



Universidade Federal
do Rio de Janeiro

Escola Politécnica

IDENTIFICAÇÃO FOTOMÉTRICA DE SUPERNOVAS ATRAVÉS DE ALGORITMOS DE MACHINE LEARNING

Felipe Matheus Fernandes de Oliveira

Projeto de Graduação apresentado ao Curso de Engenharia de Controle e Automação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadores: Amit Bhaya

Ribamar Rondon de Rezende
dos Reis

Rio de Janeiro
Julho de 2019

IDENTIFICAÇÃO FOTOMÉTRICA DE SUPERNOVAS ATRAVÉS DE
ALGORITMOS DE MACHINE LEARNING

Felipe Matheus Fernandes de Oliveira

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO DA ESCOLA
POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO
PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU
DE ENGENHEIRO DE AUTOMAÇÃO.

Examinado por:

Prof. Amit Bhaya, D.Sc.

Prof. Ribamar Rondon de Rezende dos Reis, Ph.D.

Prof. Heraldo Luís Silveira de Almeida, D.Sc.

Prof. Ramon Romankevicius Costa , D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2019

Fernandes de Oliveira, Felipe Matheus

IDENTIFICAÇÃO FOTOMÉTRICA DE
SUPERNOVAS ATRAVÉS DE ALGORITMOS DE
MACHINE LEARNING/Felipe Matheus Fernandes de
Oliveira. – Rio de Janeiro: UFRJ/ Escola Politécnica,
2019.

XI, 29 p.: il.; 29,7cm.

Orientadores: Amit Bhaya

Ribamar Rondon de Rezende dos Reis

Projeto de Graduação – UFRJ/ Escola Politécnica/
Curso de Engenharia de Controle e Automação, 2019.

Referências Bibliográficas: p. 28 – 28.

1. Machine Learning. 2. Gaussian Process Fitting.
3. Supernova Photometric Identification. I. Bhaya,
Amit *et al.* II. Universidade Federal do Rio de Janeiro,
Escola Politécnica, Curso de Engenharia de Controle e
Automação. III. Título.

*A alguém cujo valor é digno
desta dedicatória. A você Goku*

Agradecimentos

Gostaria de agradecer a todos os Sayajins por terem nos salvado das garras maléficas de Freeza, Cell e Majin Boo.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Automação.

IDENTIFICAÇÃO FOTOMÉTRICA DE SUPERNOVAS ATRAVÉS DE ALGORITMOS DE MACHINE LEARNING

Felipe Matheus Fernandes de Oliveira

Julho/2019

Orientadores: Amit Bhaya

Ribamar Rondon de Rezende dos Reis

Curso: Engenharia de Controle e Automação

Com a finalidade de estudar a expansão do universo, a cosmologia busca classificar diferentes tipos de objetos astronômicos. Entretanto, com o crescente aumento do número de objetos detectados, o método normalmente usado para a classificação (espectroscopia) torna-se muito custoso. Como consequência, utiliza-se um método com baixo custo (fotometria) embasado em algoritmos de aprendizado de máquina para a classificação desse vasto número de dados. Nesse contexto, o presente trabalho estuda otimizações para a melhoria desses algoritmos de aprendizado de máquina.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Engineer.

SUPERNOVA PHOTOMETRIC IDENTIFICATION USING MACHINE LEARNING ALGORITHMS

Felipe Matheus Fernandes de Oliveira

July/2019

Advisors: Amit Bhaya

Ribamar Rondon de Rezende dos Reis

Course: Automation and Control Engineering

To explore the expansion history of the universe, cosmology classifies different types of astronomic objects using spectroscopy. With their sample sizes increasing, spectroscopy methods cannot handle this amount of data. As a solution to this issue, photometric identification is crucial to fully exploit these large samples due to its simplicity. Once photometric identification uses machine learning algorithms, the following work tries to optimize those algorithms.

Sumário

Lista de Figuras	x
Lista de Tabelas	xi
1 Introdução	1
1.1 Tema e Contextualização	1
1.2 Problemática	2
1.3 Delimitação	2
1.4 Objetivos	3
1.4.1 Processo Gaussiano <i>Gaussian Process</i>	3
1.4.2 Aprendizagem Profunda	4
1.4.3 Tratamento de <i>Outliers</i>	4
1.5 Metodologia	6
1.6 Descrição	6
2 Fundamentos Teóricos	8
2.1 Processo Gaussiano	8
2.1.1 <i>Kernel</i> e Função de Covariância	8
2.1.2 Função Média	10
2.2 Convolutional Neural Network	12
2.3 Demais Conceitos	14
2.3.1 Análise de Componentes Principais	14
2.3.2 Transformadas de <i>Wavelet</i>	14
2.3.3 K-Fold Cross Validation	14
3 Modelo Atual	16
3.1 Pré-processamento dos Dados Brutos	16
3.2 <i>Pipeline</i> Atual	17
4 Tratamento de <i>Outliers</i>	20
4.1 Estratégias Utilizadas	20
4.2 Resultados e comparações	20

5	Rede Neural Convolutacional	21
5.1	Geração de Imagens e Parâmetros	21
5.2	<i>Pipeline</i>	22
5.3	Resultados e comparações	23
6	Interpolação através de Processo Gaussiano	25
6.1	Escolha da Biblioteca	25
6.2	Aleatoriedades e <i>Random Seeds</i>	25
6.3	<i>Kernel Functions</i>	25
6.4	Demais Observações	25
6.5	Resultados das Interpolações	25
6.6	Identificação e Justificativa do Erro	25
7	Resultados e Discussões	26
8	Conclusões	27
8.1	Conclusões Finais	27
8.2	Trabalhos Futuros	27
	Referências Bibliográficas	28
A	Repositório do Código	29

Lista de Figuras

1.1	Exemplo da interpolação não condizente com a realidade.	3
1.2	Exemplo de <i>outlier</i>	5
1.3	Exemplo de ponto com incerteza 3 vezes maior que o pico da curva. .	5
2.1	Ilustração da matriz Σ possuindo 10 e 12 pontos.	10
2.2	Ilustração do valor da função média.	11
2.3	Deep Learning Standard Architecture.	12
2.4	Convolution Operation.	13
2.5	Max Pooling.	13
3.1	Arquivo <i>.txt</i> a ser lido pelos arquivos de pré-processamento.	17
3.2	Exemplo de 4 curvas de luz de um mesmo objeto.	18
3.3	Matrizes de Confusão de um modelo final do <i>pipeline</i> original.	19
5.1	Arquitetura do modelo de <i>Deep Learning</i> utilizado.	22
5.2	Matrizes de confusão do modelo final. <i>Deep Learning</i>	23
5.3	Valores normalizados do modelo DL treinando com 80% dos dados. .	24
5.4	Modelo do <i>pipeline</i> original treinado com 80% dos dados.	24

Lista de Tabelas

Capítulo 1

Introdução

1.1 Tema e Contextualização

Dentro da cosmologia, existe a necessidade de determinar distâncias luminosas ?? para modelar seus estudos, e como ferramenta dessa tarefa, utilizam-se as curvas de luz provenientes de supernovas do tipo IA.

No passado, o conjunto de dados de supernovas era pequeno o suficiente para poder analisar a maior parte dos objetos usando o método da espectroscopia, um método lento e custoso que, no entanto, confirma precisamente o tipo de cada uma delas.

Contudo, com o avanço das pesquisas e da tecnologia de telescópios, a astronomia está entrando em uma era de conjuntos massivos de dados, tornando-se necessário a adoção de técnicas automatizadas mais simples e práticas para classificar a enorme quantidade de objetos astronômicos captados, pois, através da espectroscopia não seria possível.

