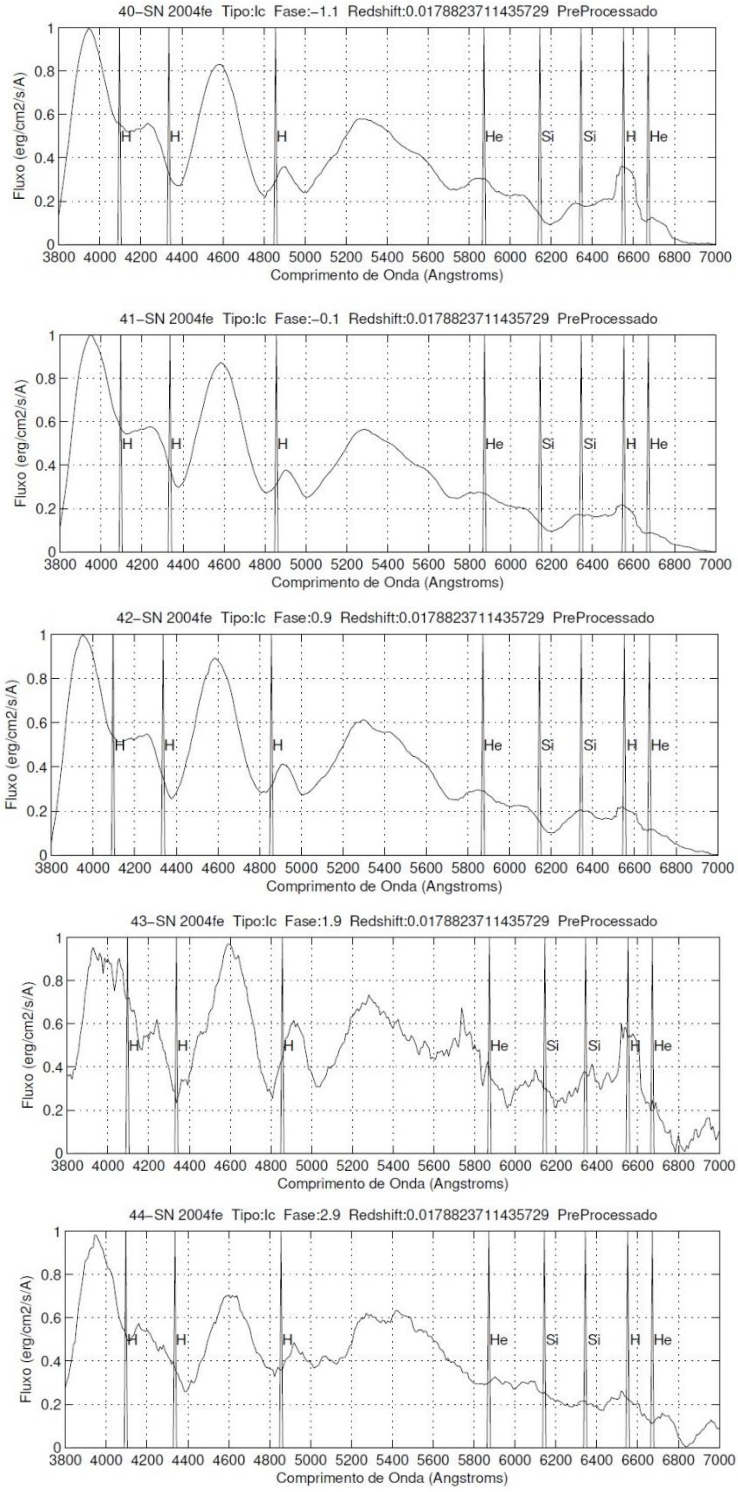
tipo Ib e Ic. Uma análise dos erros mostrados na Tabela 6.18 é feito a seguir para cada uma dessas supernovas que apresentaram erros na classificação de seus espectros: SN 2004fe, SN 2007bg e SN 2009er.

A supernova SN 2004fe é do tipo Ic e seus cinco espectros são mostrados na Figura 6.1. Dentre esses cinco espectros, somente três apresentaram erro de classificação. O espectro com fase 1,9 (padrão 110) foi o único identificado positivamente por duas redes neurais diferentes: foi classificado corretamente pela RN Ic como sendo do tipo Ic e classificado erradamente pela RN Ia como sendo do tipo Ia. Entre os erros apresentados por todas as redes neurais essa foi a única inconsistência onde um mesmo espectro foi classificado em dois tipos.

Os espectros da SN 2004fe de fase -1,1 (padrão 107) e de fase -0,1 (padrão 108) foram classificados por todas as redes neurais como não sendo do seu tipo, ou seja, segundo o CIntIa eles não são de nenhum dos quatro tipos clássicos. Os outros dois espectros dessa supernova, de fase 0,9 e de fase 2,9 foram classificados corretamente como tipo Ic e não aparecem nessa tabela.

Em resumo, dos cinco espectros testados para essa supernova: dois foram classificados corretamente como tipo Ic, dois foram classificados erradamente como de nenhum tipo e um foi classificado erradamente como tipo Ia e também corretamente como tipo Ic. Assim, temos três classificações corretas para o tipo Ic contra uma errada para o tipo Ia e duas incorretas indicando que não são do tipo Ic (Tipo_NaoIc). Fazendo uma análise do conjunto das classificações dessa supernova, pode se observar que dessas seis classificações três são para o tipo Ic enquanto duas outras não definem o tipo e uma indica tipo Ia, logo, podemos concluir pela maioria das indicações que ela é tipo Ic. Essa capacidade de análise não existe ainda no CIntIa, mas pode ser incluída usando a Lógica Nebulosa.

Figura 6.1 - Espectros pré-processados da supernova 2004fe capturados em dias diferentes



SN2004fe
Fase: -1,1

SN2004fe
Fase: -0,1

SN2004fe
Fase: 0,9

SN2004fe
Fase: 1,9

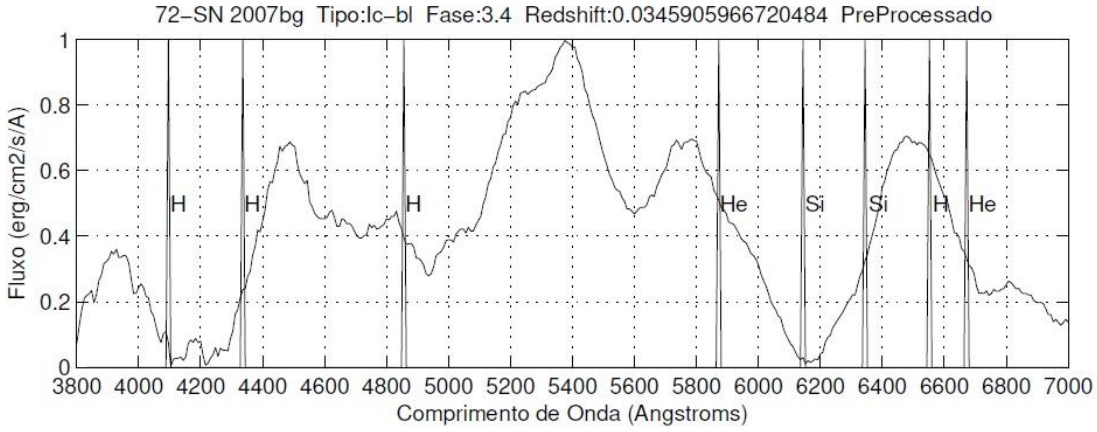
SN2004fe
Fase: 2,9

Fonte: Produção do autor.

Analisando em particular o erro de classificação do espectro de fase 1,9 (padrão110) foi feita uma análise visual comparando seu espectro pré-processado com os outros espectros pré-processados que participaram do teste. Esse erro pode ter relação com a suavização aplicada, pois esse espectro, mesmo depois de aplicada a suavização, ainda ficou com muitos picos e vales, diferente dos outros espectros dessa mesma supernova. Vale ressaltar que a diferença nos espectros não se deve a dispersão espectral que é igual a dos outros espectros dessa mesma supernova (1,47 angstroms/pixel). Assim, uma solução pode ser medir número de picos e vales por intervalo em angstroms e aplicar a suavização baseada nesse parâmetro.

A supernova SN 2007bg é do tipo Ic e teve apenas um espectro usado nos testes que é mostrado na Figura 6.2. Esse espectro de fase 3,4 (padrão 113) foi classificado erradamente como sendo do tipo II e não sendo do tipo Ic. Fazendo a análise visual em comparação com outros espectros a classificação feita pelo humano também poderia ser como tipo II.

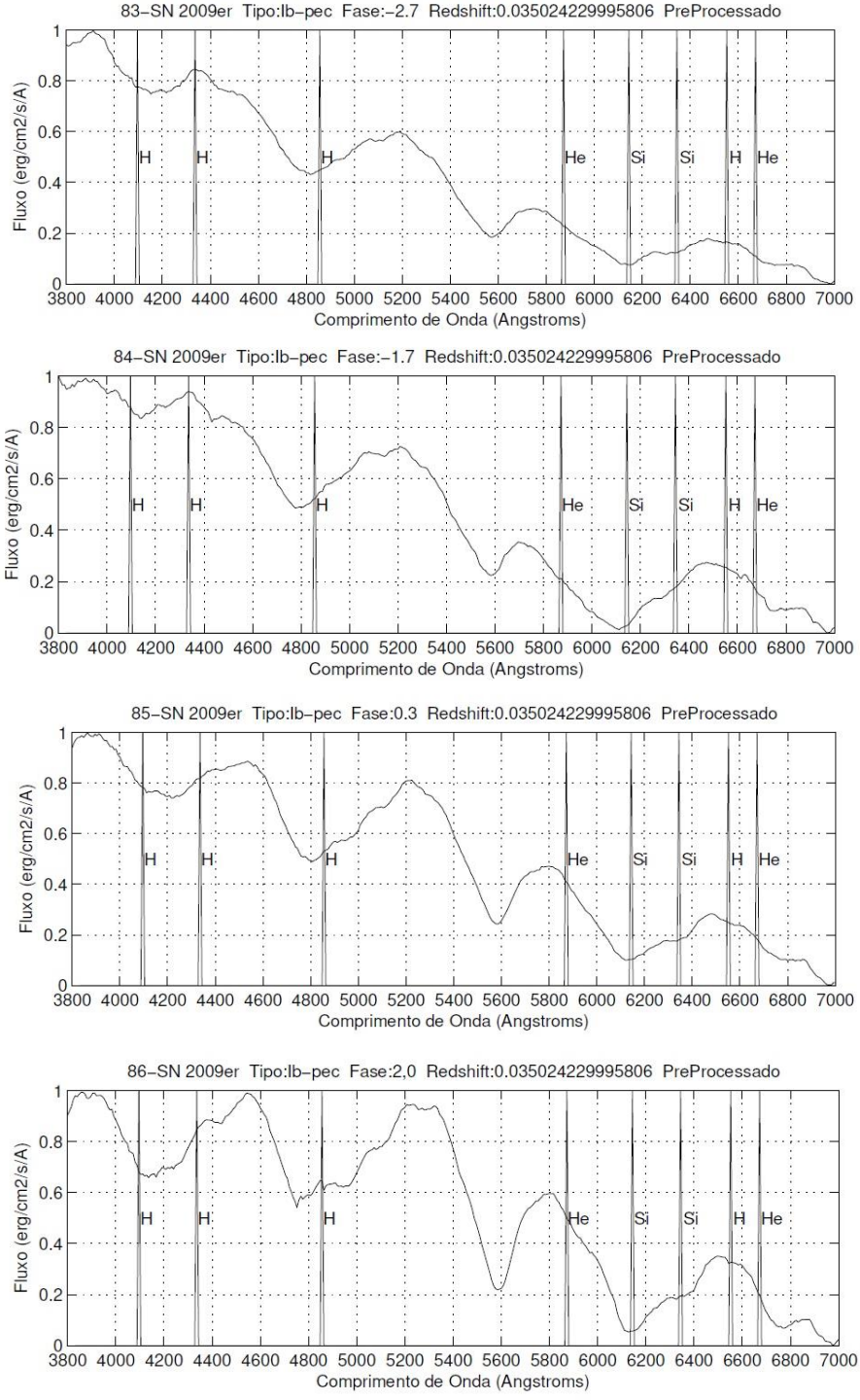
Figura 6.2 - Espectro pré-processado da supernova 2007bg



Fonte: Produção do autor.

A supernova SN 2009er é do tipo Ib e seus quatro espectros são mostrados na Figura 6.3. Ela tem quatro espectros e todos apresentaram erro de classificação.

Figura 6.3 - Espectros pré-processados da supernova 2009er capturados em dias diferentes



Fonte: Produção do autor.

Todos os quatro espectros da supernova SN 2009er foram classificados erradamente como não sendo do tipo Ib e o espectro de fase -1,7 (padrão 116) também foi classificado erradamente como sendo do tipo II. Esse erro deve ter sido ocasionado porque essa supernova está classificada na base de espectros como sendo do tipo Ib-pec e, como no esquema de classificação adotado neste trabalho não existe essa classificação, ela foi considerada como sendo do tipo Ib. Essa supernova é a única nos conjuntos de treinamento e teste que foi classificada como Ib-pec.

Se for considerado que uma supernova do tipo Ib-pec não é um dos tipos "clássicos", a classificação como não sendo tipo Ib está correta e resta apenas a classificação errada como tipo II. Analisando novamente o conjunto das classificações, quatro corretas como não sendo do tipo Ib contra uma errada como sendo tipo II, indicam que essa supernova não é de nenhum dos tipos "clássicos". Lembrando que essa análise ainda não é feita pelo CIntIa, mas pode ser implementada usando Lógica Nebulosa.

Importante observar que as supernovas SN 2007bg e SN 2009er foram classificadas erradamente como sendo do tipo II e como não sendo de qualquer dos outros três tipos. Mas que esses resultados, mesmo errados, mostram existir coerência na classificação das redes neurais, pois, apesar de trabalharem de maneira independente uma da outra, os espectros dessas supernovas foram classificados positivamente por apenas uma rede neural e negativamente pelas outras três redes neurais. Assim, nenhum desses espectros foi classificado como sendo de dois ou mais tipos diferentes de supernova.

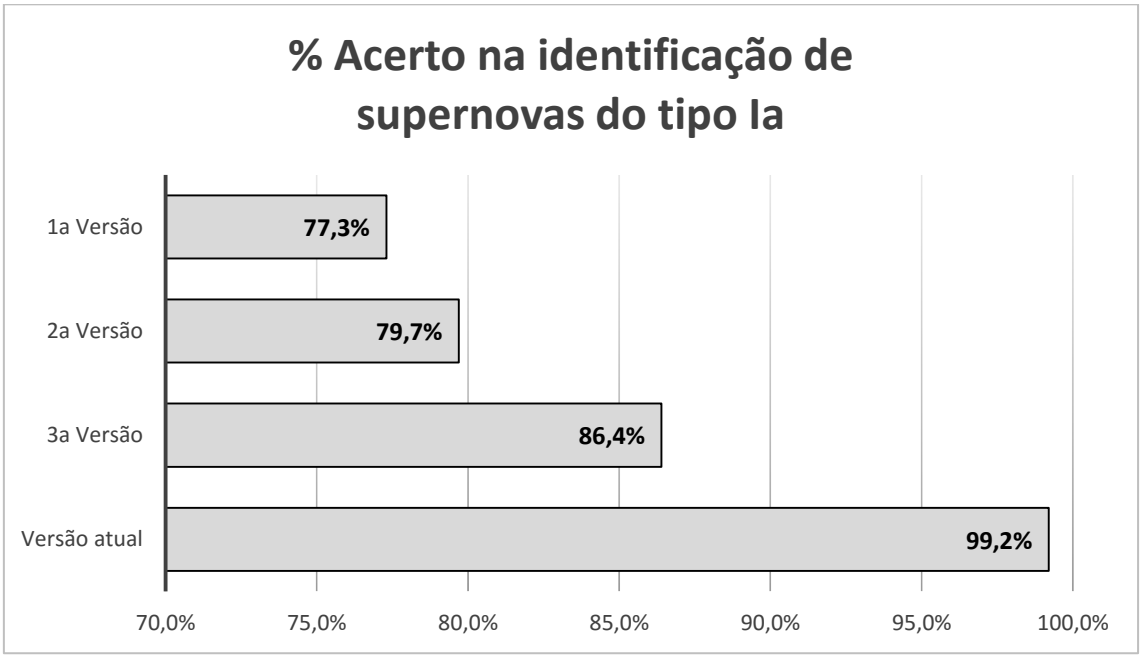
Analisando o classificador quanto ao custo computacional, pode-se afirmar que tempo necessário para treinamento das redes neurais é muito pequeno. Usando um computador Intel® Core™ i7-5500U com dois processadores de 2,40 GHz e memória de 16GB, o treinamento de cada uma das redes neurais que obtiveram os melhores resultados demorou menos de três segundos para

convergir. Mesmo entre as redes neurais com duas camadas e um número maior de neurônios, o tempo máximo de treinamento não chegou a um minuto.

6.2.8. Comparação dos resultados das quatro versões do CIntIa

Para a comparação dos resultados das quatro versões do CIntIa foi considerado apenas o percentual de acerto na separação das supernovas do tipo Ia dos outros tipos de supernovas, conforme mostrado na Figura 6.4. Isso foi feito devido ao resultado da separação das supernovas do tipo Ib para o tipo Ic ter sido considerado baixo, pois um pequeno número de espectros foi envolvido no treinamento e teste das redes neurais. Também, porque a identificação das supernovas do tipo Ia é importante para o contexto no qual está envolvido este trabalho.

Figura 6.4 - Comparação dos resultados das quatro versões do CIntIa na identificação de supernovas do tipo Ia.



Fonte: Produção do autor.

Assim, para cálculo do percentual de acerto na identificação de supernovas do tipo Ia na 1ª versão do CIntIa, foram contados os erros nos quais as supernovas do tipo Ia foram identificadas erradamente (11 erros) mais os erros nos quais outros tipos de supernovas foram identificados erradamente como sendo do tipo Ia (4 erros), que totalizaram 15 erros nos 66 espectros de teste, ou seja, 77,3% de acerto.

Na 2ª e 3ª versões do CIntIa, foi considerado o resultado da rede neural que separa o tipo Ia dos tipos Ib e Ic, que 79,7% de acerto na 2ª versão e 86,4% na 3ª versão. Na versão atual do CIntIa também foi considerado o resultado de apenas uma rede neural, aquela que identifica o tipo Ia, que teve 99,2% de acerto.

Comparando os percentuais de acerto da Figura 6.4, pode-se observar que a utilização de todo o espectro feita na 1ª versão obteve 77,3% de acertos na classificação, sendo o pior resultado entre as quatro versões. O resultado da 2ª versão foi de 79,7% de acerto, o que representa um pequeno avanço em relação ao resultado da versão anterior, mostrando que usar partes do espectro e três redes neurais ajudaram a melhorar o resultado. Na 3ª versão foi obtido 86,4% de acerto em média, mostrando que usar um intervalo em relação às linhas dos elementos foi uma estratégia melhor que a divisão em caixas. A versão atual do CIntIa, que usa as mesmas áreas do espectro que são utilizadas pelo especialista humano e quatro redes neurais independentes para identificação dos tipos de supernova, obteve em média 99,2% de respostas corretas, superando os resultados das outras versões e com percentual de acerto bastante significativo para um classificador de supernovas.

