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, 43 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. 39 – 41.

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.

"Life is tough, that's a given. When you stand up, you're gonna be shoved back down. When you're down, you're gonna be stepped on.

It's no secret, you'll fall down, you stumble, you get pushed, you land square on your face. And every time that happens, you get back on your feet. You get up just as fast as you can, no matter how many times you need to do it.

Success has been and continues to be defined as getting up one more time than you've been knocked down.

Nothing is free and living ain't easy. Life is hard, real hard, incredibly hard. You fail more often than you win, nobody is handing you anything.

It's up to you to puff up your chest, stretch your neck and overcome all that is difficult, the nasty, the mean, the unfair.

That's how winners are made."

This is how winners are made
- Autor Desconhecido.

Agradecimentos

Primeiramente agradeço aos meus pais, Antonio Paulo e Margareth. Por todo o suporte e educação, tendo sido de cunho afetivo, emocional, moral, financeiro e espiritual. Sem vocês eu não seria a pessoa que hoje sou grato de ser, e muito menos estaria buscando continuar a evoluir como ser humano. Obrigado por todos os seus erros e seus acertos, amo vocês.

Aos meus colegas, amigos e companheiros da T-17, faltam aqui palavras para poder descrever o quão **cada um** mostrou-se importante para o grupo como um todo. Em questão de união e evolução acadêmica, guardo meus comentários porque sei que os mesmos não iriam conseguir descrever tudo o que só nós sabemos o que passamos. Em especial, gostaria de agradecer em diferentes aspectos a vários nomes que puderam não somente me ajudar a passar nas matérias, mas também a moldar o caráter da pessoa que sou hoje. Através de todos esses anos de convivência com nossas diferenças e semelhanças, pude ver em vocês mais do que colegas de turma, pude ver exemplos de pessoas em quem me espelho para ser melhor.

Ao corpo docente, agradeço primeiramente ao professor Afonso, um pai-coordenador que graças à sua atenção me fez não desistir de engenharia. Ao professor Ribamar que desde o início do meu trajeto pôde me ensinar o conhecimento da ciência que mais sou apaixonado, a cosmologia. A todo o corpo docente da *École des Mines d'Alès* que me fez descobrir a paixão no meu trabalho através da ciência de dados. No escopo desse projeto, agradeço imensamente as inúmeras ajudas vindas do pós-doutorando Marcelo Vargas e dos meus orientadores professores Amit e Ribamar.

À Taís, é incomensurável minha gratidão por ser uma companheira que me ajuda sempre que caio, sempre que estou deprimido com crise de ansiedade, ou encarando dificuldades como catapora, tornozelo quebrado e tantas outras que passei durante esse fim de jornada. Na alegria ou na tristeza, na saúde ou na doença você sempre esteve comigo, muito obrigado por tudo.

Último e não menos importante agradeço ao Universo, que com sua perfeição e grandeza me motivou e me motiva diariamente a buscar meus objetivos e meus caminhos.

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 mostra-se muito custoso. Como consequência, utiliza-se um método com baixo custo 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 ALGORYTHMS

Felipe Matheus Fernandes de Oliveira

July/2019

Advisors: Amit Bhaya
Ribamar Rondon de Rezende dos Reis

Course: Automation and Control Engineering

In order to study the expansion of the universe, cosmology classifies different types of astronomical objects using spectroscopy. Given the enourmous size of current datasets, spectroscopy methods could not classify 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 3
1.4 Objetivos 4
1.4.1 Processo Gaussiano *Gaussian Process* 4
1.4.2 Aprendizagem Profunda 4
1.4.3 Tratamento de *Outliers* 5
1.5 Metodologia 6
1.6 Descrição 8

2 Fundamentos Teóricos **9**

2.1 Processo Gaussiano 9
2.1.1 *Kernel* e Função de Covariância 10
2.1.2 Função Média 11
2.2 Redes Neurais Convolucionais 12
2.3 Demais Conceitos 15
2.3.1 Análise de Componentes Principais 15
2.3.2 Transformadas de Wavelet 15
2.3.3 Validação cruzada *K-Fold* 16

3 Modelo Atual **17**

3.1 Pré-processamento dos Dados Brutos 17
3.2 *pipeline* Atual 18

4 Tratamento de *Outliers* **22**

4.1 Estratégias Utilizadas 22
4.2 Resultados e comparações 23

5 Rede Neural Convolucional **24**
5.1 Geração de Imagens e Parâmetros 24
5.2 *pipeline* 24
5.3 Resultados e comparações 26

6 Interpolação através de Processo Gaussiano **28**
6.1 Escolha da Biblioteca 28
6.2 Aleatoriedades e *Random Seeds* 29
6.3 *Kernel Functions* 30
6.4 Demais Observações 31
6.5 Resultados das Interpolações 32
6.6 Identificação e Justificativa do Erro 35

7 Conclusões **37**
7.1 Conclusões Finais 37
7.2 Trabalhos Futuros 38

Referências Bibliográficas **39**

A Repositório do Código **42**

B Avaliação das Interpolações **43**

Lista de Figuras

1.1 Exemplo de interpolação via GP não condizente com a realidade de uma explosão. 3
1.2 Exemplo de *outlier*. 5
1.3 Exemplo de ponto com incerteza 3 vezes maior que o pico da curva. . 6

2.1 Ilustração da matriz Σ possuindo 10 e 12 pontos. 11
2.2 Ilustração do valor da função média. 12
2.3 Arquitetura básica de *Deep Learning* 13
2.4 Operação de Convolução. 14
2.5 *Max Pooling.* 14

3.1 Arquivo *.txt* a ser lido pelos arquivos de pré-processamento. 18
3.2 Observação íntegra de um objeto astronômico. 19
3.3 Exemplo de 4 curvas de luz de um mesmo objeto. 20
3.4 Matrizes de Confusão de um modelo final do *pipeline* original. 21

5.1 Arquitetura do modelo de Deep Learning utilizado. 25
5.2 Matrizes de confusão do modelo final. *Deep Learning*. 26
5.3 Valores normalizados do modelo DL treinando com 80% dos dados. . 27
5.4 Modelo do *pipeline* original treinado com 80% dos dados. 27

6.1 Exemplo de distribuição com valores não-negativos. 32
6.2 Comparação dos resultados finais para um exemplo. 34
6.3 Matrizes de confusão do *Matern 5/2* pela biblioteca *george*. 35

Lista de Tabelas

4.1 Resultados do tratamento de *outliers*. 23

6.1 Resultados do *Kernel Mattern 5/2* pela biblioteca *george*. 33

Capítulo 1

Introdução

1.1 Tema e Contextualização

Dentro da cosmologia existe a necessidade de determinar distâncias (através das chamadas distâncias luminosas [1]) para modelar seus estudos, como ferramenta utilizam-se as curvas de luz provenientes de supernovas do tipo Ia.

Sabendo que curvas de luz são medidas de fluxo (energia por tempo por área [2]) em função do tempo, para determinar a distância de um objeto a partir desta medida, é preciso conhecer a potência intrínseca do mesmo. Um objeto com tal potência conhecida é chamado de vela padrão [3]. Apesar de supernovas não serem velas padrões, as do tipo Ia podem ser padronizadas através de correlações empíricas entre suas características observáveis. Esse processo de padronização funciona apenas com esse tipo de Supernova, o que torna fundamental a classificação correta desses objetos.

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 [4], um método que fornece medidas de fluxo em função do comprimento de onda ou frequência. Apesar da espetroscopia confirmar precisamente o tipo de cada supernova, com a grande quantidade dos dados possuídos atualmente ela acaba se mostrando um método lento e custoso.

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 classificar essa grande quantidade de objetos captados. Dentre essas abordagens, várias utilizam aprendizado de máquina.

Por fim, a ideia do projeto foi derivada de um artigo publicado por LOCHNER [5] e colaboradores. A autora busca criar uma maneira automática de classificação fotométrica usando as curvas de luz obtidas através da fotometria [6], um método que fornece medidas de fluxo em filtros de banda larga (tipicamente 1000 Angstrons); onde tais curvas já foram devidamente classificadas no passado utilizando espectroscopia.

Em suma, utiliza-se um método mais rápido para obter menos informações de objetos astronômicos, e tendo a classificação precisa dos mesmos, treina-se um método de aprendizado de máquina para que este possa classificar precisamente mesmo possuindo menos dados captados.

1.2 Problemática

No presente trabalho, o termo *pipeline* será utilizado diversas vezes para se referir a **sequência de passos ou procedimentos/algoritmos** utilizados ao longo do processo de classificação, caracterizando a construção das etapas do processo como um todo.

Tendo em vista que o artigo [5] testou e validou diversos *pipelines*, este trabalho adotou aquele em que ela obteve o melhor resultado de classificação, e a partir desse ponto, foram aplicadas modificações para avaliar quaisquer possíveis melhoras.

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

- Interpolação utilizando Processo Gaussiano (*Gaussian Process* ou **GP**).
- Transformada de Wavelet das funções interpoladas.
- Análise de componentes principais (*Principal Component Analysis* ou **PCA**) dos parâmetros gerados pela transformada de Wavelet.
- Floresta Aleatória (*Random Forest* ou **RF**) aplicada à classificação dos objetos astronômicos, através das suas componentes principais (resultado do PCA).