Nesse contexto, foram desenvolvidas diferentes abordagens para levantar essa grande quantidade de objetos captados através de um método chamado fotometria. Dentre essas abordagens, várias utilizam aprendizado de máquina.

Por fim, a ideia do projeto foi derivada de um artigo publicado pela cientista LOCHNER [1]. A autora busca criar uma maneira automática de classificação fotométrica usando as curvas de luz obtidas através da fotometria, essas que já foram devidamente classificadas no passado utilizando espectroscopia. Caracterizando assim um problema de classificação, visto que temos os tipos de supernovas dados pela espectroscopia, e suas representações fotométricas para que possamos criar e treinar modelos.

1.2 Problemática

Tendo em vista que o artigo [1] testou e validou diversos *pipelines*, neste trabalho adotamos aquele que ela obteve o melhor resultado de classificação, e a partir desse ponto, aplicamos modificações e avaliamos quaisquer possíveis melhoras.

O *pipeline* escolhido é constituído majoritariamente de 4 partes:

- Processo Gaussiano (*Gaussian Process* ou **GP**)
- Transformada de Wavelet
- Análise de componentes principais (*Principal Component Analysis* ou **PCA**)
- Floresta Aleatória (*Random Forest* ou **RF**)

A problemática encontra-se no método de interpolação chamado **Processo Gaussiano**. Por ser um método de interpolação, espera-se que o mesmo defina uma função que passe pelos pontos obtidos respeitando suas incertezas. Entretanto, em alguns casos o gráfico interpolado é uma constante 1.1. O que não possui sentido físico, visto que se trata do fluxo de luz após uma explosão de um objeto astronômico.

Através do Instituto de Física da UFRJ surgiu a ideia principal deste projeto que é a de tentar corrigir essas interpolações que apresentam comportamento constante durante toda a observação.

Em paralelo, outras ideias foram aplicadas na tentativa de obter uma melhora no algoritmo original da *Lochner*.

1.3 Delimitação

Todos os dados utilizados tanto neste trabalho quanto no artigo [1], foram extraídos da base de dados oferecida pelo desafio KESSLER [2]. A base de dados é de domínio público e pode ser obtida no [LINK DO DOWNLOAD](#).

Devido a granularidade dos dados brutos, é necessário um pré-tratamento visando buscar apenas os dados que iremos utilizar. Esse pré-tratamento foi aproveitado do *pipeline* do Instituto de Física da UFRJ (**IF-UFRJ**), cujo processo visa ler os arquivos de texto transformando cada informação em uma chave de dicionário.

Ao fim do uso da seleção de dados citada acima, obtivemos valores em forma de dicionários em Python para cada objeto astronômico. Esses dicionários serão os **Dados Brutos** no escopo deste trabalho.

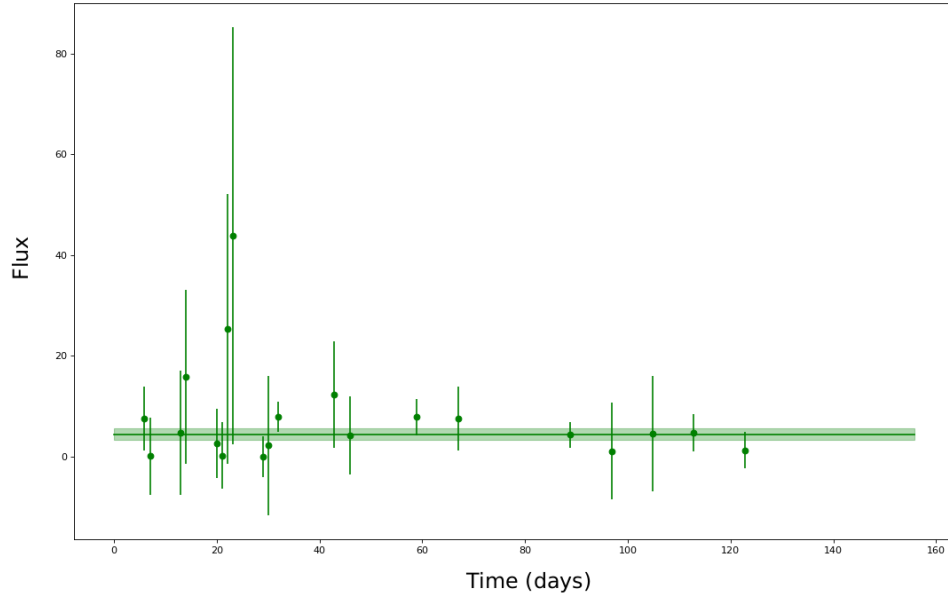


Figura 1.1: Exemplo da interpolação não condizente com a realidade.

1.4 Objetivos

O objetivo do trabalho é otimizar determinados pontos do *pipeline* utilizado atualmente, após um estudo inicial ter sido feito para identificar pontos frágeis ou passíveis de melhoras.

Esses pontos são apresentados e justificados a seguir, criando as três vertentes do trabalho

- Processo Gaussiano
- Aprendizagem Profunda
- Tratamento de *Outliers*

1.4.1 Processo Gaussiano *Gaussian Process*

O Processo Gaussiano, é a primeira parte do tratamento de dados após a transformação dos dados brutos em dicionários de Python. Iremos abordar esse procedimento no Capítulo 6.

Cada objeto astronômico irá possuir uma quantidade de pontos que representam a intensidade do fluxo de luz captado em determinado dia, bem como a incerteza desse valor. Assim, buscamos fazer uma interpolação para obter um gráfico que passe por determinados pontos.

O objetivo ao abordar esse ponto é consertar interpolações que não condizem com a interpretação física, como pode ser vista na figura 1.1.

1.4.2 Aprendizagem Profunda

A Aprendizagem Profunda (*Deep Learning* ou **DL**) vêm sendo um advento utilizado recentemente para aprimorar alguns algoritmos de Aprendizagem de Máquina (*Machine Learning* ou **ML**).

A ideia de aplicar DL nesse problema veio a partir da análise do *pipeline* inicial (visto na seção 1.2). Tendo em vista que o mesmo possui diversas partes entre os dados brutos e algoritmo de RF, buscamos aplicar DL esperando obter um *pipeline* mais simples, possuindo apenas uma etapa entre os dados brutos e o algoritmo de classificação.

Em diversos outros casos na literatura, como identificação de sons ou imagens, houveram melhorias significativas ao reduzir o número de etapas envolvendo processamento de sinais (wavelets) ou redução de componentes (PCA) em troca do uso de DL.

A segunda motivação para a aplicação de DL veio do artigo da LOCHNER [1], onde ela menciona possíveis aplicações no escopo do problema.

Assim, foi decidido aplicar algoritmos de DL após a primeira parte do tratamento de dados, visando manter uma simplicidade ao longo do *pipeline*. Estabelecendo assim, apenas a etapa do GP entre os dados brutos e o algoritmo de classificação. O detalhamento desse procedimento encontra-se no Capítulo 5.

1.4.3 Tratamento de *Outliers*

O último objetivo para buscar melhorias no algoritmo é o tratamento de pontos cuja incerteza é muito alta ou cujos valores são fora do esperado, pontos com essas características são conhecidos na literatura como *outliers*.

Analisando alguns dados brutos, observamos pontos cuja incerteza mostra-se mais de 100% maior do que a própria curva (ponto mais a esquerda da figura 1.3) e também pontos que encontram-se completamente fora da curva interpolada 1.2.