6.3. Desempenho dos classificadores pesquisados

Os classificadores que analisam a curva de luz tiveram seus desempenhos publicados que são mostrados nessa seção. Já o desempenho dos dois

classificadores que analisam também o espectro (GELATO E SNID) não são apresentados aqui porque as quantidades de acertos e erros não foram divulgadas nos trabalhos pesquisados. Essa seção apresenta também a comparação dos resultados da versão atual do CIntIa com o desempenho dos classificadores que analisam a curva de luz.

6.3.1. Classificador SN Automatic Bayesian Classifier

O classificador *SN Automatic Bayesian Classifier (SN-ABC)* desenvolvido em conjunto pela *Tel-Aviv University e o California Institute of Technology* (POZNANSKI; MAOZ; GAL-YAM, 2007) foi testado nos projetos SNLS (*Supernova Legacy Survey*) e GOODS (*Great Observatories Origins Deep Survey*), que tem um conjunto de supernovas já classificadas. O SN-ABC classifica a supernova em apenas duas classes: tipo Ia ou tipo NãoIa.

A partir do conjunto de curva de luz do projeto SNLS que tem 71 supernovas do tipo Ia e cinco supernova dos outros tipos foram extraídos 172 objetos do tipo Ia e 25 objetos dos outros tipos para uso nos testes. No conjunto do projeto GOODS tem 23 supernovas do tipo Ia e 18 supernovas dos outros tipos, devido à falta das características necessárias, foram utilizados somente 17 objetos do tipo Ia e 11 objetos dos outros tipos. O detalhamento do desempenho divulgado para os dois conjuntos de dados é mostrado a seguir.

A Tabela 6.19 mostra o desempenho e a Tabela 6.20 mostra a matriz de confusão do conjunto de dados SNLS

Tabela 6.19 - Desempenho do conjunto de dados SNLS.

Classe	Nº Objetos	Nº de Erros	% Acertos
Ia	172	6	96,5%
NãoIa	25	4	84,0%
Total	197	10	94,9%

Fonte: Adaptada de Poznanski; Maoz e Gal-Yam (2007).

Tabela 6.20 - Matriz de Confusão do conjunto de dados SNLS.

	Dados SNLS		
Classe	**Tipo**	Ia	NãoIa
	Ia	166	6
	NãoIa	4	25

Fonte: Produção do autor.

A partir da matriz de confusão pode-se calcular o coeficiente Kappa = 0,80 e os valores de Acurácia = 0,95, Precisão = 0,98 e Recall = 0,97.

A Tabela 6.21 mostra o desempenho divulgado e a Tabela 6.22 mostra a matriz de confusão para o conjunto de dados GOODS.

Tabela 6.21 - Desempenho do conjunto de dados GOODS.

Classe	Nº Objetos	Nº de Erros	% Acertos
Ia	17	1	94,1%
NãoIa	11	2	81,8%
Total	28	3	89,3%

Fonte: Adaptada de Poznanski; Maoz e Gal-Yam (2007).

Tabela 6.22 - Matriz de Confusão do conjunto de dados GOODS.

	Dados GOODS		
Classe	**Tipo**	Ia	NãoIa
	Ia	16	1
	NãoIa	2	9

Fonte: Produção do autor.

A partir da matriz de confusão pode-se calcular o coeficiente Kappa = 0,77 e os valores de Acurácia = 0,89, Precisão = 0,89 e Recall = 0,94.

O percentual de acerto e os índices calculados a partir da matriz de confusão mostram que os dois conjuntos de dados tiveram desempenho muito próximos e em ambos o maior percentual de acerto foi no tipo Ia.

6.3.2. Classificadores Bayesiano e Nebuloso

O *Institute for Astronomy da University of Hawaii* (RODNEY; TONRY, 2009) apresentou dois métodos de classificação automática: *Bayesian Adaptive Template Matching* (BATM) e *Supernova Ontology with Fuzzy Templates* (SOFT). No entanto, o método mostrado como melhoria do programa BATM não teve seu desempenho divulgado neste trabalho, pois os autores concluíram que sozinho ele é insuficiente para classificação de supernovas a partir da curva de luz. O desempenho do método SOFT foram divulgados e mostram a separação dos tipos de supernovas em apenas duas classes: tipo Ia e tipo NãoIa.

Os testes do método SOFT foram feitos em dois conjuntos de dados: um conjunto de 5000 curvas de luz sintéticas geradas por simulação para cada uma das duas classes; e um conjunto de 222 curvas de luz capturadas.

O desempenho divulgado é mostrado na Tabela 6.23 e a matriz de confusão na Tabela 6.24 para o conjunto de curvas de luz geradas.

Tabela 6.23 - Desempenho do método SOFT em curvas de luz geradas.

Classe	**Nº Objetos**	**Nº de Erros**	**% Acertos**
Ia	5.000	~500	~90,0%
NãoIa	5.000	~1.000	~80,0%
Total	10.000	~1.500	~85,0%

Fonte: Adaptada de Rodney e Tonry (2009).

Tabela 6.24 - Matriz de Confusão do método SOFT em curvas de luz geradas.

	SOFT curvas de luz geradas		
Classe	**Tipo**	Ia	NãoIa
	Ia	~4.500	~500
	NãoIa	~1.000	~4.000

Fonte: Produção do autor.

A partir da matriz de confusão pode-se calcular o coeficiente Kappa = 0,70 e os valores de Acurácia = 0,85, Precisão = 0,82 e Recall = 0,90.

Na Tabela 6.25 é mostrado o desempenho e na Tabela 6.26 a matriz de confusão para o conjunto de curvas de luz capturadas.

Tabela 6.25 - Desempenho do método SOFT em curvas de luz capturadas.

Classe	Nº Objetos	Nº de Erros	% Acertos
Ia	217	13	94,0%
NãoIa	5	Não divulgado	-
Total	222	-	-

Fonte: Adaptada de Rodney e Tonry (2009).

Tabela 6.26 - Matriz de Confusão do método SOFT em curvas de luz capturadas.

	SOFT curvas de luz capturadas		
Classe	**Tipo**	Ia	NãoIa
	Ia	204	13
	NãoIa	-	-

Fonte: Produção do autor.

A matriz de confusão está incompleta devido à falta de alguns dados. Sendo assim, não foi possível calcular o coeficiente Kappa e os valores de Acurácia e Precisão, mas apenas o Recall = 0,94.

Pode-se observar no conjunto das curvas de luz capturadas a maioria é de supernovas do tipo Ia e que não foram divulgados o desempenho para as cinco curvas de luz de outros tipos.

6.3.3. Classificadores Discriminação Logística e LogitBoost

O classificador desenvolvido na *Università Degli Studi di Padova* (PASCALE, 2011) que utiliza os algoritmos Discriminação Logística Linear e LogitBoost

apresenta o desempenho desses dois algoritmos para dois testes com número diferentes de classes. A classificação feita no primeiro teste identificou três classes, agrupando os tipos Ib e Ic em uma mesma classe. No segundo teste identificou apenas duas classes, separando as supernovas em tipo Ia e todos os outros tipos juntos que não são tipo Ia. O desempenho para cada um desses testes é mostrado nas duas próximas seções.

6.3.3.1. Desempenho para três classes: SN Ia, SN II e SN Ibc

A classificação feita nesse teste considerou apenas alguns tipos de supernovas e agrupou os tipos Ib e Ic em uma única classe. Os dois algoritmos usaram as mesmas classes: SN Ia, SN II e SN Ibc

O primeiro algoritmo, discriminação logística linear, obteve em média 72,4% de classificações certas do tipo da supernova para as três classes. O algoritmo LogitBoost obteve resultado melhor, obtendo para as três classes em média 85,4% de acerto. Esse é o melhor desempenho desse classificador em um conjunto de dados com 6.688 de instâncias selecionadas, que representam 34% do conjunto total de 19.669 instâncias. O desempenho para o conjunto total foi bem pior: o primeiro algoritmo teve 45,3% de acertos e o LogitBoost teve 50,7% de acertos.

O detalhamento do melhor desempenho divulgados para esse classificador é mostrado a seguir. A Tabela 6.27 mostra o desempenho do algoritmo por discriminação logística linear e a Tabela 6.28 mostra o desempenho do algoritmo LogitBoost.

Tabela 6.27 - Desempenho do algoritmo discriminação linear logística para três classes.

Classe	Nº Instâncias	Nº de Erros	% Acertos
Ia	1.509	689	54,3%
II	4.328	378	91,3%
Ibc	851	777	8,7%
Total	6.688	1.844	72,4%

Fonte: Adaptada de Pascale (2011).

Tabela 6.28 - Desempenho do algoritmo LogitBoost para três classes.

Classe	Nº Instâncias	Nº de Erros	% Acertos
Ia	1.509	251	83,4%
II	4.328	258	94,0%
Ibc	851	469	44,9%
Total	6.688	978	85,4%

Fonte: Adaptada de Pascale (2011).

O desempenho mostra que o maior percentual de acertos foi para o tipo II e o pior percentual de acerto foi para a junção dos tipos Ib e Ic. Os dados obtidos não são suficientes para a criação da matriz de confusão para três classes.

6.3.3.2. Desempenho para duas classes: SN Ia e SN não-Ia

A classificação desse teste agrupou os tipos Ib, Ic e II em apenas uma classe, ou seja, os dois algoritmos identificam apenas duas classes: tipo Ia e tipo NãoIa.

O algoritmo discriminação logística linear obteve em média 48,0% de classificações certas do tipo da supernova para as duas classes. O algoritmo LogitBoost obteve resultado um pouco melhor, obtendo para as duas classes em média 57,3% de acerto. Esse desempenho foi divulgado para todo o conjunto de dados com 19.669 instâncias, uma vez que não foram feitos testes para uma parte do conjunto, como no outro teste mostrado na seção anterior.

O detalhamento do melhor desempenho divulgado para esse classificador é mostrado a seguir. A Tabela 6.29 mostra o desempenho e a Tabela 6.30 a matriz de confusão do algoritmo por discriminação logística linear.

Tabela 6.29 - Desempenho do algoritmo discriminação linear logística para duas classes.

Classe	**Nº Instâncias**	**Nº de Erros**	**% Acertos**
Ia	4.380	235	94,6%
NãoIa	15.289	9.996	34,6%
Total	19,669	10.231	48,0%

Fonte: Adaptada de Pascale (2011).

Tabela 6.30 - Matriz de Confusão do algoritmo discriminação linear logística para duas classes.

	Discriminação Logística		
Classe	**Tipo**	Ia	NãoIa
	Ia	4.145	235
	NãoIa	9.996	5.293

Fonte: Produção do autor.

A partir da matriz de confusão pode-se calcular o coeficiente Kappa = 0,16 e os valores de Acurácia = 0,48, Precisão = 0,29 e Recall = 0,95.

A Tabela 6.31 mostra o desempenho e a Tabela 6.32 mostra a matriz de confusão do algoritmo LogitBoost

Tabela 6.31 - Desempenho do algoritmo LogitBoost para duas classes.

Classe	**Nº Instâncias**	**Nº de Erros**	**% Acertos**
Ia	4.380	528	87,9%
NãoIa	15.289	7.864	48,6%
Total	19,669	8.393	57,3%

Fonte: Adaptada de Pascale (2011).

Tabela 6.32 - Matriz de Confusão do algoritmo LogitBoost para duas classes.

	LogitBoost		
Classe	**Tipo**	Ia	NãoIa
	Ia	3.852	528
	NãoIa	7.864	7.425

Fonte: Produção do autor.

A partir da matriz de confusão pode-se calcular o coeficiente Kappa = 0,23 e os valores de Acurácia = 0,57, Precisão = 0,33 e Recall = 0,88.

O desempenho mostra que o maior percentual de acertos foi para o tipo Ia e que existe uma diferença muito grande entre o percentual de acerto das duas classes. Mostram também que o algoritmo Discriminação Logística obteve melhor resultado para a classe “Ia”, mas resultado pior para a classe “NãoIa”.

6.3.4. Classificador com Redes Neurais Artificiais

O classificador desenvolvido pelo *Department of Physics of the Stockholm University* e pelo *Astrophysics Group of Cavendish Laboratory* usa redes neurais Perceptron de Múltiplas Camadas (MLP). Ele foi treinado e testado usando um conjunto de 20.895 curvas de luz que é composto por 1103 curvas de luz de supernovas cujos tipos foram confirmados pela análise do espectro e 19.792 curvas de luz simuladas. O banco contém supernovas dos tipos: Ia, Ib, Ic, IIn, II-P e II-L, mas separa as supernovas em apenas duas classes: tipo Ia e tipo NãoIa.

O conjunto foi dividido de maneiras diferentes para treinamento e teste das redes neurais. Foram feitos seis testes diferentes de acordo com o tamanho do conjunto de treinamento. Cada teste usou um dos seis conjuntos de treinamento com quantidades diferentes, contendo 5%, 10%, 20%, 30%, 40% ou 50% dos dados. Os testes usaram uma rede neural MLP com uma camada escondida de 500 neurônios. O melhor resultado foi de 88% de acerto, conseguido pelo conjunto de treinamento com 50% dos dados.

O detalhamento do melhor desempenho divulgado para esse classificador é mostrado na Tabela 6.33 e a matriz de confusão na Tabela 6.34.

Tabela 6.33 - Desempenho da rede neural MLP para duas classes.

Classe	Nº instâncias	Nº de Erros	% Acertos
Ia	2.612	313	88,0%
NãoIa	7.836	Não divulgado	-
Total	10.448	-	-

Fonte: Adaptada de Karpenka; Feroz e Hobson (2013).

Tabela 6.34 - Matriz de Confusão da rede neural MLP para duas classes.

	Rede Neural MLP		
Classe	**Tipo**	Ia	NãoIa
	Ia	2.299	313
	NãoIa	-	-

Fonte: Produção do autor.

Como a matriz de confusão não está completa porque o desempenho da classificação dos outros tipos de supernovas não foi divulgado, o coeficiente Kappa e os valores de Acurácia e Precisão não puderam ser calculados, apenas o valor de Recall = 0,88.

6.3.5. Comparação do desempenho dos classificadores pesquisados com a versão atual do CIntIa

Esta seção compara o desempenho dos classificadores pesquisados, que analisam a curva de luz, com a versão atual do CIntIa, que analisa o espectro. Importante ressaltar que cada classificador usou um conjunto de teste diferente dos demais e que fizemos testes apenas para o CIntIa, pois apenas usamos o desempenho divulgados pelos autores dos outros classificadores.

Todos os trabalhos pesquisados mostram desempenho dos seus classificadores para duas classes: tipo Ia e tipo NãoIa. Assim, os tipos Ib, Ic e II estão agrupados como outros tipos em uma mesma classe. Apenas o trabalho, que usa os algoritmos Discriminação Logística Linear e LogitBoost (RODNEY; TONRY, 2009), além de desempenho para duas classes, também mostra o desempenho dos seus classificadores para três classes: Ia, II e Ibc. Ou seja, o tipo Ib e o tipo Ic estão agrupados na mesma classe.

A versão atual do CIntIa usa quatro redes neurais diferentes para classificação de cada um dos tipos “clássicos”: Ia, Ib, Ic e II. Mas para fazer a comparação com o desempenho dos classificadores pesquisados será utilizada apenas a 1ª redes neural que classifica entre tipo Ia e tipo NãoIa. Para facilitar a comparação, a Tabela 6.35 mostra os resultados dessa rede neural da versão atual do CIntIa no formato das tabelas que foram apresentadas para o desempenho dos classificadores pesquisados.

Tabela 6.35 - Resultados da rede neural que identifica espectros do tipo Ia na versão atual do CIntIa.

Classe	**Nº espectros**	**Nº de Erros**	**% Acertos**
Ia	106	0	100,0%
NãoIa	12	1	91,7%
Total	118	1	99,2%

Fonte: Produção do autor.

A Tabela 6.36 mostra a comparação do desempenho para duas classes dos sete algoritmos encontrados nos quatro trabalhos pesquisados com os resultados da versão atual do CIntIa.

Tabela 6.36 - Comparação dos resultados da versão atual do CIntIa com todos os classificadores pesquisados para duas classes: tipo Ia e tipo NãoIa.

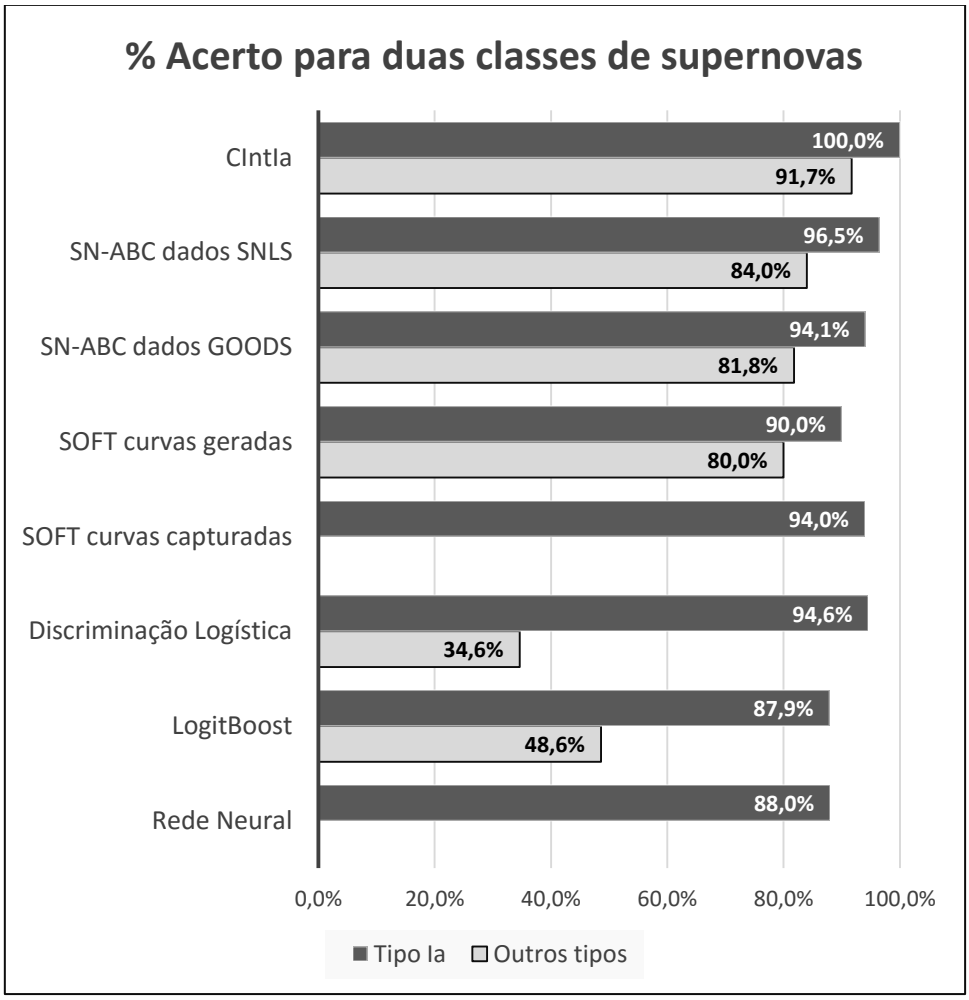
Classe	% Acertos							
	CIntIa	**SN-ABC dados SNLS**	**SN-ABC dados GOODS**	**SOFT curvas geradas**	**SOFT curvas captura-das**	**Discrimi-nação Logística**	**Logit Boost**	**Rede Neural**
Ia	100,0%	96,5%	94,1%	~90,0%	94,0%	94,6%	87,9%	88,0%
NãoIa	91,7%	84,0%	81,8%	~80,0%	Não	34,6%	48,6%	Não

Fonte: Produção do autor.