A problemática principal encontra-se no método de interpolação chamado **Processo Gaussiano**. Por ser um método de interpolação, espera-se que ele defina uma função que passe pelos pontos obtidos respeitando as incertezas em suas medições. Entretanto, onde esperava-se ver uma variação temporal de fluxo considerável, em alguns casos o gráfico interpolado é uma constante 1.1. Consequentemente, tal interpolação não possui sentido físico, visto que se trata do fluxo de luz após a explosão de um objeto astronômico; além de também violar a segunda lei da termodinâmica [7], pois uma explosão que tenha fluxo de luz constante representaria um objeto de energia infinita.

O objetivo principal deste trabalho é propor uma modificação do método de interpolação usando GP que evite o comportamento que não esteja de acordo com as leis da física.

Em paralelo, outras ideias foram aplicadas na tentativa de obter uma melhora no algoritmo utilizado atualmente pelo Instituto de Física da UFRJ (**IF-UFRJ**).

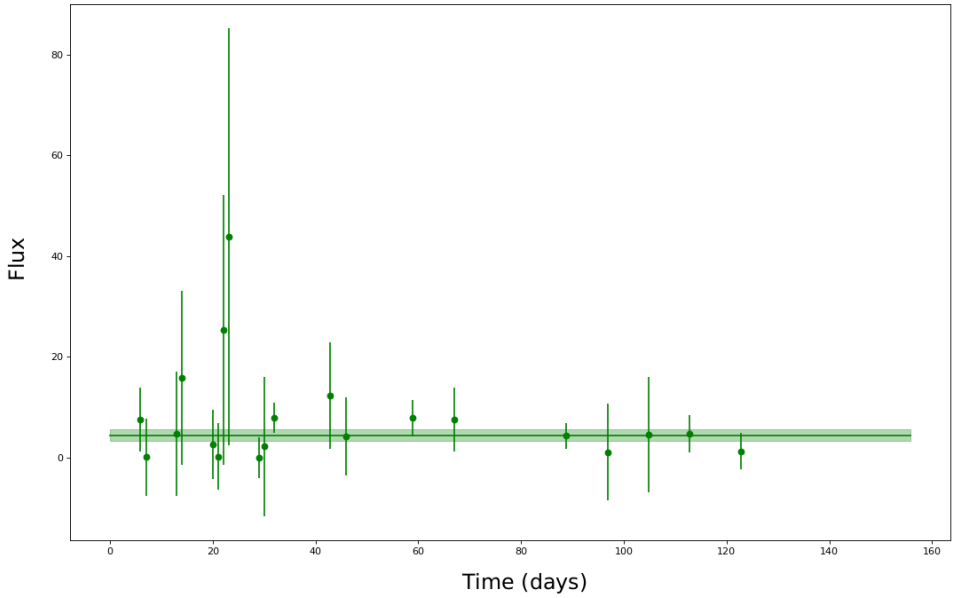


Figura 1.1: Exemplo de interpolação via GP não condizente com a realidade de uma explosão.

1.3 Delimitação

Todos os dados utilizados tanto neste trabalho quanto no artigo [5], foram extraídos da base de dados oferecida pelo desafio KESSLER [8]. Este desafio proposto pelo *Argonne National Laboratory* [9] abriu o problema de classificação de supernovas ao público para tentar obter melhores resultados e/ou novos algoritmos. A base de dados é de domínio público e pode ser obtida no repositório do desafio.

Devido à granularidade e abundância dos dados brutos, foi necessário um pré-tratamento buscando apenas os dados que serão utilizados. Esse pré-tratamento foi aproveitado do *pipeline* do Instituto de Física da UFRJ, cujo processo visa ler os arquivos de texto transformando cada informação em uma chave de dicionário.

Ao fim dessa seleção de dados foram obtidos valores em forma de dicionários em Python para cada objeto astronômico. Esses dicionários serão os dados brutos dentro do escopo deste trabalho.

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 projeto.

- Interpolação utilizando Processo Gaussiano.
- Aprendizagem Profunda
- Tratamento de *Outliers*

1.4.1 Processo Gaussiano *Gaussian Process*

O Processo Gaussiano, é a primeira parte do tratamento após a transformação dos dados brutos em dicionários de Python. Esta etapa será abordada no Capítulo 6.

Cada objeto astronômico possui uma quantidade de pontos que representam a intensidade do fluxo de luz captado em determinado dia e em determinado filtro. Associado a cada ponto, também há uma incerteza desse valor. Assim, busca-se uma interpolação que seja uma variação temporal considerável do fluxo passando 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**) vem 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. Tendo em vista que o mesmo possui diversas partes entre os dados brutos e algoritmo de Random Forest, busca-se 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 [5], 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 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*

A última vertente 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, e possível observar 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.

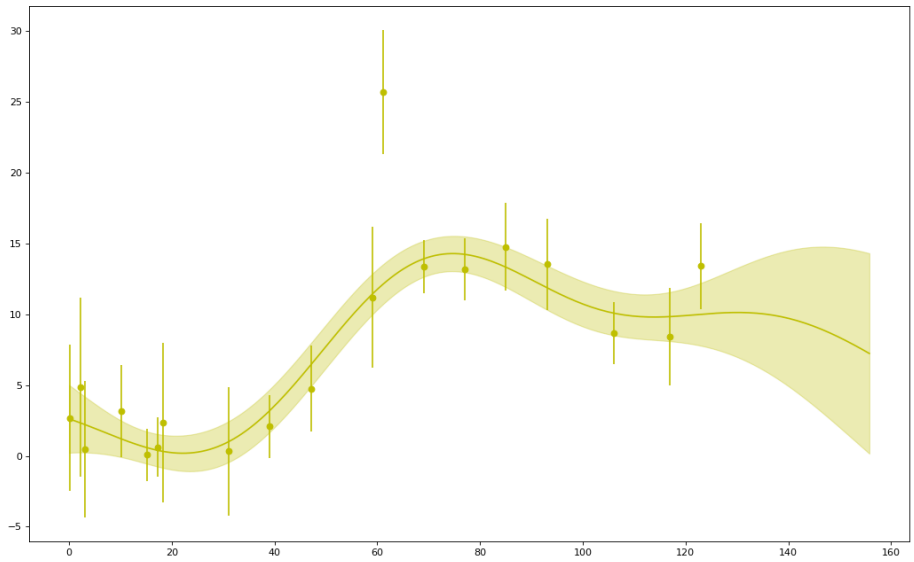


Figura 1.2: Exemplo de *outlier*. Gráfico Fluxo × Tempo (dias).

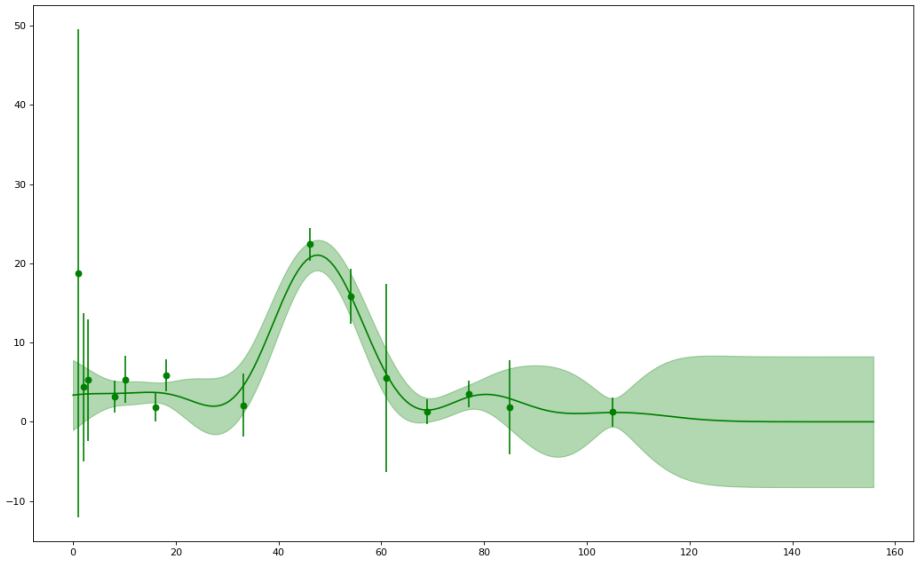


Figura 1.3: Exemplo de ponto com incerteza 3 vezes maior que o pico da curva. Gráfico Fluxo × Tempo (dias).

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 sua escolha foi a facilidade que a mesma possui para realizar experimentos através da plataforma *Jupyter*, além da grande quantidade de bibliotecas de ciência de dados e suporte para desenvolver algoritmos de ML.

As principais bibliotecas utilizadas no contexto de tratamento de dados foram *Pandas* [10] e *Numpy* [11]. Já para a aplicação dos métodos, as bibliotecas mais importantes foram *scikit-learn* [12], *Keras* [13] e *PyMC3* [14].

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, de modo a poder avaliar especificamente a eficiência e melhora dessa mudança.

Por se tratar de um algoritmo de classificação, a saída do 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, normalmente uma divisão de 80% dos dados para treinamento e 20% para o teste dos modelos. Neste projeto foram utilizados apenas 1100 objetos para treino dentre 21316 objetos totais. A justificativa é a limitação física das amostras (espera se ter apenas 1100 objetos espectroscopicamente confirmados).

Por fim, as mudanças foram verificadas através de **Matrizes de Confusão** (*Confusion Matrices* ou **CM**) ou através de métodos 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, serão explicados 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** dos mesmos, buscando averiguar a eficiência de aplicação ou explicar possíveis erros.

Por fim, no Capítulo 7 serão apresentadas as **conclusões finais do projeto e sugestões de trabalhos futuros**, junto às limitações e possíveis soluções encontradas.