As etapas do tratamento encontram-se descritas no Capítulo 4.

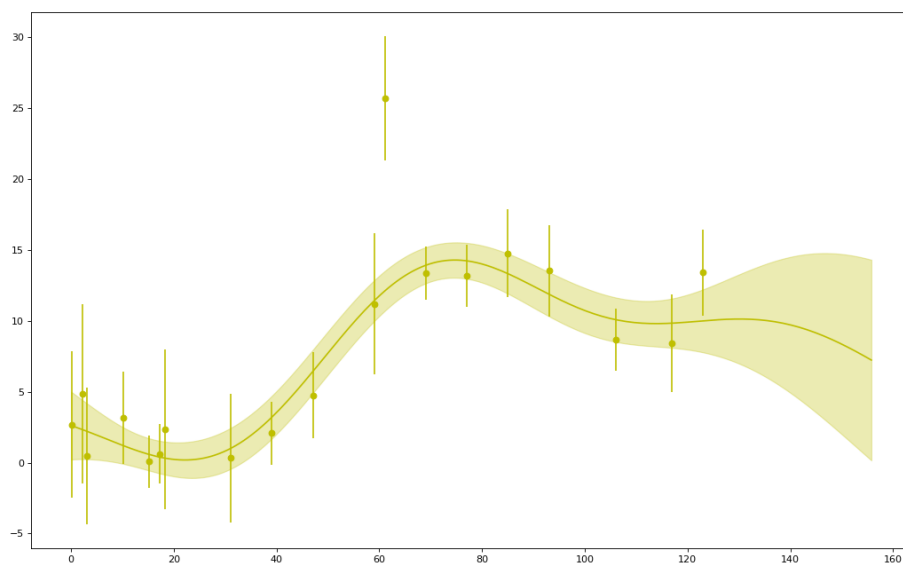


Figura 1.2: Exemplo de *outlier*. Gráfico Fluxo \times Tempo (dias).

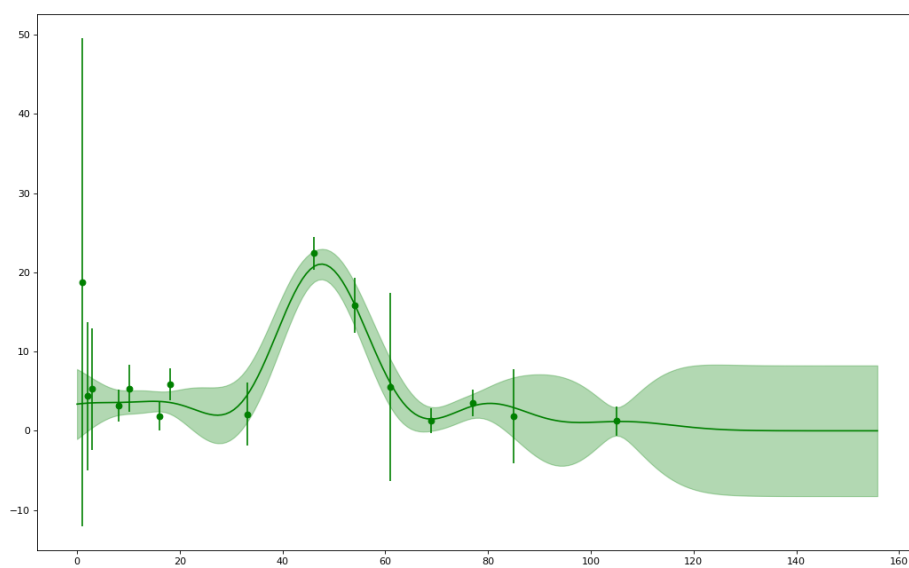


Figura 1.3: Exemplo de ponto com incerteza 3 vezes maior que o pico da curva. Gráfico Fluxo \times Tempo (dias).

1.5 Metodologia

Devido ao código de pré-processamento do IF-UFRJ estar escrito em Python, todo o trabalho também foi escrito na mesma linguagem. Outro ponto para a escolha da linguagem foi a facilidade que a mesma possui para realizar experimentos através da plataforma *Jupyter*, além da grande quantidade de bibliotecas de ciências de dados e suporte para desenvolver algoritmos de ML.

As principais bibliotecas utilizadas no contexto de tratamento de dados foram *Pandas* e *Numpy*. Já para a aplicação dos métodos as bibliotecas mais importantes foram *scikit-learn*, *Keras* e *PyMC3*.

A entrada dos modelos sempre serão os dicionários gerados a partir dos dados brutos após o pré-processamento do IF-UFRJ. Assim, pôde-se filtrar com maior facilidade as propriedades desejadas para cada uma das três vertentes do trabalho.

Cada uma dessas vertentes possuiu um trabalho específico e isolado para poder melhor averiguar suas mudanças posteriormente. Em outras palavras, a cada mudança realizada houve a comparação direta com os resultados do *pipeline* original, para poder avaliar especificamente a eficiência e melhora dessa mudança.

Por se tratar de um algoritmo de classificação, a saída do nosso modelo será booleana e classificará se um objeto astronômico é ou não do tipo IA. Entretanto, nos dados brutos, as classes são divididas em 8 tipos de objetos: *IA*, *IB*, *IBC*, *IC*, *II*, *IIL*, *IIN* e *IIP*.

É importante ressaltar também que para validar os modelos do projeto, foi usado o método *K-fold*. Todavia, este trabalho se diferenciou da divisão clássica presente na literatura, que é uma divisão de 80% dos dados para treinamento e 20% para o teste dos modelos. Neste trabalho foram utilizados apenas 1100 objetos para treino dentre 21316 objetos totais. A justificativa é a limitação física das amostras.

Por fim, as mudanças foram verificadas através de **Matrizes de Confusão** (*Confusion Matrices* ou **CM**) ou através de *score* da biblioteca *scikit-learn*.

Como última observação, a fim de estudos posteriores fora do escopo do projeto, foram feitas classificações envolvendo os 8 tipos de objetos, além de treinamentos adotando a proporção de 80% para treino e 20% para teste.

1.6 Descrição

No Capítulo 2, iremos explicar os principais conceitos teóricos utilizados na aplicação do **Processo Gaussiano** e no *Deep Learning*. Também serão levantados alguns conceitos não tão centrais do trabalho, mas de importância igualmente significativa.

O Capítulo 3 será destinado para detalhar especificamente cada etapa do *pipeline*

atual, assim como uma breve menção a maneira na qual os dados brutos são tratados.

Os capítulos 4, 5 e 6 serão destinados à explicação dos objetivos citados na seção 1.4, assim como os resultados das mudanças que os mesmos causaram, buscando averiguar a eficiência de aplicação ou explicar possíveis erros.

No Capítulo 7 serão apresentados os resultados como um todo, seguidos de análises dos métodos através uma perspectiva global do projeto, diferentemente de como será nos capítulos anteriores, nos quais teremos uma visão específica para cada objetivo.

Por fim, no Capítulo 8 tiraremos as conclusões finais do projeto e serão apresentadas sugestões de trabalhos futuros junto as limitações e possíveis soluções encontradas.

Capítulo 2

Fundamentos Teóricos

2.1 Processo Gaussiano

O Processo Gaussiano é uma ferramenta importante no escopo de *Machine Learning* que nos permite fazer predições sobre nossos dados tendo como base um conhecimento a priori. A sua aplicação mais frequente é em interpolações de funções, vide a aplicação neste projeto. Há também possíveis aplicações do conceito em classificações e agrupamentos de grande quantidades de dados. O livro base utilizado para os estudos desse projeto foi RASMUSSEN e WILLIAMS [3] .