A Figura 6.5 mostra um gráfico com os mesmos resultados da Tabela 6.36 para facilitar a visualização na comparação dos resultados.

Analisando os resultados da Tabela 6.36 e da Figura 6.5 pode-se observar que o percentual de acerto para a classe “NãoIa” varia muito entre os classificadores pesquisados. O trabalho que usa os algoritmos Discriminação Logística Linear e LogitBoost (PASCALE, 2011) teve seus dois algoritmos com o pior desempenho para essa classe. O trabalho que usa Logica Nebulosa (RODNEY; TONRY, 2009) não divulgou o percentual de acerto da classe “NãoIa” para o classificador Soft com Curvas Capturadas. O classificador Soft com Curvas Geradas obteve um percentual de acerto (80%) para essa classe que é bem próximo dos percentuais de acerto obtidos pelo método que usa Algoritmo Bayesiano (POZNANSKI; MAOZ; GAL-YAM, 2007) nos seus dois classificadores: SN-ABC com dados do conjunto GOODS (81,8%) e SN-ABC com dados do conjunto SNLS (84,0%). Esse último foi o melhor de todos os classificadores pesquisados, mas ainda foi inferior ao percentual de acerto obtido pela versão atual do CIntIa (91,7%).

Figura 6.5 - Comparação do percentual de acerto da versão atual do CIntIa com todos os classificadores pesquisados para duas classes: Ia e NãoIa



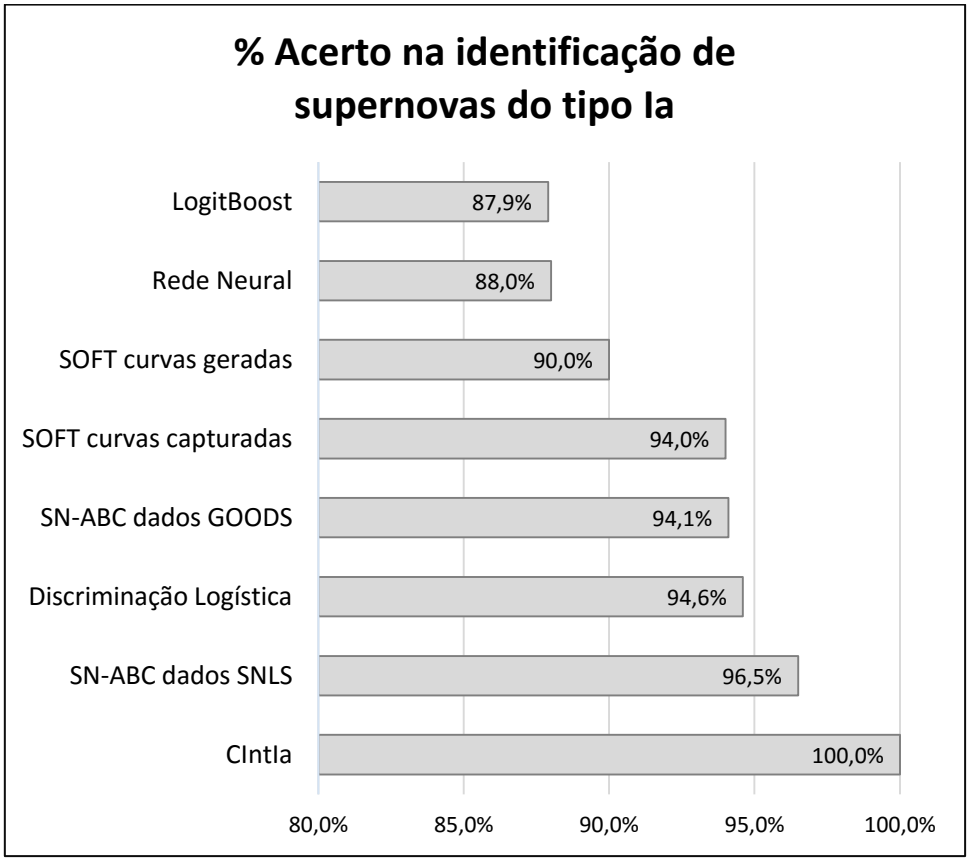
Fonte: Produção do autor.

O percentual de acerto para supernovas do tipo Ia mostrado na Tabela 6.36 e na Figura 6.5 foi isolado na Figura 6.6 porque esse tipo de supernova tem um interesse especial devido ao envolvimento deste trabalho com o Projeto KDUST (CHINESE CENTER FOR ANTARCTIC ASTRONOMY, 2010). Na Figura 6.6 os resultados são apresentados em ordem crescente de percentual de acerto na classificação da supernova tipo Ia.

A Figura 6.6 mostra que o percentual do algoritmo LogitBoost (83,4%) (PASCALE, 2011) está muito abaixo dos demais classificadores. O percentual

de acerto do classificador que usa Rede Neural (88,0%) (KARPENKA; FEROZ; HOBSON, 2013) está bem próximo do classificador SOFT (RODNEY; TONRY, 2009) com curvas geradas (90,0%), que usa Lógica Nebulosa. Enquanto que o percentual de acerto do classificador SOTF com curvas capturadas (94,0%) foi praticamente igual ao percentual do classificador SN-ABC com conjunto de dados GOODS (94,1%) (POZNANSKI; MAOZ; GAL-YAM, 2007), e ambos estão muito próximos do percentual do algoritmo Discriminação Logística (94,6%) (PASCALE, 2011).

Figura 6.6 - Comparação do percentual de acerto dos classificadores pesquisados com a versão atual do CIntIa para supernovas do tipo Ia



Fonte: Produção do autor.

O melhor resultado entre os classificadores pesquisados foi também do método SN-ABC, mas com o conjunto de dados SNLS (96,5%). No entanto, o

percentual de acerto na identificação de supernovas do tipo Ia obtido pela versão atual do CIntIa (100,0%) foi o maior entre todos os classificadores.

Foi feita também uma comparação dos índices calculados a partir da matriz de confusão de cada um dos classificadores. Essa comparação pode ser vista na Tabela 6.37.

Tabela 6.37 - Comparação dos índices da versão atual do CIntIa com todos os classificadores pesquisados para duas classes: tipo Ia e tipo NãoIa.

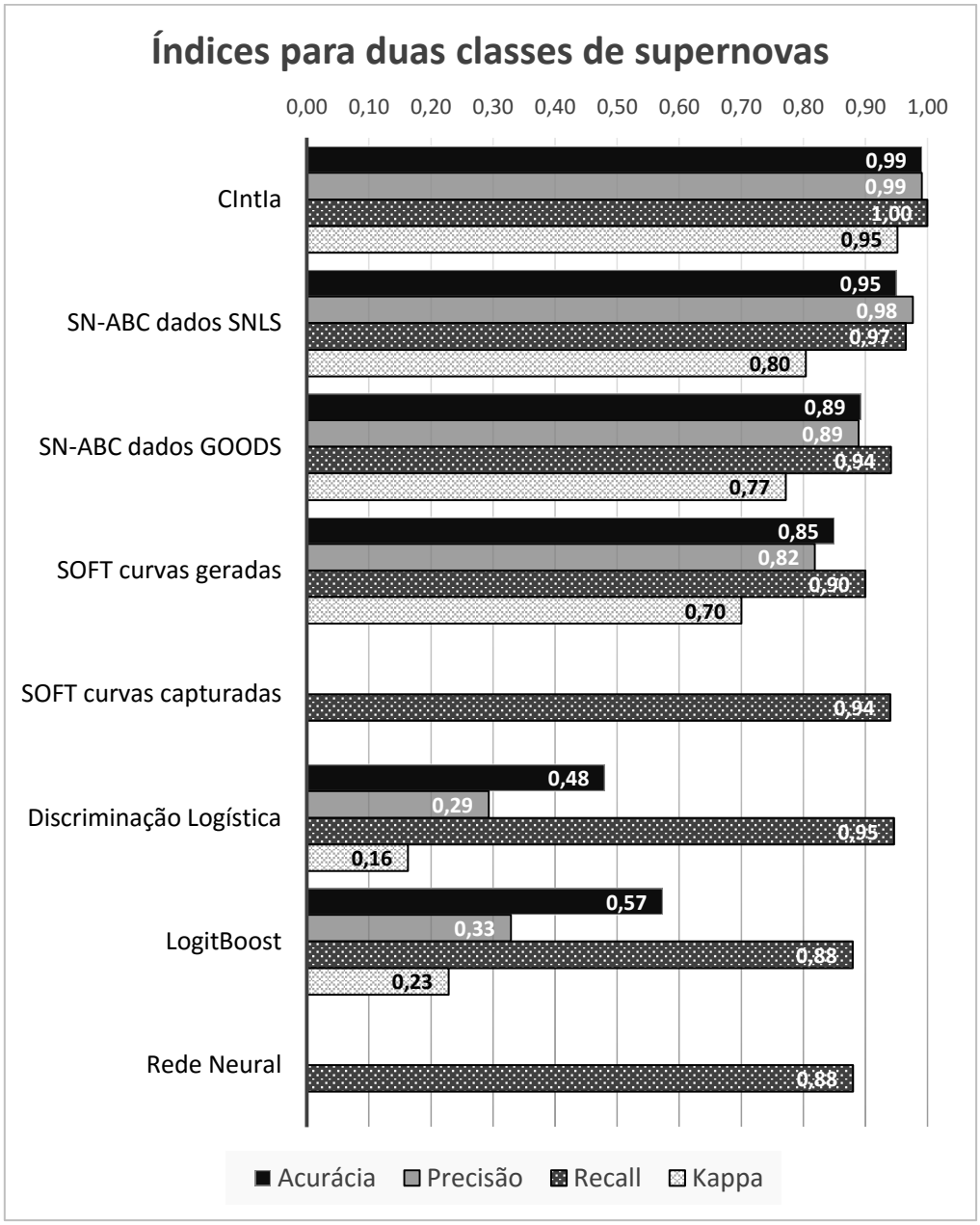
Índice	Classificador							
	CIntIa	**SN-ABC dados SNLS**	**SN-ABC dados GOODS**	**SOFT curvas geradas**	**SOFT curvas captura-das**	**Discrimi-nação Logística**	**Logit Boost**	**Rede Neural**
Acurácia	0,99	0,95	0,89	0,85	---	0,48	0,57	---
Precisão	0,99	0,98	0,89	0,82	---	0,29	0,33	---
Recall	1,00	0,97	0,94	0,90	0,94	0,95	0,88	0,88
Kappa	0,95	0,80	0,77	0,70	---	0,16	0,23	---

Fonte: Produção do autor.

Para facilitar a visualização na comparação dos resultados apresentados na Tabela 6.37, foi construído um gráfico que é mostrado na Figura 6.7.

Analisando os resultados da Tabela 6.37 e da Figura 6.7 pode-se observar que o Recall para todos os classificadores é maior que 0,80. Isso significa que todos têm um percentual de acerto maior que 80% na identificação correta das supernovas que são do tipo Ia. Entretanto, o CIntIa foi o único que identificou corretamente todas as supernovas do tipo Ia.

Figura 6.7 - Comparação dos índices da versão atual do CIntIa com todos os classificadores pesquisados para duas classes: Ia e NãoIa



Fonte: Produção do autor.

Os outros índices não foram possíveis de calcular para todos os classificadores pesquisados, devido à falta de algumas informações do seu desempenho. Mas, entre aqueles que foi possível fazer o cálculo, pode-se verificar na Figura

6.7 que para os três índices existe a mesma relação entre os classificadores. O trabalho que usa os algoritmos Discriminação Logística Linear e LogitBoost (PASCALE, 2011) teve os menores valores para os três índices, enquanto que os índices mais altos entre os classificadores pesquisados foram do trabalho usa Algoritmo Bayesiano (POZNANSKI; MAOZ; GAL-YAM, 2007) nos seus dois classificadores: SN-ABC com dados do conjunto GOODS e SN-ABC com dados do conjunto SNLS.

Finalmente, observa-se que o CIntIa obteve os maiores valores em todos os índices utilizados na avaliação dos resultados dentre todos classificadores comparados. Isso significa que os melhores resultados na classificação de supernovas do tipo Ia foram obtidos pelo CIntIa.

7 CONCLUSÃO

Neste trabalho foi proposto um método de classificação automática de supernovas que analisa espectros capturados próximos a luz máxima e utiliza Redes Neurais Artificiais para simular a análise do especialista humano na identificação de alguns elementos que determinam o tipo da supernova.

Foram construídas e testadas várias versões do classificador automático, denominado CIntIa (Classificador Inteligente de supernovas do tipo Ia). Os resultados alcançados pela última versão do CIntIa foram muito bons e indicam a viabilidade de sua utilização para os objetivos propostos.

O maior percentual de sucesso obtido foi na identificação de supernovas do tipo Ia, sendo que, todos os espectros de supernovas desse tipo foram identificados corretamente. O único erro foi em um espectro do tipo Ic que foi identificado erradamente como sendo do tipo Ia. Esse resultado é muito importante no contexto deste trabalho, pois atinge o objetivo do classificador automático proposto para o Projeto KDUST que é separar as supernovas do tipo Ia dos outros tipos com rapidez e precisão. O Projeto KDUST pretende investigar a expansão cósmica acelerada do universo a partir do estudo das supernovas do tipo Ia.

Os percentuais de acerto obtidos pela versão atual do CIntIa foram comparados com o desempenho divulgado em quatro outros trabalhos que também construíram e testaram classificadores automáticos de supernovas. A comparação foi feita para a identificação das supernovas do tipo Ia, sendo que, todos os outros tipos foram agrupados em uma única classe (Não Ia). Os resultados obtidos pela versão atual do CIntIa foram melhores em todas as comparações feitas com esses classificadores.

Analisando as quantidades de espectros utilizados no treinamento e teste da versão atual do CIntIa, verifica-se que 86% são tipo Ia e apenas 14% são outros tipos (Ib, Ic e II), ou seja, o número de espectros de supernovas do tipo

Ia é muito maior que todos os outros tipos juntos. Isso significa que um número grande de espectros do tipo Ia foi utilizado nos testes, o que valida os resultados obtidos na identificação desse tipo de supernova, mas, por outro lado, o pequeno número de espectros dos outros tipos indica a necessidade de inclusão de mais espectros para uma melhor validação dos resultados obtidos na identificação das supernovas dos tipos não Ia.

Principais Contribuições

O método de classificação automática de supernovas que analisa espectros capturados próximos a luz máxima é uma das principais contribuições deste trabalho. Mas, além da criação do método, sua implementação através do desenvolvimento do CIntIa, que usa Redes Neurais Artificiais para simular a análise do especialista humano, também é uma contribuição original importante.

Outra contribuição original é a redução da subjetividade existente na análise humana, tornando a classificação mais homogênea devido a sistematização feita durante o processo de automatização, permitindo que a classificação de supernovas seja feita em situações onde não existe um astrônomo especialista ou onde seja necessária a classificação automática, sistemática e contínua.

A sistematização feita na coleta de dados resultou em mais uma contribuição original que foi a indicação de alguns parâmetros necessários para a classificação automática dos espectros. Seguir esses parâmetros na captura de novos espectros, permite que a classificação automática possa ser feita com sucesso. Espera-se e recomenda-se que o Projeto KDUST siga essas recomendações.

Trabalhos Futuros

A inclusão de mais espectros, principalmente dos tipos Ib, Ic e II, surge como continuidade imediata para este trabalho, pois permite uma melhor validação

dos resultados obtidos na identificação desses tipos de supernova, além de propiciar que outros subtipos, identificados pela análise do espectro, possam fazer parte da classificação. Essa inclusão pode ser feita, por exemplo, com os espectros do banco SUSPECT que foi utilizado nos experimentos anteriores, ajudando também a obter resultados melhores e mais significativos.

Outro trabalho futuro tem relação aos espectros usados para treinamento e teste do CIntIa, que foram selecionados o mais próximo possível da luz máxima, ou seja, capturados no intervalo de três dias antes até sete dias depois da luz máxima. Aumentar esse intervalo de dias vai permitir usar um número maior de supernovas e acrescentar mais espectros das supernovas já utilizadas, aumentando os conjuntos de treinamento e teste. Uma proposta é aumentar gradualmente o intervalo de dias até encontrar o intervalo máximo que mantem ou até melhora os resultados, indicando para futuras pesquisas o intervalo que é o limite no qual os espectros podem ser utilizados para classificação. Ainda com relação aos espectros usados, pode-se testar a robustez da classificação utilizando dois bancos de dados diferentes, um para realizar o treinamento e outro para fazer os testes.

A variação dos parâmetros de pré-processamento também é uma das possibilidades para melhoria dos resultados, especialmente a suavização, pois visualmente alguns gráficos parecem precisar de suavização maior que outros. Assim, uma possibilidade de melhoria é a utilização de valores diferentes de suavização de acordo com a frequência das linhas do espectro.

Especificamente para o Projeto KDUST, uma sugestão de trabalho futuro é fazer uma análise das características dos instrumentos usados no projeto e utilizar essas características para gerar dados sintéticos mais específicos. O classificador usaria esses dados sintéticos, que seriam corrompidos com ruído também sintético, para um treinamento com mais restrições, voltadas às características desses instrumentos.

A possibilidade de sugerir melhoras nos esquemas de classificação existentes, também pode fazer parte da continuidade deste trabalho, pois ainda hoje subtipos têm sido introduzidos, excluídos e subdivididos. As taxonomias existentes ainda são consideradas insatisfatórias, por serem ambíguas, não exaustivas e resultarem em muitas supernovas classificadas como peculiares. Essas características levam a duas possibilidades importantes na classificação de supernovas: (1) pode surgir um novo tipo (ou subtipo) de supernova a partir de novas descobertas, quando os critérios de classificação são revisitados; e (2) o esquema de classificação pode mudar devido à identificação de novas características no espectro. Assim, a continuação deste trabalho pode também analisar as características comuns dos espectros das supernovas, usando, por exemplo, Redes Neurais de Kohonen para buscar uma taxonomia que possa incluir até mesmo as supernovas peculiares.