Capítulo 2

Fundamentos Teóricos

2.1 Processo Gaussiano

O Processo Gaussiano é uma ferramenta importante para algoritmos de Machine Learning, pois permite fazer predições sobre os dados tendo como base um conhecimento a priori. A sua aplicação mais frequente é em interpolações de funções, como é o caso deste 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 [15] .

Visto que para uma quantidade determinada de pontos existe uma infinidade de funções que podem interpolar esses valores, o Processo Gaussiano realiza sua interpolação a partir da expectativa a priori dos valores e do formato que ela pode assumir devido à relação entre seus pontos. 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 da interpolação) e um desvio padrão (incerteza).

Neste projeto não são abordados maiores detalhes para o aprendizado do método, contudo, na bibliografia são encontradas tanto referências mais específicas e técnicas [15], quanto mais práticas [16], [17], [18].

Os dois aspectos fundamentais do Processo Gaussiano são os principais responsáveis por caracterizar as interpolações. Eles serão apresentados nas subseções a seguir.

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.

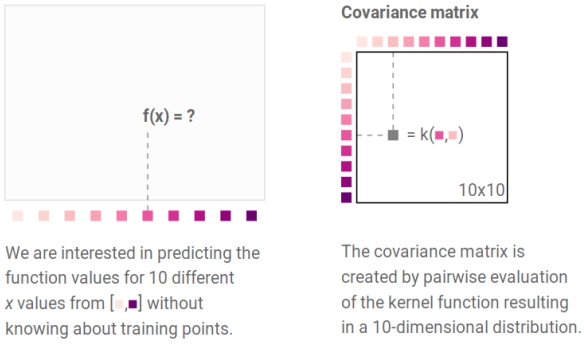
O diferencial do GP é considerar uma distribuição normal multivariável de dimensão $n + m$ durante a interpolação. Essa distribuição normal multivariável possuirá uma covariância Σ, responsável não por apenas descrever o **formato** (senoidal, linear), como também por determinar **características** (periodicidade,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

Para poder obter a matriz de covariância (Σ) desta normal multivariável, é estabelecido o *Kernel* do GP. Ele é uma função especial de duas variáveis que respeita certas restrições matemáticas (Cap. 4 RASMUSSEN e WILLIAMS [15]). Essas varíaveis são os valores da abscissa de cada ponto da função a ser interpolada.

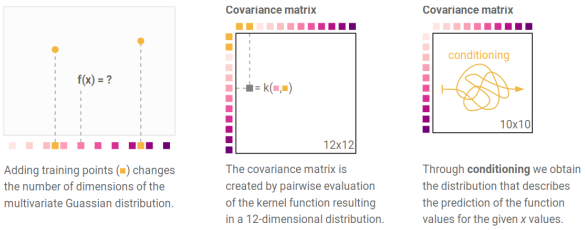
Em outras palavras, o *Kernel* é uma função que irá relacionar dois pontos que deverão estar na curva interpolada, baseando se apenas na distância entre eles (para *Kernels* estacionários) ou nos seus valores absolutos (para *Kernels* não-estacionários). É essa relação que fornecerá características como 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 por eles, através de **probabilidade condicionada** (***Posterior Distribution*** [18]). 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 na figura 2.1(b) que a probabilidade dos valores dos 10 pontos que desejam ser interpolados serão calculados através da probabilidade condicionada considerando os valores fixos dos 2 pontos pré-estabelecidos. Diferente do ilustrado na figura 2.1(a), onde não é usada probabilidade condicionada para o cálculo dos valores.



(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 [18]*

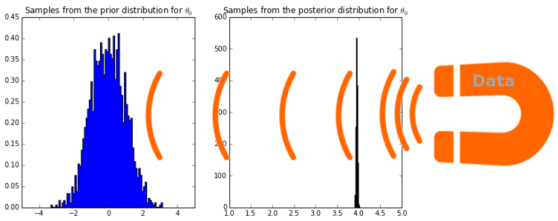
2.1.2 Função Média

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

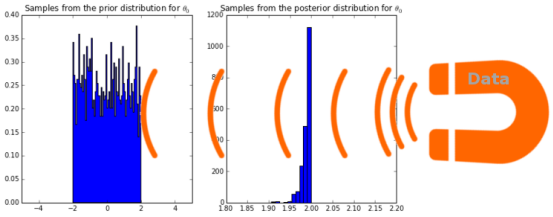
Contudo, só existe sentido em falar de Função Média caso seja associada à Função Covariância. Normalmente a Função Média é estabelecida como uma distribuição normal com média em 0 e desvio padrão 1.

É possível usar essa distribuição padrão como valor a priori, pois o valor final deste ponto será obtido após um número suficientemente grande de amostras para otimizar a distribuição. Elas são definidas através de amostragens usando **Cadeias de Markov Monte Carlo** [19], se aproximando cada vez mais do valor ideal. No fim, o valor do ponto será a média dessa distribuição a posteriori. 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 assuma alguns valores, como está ilustrado na 2.2(b).



(a) Exemplo de priori gaussiana que ao longo das amostragens passa a assumir 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 [17]*

2.2 Redes Neurais Convolucionais

Redes Neurais Convolucionais (*Convolutional Neural Network* **CNN**) são um tipo de rede neural em que *Deep Learning* é aplicado, normalmente usado para análise de imagens. O principal objetivo dentro deste projeto é usar algoritmos de Machine Learning, logo, para evitar quaisquer confusões com as nomenclaturas, vale lembrar que DL se refere a uma técnica dentro da vasta área que é ML.

Uma arquitetura básica de DL possui 3 partes, como pode ser visto na figura 2.3:

- *Input Layer*: A primeira camada; aquela responsável por receber uma amostra do dado e repassar para as camadas internas
- *Hidden Layers*: O grupo de camadas internas; nela encontram-se os nós que recebem os dados da *input layer* e conectam às camadas subsequentes através de funções de ativação não-lineares.
- *Output Layer*: A última camada; responsável pela resposta final. Em uma rede de classificação ela possui valores de saída numéricos, referindo-se as classes.

No caso deste projeto, são os números zero e um correspondendo as supernovas classificadas como *IA* ou *not IA*.

Cada camada é composta de diferentes tipos de **neurônios**. Eles são funções matemáticas com parâmetros específicos, os quais serão treinados a fim de classificar os dados.

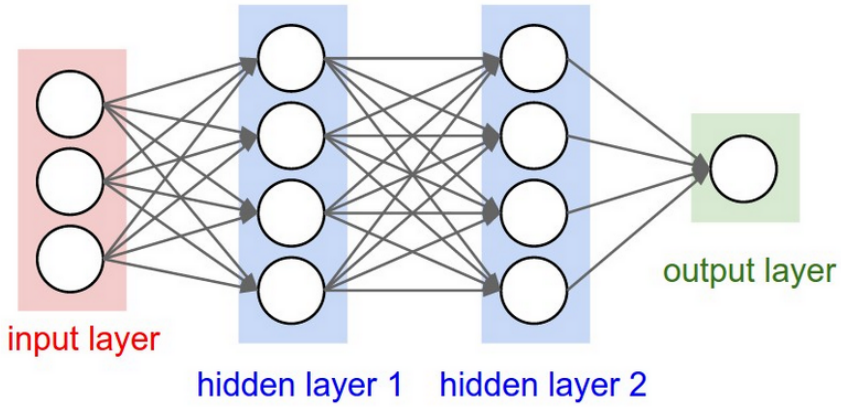


Figura 2.3: Arquitetura básica de *Deep Learning*. (retirado de: link.)

CNN é uma classe de redes neurais aplicada frequentemente na análise de imagens, neste projeto as imagens são arquivos *png* criados através da interpolação gaussiana, que serão representados em formato de matrizes de valores, como pode ser visto na matriz azul da figura 2.4.

Redes neurais convolucionais normalmente tratam-se de casos de redes totalmente conectadas, ou seja, cada neurônio em uma camada é conectado em todos os neurônios a camada seguinte. Essa alta conectividade faz com que o algoritmo demande o mínimo de pré-processamento possível quando comparada a outros métodos de classificação de imagens, pois caso sejam dadas amostras o suficiente, a rede "aprende" os filtros que em um algoritmo tradicional precisariam ser implementados manualmente [20]. Nas CNN também são usados *multilayers perceptrons* [21], a função de cada neurônio pode ser entendida como a ilustração da matriz verde na figura 2.4 e cada parâmetro sendo o número que multiplica o valor da matriz azul.

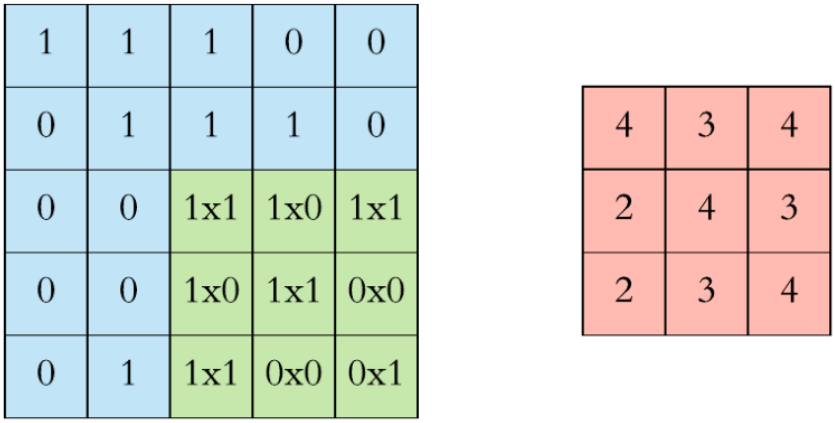


Figura 2.4: A matriz azul representa as variáveis de entrada; a vermelha os resultados da operação de convolução; e a verde possui os parâmetros para multiplicar valores da matriz azul "deslizando" através deles. (retirado de: link.)