Sabendo que para uma quantidade determinada de pontos existe uma infinidade de funções que podem interpolar esses valores, o Processo Gaussiano realiza essa interpolação a partir dos conhecimentos a priori dos valores dessa função, assim como de um provável formato que ela pode assumir como um todo. Por fim, o GP não irá obter um valor específico para cada ponto da função interpolada, mas sim uma distribuição probabilística para cada ponto, onde cada um possuirá uma média (valor que será atribuído) e um desvio padrão (incerteza).

Neste projeto não abrangeremos maiores detalhes para o aprendizado do método, contudo, pode-se encontrar na bibliografia, tanto referências mais específicas e técnicas [3], quanto mais práticas [4], [5], [6].

Iremos agora focar nos dois aspectos fundamentais do Processo Gaussiano; eles serão os principais responsáveis por caracterizar as interpolações.

2.1.1 *Kernel* e Função de Covariância

O Processo Gaussiano interpola uma função discreta de n pontos através dos m pontos conhecidos, resultando em n distribuições normais, uma para cada ponto que irá constituir essa função discreta.

Vale lembrar que uma função discreta cujos intervalos dos pontos do eixo das abscissas são significativamente pequenos pode ser considerada contínua para efeitos

práticos.

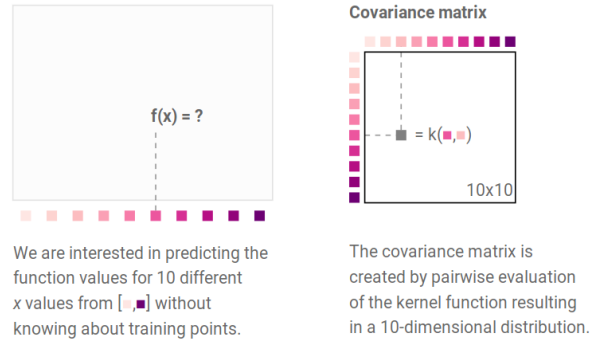
O diferencial do GP é considerar uma distribuição normal multivariável de dimensão $n + m$ durante a interpolação. Essa distribuição normal multivariável possuirá uma covariância Σ responsável não apenas por descrever o **formato**, como também por determinar **características** da função a ser prevista.

Para poder obter o Σ desta normal multivariável, é estabelecido o *Kernel* do GP. Ele é uma função de duas variáveis especial que respeita certas restrições matemáticas (Cap. 4 RASMUSSEN e WILLIAMS [3]). Essas variáveis são os valores da abscissa de cada ponto da função a ser interpolada.

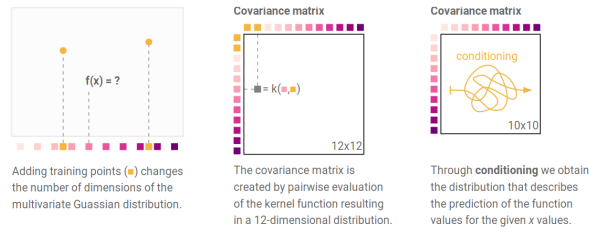
Em outras palavras, o *Kernel* é uma função que irá relacionar dois pontos que deverão estar na curva interpolada, baseando se apenas na distância entre eles. É essa relação que fornecerá características como periodicidade, picos e suavização da curva.

É importante ressaltar que apenas os valores das abscissas serão utilizados para essa relação, enquanto os valores das ordenadas dos pontos conhecidos são utilizados para forçar que essa função passe pelos pontos dados, através de **probabilidade condicionada** (*Posterior Distribution* [6]). Pode-se utilizar esse método de probabilidade condicionada devido à propriedade das distribuições gaussianas que afirma: distribuições condicionadas ou marginalizadas vindas de distribuições gaussianas também são gaussianas.

A figura 2.1 ilustra uma maneira mais fácil de visualizar o que foi dito no parágrafo acima. Pode-se ver em 2.1(b) que os mesmos 10 pontos serão calculados já com a probabilidade condicionada para satisfazer os 2 pontos pré-estabelecidos.



(a) Ilustração da matriz Σ para 10 pontos.



(b) Ilustração da matriz Σ para 12 pontos.
2 deles são pontos dados.

Figura 2.1: Ilustração da matriz Σ possuindo 10 e 12 pontos. *Imagens retiradas de JOCHEN GÖRTLER [6]*

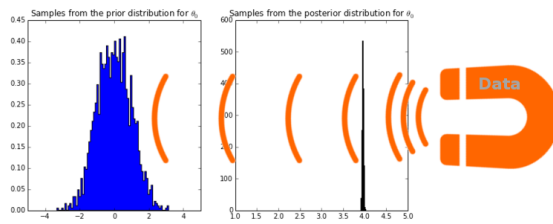
2.1.2 Função Média

A Função Média de um Processo Gaussiano é aquela responsável por oferecer a previsão inicial do ponto a ser interpolado. É o valor que determinado ponto possuiria caso só existisse ele a ser definido.

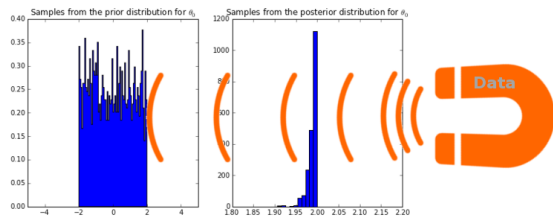
Contudo, só existe sentido em falar de função média caso a associemos à Função Covariância. Normalmente a Função Média é estabelecida como uma distribuição normal com média em 0 e desvio padrão 1.

É possível usar essa distribuição padrão como valor a priori, pois o valor final da média de um ponto será obtido após um número suficientemente grande de amostras de pontos, que serão definidos através de amostragens usando **Cadeias de Markov Monte Carlo** [7], se aproximando cada vez mais do valor ideal. A figura 2.2(a) ilustra esse caso.

Todavia, é imprescindível o uso correto da Função Média, pois caso ela tenha seu domínio delimitado, isso poderá impossibilitar que a função assumam alguns valores, como está ilustrado na 2.2(b).



(a) Exemplo de priori gaussiana que ao longo das amostragens influenciadas pelos dados assume o valor real (no caso do exemplo é 4).



(b) Exemplo de priori delimitada que mesmo ao longo das amostragens nunca assumirá o valor real (no caso do exemplo é 4 e a delimitação é de -2 a 2).

Figura 2.2: Ilustração do valor da função média durante as amostragens via Cadeias de Markov Monte Carlo. *Imagens retiradas da BAILEY [5]*

2.2 Convolutional Neural Network

Convolutional Neural Network (**CNN**) is a part of Deep Learning (**DL**) which is most commonly applied to analyze visual data. Our main goal inside this project is to use Machine Learning algorithms, however, to avoid any ambiguity with the nomenclatures, it is important to say that Deep Learning is a part of a broader Machine Learning family.

A basic DL architecture is made by 3 parts (figure 2.3):

- **Input Layer:** The first layer, being responsible for receiving one data sample and passing it to the hidden layers.
- **Hidden Layers:** The internal DL architecture structure, inside this part are set and trained the functions to process and classify the data.
- **Output Layer:** The last Layer. It will be responsible for the final answer. In this project it will have the values of 0 and 1, respectively for classified supernovas as *non IA* and *IA*.

Each layer is composed by different types of **neurons**. Those neurons are mathematical functions with specific parameters, which will be trained in order to classify our data.