Publicações realizadas

- MÓDOLO, M.; GUIMARÃES, L. N. F.; ROSA, R. R. **Supernovae Automatic Classification Method by Modeling Human Analysis Using Artificial Neural Networks**. Aceito pelo International Conference on Nonlinear Science and Complexity, 6., 2016

- MÓDOLO, M.; GUIMARÃES, L. N. F.; ROSA, R. R. **Automatic Method of Supernovae Classification by Modeling Human Procedure of Spectrum Analysis.** Aceito pelo COSPAR Scientific Assembly, 41,. 2016.

- MÓDOLO, M.; GUIMARÃES, L. N. F.; ROSA, R. R. Supernovae Automatic Classification Method through Artificial Neural Networks in Spectrum Analysis. Aceito pela revista **Journal of Computational Interdisciplinary Sciences,** v. 6, n. 2. Ago 2015.

- MÓDOLO, M.; GUIMARÃES, L. N. F.; ROSA, R. R. **Utilização de Diferentes Parâmetros para Testes de um Classificador Automático**

de Supernovas. In: Congresso Metodista de Iniciação e Produção Científica, 18., 2015, São Paulo. XVIII Congresso Metodista de Iniciação e Produção Científica, 2015. ISBN 978-85-7814-318-3.

- MÓDOLO, M.; GUIMARÃES, L. N. F.; ROSA, R. R. **Resultados de Experimentos usando Redes Neurais Artificiais na Identificação dos Tipos Clássicos de Supernovas.** In Workshop de Computação Aplicada, 15., 2015, São José dos Campos. XV WORCAP, 2015.

- MÓDOLO, M.; GUIMARÃES, L. N. F.; ROSA, R. R. **Identificação Automática do Tipo da Supernova a partir da Análise do seu Espectro.** In: Simpósio de Pesquisa do Grande ABC, 3., 2014, São Bernardo do Campo. SPGABC, 2014.

- MÓDOLO, M.; GUIMARÃES, L. N. F.; ROSA, R. R. **Classificação Automática de Supernovas usando Redes Neurais e Sistemas Especialistas.** In Workshop de Computação Aplicada, 14., 2014, São José dos Campos. XVI WORCAP, 2014.

- MÓDOLO, M.; GUIMARÃES, L. N. F.; ROSA, R. R. **Experimentos para Criação de um Controlador de Telescópios e de um Classificador de Supernovas usando Redes Neurais Artificiais.** In: Simpósio de Pesquisa do Grande ABC, 2., 2013, São Bernardo do Campo. SPGABC, 2013.

- MÓDOLO, M.; GUIMARÃES, L. N. F.; ROSA, R. R. **Primeiros Experimentos para Construção de um Classificador de Supernovas e de um Controlador de Telescópio.** In Workshop de Computação Aplicada, 13., 2013, São José dos Campos. XIII WORCAP, 2013.

- MÓDOLO, M.; GUIMARÃES, L. N. F. **Sistemas Inteligentes para Controle de Veículos Espaciais Autônomos Baseados em Paradigmas de Inteligência Computacional a Exemplo de Lógica**

Nebulosa. In Workshop de Computação Aplicada, 12., 2012, São José dos Campos. XII WORCAP, 2012.

REFERÊNCIAS BIBLIOGRÁFICAS

ANOCHI, J. A. **Previsão Climática de Precipitação por Redes Neurais Autoconfiguradas**. 2015. 159 p. (sid.inpe.br/mtc-m21b/2015/09.16.22.02-TDI). Tese (Doutorado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais. São José dos Campos. 2015. Disponível em: <http://urlib.net/8JMKD3MGP3W34P/3K98PDP>. Acesso em: 04 maio 2016.

BARBON, R. **Padova-Asiago Supernova Group**, 2010. Disponível em: <graspa.oapd.inaf.it>. Acesso em: 15 jun. 2015.

BERNAT, A. P. **An intelligent object recognizer and classification system**. Instrumentation in Astronomy. Bellingham: SPIE. 1986. p. 89-94.

BITTENCOURT, G. **Inteligência Artificial:** Ferramentas e Teorias. 3. ed. Florianópolis: Editora da UFSC, 2006. ISBN 85-328-0138-2.

BLONDIN, S. et al. The Spectroscopic Diversity of Type Ia Supernovae. **The Astronomical Journal**, v. 143, p. 126-158, maio 2012. ISSN 1538-3881.

BLONDIN, S.; TONRY, J. L. Determining the Type, Redshift, and Age of a Supernova Spectrum. **The Astrophysical Journal**, v. 666, p. 1024-1047, set. 2007.

BRAGA, A. D. P.; CARVALHO, A. C. P. D. L. F.; LUDEMIR, T. B. **Redes Neurais Artificiais:** Teoria e Aplicações. 2. ed. Rio de Janeiro: LTC, 2000.

BRANDT, T. D. et al. The Ages of Type Ia Supernova Progenitors. **The Astronomical Journal**, v. 140, p. 804-816, 2010.

CHINESE CENTER FOR ANTARCTIC ASTRONOMY. KDUST. **Kunlun Dark Universe Survey Telescopes**, 2010. Disponível em: <http://www.kdust.org/KDUST/KDUST.html>. Acesso em: 25 ago. 2013.

CHOI, C. Q. **White Dwarfs May Hold Nuclear Trigger for Explosive Supernovas**, 2014. Disponível em: <http://www.space.com/26960-exploding-white-dwarf-stars-supernova-trigger.html>. Acesso em: 07 abr. 2016.

CHRISTENSEN-DALSGAARD, J. **Lecture notes on stellar structure and evolution**. Institut for Fysik og Astronomi - Aarhus Universitet. Aarhus, p. 246. 2008.

EISBERG, R.; RESNICK, R. **Física Quantica:** Átomos, Moléculas, Sólidos, Núcleos e Partículas. Tradução de Paulo Costa Ribeiro; Enio Frota Silveira; Marta Feijó Barroso. Rio de Janeiro: Campus, 1979. ISBN 85-7001-025-7.

FOLEY, R. J. et al. Type Iax Supernovae: A New Class of Stellar Explosion. **The Astrophysical Journal**, v. 767, p. 57, 2013.

GAL-YAM, A. et al. Photometric Identification of Young Stripped-Core Supernovae. **The Astronomical Society of the Pacific**, v. 116, 2004.

GIUNTI, C.; KIM, C. W. **Fundamentals of Neutrino Physics and Astrophysics**. Nova York: Oxford University Press, 2007.

HARUTYUNYAN, A. **Automatic Objective Classification of Supernovae**. 2008. 133 f. Tese (Doutorado em Astronomia) - Dipartimento di Astronomia, Università Degli Studi di Padova. Padova. 2008.

HARVARD-SMITHSONIAN CENTER FOR ASTROPHYSICS. The CfA Supernova Data Archive. **The CfA Supernova Group**, 2010. Disponível em: <https://www.cfa.harvard.edu/supernova/SNarchive.html>. Acesso em: 06 jun. 2015.

HAYKIN, S. **Redes Neurais:** princípios e práticas. Tradução de Paulo Martins Engel. 2. ed. Porto Alegre: Bookman, 2001. ISBN 85-7307-718-2.

HILLEBRANDT, W.; NIEMEYER, J. C. Type Ia Supernova Explosion Models. **Annu. Rev. Astron. Astrophys**, v. 38, p. 191-230, 2000.

HOGAN, M.; PARRENT, J.; FELDT, A. **SUSPECT - The Online Supernova Spectrum Archive**, 2010. Disponível em: <http://www.nhn.ou.edu/~suspect/>. Acesso em: 10 jun. 2013.

HOWELL, D. A. et al. Gemini Spectroscopy of Supernovae from the Supernova Legacy Survey: Improving High-Redshift Supernova Selection and Classification. **The Astrophysical Journal**, v. 634, p. 1190, 2005.

KARPENKA, N. V.; FEROZ, F.; HOBSON, M. P. A simple and robust method for automated photometric classification of supernovae using neural networks. **Monthly Notices of the Royal Astronomical Society**, v. 429, p. 1278-1285, 2013.

KOBAYASHI, C.; KEN'ICHI; HACHISU, I. Subclassesof Type Ia Supernovae as the origin of [α/Fe] Ratios in Dwarf Sfheroidal Galaxies. **The Astrophysical Journal Letters**, v. 804, 2015. DOI: 10.1088/2041-8205/804/1/L24.

KUZNETSOVA, N. V.; CONNOLLY, B. M. A Probabilistic Approach to Classifying Supernovae Using Photometric Information. **The American Astronomical Society**, v. 659, p. 530-540, 2007.

LAGO, B. L. **Contribuições conceituais e estatisticas para a análise de supernovas do tipo Ia**. 2011. 100p. Tese (Doutorado em Física) - Instituto de Física da Universidade Federal do Rio de Janeiro. Rio de Janeiro. 2011.

LANDIS, J. R.; KOCH, G. G. The Measurement of Observer Agreement for Categorical Data. **Biometrics**, v. 33, n. 1, p. 159-174, mar. 1977.

LUGER, G. F. **Inteligência Artificial:** Estruturas e estratégias para a resolução de problemas complexos. 4. ed. Porto Alegre: Bookmann, 2004. ISBN 85-363-0396-4.

MILNE, P. A. et al. Grouping Normal Type Ia SUpernovae by UV to Optival Color DIfferencess. **The Astrophysical Journal**, v. 779, 2013. DOI: 10.1088/0004-637X/779/1/23.

MODJAZ, M. et al. Optical Spectra of 73 Stripped-envelope Core-collapse Supernovae. **The Astronomical Journal**, v. 147, p. 99-115, maio 2014. ISSN 1538-3881.

NATIONAL SCIENCE FOUNDATION. **UC Davis ChemWiki**, 2013. Disponível em: <http://chemwiki.ucdavis.edu/Core/Physical_Chemistry/Nuclear_Chemistry/Nucleosynthesis%3A_The_Origin_of_the_Elements>. Acesso em: 07 abr. 2016.

PASCALE, M. D. **Automatic Classification of Supernovae using machine learning methods**. Dissertação (Mestrado em Astronomia) - Università degli Studi di Padova. Padova, p. 75. 2011.

PERLMUTTER, S. et al. Measurements of Ω and Λ from 42 High-Redshift Supernovae. **The Astrophysical Journal**, v. 517, p. 565-586, jun. 1999.

POST, R. S. Automated Supernova Discovery. **The Journal of the American Association of Variable Star Observers**, v. 43, p. 259, dez. 2015.

POZNANSKI, D. et al. Not Color-Blind: Using Multiband Photometry to Classify Supernovae. **The Astronomical Society of the Pacific**, v. 114, n. 798, p. 833-845, 2002.

POZNANSKI, D.; MAOZ, D.; GAL-YAM, A. Bayesian Single-Epoch Photometric Classification of Supernovae. **The Astronomical Journal**, 134, 31 jul. 2007. 1285-1297. Disponível em: <http://stacks.iop.org/1538-3881/134/i=3/a=1285>. Acesso em: 24 fev. 2016.

QUILES, M. G. **Sistema de Visão Baseado em Redes Neurais para o Controle de Robôs Móveis**. 2004. Dissertação (Mestrado em CIência da Computação e Matemática Computacional) - Universidade de São Paulo. São Paulo. 2004.

REZENDE, S. O. **Sistemas Inteligentes:** Fundamentos e Aplicações. 1. ed. Barueri: Manole, 2003. ISBN 85-204-1683-7.

RIESS, A. G. et al. Type Ia Supernova Discoveries at z > 1 from the Hubble Space Telescope: Evidence for Past Deceleration and Constraints on Dark Energy Evolution. **The Astrophysical Journal**, v. 607, p. 665-687, 2004.

RIESS, A. G.; PRESS, W. H.; KIRSHNER, R. P. A precise distance indicator: Type Ia supernova multicolor light-curve shapes. **The Astrophysical Journal**, v. 473, p. 88, 1996.

RODNEY, S. A.; TONRY, J. L. Fuzzy Supernova Templates. I. Classification. **The Astrophysical Journal**, v. 707, p. 1064-1079, 02 dez. 2009. Disponível em: <http://stacks.iop.org/0004-637X/707/i=2/a=1064>. Acesso em: 24 fev. 2016.

RUSSEL, S.; NORVIG, P. **Inteligência Artificial**. Tradução de Regina Célia Simille. 3. ed. Rio de Janeiro: Elsevier, 2013. ISBN 978-85-352-3701-6.

SULLIVAN, M. et al. Photometric Selection of High-Redshift Type Ia Supernova Candidates. **The American Astronomical Society**, v. 131, p. 960-972, 2006.

SWINBURNE UNIVERSITY OF TECHNOLOGY. **COSMOS - The SAO Encyclopedia of Astronomy**, 2015. Disponível em: <http://astronomy.swin.edu.au/cosmos/T/Type+II+supernova+light+curves>. Acesso em: 30 mar. 2016.

THIEBAUT, C.; BOËR, M.; ROQUES, S. **Steps towards the development of an automatic classifier for astronomical sources**. Astronomical Data Analysis II. Bellingham : SPIE. 2002. p. 379-390.

TOMASELLA, L. et al. Asiago Supernova classification program: blowing out the first two hundreds candles. **Astronomische Nachrichten**, v. 335, p. 841-849, 2014.

TONRY, J.; DAVIS, M. A survey of galaxy redshifts. I-Data reduction techniques. **The Astronomical Journal**, v. 84, p. 1511-1525, 1979. ISSN 0004-6256.

TORRES, F. R. **Neutrinos de Supernova**. 2010. 135 f. Tese (Doutorado em Física) - Departamento de Raios Cósmicos e Cronologia - Universidade Estadual de Campinas. Campinas. 2010.

TURATTO, M. Classification of Supernovae. **Supernovae and Gamma-Ray Bursters**, Berlin, v. 598 de Lecture Notes in Physics, p. 21-36, jan. 2003. ISSN 978-3-540-44053-6.

TURATTO, M.; BENETTI, S.; PASTORELLO, A. Supernova classes and subclasses. **American Institute of Physics Conference Series**, v. 937, p. 187-197, jun. 2007.

WHEELER, J. C.; BENETTI, S. Supernovae. In: COX, A. N. **Allen's Astrophysical Quantities**. 4. ed. New York: Springer, 2002. Cap. 18. ISBN 978-1-4612-1186-0.

YARON, O.; GAL-YAM, A. WISeREP - An Interactive Supernova Data Repository. **Publications of the Astronomical Society of the Pacific**, v. 124, p. 668-681, 2012.

ZHAO, G.-B. et al. Probing Dark Energy with the Kunlun Dark Universe Survey Telescope. **Publications of the Astronomical Society of the Pacific**, v. 123, n. 904, p. 725-734, jun. 2011.

APÊNDICE A – CÓDIGO DOS PROGRAMAS ESCRITOS NO MATLAB PARA O PRÉ-PROCESSAMENTO DOS ESPECTROS

Três programas foram desenvolvidos usando o MATLAB para automatizar ações e viabilizar variação dos parâmetros nos testes. Os códigos desses programas são mostrados neste Apêndice. O primeiro realiza o pré-processamento do espectro e os outros dois dividem os espectros já pré-processados nos conjuntos de treinamento, validação e teste que serão usados pelas redes neurais.

A.1 Programa “CorrigeRedshiftSuavizaInterpolaNormaliza.m”