CNNs podem possuir camadas de conjugação *pooling*, responsáveis por particionar retângulos da imagem de entrada em um conjunto menor de pixels para cada sub região. Dentre diversas funções *pooling* não-lineares, a mais usada é a de *max pooling* 2.5.

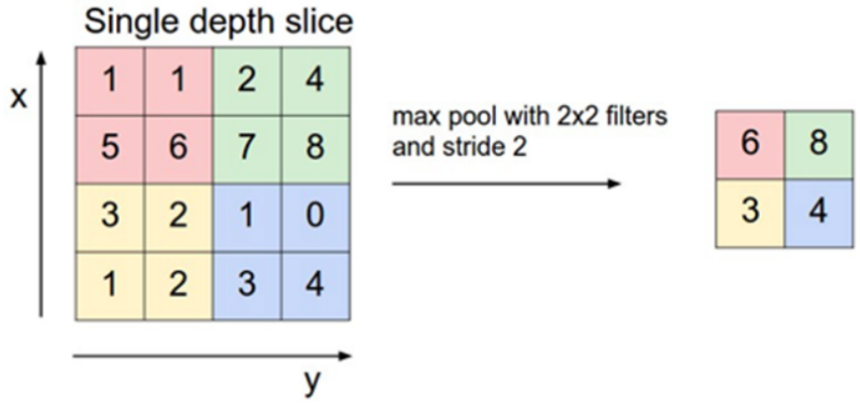


Figura 2.5: Exemplo do *Max pooling*, o maior valor dentro de um "retângulo" é selecionado e propagado para outra camada, esta possuindo sua dimensão reduzida (retirado de: link.)

Dentro das camadas internas é impossível afirmar precisamente qual o "sentido físico" de cada uma delas. Algumas podem possuir apenas significado matemático, enquanto outras podem ser responsáveis por identificar formas. Como por exemplo, na identificação de rostos, algumas camadas pondem ser responsáveis por detectar olhos, pés, curvas, etc; na identificação de animais, algumas podem detectar caudas, focinhos e patas.

É importante mencionar que existe um pequeno fator de aleatoriedade dentro dos processos de treinamento envolvendo a otimização dos parâmetros internos de cada neurônio. Para estabelecer uma comparação livre de possíveis erros causados pela aleatoriedade, foram usadas *random seeds*, funções do Python responsáveis por conservar a pseudoaleatoriedade [22].

2.3 Demais Conceitos

Dentro do vocabulário de ML, as expressões *dataframe* e *feature* são utilizadas para se referir respectivamente ao conjunto de amostras para treinar e testar o modelo, e às características delas que serão analisadas para a classificação.

2.3.1 Análise de Componentes Principais

A Análise de Componentes Principais (Cap. 12 MURPHY [23]) ou *Principal Component Analysis* (**PCA**) utiliza conceitos de ortogonalizações de vetores para poder reduzir dimensões.

Em ML essa redução de dimensões é aplicada às *features* de um *dataframe* onde as amostras possuem um grande número de características e deseja reduzi-las, com a finalidade de que o modelo de aprendizagem de máquina seja menos custoso e igualmente eficaz.

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

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

2.3.2 Transformadas de Wavelet

Transformadas de Wavelet [24] 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.

2.3.3 Validação cruzada *K-Fold*

Validação cruzada é uma família de métodos estatísticos usada para estimar como um modelo pode ser generalizado independente do conjunto de treino e teste. Em suma, é comumente utilizada para analisar se um modelo é robusto o suficiente, assim como para analisar a quantidade de falsos positivos e falsos negativos. Validação Cruzada *K-Fold* é um desses métodos.

Seu procedimento consiste em dividir os dados em K partes, **normalmente** $K-1$ são utilizados para treinar o modelo, enquanto 1 é usado para teste. Todavia, neste projeto devido as limitações físicas, apenas uma parte será usada para o treino enquanto as demais dezoito serão utilizadas para testar e validar o método.

Junto a validação cruzada existem 2 conceitos que serão usados para avaliar os resultados deste projeto

- Curva ROC e AUC (Cap. 5.7 MURPHY [23])
- *Mean Average Precision* (Cap. 9.7.4 MURPHY [23])
- *Confusion Matrix*.

Capítulo 3

Modelo Atual

Neste capítulo será detalhado o modelo atualmente utilizado para estudos cosmológicos do IF-UFRJ.

3.1 Pré-processamento dos Dados Brutos

Os dados brutos chegam para o pré-processamento da maneira ilustrada na figura 3.1.

A partir dos arquivos de pré-tratamento serão obtidas 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 observados. Será o data frame.
  - **MJD** : O momento em que a observação foi obtida. A medida está descrita em dias do calendário Juliano modificado (*Modified Julian Date*).
  - **FLT** : O filtro no qual a observação foi obtida. Pois os objetos são observados em 4 bandas de comprimento de onda de luz. Na figura 3.2 pode-se ver um exemplo de uma observação na íntegra e como a mesma é decomposta 4 bandas.
  - **FLUXCAL** : O valor do fluxo de luz obtido naquela observação.
  - **FLUXCALERR** : O valor do erro do fluxo de luz.