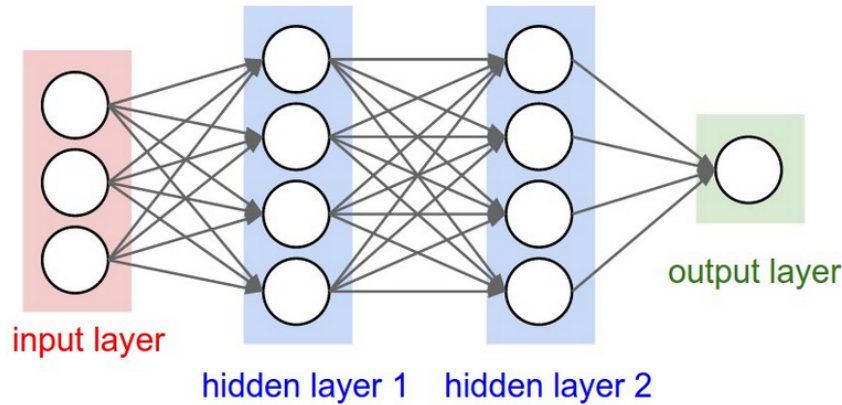


Figura 2.3: Deep Learning Standard Architecture. (adapted from: [here](#).)

In this project it is used CNN, a specific type of mathematical procedure compounding the hidden layer normally designed to process images. This thesis uses black and white png files created from a Gaussian Process interpolation.

Firstly, the CNN must receive the image in a matrix format, where each pixel is a value between 0 and 1 (0 for white and 1 for black varying through gray's intensities). Therefore, the hidden layers process the matrices using different methods 2.4(e.g. max pooling, ordinary convolution).

1	1	1	0	0
0	1	1	1	0
0	0	1x1	1x0	1x1
0	0	1x0	1x1	0x0
0	1	1x1	0x0	0x1

4	3	4
2	4	3
2	3	4

Figura 2.4: The blue matrix represents the input values; the red one is the result of convolution operation; and the green matrix contains the parameters to multiply the values from the blue one "sliding through its grid". (adapted from: [here](#).)

Through different possible existent CNN methods, there is one particular which was used inside this project. It is the Max pooling. It consists on taking the biggest value after a matrix convolution 2.5.

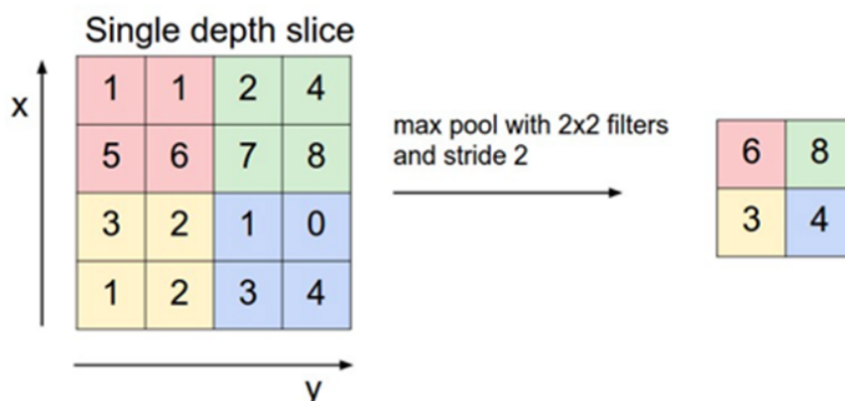


Figura 2.5: Max pooling example, it selects the biggest value of a specific matrix and reduces the dimension of it only filtering that value. (adapted from: [here](#).)

Inside the hidden layers it is impossible to say exactly what is the physical meaning of each of them. Some might be nothing but mathematical function results converging to a final classification. But sometimes it is still possible to identify one layer as an image property like edges, eyes, foots, curves, etc.

Finally, with those techniques of evaluating and processing images, the last layer will receive the results from the hidden ones and returns a value responsible for the classification (mostly like numbers).

It is important to mention that there is a little randomness inside this process of training and optimizing the internal layers' parameters.

2.3 Demais Conceitos

Aqui estão apresentados alguns conceitos menos usados ao longo do *pipeline*.

2.3.1 Análise de Componentes Principais

A Análise de Componentes Principais (Cap. 12 MURPHY [8]), ou *Principal Component Analysis* (**PCA**), utiliza conceitos de ortogonalizações de vetores para poder reduzir dimensões.

Em ML, essa redução de dimensões é aplicada as *features* de um *dataframe*, onde as amostras possuem um grande número de características e deseja-se reduzi-las para que o modelo de aprendizagem de máquina seja menos pesado mas igualmente eficaz.

Através de técnicas vindas da álgebra linear o método "cria" novas dimensões virtuais compostas por combinações lineares das existentes, de maneira que estas novas dimensões poderão melhor distinguir a quantidade de dados existentes no *dataframe*.

Por fim pode-se também interpretar PCA por uma perspectiva de filtros, onde se decompõe determinado sinal, e selecionando determinadas frequências conserva-se a potência do sinal. Por esta perspectiva, as frequências são as *features* e a potência é a porcentagem que as novas *features* têm de representar o *dataframe*.

2.3.2 Transformadas de *Wavelet*

Transformadas de *Wavelet* [9] ou onduletas são funções capazes de descrever ou representar outras funções em diferentes escalas de frequência e de tempo.

São normalmente usadas no domínio de processamento de sinais, sendo úteis para eliminar ruídos e comprimir dados por exemplo.

Neste projeto as Transformadas de *Wavelet* servem para processar os gráficos interpolados do GP. Resultando assim em valores que podem ter possíveis erros de interpolação eliminados, por exemplo.

2.3.3 K-Fold Cross Validation

Cross-validation is a family of statistical methods used to estimate how a model can be generalized regardless the train and test sets.

In ML, it is commonly used to analyze if a model is robust enough or if it gives more false-positives and false-negatives (inside the classification cases).

K-Fold Cross Validation is one of these methods. The procedure consists in dividing the data set in K folds, **normally** $K - 1$ are used to train the model while 1 is used to test. However in this project, due to physical limitation reasons only

1 fold is used to train while the other 18 are used to test and validate the method (the value of K is set to be 19).

With the K-Fold idea there are 2 concepts which will be used to evaluate the results in this project:

- AUC Curve (Cap. 5.7 MURPHY [8])
- Mean Average Precision (Cap. 9.7.4 MURPHY [8])

Capítulo 3

Modelo Atual

Neste capítulo iremos detalhar o modelo atual utilizado para estudos cosmológicos do IF-UFRJ.

3.1 Pré-processamento dos Dados Brutos

Como pode ser visto na figura 3.1, os dados brutos chegam da seguinte forma para o pré-processamento.

Através dos arquivos de pré-tratamento iremos obter todas essas informações em forma de dicionário, entretanto apenas as seguintes serão utilizadas durante o projeto.

- **SN TYPE** : Informa o tipo de supernova
- **TERSE LIGHT CURVE OUTPUT** : Informa os pontos de observações obtidos. Será nosso *Data Frame*.
 - **MJD** : O momento em que a observação foi obtida. A medida está descrita em dias do calendário Juliano.
 - **FLT** : O filtro em que a observação foi obtida, pois cada observação é obtida a partir de um determinado espectro de luz.
 - **FLUXCAL** : O valor do fluxo de luz obtido naquela observação.
 - **FLUXCALERR** : O valor do erro do fluxo de luz obtido naquela observação.