Esse programa realiza automaticamente o pré-processamento nos espectros e gera um arquivo que ainda precisa de algum processamento para servir de entrada das redes neurais.

```
%
% Normaliza para vetor de magnitude 1
% Corrige o redshift: y0 = y/(z + 1) (cálculo do redshift: z = (y - y0)/y0)
% Suaviza segundo parâmetro 'suavy' em angstroms
% Faz com que o tamanho de cada vetor varie segundo parâmetros 'inix' e 'fimx' em angstroms
% Interpola os espectros segundo parãmetro interpx em angstroms
% Gera os gráficos e salva em um arquivo
%

%%%%% Parametros %%%%%
inix = 3800 %inicio do espectro
fimx = 7000 %final do espectro
faseMinMax = '-3+7' %intervalo de dias dos espectros em relação ao dia com luz máxima
interpx = 8 %parâmetro de interpolação
suavy = 70 %parâmetro de suavização
%caminho = 'C:\Users\modolo.colaborador\Documents\Marcelo Módolo\Dropbox\Doutorado\Pesquisa\Classificador2-EspectrosdoCfA\'
caminho = 'C:\Users\Marcelo\Dropbox\Doutorado\Pesquisa\Classificador2-EspectrosdoCfA\'
cd (caminho);
```

```
nomePadrao = strcat('Int', int2str(inix));
nomePadrao = strcat(nomePadrao, '-');
nomePadrao = strcat(nomePadrao, int2str(fimx));
nomePadrao = strcat(nomePadrao, 'Fase');
nomePadrao = strcat(nomePadrao, faseMinMax);
nomePadrao = strcat(nomePadrao, 'Interp');
nomePadrao = strcat(nomePadrao, int2str(interpx));
nomePadrao = strcat(nomePadrao, 'Suav');
nomePadrao = strcat(nomePadrao, int2str(suavy));

pastaSaida = 'EspectrosPreProcessados';
pastaSaida = strcat(pastaSaida, nomePadrao);
pastaSaida = strcat(pastaSaida, '\');
if exist(pastaSaida, 'dir') == 0
   mkdir(caminho, pastaSaida);
end;
nomeSai = strcat(caminho, pastaSaida);
nomeSai = strcat(nomeSai, 'EntradaRNTodoEspec');
nomeSai = strcat(nomeSai, nomePadrao);
nomeSai = strcat(nomeSai, '.csv');
saida = fopen(nomeSai, 'w');
xx = (inix:interpx:fimx)';
fprintf(saida, '%d;', xx);
fprintf(saida, 'Tipo;');
fprintf(saida, 'Nome;');
fprintf(saida, 'Data;');
fprintf(saida, 'Inicio espectro;');
fprintf(saida, 'Final espectro;');
fprintf(saida, 'Fase;');
fprintf(saida, 'Redshift;');
fprintf(saida, 'Nome arquivo\n');

nomeArqNomesPre = strcat(caminho, pastaSaida);
nomeArqNomesPre = strcat(nomeArqNomesPre, 'NomesDosArquivosPre');
nomeArqNomesPre = strcat(nomeArqNomesPre, nomePadrao);
nomeArqNomesPre = strcat(nomeArqNomesPre, '.txt');

% salva parâmetros no arquivo de nomes dos arquivos PreProcessados
arquivoParametros = fopen('ParametrosPreProcessamento.txt', 'w');
fprintf(arquivoParametros, '%s\n', nomeArqNomesPre);
fprintf(arquivoParametros, '%d\n', inix);
fprintf(arquivoParametros, '%d\n', fimx);
fprintf(arquivoParametros, '%d\n', interpx);
fprintf(arquivoParametros, '%d\n', suavy);
fprintf(arquivoParametros, '%s\n', faseMinMax);
```

```
fprintf(arquivoParametros, '%s\n', caminho);
fprintf(arquivoParametros, '%s\n', pastaSaida);
fprintf(arquivoParametros, '%s', nomePadrao);
fclose(arquivoParametros);

% cria o nome do arquivo para salvar os gráficos e verifica se esse arquivo
% ja existe
nomeGrafico = strcat(caminho, pastaSaida);
nomeGrafico = strcat(nomeGrafico, 'GraficosSupernova');
nomeGrafico = strcat(nomeGrafico, nomePadrao);
nomeGrafico = strcat(nomeGrafico, '.ps');
if exist(nomeGrafico, 'file') > 0
   delete(nomeGrafico);
end;

% cria um vetor com as linhas dos elementos para visualização nos gráficos
vetLxx = zeros(length(xx),1);
%%%hidrogênio
ind = round((4102-xx(1))/interpx);
vetLxx(ind) = 1;
ind = round((4341-xx(1))/interpx);
vetLxx(ind) = 1;
ind = round((4861-xx(1))/interpx);
vetLxx(ind) = 1;
ind = round((6563-xx(1))/interpx);
vetLxx(ind) = 1;
%%%silício
ind = round((6150-xx(1))/interpx);
vetLxx(ind) = 1;
ind = round((6355-xx(1))/interpx);
vetLxx(ind) = 1;
%%%hélio
ind = round((5876-xx(1))/interpx);
vetLxx(ind) = 1;
ind = round((6678-xx(1))/interpx);
vetLxx(ind) = 1;
%ind = round((7065-xx(1))/interpx);
%vetLxx(ind) = 1;

arquivoPre = fopen(nomeArqNomesPre, 'w');
for contTipo = 1:2
   if contTipo == 1
      tipo = 'Ia'; %tipo da supernova (entre Ia ou NaoIa)
      %minY = -2.30498e-015; %valor mínimo entre o fluxo de todos espectros
tipo Ia
```

```
        %maxY = 2.0723012e-013; %valor máximo entre o fluxo de todos
espectros tipo Ia
    else
        tipo = 'NaoIa'; %tipo da supernova (entre Ia ou NaoIa)
        %minY = -3.3516412; %valor mínimo entre o fluxo de todos espectros tipo
Ia
        %maxY = 54.1251; %valor máximo entre o fluxo de todos espectros tipo Ia
    end;

    nomeArqNomes = strcat(caminho, 'EspectrosTipo');
    nomeArqNomes = strcat(nomeArqNomes, tipo);
    nomeArqNomes = strcat(nomeArqNomes, '\NomesDosArquivosTipo');
    nomeArqNomes = strcat(nomeArqNomes, tipo);
    nomeArqNomes = strcat(nomeArqNomes, faseMinMax);
    nomeArqNomes = strcat(nomeArqNomes, '.txt');
    arquivo = fopen(nomeArqNomes, 'r');

    nomeArqMinMax = strcat(caminho, 'MinMax');
    nomeArqMinMax = strcat(nomeArqMinMax, tipo);
    nomeArqMinMax = strcat(nomeArqMinMax, '.csv');
    saidaMinMax = fopen(nomeArqMinMax, 'w');

    cont = 0;
    while ~feof(arquivo)
        cont = cont+1;
        nomeArq = fgetl(arquivo);

        nomeArqPre = strcat('Pre', tipo);
        nomeArqPre = strcat(nomeArqPre, '_');
        nomeArqPre = strcat(nomeArqPre, nomePadrao);
        nomeArqPre = strcat(nomeArqPre, '_');
        nomeArqPre = strcat(nomeArqPre, nomeArq);
        fprintf(arquivoPre, '%s\n', nomeArqPre);
        nomeSN = fgetl(arquivo);
        fprintf(arquivoPre, '%s\n', nomeSN);
        data = fgetl(arquivo);
        fprintf(arquivoPre, '%s\n', data);
        ondaini = fgetl(arquivo);
        fprintf(arquivoPre, '%s\n', ondaini);
        ondafin = fgetl(arquivo);
        fprintf(arquivoPre, '%s\n', ondafin);
        tipoSN = fgetl(arquivo);
        fprintf(arquivoPre, '%s\n', tipoSN);
        nome_tipo_fase = strcat(int2str(cont), '- ');
        nome_tipo_fase = strcat(nome_tipo_fase, nomeSN);
        nome_tipo_fase = strcat(nome_tipo_fase,'  Tipo: ');
```

```
nome_tipo_fase = strcat(nome_tipo_fase,tipoSN);
fase = fgetl(arquivo);
fprintf(arquivoPre, '%s\n', fase);
nome_tipo_fase = strcat(nome_tipo_fase,'  Fase: ');
nome_tipo_fase = strcat(nome_tipo_fase,fase);
redshift = fgetl(arquivo);
fprintf(arquivoPre, '%s\n', redshift);
nome_tipo_fase = strcat(nome_tipo_fase,'  Redshift: ');
nome_tipo_fase = strcat(nome_tipo_fase,redshift);

nomeArqDados = strcat(caminho, '\EspectrosTipo');
nomeArqDados = strcat(nomeArqDados, tipo);
nomeArqDados = strcat(nomeArqDados, '\');
nomeArqDados = strcat(nomeArqDados, nomeArq);
arqDados = fopen(nomeArqDados, 'r');
if strcmp(tipo,'Ia')
   invDados = fscanf(arqDados, '%g %g %g', [3 inf]);
else
   invDados = fscanf(arqDados, '%g %g', [2 inf]);
end;
fclose(arqDados);

dados = invDados';
x = dados(:,1);
y = dados(:,2);

subplot(2,1,1);
set(gcf, 'position', [50 50 1200 650]);
plot(x, y);
xlim([inix, fimx]);
set(gca,'XTick',(inix:200:fimx));
title(nome_tipo_fase);
xlabel('Comprimento de Onda (Angstroms)');
ylabel('Fluxo (erg/cm2/s/A)');
grid;

% Suaviza segundo parâmetro definido em angstroms
ys = smooth(y, suavy);

%minY = min(ys);
%maxY = max(ys);
% Encontra o valor mínimo e máximo de cada vetor e salva na
% planilha para encontrar o mínimo e máximo geral
%fprintf(saidaMinMax, '%s;', nomeSN);
%fprintf(saidaMinMax, '%s;', fase);
%fprintf(saidaMinMax, '%s;', tipo);
```

```
%fprintf(saidaMinMax, '%22.20f;', minY);
%fprintf(saidaMinMax, '%22.20f\n', maxY);

% Corrige o redshift
z = str2double(redshift);
xr = x/(z + 1);

% Interpola em 1 (um) angstrom
ini = round(xr(1));
fim = round(x(length(xr)));
x1 = (ini:1:fim)';
y1 = interp1(xr, ys, x1, 'cubic');

% Faz com que o tamanho dos vetores variem segundo parâmetros de início e fim em angstroms
novoini = find(x1 == inix);
novofim = find(x1 == fimx);
yy1 = y1(novoini:novofim);
xx1 = x1(novoini:novofim);

% Normaliza para valores de 0 a 1 usando os valores mínimo e
% máximo do vetor
yn = (yy1 - min(yy1))/(max(yy1) - min(yy1));

% Interpola segundo parâmetro definido em angstroms
yy = interp1(xx1, yn, xx, 'cubic');

% Normaliza para vetor de 0 a 0.9999999999 usando os valores mínimo e
% máximo do vetor
%yy = (yi - min(yi))*0.9999999999/(max(yi) - min(yi));

% Plota o gráfico pré-processado
subplot(2,1,2);
plot(xx, yy, xx, vetLxx);
%%%hidrogênio
text(4102, 0.5, 'H');
text(4341, 0.5, 'H');
text(4861, 0.5, 'H');
text(6563, 0.5, 'H');
%%%silício
text(6150, 0.5, 'Si');
text(6355, 0.5, 'Si');
%%%hélio
text(5876, 0.5, 'He');
text(6678, 0.5, 'He');
%text(7065, 0.5, 'He');
```

```
        xlim([inix, fimx]);
        set(gca,'XTick',(inix:200:fimx));
        %%%ylim([0, 1]);
        nome_tipo_fase = strcat(nome_tipo_fase, '  PreProcessado');
        title(nome_tipo_fase);
        xlabel('Comprimento de Onda (Angstroms)');
        ylabel('Fluxo (erg/cm2/s/A)');
        grid;
        print ('-append', nomeGrafico);


        % Grava dados no arquivo CSV
        fprintf(saida, '%12.10f;', yy');
        fprintf(saida, '%s;', tipoSN);
        fprintf(saida, '%s;', nomeSN);
        fprintf(saida, '%s;', data);
        fprintf(saida, '%s;', ondaini);
        fprintf(saida, '%s;', ondafin);
        fprintf(saida, '%s;', fase);
        fprintf(saida, '%s;', redshift);
        fprintf(saida, '%s\n', nomeArq);

        nomeArqSai = strcat(caminho, pastaSaida);
        nomeArqSai = strcat(nomeArqSai, nomeArqPre);

        saiArq = fopen(nomeArqSai, 'w');
        for i = 1:length(xx);
           fprintf(saiArq, '%d\t%12.10f\n', xx(i), yy(i));
        end;
        fclose(saiArq);
    end;
    fclose(arquivo);
    fclose(saidaMinMax);
end;
fclose(arquivoPre);
fclose(saida);
```

A.2 Programa “CriaCSVcomIntervalosDoEspectro.m”

Esse programa realiza automaticamente a seleção das partes do espectro que serão utilizadas pela 3ª versão do CIntIa e divide os espectros nos conjuntos de treinamento, validação e teste para serem usados como entrada das redes neurais.

```
%
% Cria um arquivo .csv com as linhas do elemento usado para
% treinar ou testar a Rede Neural que verifica a presença do elemento
%

intervalo = 150; %parâmetro dos limites em relação aos picos e vale

arquivoParam = fopen('ParametrosPreProcessamento.txt', 'r');
nomeArqNomesPre = fgetl(arquivoParam); %nome do arquivo com os nomes dos arquivos preprocessados
inix = str2double(fgetl(arquivoParam)); %inicio do espectro
fimx = str2double(fgetl(arquivoParam)); %final do espectro
interpx = str2double(fgetl(arquivoParam)); %parâmetro de interpolação
suavy = fgetl(arquivoParam); %parâmetro de suavização
faseMinMax = fgetl(arquivoParam); %fase inicial e fase final do espectro
caminho = fgetl(arquivoParam); %caminho base dos arquivos
pastaSaida = fgetl(arquivoParam); %nome da pasta onde foram salvos os arquivos preprocessados
nomePadrao = fgetl(arquivoParam); %padrao de nome usado nos arquivos(fase, interpolação e suavização)
fclose(arquivoParam);

cont = 0;
vetIa = {''};
contIa = 1;
vetIb = {''};
contIb = 1;
vetIc = {''};
contIc = 1;
vetIIc = {''};
contIIc = 1;

arquivo = fopen(nomeArqNomesPre, 'r');
while ~feof(arquivo)
    cont = cont + 1
    nomeArq = fgetl(arquivo);
```

```
   nomeSN = fgetl(arquivo);
   data = fgetl(arquivo);
   ondaini = fgetl(arquivo);
   ondafin = fgetl(arquivo);
   tipoSN = fgetl(arquivo);
   fase = fgetl(arquivo);
   redshift = fgetl(arquivo);

   %Cria um vetor com os nomes das supernovas para cada tipo de supernova
   if strcmp(tipoSN(1:2), 'Ia')
      if  any(strcmp(vetIa,nomeSN)) == 0 %retorna 1 se qualquer elemento do
vetor for diferente de zero
         vetIa(contIa,1) = {nomeSN};
         contIa = contIa+1
      end;
   else
      if strcmp(tipoSN(1:2), 'Ib')
         if  any(strcmp(vetIb,nomeSN)) == 0 %retorna 1 se qualquer elemento do
vetor for diferente de zero
            vetIb(contIb,1) = {nomeSN};
            contIb = contIb+1
         end;
      else
         if strcmp(tipoSN(1:2), 'Ic')
            if  any(strcmp(vetIc,nomeSN)) == 0 %retorna 1 se qualquer elemento
do vetor for diferente de zero
               vetIc(contIc,1) = {nomeSN};
               contIc = contIc+1
            end;
         else %if tipo = IIc
            if  any(strcmp(vetIIc,nomeSN)) == 0 %retorna 1 se qualquer elemento
do vetor for diferente de zero
               vetIIc(contIIc,1) = {nomeSN};
               contIIc = contIIc+1
            end;
         end;
      end;
   end;
end;
fclose(arquivo);

%Hidrogênio: Distinção entre tipos I e II
    %Série de Balmer: 6563(H?), 4861(H?), 4341(H?) e 4102(H?) angstroms
%Silício: Distinção entre Ia e outros tipo I
    %Pico: 6150 e Vale: 6355 angstroms
%Helio: Distinção entre Ib e Ic
```

```
    % usa He I 5876 e também He I 6678 e He I 7065 visíveis da fase -10 até a
fase +15
    % (2014 Artigo - OPTICAL SPECTRA OF 73 STRIPPED-ENVELOPE
CORE-COLLAPSE SUPERNOVAE)

for contElem = 1:3
   if contElem == 1 % if H
      elemento = 'H'
      picosVales = [4102; 4341; 4861; 6563];
   elseif contElem == 2 %if Si
      elemento ='Si'
      picosVales = [6150; 6355];
   else %if He
      elemento = 'He'
      picosVales = [5876; 6678];
   end;

   % cria intervalos referentes as linhas dos elementos de acordo com
   % parâmetro intervalo
   numIntervalos = length(picosVales);
   limites = [picosVales-intervalo picosVales+intervalo];
   for lin = 1:numIntervalos-1
      if limites(lin,2) >= limites(lin+1, 1)
         limites(lin+1, 1) = limites(1,2) + 1;
      end;
   end;
   for lin = 1:numIntervalos
      if limites(lin,1) < inix
         limites(lin,1) = inix;
      elseif limites(lin,2) > fimx
         limites(lin,2) = fimx;
      end;
   end;
   posicao = round((limites - limites(1,1))/interpx + 1);
   x = limites(1,1):interpx:limites(1,2);
   for lin = 2:numIntervalos
      x = [x (limites(lin,1):interpx:limites(lin,2))];
   end;

   % cria linha de títulos do arquivo CSV de saída
   nomeSaida = strcat(caminho, pastaSaida);
   nomeSaida = strcat(nomeSaida, 'EntradaRNLinhas_');
   nomeSaida = strcat(nomeSaida, elemento);
   nomeSaida = strcat(nomeSaida, '_');
   nomeSaida = strcat(nomeSaida, nomePadrao);
```

```
nomeSaidaGeral = strcat(nomeSaida, '.csv')
saidaGeral = fopen(nomeSaidaGeral, 'w');
imprimeTitulo(saidaGeral, x, elemento);

nomeSaidaTreinoEstima = strcat(nomeSaida, 'TreinoEstima');
nomeSaidaTreinoEstima = strcat(nomeSaidaTreinoEstima, '.csv')
saidaTreinoEstima = fopen(nomeSaidaTreinoEstima, 'w');
imprimeTitulo(saidaTreinoEstima, x, elemento);

nomeSaidaTreinoValida = strcat(nomeSaida, 'TreinoValida');
nomeSaidaTreinoValida = strcat(nomeSaidaTreinoValida, '.csv')
saidaTreinoValida = fopen(nomeSaidaTreinoValida, 'w');
imprimeTitulo(saidaTreinoValida, x, elemento);

nomeSaidaTeste = strcat(nomeSaida, 'Teste');
nomeSaidaTeste = strcat(nomeSaidaTeste, '.csv')
saidaTeste = fopen(nomeSaidaTeste, 'w');
imprimeTitulo(saidaTeste, x, elemento);

% salva em cada linha do arquivo CSV os dados referentes a cada
% espectro
cont = 0;
contIa = 0;
contIaTreino = 0;
contIb = 0;
contIbTreino = 0;
contIc = 0;
contIcTreino = 0;
contIIc = 0;
contIIcTreino = 0;
nomeSNAnt = '';
arquivo = fopen(nomeArqNomesPre, 'r');
while ~feof(arquivo)
    cont = cont + 1;
    nomeArq = fgetl(arquivo);
    nomeSN = fgetl(arquivo);
    data = fgetl(arquivo);
    ondaini = fgetl(arquivo);
    ondafin = fgetl(arquivo);
    tipoSN = fgetl(arquivo);
    fase = fgetl(arquivo);
    redshift = fgetl(arquivo);

    nomeArqDados = strcat(caminho, pastaSaida);
    nomeArqDados = strcat(nomeArqDados, nomeArq);
    arqDados = fopen(nomeArqDados,'r');
```

```
        dados = textscan(arqDados, '%d%f');
        fclose(arqDados);

        % cria arquivos diferentes de acordo com o elemento que vai ser
        % verificado na rede neural

        salvar = 1; %salvar os espectros de todos os tipos de supernovas
        if strcmp(elemento,'H') %presença de H -> tipo I; ausência de H -> tipo II
           if strcmp(tipoSN(1:2),'II')
              presenca = 1;
           else
              presenca = 0;
           end;
        else
           if strcmp(tipoSN(1:2),'II')
              salvar = 0; %não salvar espectros do tipo II para presença de Si
           end;
           if strcmp(elemento,'Si') %presença de Si -> tipo Ia; ausência de Si ->
tipo Ib ou Ic
              if strcmp(tipoSN(1:2),'Ia')
                 presenca = 1;
              else
                 presenca = 0;
              end;
           else %if HE: rico em He -> tipo Ib; pobre em He -> tipo Ic
              if strcmp(tipoSN(1:2), 'Ia')
                 salvar = 0; %não salvar espectros do tipo Ia par apresença de HE
              end;
              if strcmp(tipoSN(1:2),'Ib')
                 presenca = 1;
              else
                 presenca = 0;
              end;
           end;
        end;
        if salvar == 1
           y = dados{2}(posicao(1,1):posicao(1,2))';
           for lin = 2:numIntervalos
              y = [y (dados{2}(posicao(lin,1):posicao(lin,2))')];
           end;

imprimeDadosEspectro(saidaGeral,y,presenca,tipoSN,nomeSN,data,ondaini,on
dafin,fase,redshift,nomeArq)

           if strcmp(nomeSN, nomeSNAnt) == 0
              nomeSNAnt = nomeSN;
```

```
if strcmp(tipoSN(1:2),'Ia')
   contIa = contIa + 1;
   if mod(contIa,5) == 0
      treinoTeste = 'teste';
   else
      contIaTreino = contIaTreino + 1;
      if mod(contIaTreino,5) == 0;
         treinoTeste = 'treinoValida';
      else
         treinoTeste = 'treinoEstima';
      end;
   end;
else
   if strcmp(tipoSN(1:2), 'Ib')
      contIb = contIb + 1;
      if mod(contIb,5) == 0
         treinoTeste = 'teste';
      else
         contIbTreino = contIbTreino + 1;
         if mod(contIbTreino,5) == 0;
            treinoTeste = 'treinoValida';
         else
            treinoTeste = 'treinoEstima';
         end;
      end;
   else
      if strcmp(tipoSN(1:2), 'Ic')
         contIc = contIc + 1;
         if mod(contIc,5) == 0
            treinoTeste = 'teste';
         else
            contIcTreino = contIcTreino + 1;
            if mod(contIcTreino,5) == 0;
               treinoTeste = 'treinoValida';
            else
               treinoTeste = 'treinoEstima';
            end;
         end;
      else
         contIIc = contIIc + 1;
         if mod(contIIc,5) == 0
            treinoTeste = 'teste';
         else
            contIIcTreino = contIIcTreino + 1;
            if mod(contIIcTreino,5) == 0;
               treinoTeste = 'treinoValida';
```

```
                    else
                        treinoTeste = 'treinoEstima';
                    end;
                end;
            end;
        end;
    end;

    if strcmp(treinoTeste, 'teste')

imprimeDadosEspectro(saidaTeste,y,presenca,tipoSN,nomeSN,data,ondaini,on
dafin,fase,redshift,nomeArq);
    else
        if strcmp(treinoTeste, 'treinoValida')

imprimeDadosEspectro(saidaTreinoValida,y,presenca,tipoSN,nomeSN,data,on
daini,ondafin,fase,redshift,nomeArq);
        else %strcmp(treinoTeste, treinoEstima)

imprimeDadosEspectro(saidaTreinoEstima,y,presenca,tipoSN,nomeSN,data,on
daini,ondafin,fase,redshift,nomeArq);
        end;
    end;
  end;
  fclose(arquivo);
  fclose(saidaGeral);
  fclose(saidaTreinoEstima);
  fclose(saidaTreinoValida);
  fclose(saidaTeste);
end
```