Em relação ao **MJD** vale explicar 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 zero o menor valor do MJD para que o eixo das abscissas tenha sempre início em zero.

```
FAKE:    2   (=> simulated LC with snlc_sim.exe)
MWEBV:   0.0111    MW E(B-V)
REDSHIFT_HELIO:   0.35020 +- 0.04680  (Helio, 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.

Pela perspectiva 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 3.3 ilustra as 4 curvas de luz para o mesmo objeto.

3.2 *pipeline* Atual

Dispondo dos dados obtidos pelo pré-processamento explicado na seção anterior, inicialmente o *pipeline* 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**), sendo n o número de amostras.

Em seguida, esses 4 *numpy arrays* serão a entrada do Processo Gaussiano, cuja função é interpolar uma curva que passe pelos pontos de cada filtro 3.3.

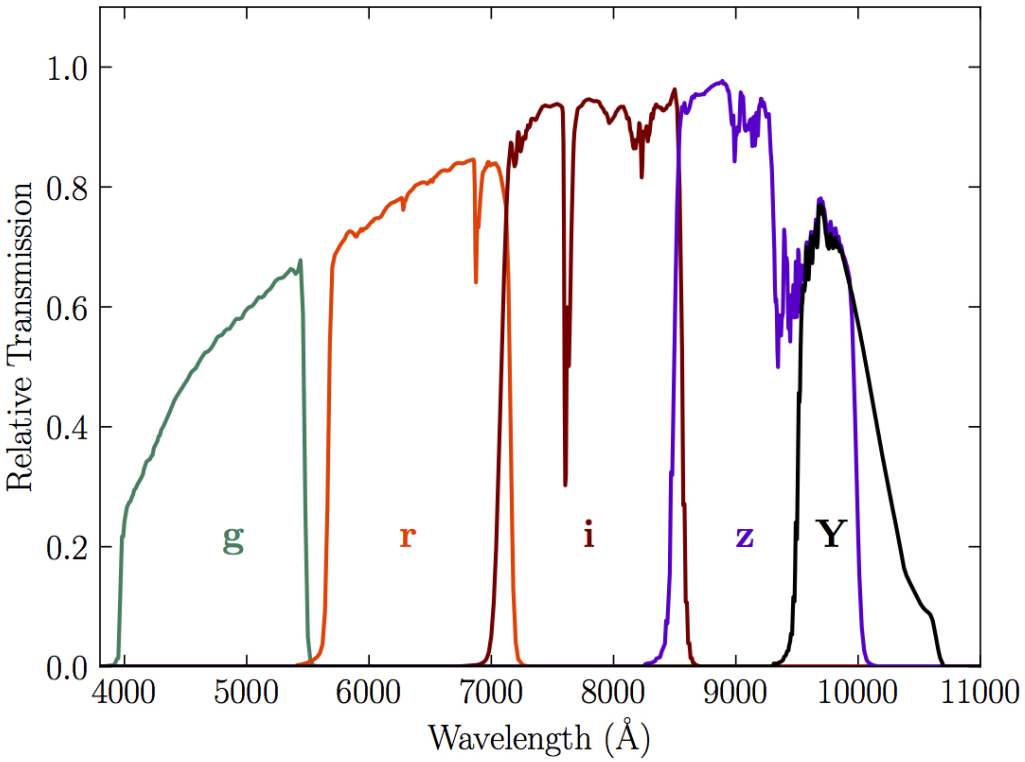


Figura 3.2: Observação íntegra de um objeto astronômico. Não é referente a nenhum exemplo específico do *dataframe*

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 esse procedimento se repetirá quatro vezes para cada objeto, uma para cada filtro. Assim, como saída do processo de wavelets serão 1600 coeficientes para cada objeto astronômico.

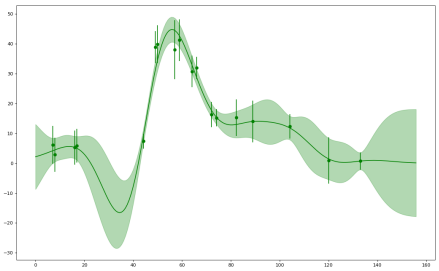
Em seguida é realizada uma redução de variáveis através da **análise de componentes principais** resultando em cada objeto com 20 *features*.

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

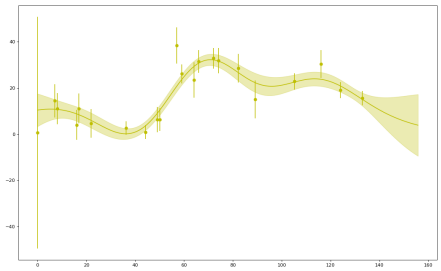
Em paralelo, usa-se o **SN TYPE** do dicionário para armazenar os tipos (labels) de cada objeto astronômico em forma de lista.

Finalmente serão estabelecidos os parâmetros que caracterizam o modelo de classificação baseado em *Random Forest*. Neste ponto é usado o módulo *sklearn.pipeline* da biblioteca *scikit-learn*.

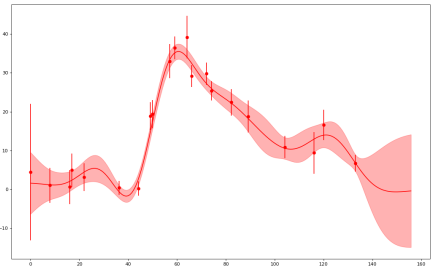
Tendo em mãos o *dataframe*, a lista com as classificações de cada objeto (os



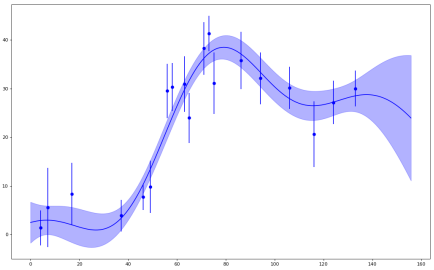
(a) Filtro *desg*



(b) Filtro *desi*



(c) Filtro *desr*

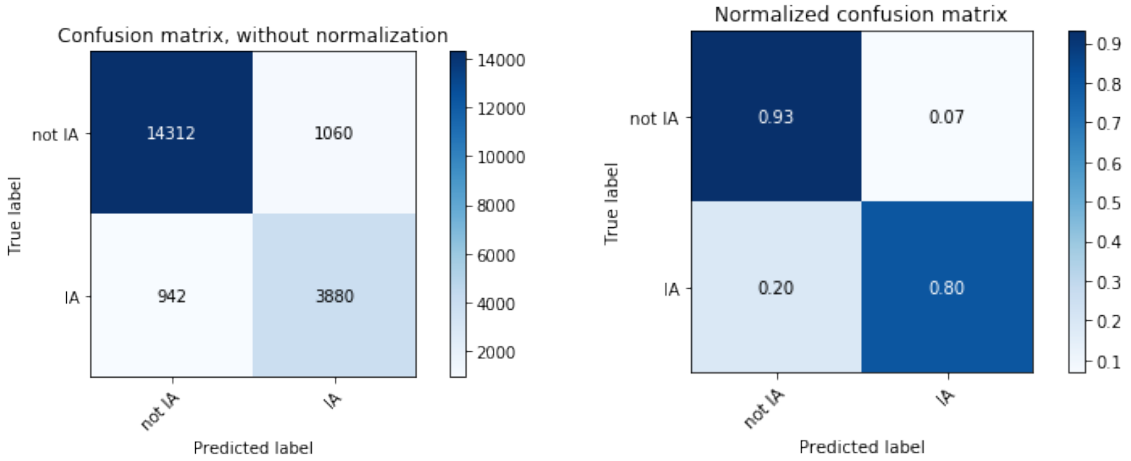


(d) Filtro *desz*

Figura 3.3: Exemplo de 4 curvas de luz de um mesmo objeto, uma para cada filtro. Gráfico Fluxo × Tempo (dias).

labels) e o modelo de classificação, é feita a divisão através do método *k-fold* de validação cruzada, dividindo os objetos em 19 partições e usando 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*, o **método AUC** (a área abaixo da curva ROC) e o **método de precisão média** 2.3. As matrizes de confusão de cada um dos 19 modelos também podem ser analisadas para ilustrar seus acertos, falsos positivos e falsos negativos 3.4.



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

(b) Matriz de Confusão, valores normalizados.

Figura 3.4: Matrizes de Confusão do modelo final do *pipeline* original.

Capítulo 4

Tratamento de *Outliers*

A primeira forma de buscar melhorias neste projeto foi o tratamento de *Outliers*. Todos os tipos de dados brutos estão sujeitos a amostras com valores absurdos ou incertezas irreais, sejam elas causadas por erros de medição ou quaisquer outros fatores [25].

4.1 Estratégias Utilizadas

O primeiro tratamento realizado foi a retirada de pontos negativos. Visto que os valores a serem interpolados são fluxos de luz em função do tempo, não há sentido físico em haver pontos negativos. Entretanto, o motivo para que eles existam nos registros são as imprecisões nos sensores, onde devido à agitação térmica, é possível que um elétron passe espontaneamente de um receptor para o outro, resultando em uma contagem negativa de elétrons no receptor original, acarretando o valor negativo.

A segunda estratégia foi analisar a incerteza das amostras de cada ponto referente aos 4 filtros dos objetos astronômicos. Em seguida, foi calculada a **média** e o **desvio padrão** desses conjuntos de incertezas e por fim, houve a exclusão daqueles pontos cujo erro era superior à média mais um desvio padrão da distribuição de incertezas.

Outra tentativa foi a aplicação de limiares baseados nos valores das incertezas de cada ponto em relação ao pico da curva. Caso o valor da incerteza fosse maior que o pico da curva multiplicado por um limiar (valor entre 0 e 1), este ponto seria excluído.

O código referente a implementação deste tratamento de *outlier*, assim como todo o código do projeto, encontra-se disponível no *GitHub* A.

4.2 Resultados e comparações

Serão apresentados 3 tipos diferentes de *score*s para cada um dos 19 modelos gerados pela validação cruzada K-Fold: curva AUC (**AUC**), *Average Precision* (**AP**) e o padrão da biblioteca *scikit*; tendo sido aplicadas as seguintes estratégias:

- Retirada de valores negativos e daqueles com incerteza maior que a média mais desvio padrão. Representada na tabela 4.1 como **StdDev**.
- Retirada de valores negativos e daqueles com incerteza maior do que 70% do valor do pico. Representada na tabela 4.1 como **Threshold**.

Fold	**AUC**	**AUC StdDev**	**AUC Threshold**	**AP**	**AP StdDev**	**AP Threshold**	**Padrão**	**Padrão StdDev**	**Padrão Threshold**
Fold 1	95,93%	95,18%	95,99%	86,34%	83,86%	86,41%	90,33%	89,65%	90,35%
Fold 2	95,76%	94,88%	95,75%	84,51%	81,72%	84,36%	90,11%	89,00%	90,09%
Fold 3	96,02%	94,89%	96,01%	87,25%	82,93%	87,27%	90,70%	89,14%	90,51%
Fold 4	95,65%	94,77%	95,71%	84,73%	81,99%	84,96%	89,68%	88,47%	89,46%
Fold 5	95,72%	95,01%	95,86%	85,34%	83,42%	86,34%	90,19%	89,51%	90,11%
Fold 6	95,79%	94,81%	95,68%	84,27%	82,30%	84,04%	90,42%	88,97%	90,26%
Fold 7	95,56%	94,82%	95,93%	85,20%	83,75%	86,36%	89,73%	88,96%	90,36%
Fold 8	95,82%	95,19%	95,90%	85,49%	84,12%	86,10%	90,00%	89,41%	90,25%
Fold 9	95,74%	95,24%	95,93%	85,71%	83,66%	86,87%	90,27%	89,45%	90,40%
Fold 10	95,91%	94,96%	95,94%	87,38%	85,24%	87,52%	90,65%	89,76%	90,69%
Fold 11	95,89%	95,42%	95,95%	85,90%	84,63%	86,77%	90,19%	89,61%	90,33%
Fold 12	95,72%	94,81%	95,93%	85,02%	81,56%	86,06%	90,19%	89,06%	90,46%
Fold 13	95,69%	95,29%	95,85%	85,19%	83,69%	85,86%	89,82%	89,55%	89,99%
Fold 14	95,48%	94,81%	95,51%	84,65%	82,17%	85,06%	89,83%	88,84%	89,94%
Fold 15	95,65%	95,29%	95,66%	84,84%	84,56%	85,68%	89,91%	90,18%	90,26%
Fold 16	95,83%	95,55%	95,95%	86,48%	85,30%	86,56%	90,12%	89,79%	90,00%
Fold 17	96,01%	95,41%	95,97%	86,19%	84,58%	85,95%	90,28%	89,66%	90,19%
Fold 18	95,96%	95,47%	95,86%	85,84%	83,88%	85,55%	90,63%	90,02%	90,60%
Fold 19	95,35%	95,37%	95,47%	84,34%	84,16%	84,47%	89,87%	89,61%	89,70%
Média	**95,76%**	**95,11%**	**95,83%**	**85,51%**	**83,55%**	**85,90%**	**90,15%**	**89,40%**	**90,21%**