Em relação ao **MJD**, vale ressaltar que é um método usado na astronomia para contar os dias sequencialmente, começando em uma data arbitrária no passado. Neste projeto, essas datas serão normalizadas, tendo como 0 o menor valor do MJD para que nosso eixo das abscissas tenha sempre início em 0.

```

FAKE:      2      (=> simulated LC with snlc_sim.exe)
MWEBV:    0.0111      MW E(B-V)
REDSHIFT_HELIO: 0.35020 +- 0.04680 (Hello, z_best)
REDSHIFT_FINAL: 0.35020 +- 0.04680 (CMB)
REDSHIFT_SPEC: -9.00000 +- 9.00000
REDSHIFT_STATUS: OK

HOST_GALAXY_GALID: 18943
HOST_GALAXY_PHOTO-Z: 0.3502 +- 0.0468

SIM_MODEL: NONIA 10 (name index)
SIM_NON1a: 31 (non1a index)
SIM_COMMENT: SN Type = II , MODEL = SDSS-017862
SIM_LIBID: 4
SIM_REDSHIFT: 0.3972
SIM_HOSTLIB_TRUEZ: 0.4000 (actual Z of hostlib)
SIM_HOSTLIB_GALID: 18943
SIM_DLMU: 41.666473 mag [ -5*log10(10pc/dL) ]
SIM_RA: 0.500000 deg
SIM_DECL: -43.000000 deg
SIM_MWEBV: 0.0113 (MilkyWay E(B-V))
SIM_PEAKMAG: 26.78 28.54 28.42 26.27 (griz obs)
SIM_EXPOSURE: 1.0 1.0 1.0 1.0 (griz obs)
SIM_PEAKMJD: 56214.511719 days
SIM_SALT2x0: 2.155e-17
SIM_MAGDIM: 0.000
SIM_SEARCHEFF_MASK: 3 (bits 1,2=> found by software,humans)
SIM_SEARCHEFF: 1.0000 (spectro-search efficiency (ignores pipelines))
SIM_TRESTMIN: -31.11 days
SIM_TRESTMAX: 64.81 days
SIM_RISETIME_SHIFT: 0.0 days
SIM_FALLTIME_SHIFT: 0.0 days

SEARCH_PEAKMJD: 56214.434

# =====
# TERSE LIGHT CURVE OUTPUT:
#
NOBS: 93
NVAR: 9
VARLIST: MJD FLT FIELD FLUXCAL FLUXCALERR SNR MAG MAGERR SIM_MAG
OBS: 56171.051 g NULL -3.679e+01 7.809e+01 -0.47 99.000 5.000 99.055
OBS: 56172.039 r NULL 4.410e+00 1.758e+01 0.25 99.000 5.000 98.914
OBS: 56172.051 i NULL 6.447e-01 4.995e+01 0.01 99.000 5.000 98.994
OBS: 56176.160 z NULL 1.337e+00 3.581e+00 0.37 27.184 100.816 59.034
OBS: 56179.023 g NULL 6.122e+00 6.377e+00 0.96 25.533 102.467 60.656
OBS: 56179.031 r NULL -3.697e+00 4.331e+00 -0.85 128.000 0.000 63.358
OBS: 56179.047 i NULL 1.439e+01 7.145e+00 2.01 24.605 0.745 59.327
OBS: 56179.062 z NULL 5.544e+00 8.139e+00 0.68 25.640 102.359 54.103
OBS: 56180.031 g NULL 2.865e+00 5.671e+00 0.51 26.357 101.643 59.296
OBS: 56180.047 r NULL 1.002e+00 4.492e+00 0.22 27.498 100.502 61.984
OBS: 56180.062 i NULL 1.095e+01 6.740e+00 1.62 24.902 1.038 57.955
OBS: 56180.078 z NULL -1.652e+00 1.016e+01 -0.16 128.000 0.000 52.927
OBS: 56188.004 g NULL 5.279e+00 5.677e+00 0.93 25.694 102.306 52.990
OBS: 56188.016 r NULL 5.595e-01 4.368e+00 0.13 28.130 99.870 52.880
OBS: 56188.027 i NULL 3.962e+00 6.462e+00 0.61 26.005 101.995 50.157

```

Figura 3.1: Arquivo *.txt* a ser lido pelos arquivos de pré-processamento.

Em relação ao **FLT**, cada objeto astronômico é visto através de 4 filtros de luz diferentes, obtendo assim 4 curvas de luz para cada um deles. Para efeitos de aprendizagem de máquina, cada objeto possuirá 4 *data frames*, um para cada filtro, e posteriormente essas propriedades serão convertidas em *features* de cada objeto.

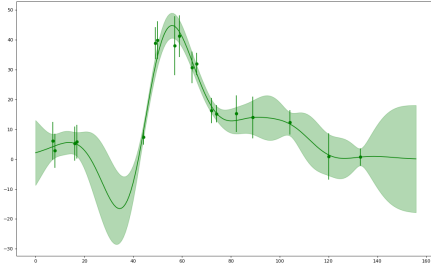
A figura 3.2 ilustra as 4 curvas de luz para o mesmo objeto.

3.2 Pipeline Atual

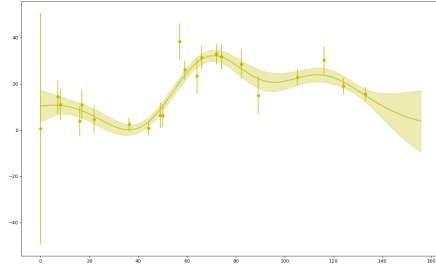
A partir do pré-processamento explicado na seção anterior, o *pipeline* atual irá dispor destes dados para realizar as etapas do processo de classificação.

Inicialmente ele irá separar os dados do *Terse Light Curve Output* pelos filtros e irá criar 4 *numpy arrays* possuindo $n \times 3$ dimensões cada (**MJD**, **FLUXCAL** e **FLUXCALERR**) e sendo n o número de amostras.

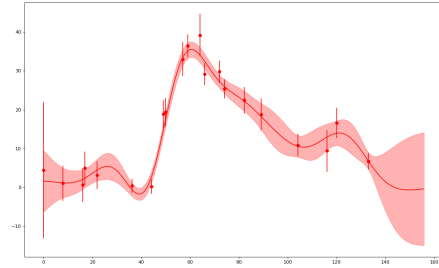
Em seguida, esses 4 *numpy arrays* serão a entrada do modelo de *Gaussian Process*, cuja função é interpolar uma curva que passe pelos pontos de cada filtro, como pode ser visto na figura 3.2.



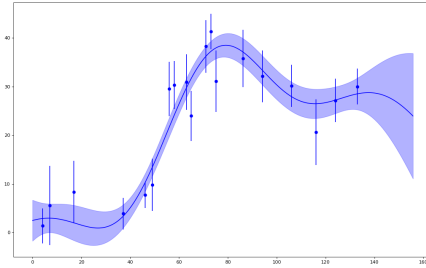
(a) Filtro *desg*



(b) Filtro *desi*



(c) Filtro *desr*



(d) Filtro *desz*

Figura 3.2: Exemplo de 4 curvas de luz de um mesmo objeto, uma para cada filtro. Gráfico Fluxo \times Tempo (dias).

A saída do Processo Gaussiano será o gráfico interpolado em forma de *numpy array* 100×3 , contendo 100 pontos de abscissa, 100 pontos de ordenada e 100 pontos do erro da ordenada.

Essa saída será a entrada do processo de *wavelets*, que irá retornar 400 valores para cada entrada.