A.3 Programa “CriaCSVcomLinhasElementos.m”

Esse programa realiza automaticamente a seleção das partes do espectro que serão utilizadas pela versão atual do CIntIa e divide os espectros nos conjuntos de treinamento, validação e teste para serem usados como entrada das redes neurais.

```
%
% Cria um arquivo .csv com intervalos do espectro usado para
% treinar ou testar a Rede Neural que identifica o tipo de supernova
%

% define os intervalos do espectro que serão usados de acordo com o tipo da
% supernova que se deseja identificar
%intervalos = [5000 6500]; % Tipo Ia
%nomeClasse = 'Ia';
%intervalos = [4000 5000; 6000 7000]; % Tipo II
%nomeClasse = 'IIb';
%intervalos = [5500 7000]; % Tipo Ib
%nomeClasse = 'Ib';
intervalos = [5500 6500]; % Tipo Ic
nomeClasse = 'Ic';

arquivoParam = fopen('ParametrosPreProcessamento.txt', 'r');
nomeArqNomesPre = fgetl(arquivoParam); %nome do arquivo com os nomes dos arquivos preprocessados
inix = str2double(fgetl(arquivoParam)); %inicio do espectro
fimx = str2double(fgetl(arquivoParam)); %final do espectro
interpx = str2double(fgetl(arquivoParam)); %parâmetro de interpolação
suavy = fgetl(arquivoParam); %parâmetro de suavização
faseMinMax = fgetl(arquivoParam); %fase inicial e fase final do espectro
caminho = fgetl(arquivoParam); %caminho base dos arquivos
pastaSaida = fgetl(arquivoParam); %nome da pasta onde foram salvos os arquivos preprocessados
nomePadrao = fgetl(arquivoParam); %padrao de nome usado nos arquivos(fase, interpolação e suavização)
fclose(arquivoParam);


vetx = inix:interpx:fimx;
x = [];
for i = 1:size(intervalos,1)
    ini = find(vetx >= intervalos(i,1),1);
```

```
    fim = find(vetx >= intervalos(i,2),1);
    x = [x vetx(ini:fim)];
end;

% cria linha de títulos do arquivo CSV de saída
nomeSaida = strcat(caminho, pastaSaida);
nomeSaida = strcat(nomeSaida, 'EntradaRN_Int');
nomeSaida = strcat(nomeSaida, int2str(x(1)));
nomeSaida = strcat(nomeSaida, '-');
nomeSaida = strcat(nomeSaida, int2str(x(length(x))));
nomeSaida = strcat(nomeSaida, 'Fase');
nomeSaida = strcat(nomeSaida, faseMinMax);
nomeSaida = strcat(nomeSaida, 'Interp');
nomeSaida = strcat(nomeSaida, int2str(interpx));
nomeSaida = strcat(nomeSaida, 'Suav');
nomeSaida = strcat(nomeSaida, suavy);
nomeSaida = strcat(nomeSaida, 'Tipo');
nomeSaida = strcat(nomeSaida, nomeClasse);

nomeSaidaGeral = strcat(nomeSaida, '.csv')
saidaGeral = fopen(nomeSaidaGeral, 'w');
fprintf(saidaGeral, '%d;', x);
imprimeTitulo(saidaGeral, nomeClasse);

nomeSaidaTreinoEstima = strcat(nomeSaida, '_Treino');
nomeSaidaTreinoEstima = strcat(nomeSaidaTreinoEstima, '.csv')
saidaTreinoEstima = fopen(nomeSaidaTreinoEstima, 'w');
fprintf(saidaTreinoEstima, '%d;', x);
imprimeTitulo(saidaTreinoEstima, nomeClasse);

nomeSaidaTreinoValida = strcat(nomeSaida, '_Valida');
nomeSaidaTreinoValida = strcat(nomeSaidaTreinoValida, '.csv')
saidaTreinoValida = fopen(nomeSaidaTreinoValida, 'w');
fprintf(saidaTreinoValida, '%d;', x);
imprimeTitulo(saidaTreinoValida, nomeClasse);

nomeSaidaTeste = strcat(nomeSaida, '_Teste');
nomeSaidaTeste = strcat(nomeSaidaTeste, '.csv')
saidaTeste = fopen(nomeSaidaTeste, 'w');
fprintf(saidaTeste, '%d;', x);
imprimeTitulo(saidaTeste, nomeClasse);

% salva em cada linha do arquivo CSV os dados referentes a cada
% espectro
cont = 0;
contTreino = 0;
```

```
contValida = 0;
contTeste = 0;
contIa = 0;
contIaTreino = 0;
contIb = 0;
contIbTreino = 0;
contIc = 0;
contIcTreino = 0;
contIIb = 0;
contIIbTreino = 0;
nomeSNAnt = '';
arquivo = fopen(nomeArqNomesPre, 'r');
while ~feof(arquivo)
    cont = cont + 1;
    nomeArq = fgetl(arquivo);
    nomeSN = fgetl(arquivo);
    data = fgetl(arquivo);
    ondaini = fgetl(arquivo);
    ondafin = fgetl(arquivo);
    tipoSN = fgetl(arquivo);
    fase = fgetl(arquivo);
    fase = str2double(fase);
    redshift = fgetl(arquivo);

    nomeArqDados = strcat(caminho, pastaSaida);
    nomeArqDados = strcat(nomeArqDados, nomeArq);
    arqDados = fopen(nomeArqDados,'r');
    dados = textscan(arqDados, '%d%f');
    fclose(arqDados);

    vety = dados{2}';
    y = [];
    for i = 1:size(intervalos,1)
        ini = find(vetx >= intervalos(i,1),1);
        fim = find(vetx >= intervalos(i,2),1);
        y = [y vety(ini:fim)];
    end;

    % cria arquivos diferentes para Treinamento Estimação, Treinamento
    % Validação e Teste
    if strcmp(nomeSN, nomeSNAnt) == 0
        nomeSNAnt = nomeSN;
        if strcmp(tipoSN(1:2),'Ia')
            contIa = contIa + 1;
            if strcmp(nomeClasse,'Ia')
                classe = 1;
```

```
    else
        classe = 0;
    end;
    %classe = [0 0];
    if mod(contIa,5) == 0
        treinoTeste = 'teste';
    else
        contIaTreino = contIaTreino + 1;
        if mod(contIaTreino,5) == 0;
            treinoTeste = 'treinoValida';
        else
            treinoTeste = 'treinoEstima';
        end;
    end;
else
    if strcmp(tipoSN(1:2), 'Ib')
        contIb = contIb + 1;
        if strcmp(nomeClasse,'Ib')
            classe = 1;
        else
            classe = 0;
        end;
        %classe = [0 1];
        if mod(contIb,5) == 0
            treinoTeste = 'teste';
        else
            contIbTreino = contIbTreino + 1;
            if mod(contIbTreino,5) == 0;
                treinoTeste = 'treinoValida';
            else
                treinoTeste = 'treinoEstima';
            end;
        end;
    else
        if strcmp(tipoSN(1:2), 'Ic')
            contIc = contIc + 1;
            if strcmp(nomeClasse,'Ic')
                classe = 1;
            else
                classe = 0;
            end;
            %classe = [1 0];
            if mod(contIc,5) == 0
                treinoTeste = 'teste';
            else
                contIcTreino = contIcTreino + 1;
```

```
                if mod(contIcTreino,5) == 0;
                    treinoTeste = 'treinoValida';
                else
                    treinoTeste = 'treinoEstima';
                end;
            end;
        else %tipo == IIb
            contIIb = contIIb + 1;
            if strcmp(nomeClasse,'IIb')
                classe = 1;
            else
                classe = 0;
            end;
            %classe = [1 1];
            if mod(contIIb,5) == 0
                treinoTeste = 'teste';
            else
                contIIbTreino = contIIbTreino + 1;
                if mod(contIIbTreino,5) == 0;
                    treinoTeste = 'treinoValida';
                else
                    treinoTeste = 'treinoEstima';
                end;
            end;
        end;
    end;
end;


imprimeDadosEspectro(saidaGeral,y,classe,tipoSN,nomeSN,data,ondaini,ondaf
in,fase,redshift,nomeArq)

   if strcmp(treinoTeste, 'teste')

imprimeDadosEspectro(saidaTeste,y,classe,tipoSN,nomeSN,data,ondaini,onda
fin,fase,redshift,nomeArq);
       contTeste = contTeste + 1;
   else
       if strcmp(treinoTeste, 'treinoValida')

imprimeDadosEspectro(saidaTreinoValida,y,classe,tipoSN,nomeSN,data,ondai
ni,ondafin,fase,redshift,nomeArq);
           contValida = contValida + 1;
       else %strcmp(treinoTeste, 'treinoEstima')
```

```
imprimeDadosEspectro(saidaTreinoEstima,y,classe,tipoSN,nomeSN,data,ondai
ni,ondafin,fase,redshift,nomeArq);
          contTreino = contTreino + 1;
       end;
   end;
end;

fprintf(saidaTreinoEstima, '%d\n', 1);
fprintf(saidaTreinoEstima, '%d\n', 0);
fprintf(saidaTreinoEstima, 'Tipo_%s\n', nomeClasse);
fprintf(saidaTreinoEstima, 'Tipo_Nao%s\n', nomeClasse);
fprintf(saidaTreinoEstima, '%d\n', length(x)); %número de entradas da RN
fprintf(saidaTreinoEstima, '%d\n', 1); %número de saídas da RN
fprintf(saidaTreinoEstima, '%d\n', contTreino); %número de padrões de Treino
da RN
fprintf(saidaTreinoEstima, '%d\n', 2); %número de classes da RN

fprintf(saidaTreinoValida, '%d\n', contValida); %número de padrões de
Validação da RN

fprintf(saidaTeste, '%d\n', contTeste); %número de padrões de Teste da RN

fclose(arquivo);
fclose(saidaGeral);
fclose(saidaTreinoEstima);
fclose(saidaTreinoValida);
fclose(saidaTeste);
```

APÊNDICE B – CÓDIGOS DOS PROGRAMAS ESCRITOS EM LINGUAGEM C PARA DESENVOLVIMENTO DAS REDES NEURAIS

Um sistema foi construído usando a linguagem C para realizar treinamento e teste de Redes Neurais Perceptron de Múltiplas Camadas. O programa principal é mostrado neste Apêndice.

B.1 Programa “cintia.cpp”