Tabela 4.1: Resultados do tratamento de *outliers*. 'Padrão' se refere ao *score default* do *scikit*, e as colunas que não apresentam nem *StdDev* nem *Threshold* representam os *score*s do *pipeline* original.

A conclusão final para o tratamento de *outliers* é que o mesmo não influencia significativamente no aumento dos *score*s, a possível explicação para isto encontra-se na robustez que as transformadas de wavelets oferecem ao método.

Outra explicação é que a fração de *outliers* é muito baixa, no método ***StdDev***, apenas 27 *outliers* foram retirados, enquanto pelo ***Threshold*** o número foi de 85. Resultando numa limpeza percentual de 0,13% e 0,4% de todos os dados respectivamente.

Capítulo 5

Rede Neural Convolucional

5.1 Geração de Imagens e Parâmetros

A primeira etapa desta abordagem consiste em gerar gráficos a partir da interpolação gaussiana. Para poder comparar o resultado deste método com o do *pipeline* inicial, a interpolação realizada para gerar os gráficos fornecidos ao modelo de Deep Learning foi a mesma interpolação usada no *pipeline* original.

Estes gráficos serão imagens em escalas de cinza na extensão *png*, cujas dimensões são: 64×40 pixels. Ao gerar essas imagens eram criadas margens com 5 pixels na dimensão y e 9 pixels na dimensão x. No fim, elas 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 *numpy array*, 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 (MaxPooling2	(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

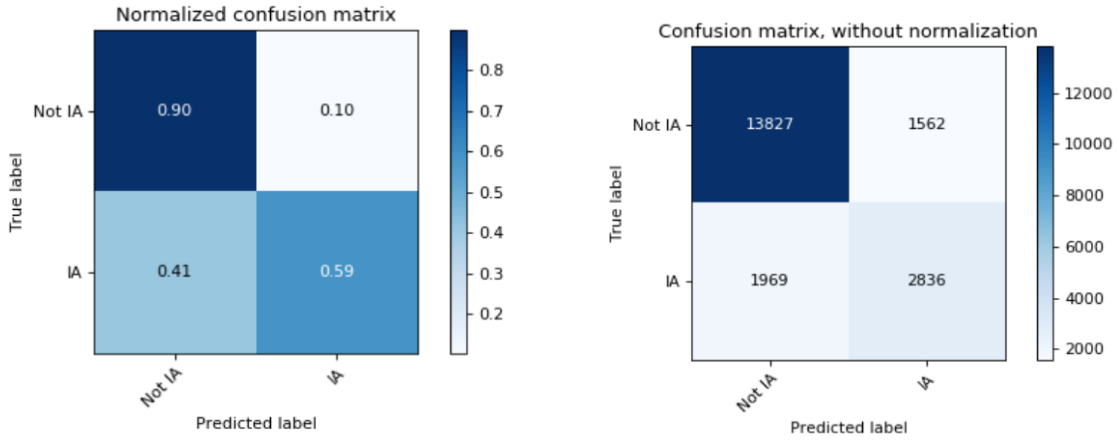
Figura 5.1: Arquitetura do modelo de Deep Learning utilizado.

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 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 para o valor de 10 épocas, pois sua acurácia foi alta sem deixar o algoritmo dependente.

5.3 Resultados e comparações

Os resultados em forma de matriz de confusão do melhor modelo estão descritos a seguir. 5.2



(a) Valores normalizados.

(b) Valores absolutos.

Figura 5.2: Matrizes de confusão do melhor modelo de Deep Learning.

Ao compararmos os resultados da figura 5.2 com o da figura 3.4 percebemos que o modelo de Deep Learning apresentou piora em seu desempenho.

Dois motivos podem explicar tais resultados, o primeiro deles é o fato das imagens treinadas não terem sofrido 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*.

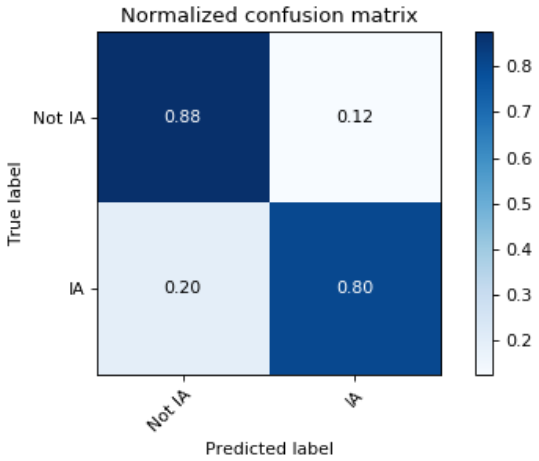


Figura 5.3: Valores normalizados do modelo DL treinando com 80% dos dados.

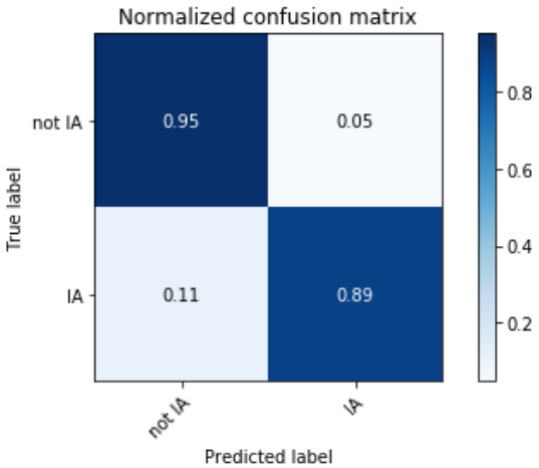


Figura 5.4: Modelo do *pipeline* original treinado com 80% dos dados.

Capítulo 6

Interpolação através de Processo Gaussiano

A abordagem feita pelo IF-UFRJ ao usar o Processo Gaussiano foi através da biblioteca *george* [26]. Uma biblioteca focada em avaliar a probabilidade marginal [27] [15] da distribuição dos dados.

Contudo, ao tentar editar alguns parâmetros internos referentes a interpolação nesta biblioteca, como os *Kernels* ou funções a priori 2.1, houve certa dificuldade ou mesmo impossibilidade de modelá-los da maneira desejada. Como, por exemplo, somar e multiplicar *Kernels*, sobretudo aqueles não-estacionários (Cap. 3 [15]).

Um motivo que levou o projeto a usar uma outra biblioteca foi a citação do capítulo 4 seção 2 do RASMUSSEN e WILLIAMS [15], argumentando que o *Kernel* de tipo *Squared Exponential Covariance Funtion* (utilizado no *pipeline* original) propicia à função interpolada uma suavização irreal para diversos fenômenos físicos, e recomenda usar *Kernels* do tipo *Matern*. Apesar de a biblioteca *george* também conter esse tipo de *Kernel*, outras bibliotecas possuíam uma maior possibilidade de edição.

Na seção abaixo serão vistos os tipos de bibliotecas analisadas.

6.1 Escolha da Biblioteca

A biblioteca utilizada no artigo da LOCHNER [5] foi a *Gapp*, uma biblioteca para reconstrução de funções usada em outros trabalhos de cosmologia [28].

Entretanto, o próprio *pipeline* desenvolvido pelo IF-UFRJ já buscava otimizar o artigo original da M. Lochner, logo, este projeto partiu dos estudos já feitos, tendo como ponto de partida a biblioteca *george*.

Vale mencionar também que embora seja possível implementar manualmente os processos gaussianos, as várias bibliotecas disponíveis já possuem especificações e ajustes para os modelos de maneira mais automatizada. As três bibliotecas analisadas foram:

- SciKit-Learn
- GPflow
- PyMC3

Em particular, cada um desses pacotes inclui um conjunto de funções de covariância que podem ser flexivelmente combinadas para descrever adequadamente os padrões específicos dos dados, juntamente aos métodos para ajustar os parâmetros do GP.

A maior parte da análise foi baseada no blog *Domino*, onde vários experimentos foram realizados comparando o desempenho de cada pacote [29].

As minuciosidades destas análises podem ser vistas em todo o projeto realizado no blog citado acima, para não fugir do escopo deste trabalho não serão mencionados mais detalhes sobre a especificidade de cada biblioteca.

Como explicação breve para a escolha final, pode se dizer globalmente que o *scikit-learn* possui uma abordagem mais simples e focada menos nos modelos probabilísticos sofisticados. Enquanto tanto o *GPFlow* quanto o PyMC3 possuem um próprio *backend* computacional dando suporte a esses modelos. Por fim, o motivo da escolha do PyMC3 foi devido a maior comunidade de suporte e ao vasto domínio de estudo da biblioteca. Pois, por ser um pacote de programação probabilística, o PyMC3 abrange mais ferramentas que podem vir a ser úteis no desenvolvimento das distribuições de probabilidade usadas no GP.

6.2 Aleatoriedades e *Random Seeds*

Antes de adentrar nas escolhas dos *Kernels*, é muito importante explicar o fator da aleatoriedade dentro do projeto. Normalmente a biblioteca PyMC3 realizaria métodos de otimização da função a posteriori em seu GP, entretanto, por ser uma biblioteca que exige um alto custo computacional, foi decidido não executar a linha de código que otimizaria o código em prol de escolher uma função *Kernel* que melhor faria a interpolação.

Isto não significa que os parâmetros do GP não serão otimizados, eles apenas não irão convergir para um valor final, o que os torna dependentes das condições iniciais estabelecidas internamente pelo computador durante as cadeias de Markov Monte Carlo [19].