No total, cada objeto terá esse procedimento feito 4 vezes, uma para cada filtro. Assim, como saída do processo de *wavelets* teremos 1600 coeficientes para cada objeto astronômico.

Em seguida, é realizada uma redução de variáveis através da **análise de componentes principais** ou **PCA**. Após esse procedimento, cada objeto possuirá 20 *features*.

Após essa etapa, o *dataframe* já estará composto por 21316 amostras com 20 *features* cada.

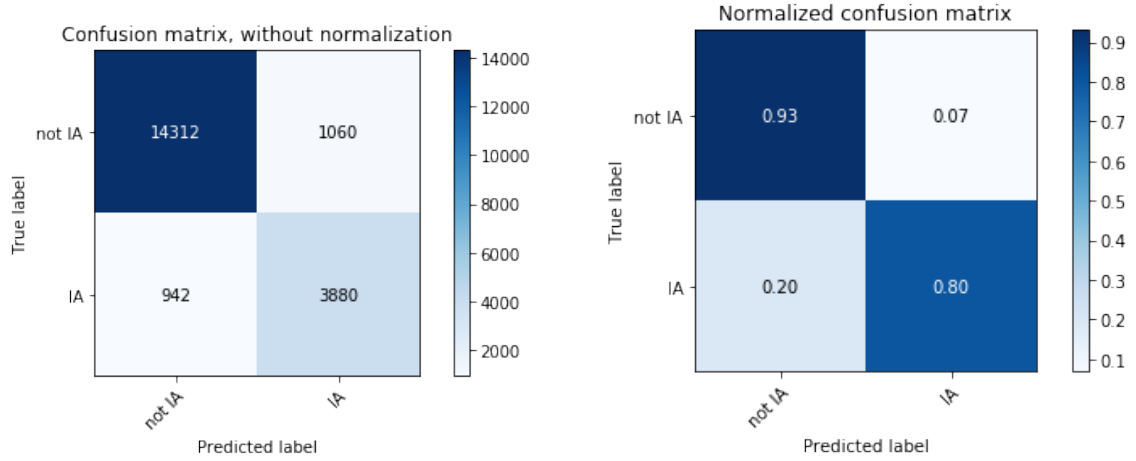
Em paralelo, usamos o **SN TYPE** do dicionário para guardar os *labels* de cada objeto astronômico em forma de lista.

Finalmente será criado o modelo de classificação baseado em *Random Forest* para classificar o tipo de objeto astronômico. Usamos o módulo *sklearn.pipeline* da biblioteca *scikit learn*.

Tendo em mãos o *dataframe*, a lista com as classificações de cada objeto (os

labels) e o modelo de classificação, é feito a divisão usando o método *k-fold* de validação cruzada. Nele dividimos os objetos em 19 partições e usa-se 1 delas para treino e as demais para teste.

Por fim, cada modelo dos 19 *folds* existentes é avaliado através de 2 maneiras de obtenção de *score*, a **curva AUC** e o **método de precisão média** (seção 2.3). Podemos também analisar as matrizes de confusão de cada um dos 19 modelos para ilustrar seus acertos, seus falsos positivos e falsos negativos (figura 3.3).



(a) Exemplo de Matriz de Confusão, valores absolutos.

(b) Matriz de Confusão, valores normalizados.

Figura 3.3: Matrizes de Confusão do modelo final do *pipeline* original.

Capítulo 4

Tratamento de *Outliers*

4.1 Estratégias Utilizadas

4.2 Resultados e comparações

Capítulo 5

Rede Neural Convolucional

A abordagem utilizando *Deep Learning* teve como base uma análise das interpolações, nela foram notadas diferenças entre gráficos de diferentes tipos assim como pequenas semelhanças entre aqueles do mesmo tipo.

Pode-se citar também que recentemente diversos algoritmos vêm tendo seus *pipelines* reduzidos e seu desempenho melhorado graças a adição do *Deep Learning* (como processamento de imagens e identificação de voz).

5.1 Geração de Imagens e Parâmetros

A primeira etapa desta abordagem consiste em gerar gráficos a partir da interpolação gaussiana. Para comparar o resultado deste método com o *pipeline* inicial, a interpolação realizada para gerar os gráficos fornecidos ao DL é a mesma interpolação usada no *pipeline* original.

Optou-se por gerar gráficos ao invés de utilizar os pontos diretamente, devido à interpolação possuir mais informações ao aprendizado da máquina do que apenas pontos soltos em uma imagem.

Estas imagens serão gráficos em escalas de cinza na extensão *png* cujas dimensões são: 64×40 pixels. Ao analisá-las, notou-se que existe uma margem ao gerar as imagens, esta margem contém 5 pixels na dimensão y e 9 pixels na dimensão x . No fim, essas margens foram eliminadas resultando em um formato de dimensão 46×30 .

Esses tamanhos foram escolhidos baseando-se em exemplos clássicos e bem conhecidos da literatura que possuem dimensões com valores bem próximos, como o MNIST e Fashion MNIST.

5.2 Pipeline

Após a transformação dos arquivos de imagem para matrizes de *nparray*, foi construída uma estrutura de dados para cada objeto como um compilado de suas 4 figuras. Obtendo o formato final de **(21316, 4, 30, 46)**, onde:

- Número de objetos: 21316
- Número de imagens ou matrizes (referente a cada filtro): 4
- Número de pixels por coluna (ou número de linhas): 30
- Número de pixels por linha (ou número de colunas): 46

Dentre diversos possíveis tipos de camadas internas de *Deep Learning*, foram escolhidas camadas do tipo **Convolucionais 2D** e **Max Pooling 2D** (seção 2.2).

A arquitetura escolhida também foi baseada em exemplos clássicos de problemas de identificação de imagens, em especial o Fashion MNIST.

Tendo como base esses exemplos, algumas camadas foram adicionadas e modificadas de forma a buscar empiricamente obter um melhor resultado. A arquitetura final encontra-se na figura 5.1.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 46, 30, 64)	4160
conv2d_1 (Conv2D)	(None, 46, 30, 64)	36928
max_pooling2d (MaxPooling2D)	(None, 23, 15, 64)	0
conv2d_2 (Conv2D)	(None, 23, 15, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 11, 7, 64)	0
flatten (Flatten)	(None, 4928)	0
dense (Dense)	(None, 2)	9858
Total params: 87,874		
Trainable params: 87,874		
Non-trainable params: 0		

Figura 5.1: Arquitetura do modelo de *Deep Learning* utilizado.

O último valor do *Shape* de cada camada é a quantidade de neurônios que ela possui, enquanto os demais valores são o número de linhas e colunas da imagem. O valor '4' que era de se esperar devido aos 4 filtros não aparece explicitamente na estrutura da arquitetura do modelo, todavia, é um comportamento normal visto que neste caso ele funciona como se fosse uma imagem em "RGBA", onde teria uma matriz para vermelho, azul, verde e transparência.

Em seguida, foi definida uma *random seed* para poder fixar quaisquer tipos de aleatoriedades intrínsecas ao modelo. Assim, o mesmo foi treinado em 2,3,5 e 10 épocas (número este também baseado nos exemplos citados), obtendo um resultado satisfatório quando o valor foi de 10 épocas, pois sua acurácia foi alta sem deixar o algoritmo dependente.