Esse programa recebe como entrada os parâmetros da rede e os arquivos gerados no pré-processamento e tem como saída informações sobre os acertos e erros dos testes.

```
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include <sstream>

#define NOMECONF "Rede40";
#define NOMEARQ "EntradaRN_Int5000-6504Fase-
3+7Interp8Suav70TipoIa_";
//#define NOMEARQ "EntradaRN_Int5504-7000Fase-
3+7Interp8Suav70TipoIb_";
//#define NOMEARQ "EntradaRN_Int5504-6504Fase-
3+7Interp8Suav70TipoIc_";
//#define NOMEARQ "EntradaRN_Int4000-7000Fase-
3+7Interp8Suav70TipoIIb_";

using namespace std;

int abreArquivosTreino();
void defineRede();
int treinamento(double, double, int, double, int);
void geraVetorPesos(double[], int);
void propagaCamada(int, double*, double*, int, double*, double**, int);
void calculaErrosCamInterna(int , double*, double*, int, double*, double**);
void calculaNovosPesos(double, double, int, double*, int, double*, double**,
double**, int);
void criaVetoresMatrizes();
```

```
void salvaTreinamento();
int leTreinamento();
void testaRede(string);
void mostraVetorDouble(double[], int);

double **matEntradas, *vetCamada1, *vetCamada2, *vetSaidas,
**matSaidasEsperadas;
double *vetErrosCam1, *vetErrosCam2, *vetErrosSaidas;
double *vetFAtivaCam1, *vetFAtivaCam2, *vetFAtivaSai;
double **matPesosCam1, **matPesosCam2, **matPesosSaidas, **matTestes,
**matSaidaEsperada, **matClasses;
double **matPesosCam1Ant, **matPesosCam2Ant, **matPesosSaidasAnt;
double erroTolerado;
char **vetNomeClasses, **vetNomePadrao, **vetErrosTeste, **vetNomeSN,
**vetFaseSN;
int quantCamadas, quantNeuroniosCam1, quantNeuroniosCam2,
quantNeuroniosUltCam;
int quantEntradas, quantSaidas, quantPadroes, quantClasses, bias;
ifstream arqConf, arqEnt, arqTestes, arqTreinado;
ofstream arqSai, arqNome, arqTreino;

int main(int argc, char *argv[]){

    int opcao, treinoOk = 0;
    double taxaAprendizagem = 0.5, momentum = 0.3;
    int quantidadeEpocas = 1000000;
    erroTolerado = 0.001;
    bias = 1;
    string nomeValida, nomeTeste, nomeTreino = "Treino", nomeTreinado =
"Treino";

    do{
        cout << "Menu:" << endl;
        cout << "1- Treinar a Rede Neural" << endl;
        cout << "2- Ler um treinamento feito da Rede Neural" << endl;
        cout << "3- Validar treinamento da Rede Neural" << endl;
        cout << "4- Testar a Rede Neural" << endl;
        cout << "0- Sair do Programa" << endl;
        cout << "Escolha sua opcao: ";
        cin >> opcao;
        switch(opcao){
            case 1:
                treinoOk = 0;
                if(abreArquivosTreino() == 0){
                    return 1;
                }
```

```
            defineRede();
            cout << "\nREDE CONSTRUIDA COM SUCESSO\n";
            cout << "\nClique qualquer tecla para iniciar o TREINAMENTO\n";
            system("PAUSE");
            treinoOk = treinamento(taxaAprendizagem, momentum,
quantidadeEpocas, erroTolerado, bias);
            if(treinoOk == 1){
               nomeTreino = nomeTreino + NOMEARQ;
               nomeTreino = nomeTreino + NOMECONF;
               nomeTreino = nomeTreino + ".txt";
               arqTreino.open(nomeTreino.c_str());
               if(arqTreino.fail()){
                  cout << "\nERRO AO ABRIR O ARQUIVO PARA SALVAR
TREINAMENTO: " << nomeTreino << "\n\n";
                  nomeTreino = "Treino";
                  system("PAUSE");
                  return 0;
               }
               cout << "Nome do arquivo treinado: " << nomeTreino << endl;
               salvaTreinamento();
               arqTreino.close();
            }
            arqEnt.close();
            arqConf.close();
            arqSai.close();
            arqNome.close();
            break;

         case 2:
            treinoOk = 0;
            //cout << "\nDigite o nome do arquivo com os dados do treinamento:
";
            //cin >> nomeTreinado;
            nomeTreinado = nomeTreinado + NOMEARQ;
            nomeTreinado = nomeTreinado + NOMECONF;
            nomeTreinado = nomeTreinado + ".txt";
            //nomeTreinado = "EspectrosTreinado.txt";
            arqTreinado.open(nomeTreinado.c_str());
            if(arqTreinado.fail()){
               cout << "\nERRO AO ABRIR O ARQUIVO COM O
TREINAMENTO: " << nomeTreinado << "\n\n";
               nomeTreinado = "Treino";
            } else{
               treinoOk = leTreinamento();
               if(treinoOk == 1){
```

```
                    cout << "\nDADOS DO TREINAMENTO LIDOS COM
SUCESSO. PRONTO PARA OS TESTES." << "\n\n";
                } else {
                    cout << "\nERRO AO LER OS DADOS DO ARQUIVO COM O
TREINAMENTO: " << nomeTreinado << "\n\n";
                    nomeTreinado = "Treino";
                }
            }
            arqTreinado.close();
            break;

        case 3:
            if(treinoOk == 1){
                //cout << "\nDigite o nome do arquivo com os dados de teste: ";
                //cin >> nomeEnt;
                nomeValida = NOMEARQ;
                nomeValida = nomeValida + "Valida.csv";
                arqTestes.open(nomeValida.c_str());
                if(arqTestes.fail()){
                    cout << "\nERRO AO ABRIR O ARQUIVO DE VALIDACAO: " <<
nomeValida << "\n\n";
                    nomeValida = "";
                } else{
                    testaRede("Valida");
                }
                arqTestes.close();
            }
            else{
                cout << "\nTREINAMENTO NAO ESTA DISPONIVEL. TREINE A
REDE OU LEIA UM TREINAMENTO.\n\n";
            }
            break;
        case 4:
            if(treinoOk == 1){
                //cout << "\nDigite o nome do arquivo com os dados de teste: ";
                //cin >> nomeEnt;
                nomeTeste = NOMEARQ;
                nomeTeste = nomeTeste + "Teste.csv";
                arqTestes.open(nomeTeste.c_str());
                if(arqTestes.fail()){
                    cout << "\nERRO AO ABRIR O ARQUIVO DE TESTES: " <<
nomeTeste << "\n\n";
                    nomeTeste = "";
                } else{
                    testaRede("Teste");
                }
```

```
                arqTestes.close();
            }
            else{
                cout << "\nTREINAMENTO NAO ESTA DISPONIVEL. TREINE A
REDE OU LEIA UM TREINAMENTO.\n\n";
            }
            break;
        case 0:
            cout << "\nFIM DO PROGRAMA\n";
            break;
        default:
            cout << "OPCAO INVALIDA\n\n" << endl;
      }
   } while(opcao != 0);

   //system("PAUSE");
   return 0;
}

/*****************************************************************************/
   // função para abrir os arquivos usados para:
   // definir a topologia da rede MLP, construir a rede e salvar o treinamento
/*****************************************************************************/
int abreArquivosTreino(){
   //char nomeConf[50], nomeEnt[50], nomeSai[100]=
{'E','r','r','o','s'};//{'D',':','/','R','e','d','e','M','L','P','/','E','r','r','o','s'};
   string nomeConf, nomeEnt, nomeErros =
"Erros";//{'D',':','/','R','e','d','e','M','L','P','/','E','r','r','o','s'};

   //cout << "Digite o nome do arquivo com a configuracao da rede: ";
   //cin >> nomeConf;
   nomeConf = NOMECONF;
   nomeConf = nomeConf + ".txt";
   arqConf.open(nomeConf.c_str());
   if(arqConf.fail()){
      cout << "\nERRO AO ABRIR O ARQUIVO DE CONFIGURACAO DA
REDE: " << nomeConf << "\n\n";
      nomeConf = "";
      system("PAUSE");
      return 0;
   }
   //cout << "Digite o nome do arquivo com os dados de entrada: ";
   //cin >> nomeEnt;
   nomeEnt = NOMEARQ;
   nomeEnt = nomeEnt + "Treino.csv";
   arqEnt.open(nomeEnt.c_str());
```

```
    if(arqEnt.fail()){
        cout << "\nERRO AO ABRIR O ARQUIVO DE ENTRADA: " << nomeEnt
<< "\n\n";
        nomeEnt = "";
        system("PAUSE");
        return 0;
    }
    nomeErros = nomeErros + NOMEARQ;
    nomeErros = nomeErros + NOMECONF;
    nomeErros = nomeErros + ".txt";
    arqSai.open(nomeErros.c_str());
    if(arqSai.fail()){
        cout << "\nERRO AO ABRIR O ARQUIVO DE SAIDA: " << nomeErros <<
"\n\n";
        nomeErros = "Erros";
        system("PAUSE");
        return 0;
    }

    arqNome.open("NomeArquivoEntrada.txt");
    if(arqNome.fail()){
        cout << "\nERRO AO ABRIR O ARQUIVO NomeArquivoEntrada.txt\n\n";
        system("PAUSE");
        return 0;
    }
    arqNome << nomeEnt;

    return 1;
}

/*****************************************************************************/
    // função para ler o arquivo que define a topologia da rede MLP e construir a
rede
/*****************************************************************************/
void defineRede(){
    int i, j;
    string linha, celula, caracter;
    srand(time(NULL));
    /*
    formato do arquivo com dados de configuração da rede:
        1- quantidade de camadas
        2- quantidade de neuronios da camada 1
        3- quantidade de neuronios da camada 2
    */
    /*
    formato do arquivo com dados de entrada:
```

```
        1- quantidade de entradas
        2- quantidade de valores de saida
        3- quantidade de padrões de treinamento
        4- quantidade de classes
        5- matriz de entradas com últimas colunas com resultado esperado
        6- matriz com os valores de saída
        7- classes
    */
    cout << "\nINFORMACOES LIDAS NO ARQUIVO DE CONFIGURACAO:\n";
    arqConf >> quantCamadas;
    cout << "Quantidade de Camadas = " << quantCamadas << "\n";
    arqConf >> quantNeuroniosCam1;
    cout << "Quantidade de Neuronios da Camada 1 = " <<
quantNeuroniosCam1 << "\n";
    arqConf >> quantNeuroniosCam2;
    cout << "Quantidade de Neuronios da Camada 2 = " <<
quantNeuroniosCam2 << "\n";

    cout << "\nINFORMACOES LIDAS NO ARQUIVO DE ENTRADA:\n";
    //arqEnt >> quantEntradas;
    getline (arqEnt,linha);
    quantEntradas = atoi(linha.c_str());
    cout << "Quantidade de Entradas = " << quantEntradas << "\n";
    //arqEnt >> quantSaidas;
    getline (arqEnt,linha);
    quantSaidas = atoi(linha.c_str());
    cout << "Quantidade de Saidas = " << quantSaidas << "\n";
    //arqEnt >> quantPadroes;
    getline (arqEnt,linha);
    quantPadroes = atoi(linha.c_str());
    cout << "Quantidade de Padroes = " << quantPadroes << "\n";
    //arqEnt >> quantClasses;
    getline (arqEnt,linha);
    quantClasses = atoi(linha.c_str());
    cout << "Quantidade de Classes = " << quantClasses << "\n";

    arqSai << "INFORMACOES DE CONFIGURAÇÃO DA REDE MLP:\n";
    arqSai << "Quantidade de Entradas = " << quantEntradas << "\n";
    arqSai << "Quantidade de Camadas = " << quantCamadas << "\n";
    arqSai << "Quantidade de Neuronios da Camada 1 = " <<
quantNeuroniosCam1 << "\n";
    arqSai << "Quantidade de Neuronios da Camada 2 = " <<
quantNeuroniosCam2 << "\n";
    arqSai << "Quantidade de Saidas = " << quantSaidas << "\n";
    arqSai << "Quantidade de Padroes = " << quantPadroes << "\n";
    arqSai << "Quantidade de Classes = " << quantClasses << "\n";
```

```
    criaVetoresMatrizes();
    cout << "VETORES CRIADOS\n";

    for(i = 0; i < quantPadroes; i++){
        getline (arqEnt,linha);
        istringstream ss(linha);
        for(j = 0; j < quantEntradas; j++){
            if (!getline(ss,celula,';')) break;
            matEntradas[i][j] = atof(celula.c_str());
            //arqEnt >> matEntradas[i][j];
        }
        for(j = 0; j < quantSaidas; j++){
            if (!getline(ss,celula,';')) break;
            matSaidasEsperadas[i][j] = atof(celula.c_str());
            //arqEnt >> matSaidasEsperadas[i][j];
        }
    }
//    cout << "Matriz de Entradas:\n";
//    for(i = 0; i < quantPadroes; i++){
//        mostraVetorDouble(matEntradas[i], quantEntradas);
//    }
//    cout << "Matriz de Saidas Esperadas:\n";
//    for(i = 0; i < quantPadroes; i++){
//        mostraVetorDouble(matSaidasEsperadas[i], quantSaidas);
//    }

    for(i = 0; i < quantClasses; i++){
        getline (arqEnt,linha);
        istringstream ss(linha);
        for(j = 0; j < quantSaidas; j++){
            if (!getline(ss,celula,';')) break;
            matClasses[i][j] = atof(celula.c_str());
        }
        //arqEnt >> vetClasses[i];
    }
    for(i = 0; i < quantClasses; i++){
        getline (arqEnt,linha);
        istringstream ss(linha);
        if (!getline(ss,celula,';')) break;
        strcpy(vetNomeClasses[i], (char*)celula.c_str());
        //arqEnt >> vetNomeClasses[i];
    }

    cout << "Classes: \n";
    for(i = 0; i < quantClasses; i++){
```

```
        mostraVetorDouble(matClasses[i], quantSaidas);
        cout << vetNomeClasses[i] << "\n";
    }
}

// função para criar as vetores e matrizes a partir dos parâmetros lidos
void criaVetoresMatrizes(){
    int i, j;
    matEntradas = (double **)calloc(quantPadroes,sizeof(double *));
    for(i = 0; i < quantPadroes; i++){
        matEntradas[i] = (double *)calloc(quantEntradas,sizeof(double));
    }
    matSaidasEsperadas = (double **)calloc(quantPadroes,sizeof(double*));
    for(i = 0; i < quantPadroes; i++){
        matSaidasEsperadas[i] = (double *)calloc(quantSaidas,sizeof(double));
    }
    matClasses = (double **)calloc(quantClasses,sizeof(double *));
    for(i = 0; i < quantClasses; i++){
        matClasses[i] = (double *)calloc(quantSaidas,sizeof(double));
    }
    vetNomeClasses = (char **)calloc(quantClasses,sizeof(char *));
    for(i = 0; i < quantClasses; i++){
        vetNomeClasses[i] = (char *)calloc(50,sizeof(char));
    }
    vetCamada1 = (double *)calloc(quantNeuroniosCam1,sizeof(double));
    vetFAtivaCam1 = (double *)calloc(quantNeuroniosCam1,sizeof(double));
    vetErrosCam1 = (double *)calloc(quantNeuroniosCam1,sizeof(double));
    matPesosCam1 = (double **)calloc(quantEntradas+1,sizeof(double *));
    for(i = 0; i < quantEntradas+1; i++){
        matPesosCam1[i] = (double *)calloc(quantNeuroniosCam1,sizeof(double));
    }
    for(i = 0; i < quantEntradas+1; i++){
        geraVetorPesos(matPesosCam1[i], quantNeuroniosCam1);
    }
    matPesosCam1Ant = (double **)calloc(quantEntradas+1,sizeof(double *));
    for(i = 0; i < quantEntradas+1; i++){
        matPesosCam1Ant[i] = (double
*)calloc(quantNeuroniosCam1,sizeof(double));
    }
    for(i= 0; i < quantEntradas+1; i++){
        for(j = 0; j < quantNeuroniosCam1; j++){
            matPesosCam1Ant[i][j] = matPesosCam1[i][j];
        }
    }
//    cout << "Matriz de Pesos da Camada 1:\n";
//    for(i = 0; i < quantEntradas+1; i++){
```

```
//        mostraVetorDouble(matPesosCam1[i], quantNeuroniosCam1);
//    }

    if(quantCamadas == 2){
        vetCamada2 = (double *)calloc(quantNeuroniosCam2,sizeof(double));
        vetErrosCam2 = (double *)calloc(quantNeuroniosCam2,sizeof(double));
        vetFAtivaCam2 = (double *)calloc(quantNeuroniosCam2,sizeof(double));
        matPesosCam2 = (double
**)calloc(quantNeuroniosCam1+1,sizeof(double*));
        for(i = 0; i < quantNeuroniosCam1+1; i++){
            matPesosCam2[i] = (double
*)calloc(quantNeuroniosCam2,sizeof(double));
        }
        for(i = 0; i < quantNeuroniosCam1+1; i++){
            geraVetorPesos(matPesosCam2[i], quantNeuroniosCam2);
        }
        matPesosCam2Ant = (double
**)calloc(quantNeuroniosCam1+1,sizeof(double*));
        for(i = 0; i < quantNeuroniosCam1+1; i++){
            matPesosCam2Ant[i] = (double
*)calloc(quantNeuroniosCam2,sizeof(double));
        }
        for(i = 0; i < quantNeuroniosCam1+1; i++){
            for(j = 0; j < quantNeuroniosCam2; j++){
                matPesosCam2Ant[i][j] = matPesosCam2[i][j];
            }
        }
//        cout << "Matriz de Pesos da Camada 2:\n";
//        for(i = 0; i < quantNeuroniosCam1+1; i++){
//            mostraVetorDouble(matPesosCam2[i], quantNeuroniosCam2);
//        }
        quantNeuroniosUltCam = quantNeuroniosCam2;
    }else{
        quantNeuroniosUltCam = quantNeuroniosCam1;
    }

    matPesosSaidas = (double
**)calloc(quantNeuroniosUltCam+1,sizeof(double*));
    for(i = 0; i < quantNeuroniosUltCam+1; i++){
        matPesosSaidas[i] = (double *)calloc(quantSaidas,sizeof(double));
    }
    for(i = 0; i < quantNeuroniosUltCam+1; i++){
        geraVetorPesos(matPesosSaidas[i], quantSaidas);
    }
    matPesosSaidasAnt = (double
**)calloc(quantNeuroniosUltCam+1,sizeof(double*));
```

```
    for(i = 0; i < quantNeuroniosUltCam+1; i++){
        matPesosSaidasAnt[i] = (double *)calloc(quantSaidas,sizeof(double));
    }
    for(i = 0; i < quantNeuroniosUltCam+1; i++){
        for(j = 0; j < quantSaidas; j++){
            matPesosSaidasAnt[i][j] = matPesosSaidas[i][j];
        }
    }
//    cout << "Matriz de Pesos da Camada de Saida:\n";
//    for(i = 0; i < quantNeuroniosUltCam+1; i++){
//       mostraVetorDouble(matPesosSaidas[i], quantSaidas);
//    }

    vetSaidas = (double *)calloc(quantSaidas,sizeof(double));
    vetFAtivaSai = (double *)calloc(quantSaidas,sizeof(double));
    vetErrosSaidas = (double *)calloc(quantSaidas,sizeof(double));
}

/*****************************************************************************/
// função para realizar o treinamento da rede
/*****************************************************************************/
int treinamento(double taxaAprend, double moment, int quantEpocas, double
erroTolerado, int bias){

    int  erro, temErro, sucesso, i, j;
    long contEpocas = 0;
    double exponencial, erroTotal, erroPadrao, erroMaior, erroSaida,
erroMaiorEpoca;

    arqSai << "\nERROS DA REDE MLP\n";
    cout << "\nREALIZANDO O TREINAMENTO...\n";

    // fazer o treinamento até erro <= tolerancia ou numero limite de epocas
    do{
        if(contEpocas % 1000 == 0){
            cout << "\nEPOCA " << contEpocas+1 << "\n";
            //if(taxaAprend > 0.3){
            //    taxaAprend -= 0.001;
            //}
            //cout << taxaAprend << endl;
        }
        erroSaida = 0;
        temErro = 0;
        erroMaiorEpoca = 0;
            // fazer o treinamento para cada padrão
        for(i = 0; i < quantPadroes; i++){
```

```
        // propagar os valores de entrada até a camada de saída
        propagaCamada(quantNeuroniosCam1, vetCamada1, vetFAtivaCam1,
quantEntradas, matEntradas[i], matPesosCam1, bias);
        if(quantCamadas == 2){
            propagaCamada(quantNeuroniosCam2, vetCamada2,
vetFAtivaCam2, quantNeuroniosCam1, vetFAtivaCam1, matPesosCam2, bias);
            propagaCamada(quantSaidas, vetSaidas, vetFAtivaSai,
quantNeuroniosCam2, vetFAtivaCam2, matPesosSaidas, bias);
        }else{
            propagaCamada(quantSaidas, vetSaidas, vetFAtivaSai,
quantNeuroniosCam1, vetFAtivaCam1, matPesosSaidas, bias);
        }

        // calcular o vetor de erros para camada de saida
        // erro Ej = (tj - xj)*F'(yj) (Derivada da Função de Ativação)
        erroTotal = 0;
        erroMaior = 0;
        for(j = 0; j < quantSaidas; j++){
            exponencial = exp(vetSaidas[j]*(-1.0));
            vetErrosSaidas[j] = (matSaidasEsperadas[i][j] -
vetFAtivaSai[j])*(exponencial/((1+exponencial)*(1+exponencial)));
            erroTotal += fabs(vetErrosSaidas[j]);
            if(fabs(vetErrosSaidas[j]) > erroMaior){
                erroMaior = fabs(vetErrosSaidas[j]);
            }