A alternativa para esta escolha de projeto foi comparar determinadas sementes aleatórias [22]. Através da análise de 11 sementes pôde-se escolher aquelas cujas condições iniciais resultavam em um maior *overfitting* das interpolações. As figuras das comparações das interpolações de cada semente encontram-se no apêndice B. Neste contexto, *overfitting* se refere àquela interpolação que "força"o gráfico resultante a passar pelos pontos, podendo assim ter taxas de variações mais bruscas e suavidades menores. No contexto do presente projeto, é algo desejado, visto que variações bruscas do fluxo de luz correspondem a um comportamento esperado de uma explosão.

Buscando sempre aquelas sementes que eliminam interpolações constantes e proporcionam um *overfitting*, as melhores foram:

- Random Seed 8.
- Random Seed 6.
- Random Seed 5.
- Random Seed 4.
- Random Seed 9.
- Random Seed 7.

Por fim, foram escolhidas duas delas para as interpolações finais, as sementes 4 e 9. A justificativa encontra-se no fato de elas não apresentarem nenhum caso de interpolação mantendo valores constantes, enquanto as demais, mesmo que menos do que o *pipeline* original, ainda a possuíam.

6.3 *Kernel Functions*

Para escolher as funções *Kernel* que seriam utilizadas nas interpolações, foi seguido o suporte da biblioteca PyMC3 [14]. Nesse suporte existem diferentes maneiras de interpolações e análises sobre como os *Kernels* afetam diretamente no formato das funções.

A partir dos dados da referência acima, foram escolhidos os seguintes *Kernels* para as interpolações:

- Exponencial quadrática:

$$k(x, x') = \exp\left[-\frac{(x - x')^2}{2\ell^2}\right] \tag{6.1}$$

- Racional quadrática:

$$k(x, x') = \left(1 + \frac{(x - x')^2}{2\alpha\ell^2}\right)^{-\alpha} \tag{6.2}$$

- *Matern* 5/2:

$$k(x, x') = \left(1 + \frac{\sqrt{5(x - x')^2}}{\ell} + \frac{5(x - x')^2}{3\ell^2}\right) \exp\left[-\frac{\sqrt{5(x - x')^2}}{\ell}\right] \tag{6.3}$$

- *Matern* 3/2:

$$k(x, x') = \left(1 + \frac{\sqrt{3(x - x')^2}}{\ell}\right) \exp\left[-\frac{\sqrt{3(x - x')^2}}{\ell}\right] \tag{6.4}$$

Nestas funções temos x e x' como os valores das abscissas e α e l como hiperparâmetros a serem ajustados. Empiricamente percebeu-se que o l possui um efeito de aumentar *overfitting* em troca do aumento do erro da interpolação. O parâmetro l é possui um efeito de "distância de correlação", pois pontos separados por muito mais do que essa distância, tem pouca influência uns sobre os outros. Enquanto α determina como a correlação diminui com a distância $x - x'$.

Esses hiperparâmetros podem assumir valores como funções de distribuições probabilísticas ou funções constantes. Após diversos testes utilizando constantes, foi notório que os melhores resultados foram obtidos usando distribuições. Outro fator que levou a essa escolha, foram os exemplos oferecidos no suporte da biblioteca PyMC3 [30] [31].

No apêndice B são ilustradas as interpolações realizadas para escolher qual *Kernel* seria utilizado. Os critérios principais para a escolha foram evitar interpolações constantes e buscar *overfitting*. Como resultado, os dois melhores *Kernels* foram o *Matern* 3/2 e o *Matern* 5/2; uma conclusão alinhada com a teoria exposta no capítulo 4 seção 2 do RASMUSSEN e WILLIAMS [15].

6.4 Demais Observações

Antes de adentrar nos resultados das Interpolações, duas observações importantes são feitas.

A primeira é a solução para o problema de possíveis valores negativos. Por mais que tal abordagem não tenha sido adotada neste trabalho devido à tentativa de deixar o código com menos custo computacional possível; para trabalhos futuros nos quais deseje-se impossibilitar interpolações com valores negativos, o uso de uma

mean function não-negativa resolve este problema [32].

Vários tipos de funções que atendem esse pré-requisito estão descritos no PYMC3 [32].

Um exemplo pode ser a *Half-Cauchy log-likelihood*.

$$f(x \mid \beta) = \frac{2}{\pi\beta[1 + (\frac{x}{\beta})^2]} \tag{6.5}$$

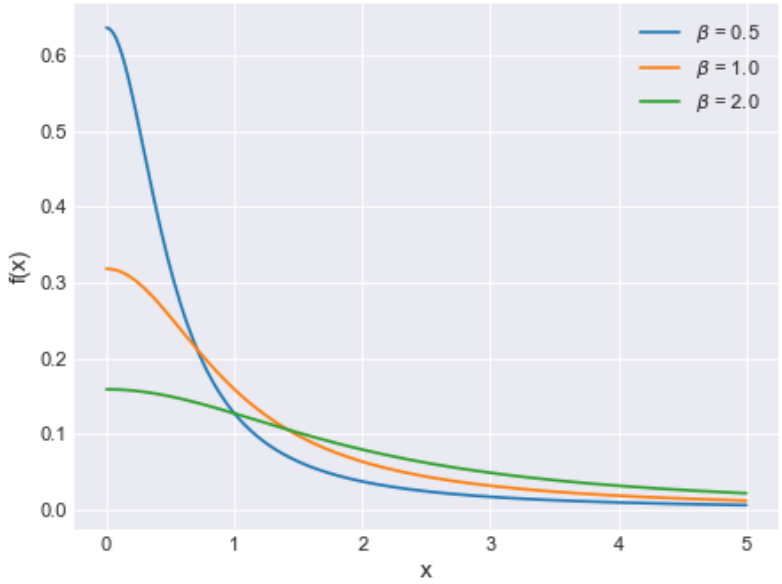


Figura 6.1: Exemplo de distribuição com valores não-negativos.

A segunda observação é o uso da função *Matern* dentro da própria biblioteca usada no *pipeline* original do *george*.

Os resultados da troca do *Kernel ExpQuadr* pelo *Matern* 5/2 na biblioteca *george* podem ser vistos em 6.1 e 6.3

A justificativa para tal desempenho ser pior pode ser atribuída aos hiperparâmetros terem sido constantes e não distribuições. Assim, caso fosse analisado apenas o resultado do *george*, concluiria-se que o uso do *Matern* é inferior ao do *Exp Quadr*. Todavia, ao realizar comparações dentro de uma biblioteca com mais suporte, a melhora é visivelmente notada ao interpolar pelo *Kernel* proposto do que pelo antigo 6.2.

6.5 Resultados das Interpolações

O resultado global das interpolações não pôde ser concluído devido a um erro de processamento de código, mais precisamente um *memory leak* interno durante a

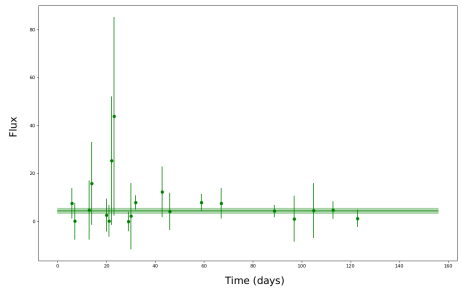
Fold	**Padrão**	**AUC**	**AP**
Fold 1	88,88%	83,15%	94,79%
Fold 2	88,50%	81,28%	94,42%
Fold 3	88,74%	81,86%	94,32%
Fold 4	88,26%	80,44%	93,92%
Fold 5	88,67%	83,51%	94,53%
Fold 6	88,32%	79,85%	93,92%
Fold 7	88,72%	82,05%	94,09%
Fold 8	88,22%	81,51%	94,10%
Fold 9	88,18%	82,14%	93,74%
Fold 10	88,43%	82,67%	94,04%
Fold 11	89,33%	83,75%	94,82%
Fold 12	89,14%	83,03%	93,97%
Fold 13	88,35%	82,31%	94,36%
Fold 14	88,56%	83,42%	94,38%
Fold 15	88,01%	81,28%	93,83%
Fold 16	88,29%	82,24%	94,12%
Fold 17	88,90%	83,30%	94,72%
Fold 18	88,63%	82,23%	94,21%
Fold 19	88,38%	81,63%	94,26%
Média	**94,24%**	**82,19%**	**88,55%**

Tabela 6.1: Resultados do *Kernel Mattern 5/2* pela biblioteca *george*. 'Padrão' se refere ao *score default* do *scikit*.

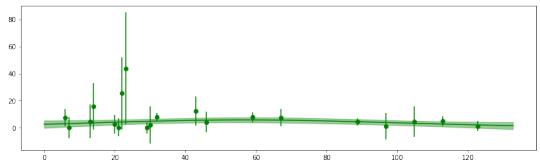
execução dos modelos do PyMC3 em looping. A explicação e detalhamento desse erro será apresentado na seção a seguir.

Nesta seção serão apresentados alguns exemplos de interpolações entre os algoritmos, onde pôde se notar uma melhor interpolação usando o *Kernel Matern* no PyMC3 através das sementes. Justificando o porquê de se esperar que caso o processamento de todos os dados fosse concluído, provavelmente obter-se-iam resultados melhores 6.2.

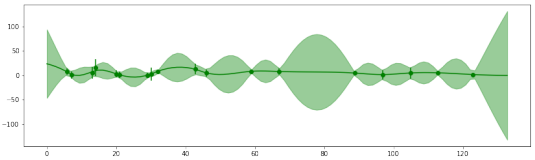
Demais exemplos de comparações estão descritos no *notebook* do apêndice B.



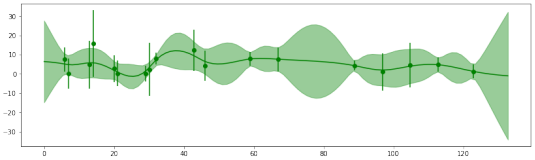
(a) *pipeline* original



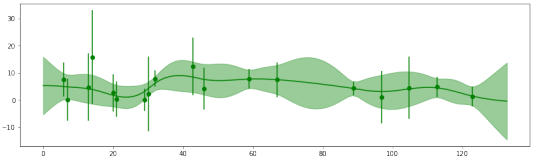
(b) PyMC3 Exponential Quadratic semente 9



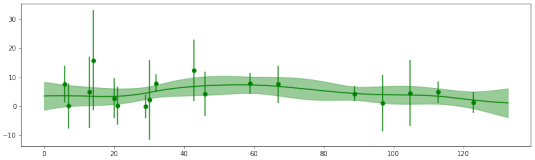
(c) PyMC3 *Matern* 5/2 semente 6, *overfitting*



(d) PyMC3 *Matern* 5/2 semente 9, *overfitting*

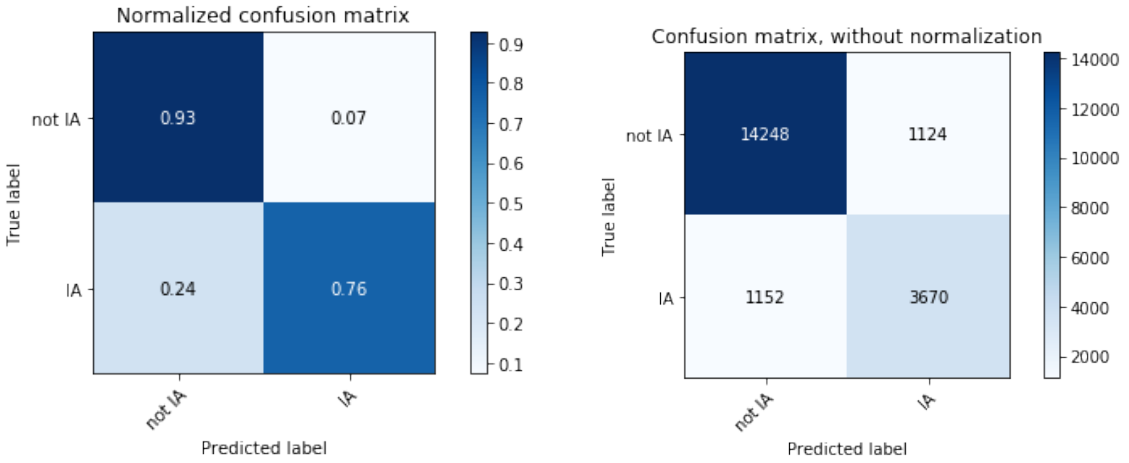


(e) PyMC3 *Matern* 5/2 semente 6, menos *overfitting*



(f) PyMC3 *Matern* 5/2 semente 9, menos *overfitting*