5.3 Resultados e comparações

Os resultados obtidos para a classificação do melhor modelo podem ser vistos nas matrizes de confusão da figura 5.2

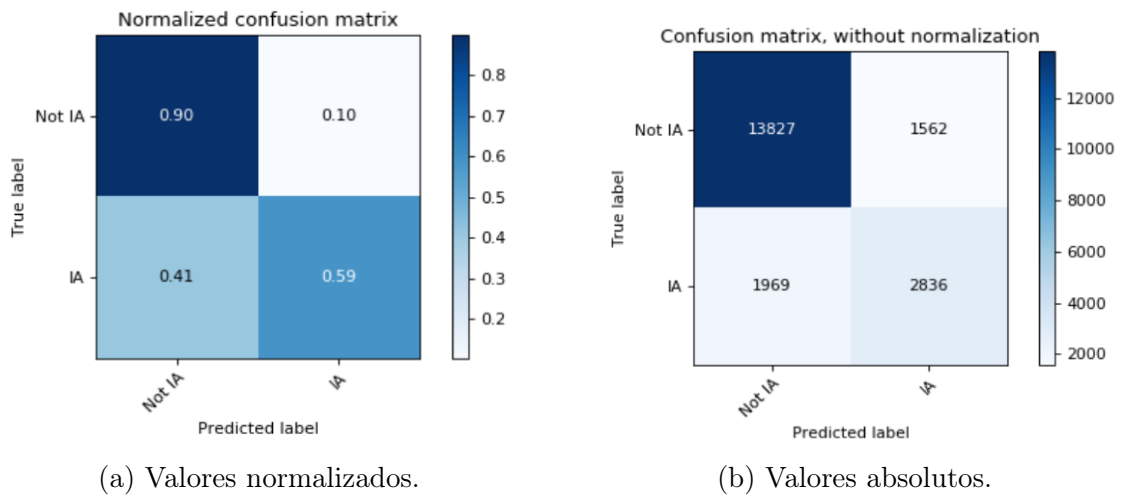


Figura 5.2: Matrizes de confusão do modelo final.

Ao compararmos os resultados da figura 5.2 com o da figura 3.3 percebemos que o modelo de *Deep Learning* apresentou piora em seu desempenho.

Dois motivos podem explicar tal desempenho, o primeiro deles é que as imagens treinadas não sofreram o chamado *data augmentation*, uma técnica que busca aumentar a quantidade de dados treinados e variar suas formas de identificação, focando em rotacionar, deslocar, reduzir e aumentar as figuras. Devido as imagens deste projeto serem gráficos, haveria uma perda de sentido caso os mesmos fossem rotacionados ou deslocados.

O segundo motivo é a limitação física que faz com que sejam treinados apenas 1100 objetos astronômicos. Normalmente algoritmos de *Deep Learning* mostram-se mais vantajosos do que algoritmos de ML devido a grande quantidade de dados que os faz ter um desempenho melhor.

Por fim, também foram feitos experimentos envolvendo uma distribuição de dados de 80% para treino de 20% para teste. A análise dessas matrizes de confusão (figuras 5.3 e 5.4) confirmam que para este problema a aplicação de *Deep Learning*

analisando o formato dos gráficos não se mostra eficiente, reiterando a justificativa da ausência do *data augmentation*.

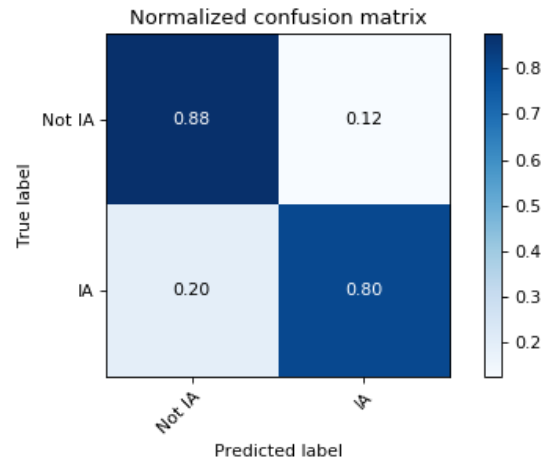


Figura 5.3: Valores normalizados do modelo DL treinando com 80% dos dados.

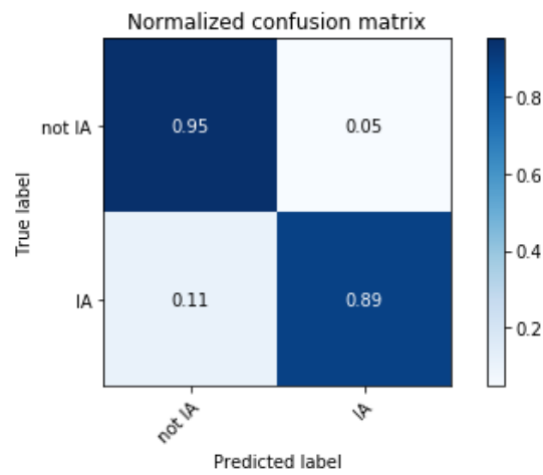


Figura 5.4: Modelo do *pipeline* original treinado com 80% dos dados.

Capítulo 6

Interpolação através de Processo Gaussiano

6.1 Escolha da Biblioteca

6.2 Aleatoriedades e *Random Seeds*

6.3 *Kernel Functions*

6.4 Demais Observações

Valor negativo e Mean function, erro grande, melhor overfittado que underfittado

6.5 Resultados das Interpolações

6.6 Identificação e Justificativa do Erro

Capítulo 7

Resultados e Discussões

Capítulo 8

Conclusões

8.1 Conclusões Finais

Falar que o Algoritmo de hoje já é bem robusto

8.2 Trabalhos Futuros

Falar de consertar o Erro e pah

Referências Bibliográficas

- [1] LOCHNER, M. “Photometric Supernova Classification with Machine Learning”, *ApJS (2016)*, v. 225, n. 2, pp. 31, set. 2016.
- [2] KESSLER, R. “Supernova Photometric Classification Challenge”, *arXiv*, v. 6, n. 5210, pp. 4, abr. 2010.
- [3] RASMUSSEN, C., WILLIAMS, C. K. I. *Gaussian Processes for Machine Learning*. 1 ed. New York, Thomas Dietterich, Editor, 2004.
- [4] BAILEY, K. “From both sides now: the math of linear regression”, <http://katbailey.github.io/post/from-both-sides-now-the-math-of-linear-regression/>, v. 1, n. 1, pp. 1, jun. 2016.
- [5] BAILEY, K. “Gaussian Processes for Dummies”, <http://katbailey.github.io/post/gaussian-processes-for-dummies/>, v. 1, n. 1, pp. 1, jun. 2016.
- [6] JOCHEN GÖRTLER, REBECCA KEHLBECK, O. D. “A Visual Exploration of Gaussian Processes”, <https://distill.pub/2019/visual-exploration-gaussian-processes/>, v. 1, n. 1, pp. 1, 2019.
- [7] DANI GAMERMAN, H. F. L. *Markov Chain Monte Carlo Stochastic Simulation for Bayesian Inference*. 2 ed. New York, Chapman and Hall/CRC, 2006.
- [8] MURPHY, K. P. *Machine Learning A Probabilistic Perspective*. 1 ed. Massachusetts Institute of Technology, MIT, 2012.
- [9] CELSO, A. “Notas de Aula de Sinais e Sistemas”, *UFRJ - COE350 - Sinais e Sistemas - ECA*, jun. 2014.

Apêndice A

Repositório do Código

Todo o código desenvolvido encontra-se disponível no repositório público do *GitHub*, a fim de poder ser desenvolvido e modificado por qualquer pessoa que deseje contribuir ou estudar este projeto, podendo ser acessado no seguinte link: https://github.com/FelipeMFO/supernova_identification.