        }

        erroPadrao = erroTotal/quantSaidas;
        //if (erroPadrao > erroTolerado){
        if (erroMaior > erroTolerado){
            erro = 1;
        } else {
            erro = 0;
        }
        if (erroMaior > erroMaiorEpoca){
            erroMaiorEpoca = erroMaior;
        }

        erroSaida += erroPadrao;
        /*
        if(contEpocas % 10000 == 0){
            cout << "ERRO DO PADRAO " << i << " = " << erroPadrao << endl;
        }
        */
```

```
            // retropropagar o erro (se existir) da saída até o vetor de pesos da
entrada
            if(erro == 1){
               temErro = 1;
               if(quantCamadas == 2){
                  calculaNovosPesos(taxaAprend, moment, quantNeuroniosCam2,
vetFAtivaCam2, quantSaidas, vetErrosSaidas, matPesosSaidas,
matPesosSaidasAnt, bias);
                  calculaErrosCamInterna(quantNeuroniosCam2, vetErrosCam2,
vetCamada2, quantSaidas, vetErrosSaidas, matPesosSaidas);
                  calculaNovosPesos(taxaAprend, moment, quantNeuroniosCam1,
vetFAtivaCam1, quantNeuroniosCam2, vetErrosCam2, matPesosCam2,
matPesosCam2Ant, bias);
                  calculaErrosCamInterna(quantNeuroniosCam1, vetErrosCam1,
vetCamada1, quantNeuroniosCam2, vetErrosCam2, matPesosCam2);
               }else{ // se tem só uma camada
                  calculaNovosPesos(taxaAprend, moment, quantNeuroniosCam1,
vetFAtivaCam1, quantSaidas, vetErrosSaidas, matPesosSaidas,
matPesosSaidasAnt, bias);
                  calculaErrosCamInterna(quantNeuroniosCam1, vetErrosCam1,
vetCamada1, quantSaidas, vetErrosSaidas, matPesosSaidas);
               }
               calculaNovosPesos(taxaAprend, moment, quantEntradas,
matEntradas[i], quantNeuroniosCam1, vetErrosCam1, matPesosCam1,
matPesosCam1Ant, bias);
            }
         } // fim do for de um padrao


         if(contEpocas % 1000 == 0){
            cout << "ERRO MEDIO DA SAIDA: " << erroSaida/quantPadroes <<
"\n";
            cout << "MAIOR ERRO DA SAIDA: " << erroMaiorEpoca << "\n";
            //system("PAUSE");
         }
         //arqSai << erroSaida/quantPadroes << "\n";
         arqSai << erroMaior << "\n";
         contEpocas++;
      }while(temErro == 1 && contEpocas < quantEpocas); // final de uma epoca

      cout << "\nQUANTIDADE TOTAL DE EPOCAS: " << contEpocas << "\n";
      arqSai << "\nQUANTIDADE TOTAL DE EPOCAS: " << contEpocas << "\n";

      if (contEpocas < quantEpocas){
         cout << "\nTREINAMENTO REALIZADO COM SUCESSO\n\n";
         sucesso = 1;
```

```
    }else{
        cout << "\nTREINAMENTO ENCERRADO PELO NUMERO DE
EPOCAS\n\n";
        sucesso = 0;
    }
/*
    cout << "Matriz de Pesos da Camada 1:\n";
    for(i = 0; i < quantEntradas+1; i++){
         mostraVetorDouble(matPesosCam1[i], quantNeuroniosCam1);
    }
    if(quantCamadas == 2){
        cout << "Matriz de Pesos da Camada 2:\n";
        for(i = 0; i < quantNeuroniosCam1+1; i++){
            mostraVetorDouble(matPesosCam2[i], quantNeuroniosCam2);
        }
    }
    cout << "Matriz de Pesos da Camada de Saida:\n";
    for(i = 0; i < quantNeuroniosUltCam+1; i++){
        mostraVetorDouble(matPesosSaidas[i], quantSaidas);
    }
*/

    return sucesso;
}


// função para propagar valores de uma camada
void propagaCamada(int quantSai, double *vetSai, double *vetAtiva, int
quantEnt,
                   double *vetEnt, double **matPesos, int bias){
    int i,j;
    for(j = 0; j < quantSai; j++){ // inicio de um padrao para um neuronio
        vetSai[j] = bias * matPesos[0][j];
        for(i = 0; i < quantEnt; i++){
            vetSai[j] += vetEnt[i] * matPesos[i+1][j];
        }
    }
    // função de ativação f(yi) = 1/(1+exp(-yi))
    for(j = 0; j < quantSai; j++){
        vetAtiva[j] = 1.0/(1.0 + exp(vetSai[j]*(-1))); //Função de Ativação Sigmoid
Logística
    }
}


// função para calcular vetor de erros das camadas internas
```

```
// erro Ej = F'(yj)*somatorio(erro K * pesos jk) (Derivada da Função de Ativação
* somatorio de erros por pesos)
//calculaErrosCamInterna(quantNeuroniosCam1, vetErrosCam1, vetCamada1,
quantSaidas, vetErrosSaidas, matPesosSaidas);
void calculaErrosCamInterna(int quantErros, double *vetErros, double *vetCam,
                            int quantErrosProx, double *vetErrosProx, double
**matPesos){
    double somatorio, exponencial;
    int j, k;
    for(j = 0; j < quantErros; j++){
        somatorio = 0;
        for(k = 0; k < quantErrosProx; k++){
            somatorio += vetErrosProx[k] * matPesos[j+1][k];
        }
        exponencial = exp(vetCam[j]*(-1.0));
        vetErros[j] = (exponencial/((1+exponencial)*(1+exponencial)))*somatorio;
    }
}

// função para calcular novos pesos
// calculaNovosPesos(taxaAprend, quantNeuroniosCam1, vetFAtivaCam1,
quantSaidas, vetErrosSaidas, matPesosSaidas, bias);
// calculaNovosPesos(taxaAprend, quantEntradas, matEntradas[i],
quantNeuroniosCam1, vetErrosCam1, matPesosCam1, bias);
void calculaNovosPesos(double taxa, double momentum, int
quantNeuroniosEsq, double *vetNeuroniosEsq,
                       int quantErros, double *vetErros, double **matPesos, double
**matPesosAnt, int bias){
    int i, j;
    double nxE;
    for(j = 0; j < quantErros; j++){
        nxE = (taxa * bias * vetErros[j]) + momentum*(matPesos[0][j] -
matPesosAnt[0][j]);
        matPesosAnt[0][j] = matPesos[0][j];
        matPesos[0][j] += nxE;
    }
    for(i = 0; i < quantNeuroniosEsq; i++){
        for(j = 0; j < quantErros; j++){
            nxE = (taxa * vetNeuroniosEsq[i] * vetErros[j]) +
momentum*(matPesos[i+1][j] - matPesosAnt[i+1][j]);
            matPesosAnt[i+1][j] = matPesos[i+1][j];
            matPesos[i+1][j] += nxE;
        }
    }
}
```

```
// função para gerar pesos aleatórios entre -0.1 e 0.1
void geraVetorPesos(double vetor[], int quant){
    int i;
    for(i = 0; i < quant; i++){
        vetor[i] = rand()%2000/10000.0-0.1;
    }
}

// função para salvar um treinamento em arquivo
void salvaTreinamento(){
    int i, j;

    // Salvando os valores lidos no arquivo com os parâmetros da Rede Neural
    arqTreino << quantClasses << "\n";
    arqTreino << quantEntradas << "\n";
    arqTreino << quantSaidas << "\n";
    arqTreino << bias << "\n";
    arqTreino << quantCamadas << "\n";
    arqTreino << quantNeuroniosCam1 << "\n";
    if (quantCamadas == 2){
        arqTreino << quantNeuroniosCam2 << "\n";
    }
    for(i = 0; i < quantClasses; i++){
        for(j = 0; j < quantSaidas; j++){
            arqTreino << matClasses[i][j] << "\t";
        }
        arqTreino << "\n";
    }
    for(i = 0; i < quantClasses; i++){
        arqTreino << vetNomeClasses[i] << "\n";
    }
    /*
    for(i = 0; i < quantNeuroniosCam1; i++){
        arqTreino << vetCamada1[i] << "\t";
    }
    arqTreino << "\n";
    for(i = 0; i < quantNeuroniosCam1; i++){
        arqTreino << vetFAtivaCam1[i] << "\t";
    }
    arqTreino << "\n";
    */
    for(i = 0; i < quantEntradas+1; i++){
        for(j = 0; j < quantNeuroniosCam1; j++){
            arqTreino << matPesosCam1[i][j] << "\t";
        }
        arqTreino << "\n";
```

```
    }
    /*
    for(i = 0; i < quantSaidas; i++){
        arqTreino << vetSaidas[i] << "\t";
    }
    arqTreino << "\n";
    for(i = 0; i < quantSaidas; i++){
        arqTreino << vetFAtivaSai[i] << "\t";
    }
    arqTreino << "\n";
    */
    for(i = 0; i < quantNeuroniosUltCam+1; i++){
        for(j = 0; j < quantSaidas; j++){
            arqTreino << matPesosSaidas[i][j] << "\t";
        }
        arqTreino << "\n";
    }

    // Salvando caso a Rede Neural tenha duas camadas
    if (quantCamadas == 2){
        /*
        for(i = 0; i < quantNeuroniosCam2; i++){
            arqTreino << vetCamada2[i] << "\t";
        }
        arqTreino << "\n";
        for(i = 0; i < quantNeuroniosCam2; i++){
            arqTreino << vetFAtivaCam2[i] << "\t";
        }
        arqTreino << "\n";
        */
        for(i = 0; i < quantNeuroniosCam1+1; i++){
            for(j = 0; j < quantNeuroniosCam2; j++){
                arqTreino << matPesosCam2[i][j] << "\t";
            }
            arqTreino << "\n";
        }
    }
}


// função para ler um treinamento salvo em arquivo
int leTreinamento(){
    int i, j, sucesso = 0;

    // Lendo os valores no arquivo com os parâmetros da Rede Neural
    arqTreinado >> quantClasses;
```

```
    cout << "\nQuantidade de Classes: " << quantClasses << "\n";
    arqTreinado >> quantEntradas;
    cout << "Quantidade de Entradas: " << quantEntradas << "\n";
    arqTreinado >> quantSaidas;
    cout << "Quantidade de Saidas: " << quantSaidas << "\n";
    arqTreinado >> bias;
    cout << "Bias: " << bias << "\n";
    arqTreinado >> quantCamadas;
    cout << "Quantidade de Camandas: " << quantCamadas << "\n";
    arqTreinado >> quantNeuroniosCam1;
    cout << "Quantidade de Neuronios da Camada 1: " << quantNeuroniosCam1
<< "\n";
    if (quantCamadas == 2){
        arqTreinado >> quantNeuroniosCam2;
        cout << "Quantidade de Neuronios da Camada 2: " <<
quantNeuroniosCam2 << "\n";
    }

    criaVetoresMatrizes();

    for(i = 0; i < quantClasses; i++){
        for(j = 0; j < quantSaidas; j++){
            arqTreinado >> matClasses[i][j];
            //cout << matClasses[i][j] << "\t";
        }
        //cout << "\n";
    }
    for(i = 0; i < quantClasses; i++){
        arqTreinado >> vetNomeClasses[i];
        //cout << vetNomeClasses[i] << "\n";
    }
    /*
    for(i = 0; i < quantNeuroniosCam1; i++){
        arqTreinado >> vetCamada1[i];
        cout << vetCamada1[i] << "\t";
    }
    cout << "\n";
    for(i = 0; i < quantNeuroniosCam1; i++){
        arqTreinado >> vetFAtivaCam1[i];
        cout << vetFAtivaCam1[i] << "\t";
    }
    cout << "\n";
    */
    for(i = 0; i < quantEntradas+1; i++){
        for(j = 0; j < quantNeuroniosCam1; j++){
            arqTreinado >> matPesosCam1[i][j];
```

```
        //cout << matPesosCam1[i][j] << "\t";
    }
    //cout << "\n";
}
/*
for(i = 0; i < quantSaidas; i++){
    arqTreinado >> vetSaidas[i];
    cout << vetSaidas[i] << "\t";
}
cout << "\n";
for(i = 0; i < quantSaidas; i++){
    arqTreinado >> vetFAtivaSai[i];
    cout << vetFAtivaSai[i] << "\t";
}
cout << "\n";
*/
for(i = 0; i < quantNeuroniosUltCam+1; i++){
    for(j = 0; j < quantSaidas; j++){
        arqTreinado >> matPesosSaidas[i][j];
        //cout << matPesosSaidas[i][j] << "\t";
    }
    //cout << "\n";
}

// Lendo caso a Rede Neural tenha duas camadas
if (quantCamadas == 2){
    /*
    for(i = 0; i < quantNeuroniosCam2; i++){
        arqTreinado >> vetCamada2[i];
        cout << vetCamada2[i] << "\t";
    }
    cout << "\n";
    for(i = 0; i < quantNeuroniosCam2; i++){
        arqTreinado >> vetFAtivaCam2[i];
        cout << vetFAtivaCam2[i] << "\t";
    }
    cout << "\n";
    */
    for(i = 0; i < quantNeuroniosCam1+1; i++){
        for(j = 0; j < quantNeuroniosCam2; j++){
            arqTreinado >> matPesosCam2[i][j];
            //cout << matPesosCam2[i][j] << "\t";
        }
        //cout << "\n";
    }
}
```

```
    if(quantClasses > 1 && quantNeuroniosCam1 > 0){
        sucesso = 1;
    }
    return sucesso;
}

// função para testar um treinamento realizado
void testaRede(string nomeTeste){

    int quantTestes, i, j, k, ind, indEsperada = -1, cont, contErros = 0;
    //double vetErrosTestes[quantSaidas], exponencial, correto;
    double menor, soma;
    string linha, celula, nomeAcertos = "Acertos", erro = "";

    cout << "\nINFORMACOES LIDAS NO ARQUIVO DE TESTE:\n";
    getline (arqTestes,linha);
    quantTestes = atoi(linha.c_str());
    //arqTestes >> quantTestes;
    cout << "Quantidade de Testes = " << quantTestes << "\n";
    matTestes = (double **)calloc(quantTestes,sizeof(double *));
    for(i = 0; i < quantTestes; i++){
        matTestes[i] = (double *)calloc(quantEntradas,sizeof(double));
    }
    matSaidaEsperada = (double **)calloc(quantTestes,sizeof(double *));
    for(i = 0; i < quantTestes; i++){
        matSaidaEsperada[i] = (double *)calloc(quantSaidas,sizeof(double));
    }
    vetNomePadrao = (char **)calloc(quantTestes,sizeof(char *));
    for(i = 0; i < quantTestes; i++){
        vetNomePadrao[i] = (char *)calloc(50,sizeof(char));
    }
    vetNomeSN = (char **)calloc(quantTestes,sizeof(char *));
    for(i = 0; i < quantTestes; i++){
        vetNomeSN[i] = (char *)calloc(50,sizeof(char));
    }
    vetFaseSN = (char **)calloc(quantTestes,sizeof(char *));
    for(i = 0; i < quantTestes; i++){
        vetFaseSN[i] = (char *)calloc(50,sizeof(char));
    }

    for(i = 0; i < quantTestes; i++){
        getline (arqTestes,linha);
        istringstream ss(linha);
        for(j = 0; j < quantEntradas; j++){
            if (!getline(ss,celula,';')) break;
            matTestes[i][j] = atof(celula.c_str());
```

```
            //arqTestes >> matTestes[i][j];
        }
        for(j = 0; j < quantSaidas; j++){
            if (!getline(ss,celula,';')) break;
            matSaidaEsperada[i][j] = atof(celula.c_str());
            //arqTestes >> matSaidaEsperada[i][j];
        }
        if (!getline(ss,celula,';')) break;
        strcpy(vetNomePadrao[i], (char*)celula.c_str());
        if (!getline(ss,celula,';')) break;
        strcpy(vetNomeSN[i], (char*)celula.c_str());
        if (!getline(ss,celula,';')) break;
        celula.c_str();
        if (!getline(ss,celula,';')) break;
        celula.c_str();
        if (!getline(ss,celula,';')) break;
        celula.c_str();
        if (!getline(ss,celula,';')) break;
        strcpy(vetFaseSN[i], (char*)celula.c_str());
    }

    cout << "Classes: \n";
    for(i = 0; i < quantClasses; i++){
        mostraVetorDouble(matClasses[i], quantSaidas);
        cout << vetNomeClasses[i] << "\n";
    }
/*
    cout << "Matriz de Testes:\n";
    for(i = 0; i < quantTestes; i++){
        mostraVetorDouble(matTestes[i], quantEntradas);
        mostraVetorDouble(matSaidaEsperada[i], quantSaidas);
    }
*/

    cout << "\nREALIZANDO OS TESTES:\n";

    vetErrosTeste = (char **)calloc(quantTestes,sizeof(char *));
    for(i = 0; i < quantTestes; i++){
        vetErrosTeste[i] = (char *)calloc(50,sizeof(char));
    }

    nomeAcertos = nomeAcertos + NOMEARQ;
    nomeAcertos = nomeAcertos + NOMECONF;
    nomeAcertos = nomeAcertos + nomeTeste;
    nomeAcertos = nomeAcertos + ".txt";
    arqSai.open(nomeAcertos.c_str());
```

```
   if(arqSai.fail()){
      cout << "\nERRO AO ABRIR O ARQUIVO PARA SALVAR ACERTOS DO
TESTE: " << nomeAcertos << "\n\n";
      nomeAcertos = "Acertos";
      system("PAUSE");
   }
   for(i = 0; i < quantTestes; i++){
      // propagar os valores de entrada até a camada de saída
      propagaCamada(quantNeuroniosCam1, vetCamada1, vetFAtivaCam1,
quantEntradas, matTestes[i], matPesosCam1, bias);
      if(quantCamadas == 2){
         propagaCamada(quantNeuroniosCam2, vetCamada2, vetFAtivaCam2,
quantNeuroniosCam1, vetFAtivaCam1, matPesosCam2, bias);
         propagaCamada(quantSaidas, vetSaidas, vetFAtivaSai,
quantNeuroniosCam2, vetFAtivaCam2, matPesosSaidas, bias);
      }else{
         propagaCamada(quantSaidas, vetSaidas, vetFAtivaSai,
quantNeuroniosCam1, vetFAtivaCam1, matPesosSaidas, bias);
      }
      cout << endl << "Padrao " << i+1 << ": " << vetNomePadrao[i] << endl;
      arqSai << endl << "Padrao " << i+1 << ": "  << vetNomePadrao[i] << endl;
      cout << "Supernova: " << vetNomeSN[i] << endl;
      arqSai << "Supernova: "  << vetNomeSN[i] << endl;
      cout << "Fase: " << vetFaseSN[i] << endl;
      arqSai << "Fase: "  << vetFaseSN[i] << endl;
      menor = quantSaidas;
      ind = 0;
      for(j = 0; j < quantClasses; j++){
         soma = 0;
         for(k = 0; k < quantSaidas; k++){
            soma = soma + fabs(matClasses[j][k] - vetFAtivaSai[k]);
         }
         if(soma < menor){
            menor = soma;
            ind = j;
         }
         cont = 0;
         for(k = 0; k < quantSaidas; k++){
            if(matSaidaEsperada[i][k] == matClasses[j][k]) cont++;
         }
         if(cont == quantSaidas){
            cout << "Classe Esperada = " << vetNomeClasses[j] << "\n";
            arqSai << "Classe Esperada = " << vetNomeClasses[j] << "\n";
            indEsperada = j;
         }
      }
```

```
        cout << "Saida = ";
        arqSai << "Saida = ";
        for(j = 0; j < quantSaidas; j++){
            cout << vetFAtivaSai[j] << "\t";
            arqSai << vetFAtivaSai[j] << "\t";
        }
        cout << "\n";
        arqSai << "\n";
        cout << "Classe = " << vetNomeClasses[ind] << "\n";
        arqSai << "Classe = " << vetNomeClasses[ind] << "\n";
        if(ind != indEsperada){
            itoa((i+1), vetErrosTeste[contErros], 10);
            strcat(vetErrosTeste[contErros], "\t");
            strcat(vetErrosTeste[contErros], vetNomePadrao[i]);
            strcat(vetErrosTeste[contErros], "\t");
            strcat(vetErrosTeste[contErros], vetNomeSN[i]);
            strcat(vetErrosTeste[contErros], "\t");
            strcat(vetErrosTeste[contErros], vetFaseSN[i]);
            strcat(vetErrosTeste[contErros], "\tClasse: ");
            strcat(vetErrosTeste[contErros], vetNomeClasses[ind]);
            contErros++;
        }
    }

    arqSai << "\nINFORMACOES DE CONFIGURAÇÃO DA REDE MLP:\n";
    arqSai << "Quantidade de Entradas = " << quantEntradas << "\n";
    arqSai << "Quantidade de Camadas = " << quantCamadas << "\n";
    arqSai << "Quantidade de Neuronios da Camada 1 = " <<
quantNeuroniosCam1 << "\n";
    arqSai << "Quantidade de Neuronios da Camada 2 = " <<
quantNeuroniosCam2 << "\n";
    arqSai << "Quantidade de Saidas = " << quantSaidas << "\n";
    arqSai << "Quantidade de Padroes de Treinamento = " << quantPadroes <<
"\n";
    arqSai << "Quantidade de Classes = " << quantClasses << "\n\n";

    arqSai << "RESULTADOS DA REDE MLP:\n";
    cout << "\nQuantidade de padroes de teste = " << quantTestes << endl;
    arqSai << "\nQuantidade de padroes de teste = " << quantTestes << endl;
    cout << "\nQuantidade de erros = " << contErros << endl;
    arqSai << "\nQuantidade de erros = " << contErros << endl;
    cout << "\nPercentual de acertos = " << (quantTestes -
(double)contErros)/quantTestes*100 << "%" << endl;
    arqSai << "\nPercentual de acertos = " << (quantTestes -
(double)contErros)/quantTestes*100 << "%" << endl;
    cout << "\nErro de classificacao dos padroes: " << endl;
```

```
    arqSai << "\nErro de classificacao dos padroes: " << endl;
    for(i = 0; i < contErros; i++){
        cout << vetErrosTeste[i] << endl;
        arqSai << vetErrosTeste[i] << endl;
    }
    cout << endl;
    arqSai << endl;
    arqSai.close();
}

// função para mostrar vetores de double
void mostraVetorDouble(double vetor[], int quant){
     int i;
     for(i = 0; i < quant; i++){
          cout << vetor[i] << "\t";
     }
     cout << "\n";
}
```

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programa de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Programas de Computador (PDC)

São as sequências de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. São aceitos tanto programas fonte quanto executáveis.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.