Figura 6.2: Interpolações do objeto SN013742 usando 6 configurações diferentes. Gráfico Fluxo × Tempo (dias).



(a) Valores normalizados.

(b) Valores absolutos.

Figura 6.3: Matrizes de confusão do *Matern 5/2* pela biblioteca *george*. Resultados piores do que os apresentados em 3.4

6.6 Identificação e Justificativa do Erro

O erro ocorreu durante o looping de execução da interpolação modelada causando a parada do computador. Ao inspecionar o consumo de memória através do comando *htop* do Ubuntu, foi identificado um *memory leak* do sistema.

Durante uma árdua investigação do problema, houveram várias tentativas buscando identificá-lo e posteriormente corrigi-lo, dentre elas as principais foram:

- Buscar possíveis erros de construção no código.
- Forçar manualmente a liberação de memória utilizando pacotes de *Garbage Collector*.
- Inspeção através de métodos *magic* do Jupyter para localizar a linha de código com o vazamento de memória (%%time, %%lprun, %%mprun).
- Tentativa de uso de GPU para o processamento.
- Uso de pacotes envolvendo tabelas *hash* para reduzir a memória consumida pela estrutura de dados em forma de dicionários.
- Uso de bibliotecas próprias para gerenciar a alocação e consumo de memória.

Após essa inspeção, o problema foi descoberto como um erro interno da biblioteca ao longo da criação modelos probabilísticos dentro de um looping. Em resumo, quando se instancia um modelo do PyMC3 dentro de um looping, a biblioteca continua alocando memória em lugares diferentes sem 'esvaziar' os antigos, ocasionando o vazamento.

Uma discussão prolongada sobre esse erro foge do assunto abordado no presente trabalho, e devido ao grande tempo investido no estudo de identificação e solução do problema, tomou-se a decisão de abortar essa abordagem. Todavia, como o problema já foi identificado [33], estão disponíveis na bibliografia os links das discussões em forums onde ele vem sendo discutido.

Capítulo 7

Conclusões

7.1 Conclusões Finais

O presente trabalho teve como objetivo buscar otimizações e melhorias nas técnicas de ML dentro dos modelos de classificação de supernovas usando espectroscopia. A partir da análise dos resultados, foi possível verificar que o modelo atual possui uma robustez muito eficaz tendo em vista a pouca quantidade de dados treinados.

Como ponto positivo para o método, este projeto pôde agregar conhecimento testando diferentes abordagens e concluindo que suas aplicações não ofereceriam melhoras ao modelo.

Dentro do escopo de um projeto final, cabe também mencionar a evolução e aquisição de conhecimento do aluno, que ao realizar pesquisas e discutir novas ideias para o algoritmo, teve um aprendizado significativo a medida em que foram sendo descobertas e estudadas novas tecnologias e conceitos.

Ao aplicar a estratégia de tratamento de dados removendo os *outliers*, foi possível observar que a diferença dos resultados finais mostrou-se tão pequena que pode ser desprezada, levando a conclusão de que a etapa das transformadas de wavelets proporciona ao algoritmo uma robustez significativa, a ponto de que *outliers* 'grosseiros' não comprometeram o modelo.

Em relação ao Deep Learning concluiu-se que no local onde as redes neurais convolucionais foram aplicadas, essa técnica não se mostrou eficaz devido aos modelos serem treinados com apenas um dezenove avos dos dados. Além de concluir que a ausência de um possível *data augmentation* compromete o desempenho mesmo quando são utilizados 80% dos dados para treino.

Por fim, a abordagem principal que foi a busca de uma melhor execução do Processo Gaussiano não pôde ser concluída devido a problemas que fugiram da natureza do projeto. Entretanto, a partir da análise de outros métodos e dos resultados das interpolações dos gráficos através do *Kernel Mattern 5/2* pelo *george*, conclui-se que o método atual já se encontra bem otimizado, sustentado principalmente pela robustez que a transformada de wavelets o oferece.

7.2 Trabalhos Futuros

Em relação à utilização de aprendizado de máquina na temática em questão, muitos outros métodos são usados para a identificação de objetos astronômicos para demais fins [34]. Tendo em vista esses outros objetivos, o trabalho aqui desenvolvido mostra-se útil para projetar novas ideias nessas outras abordagens.

Em relação ao erro de *memory leak*, a resolução fica em aberto para trabalhos futuros que busquem finalizar a ideia aqui já construída. Visto que a parte árdua de identificar o problema já foi feita, o mesmo pode ser publicado em repositórios [35] [33] [36] e comunidades de programadores buscando suporte direcionado.

Sobre as outras bibliotecas estudadas para realizar a interpolação, após a análise e detalhamento de cada uma delas e suas propriedades [29], elas se mostram interessantes em trabalhos cujo uso de Processo Gaussiano seja necessário.

Após a utilização da biblioteca PyMC3, apesar do problema ocorrido, ela se mostrou extremamente eficaz e matematicamente completa, estando alinhada com os conceitos estatísticos proporcionando uma flexibilidade na modelagem das distribuições. Logo, é aconselhável que novos interessados no projeto usem a biblioteca, devido ao seu grande suporte que facilita o entendimento teórico. Assim como àqueles que já possuem determinado conhecimento e queiram tentar modelos mais específicos e customizados.

Referências Bibliográficas

[1] WIKIPEDIA. "Luminosity distance", *https://en.wikipedia.org/wiki/Luminosity_-distance*, 2019.

[2] WIKIPEDIA. "Luminous Flux", *https://en.wikipedia.org/wiki/Luminous_flux*, jun. 2019.

[3] WIKIPEDIA. "Vela padrão", *https://pt.wikipedia.org/wiki/Vela_-padr%C3%A3o*, mar. 2013.

[4] WIKIPEDIA. "Espectroscopia astronômica", *https://pt.wikipedia.org/wiki/Espectroscopia_astron%C3%B4mica*, mar. 2018.

[5] LOCHNER, M. "Photometric Supernova Classification with Machine Learning", *ApJS (2016)*, v. 225, n. 2, pp. 31, set. 2016.

[6] WIKIPEDIA. "Fotometria astronômica", *https://pt.wikipedia.org/wiki/Fotometria_-(astronomia)*, jun. 2018.

[7] WIKIPEDIA. "Segunda lei da Termodinâmica", *https://pt.wikipedia.org/wiki/Segunda_lei_da_termodin%C3%A2mica*, jul. 2019.

[8] KESSLER, R. "Supernova Photometric Classification Challenge", *arXiv*, v. 6, n. 5210, pp. 4, abr. 2010.

[9] LABORATORY, A. N. "Argonne National Laboratory", *https://www.anl.gov/hep*.

[10] PANDAS. "Pandas - Python Data Analysis Library", *https://pandas.pydata.org/*, jul. 2019.

[11] DEVELOPERS, N. "Numpy", *https://numpy.org/*, 2019.

[12] INRIA, OTHERS. "Scikit-Learn", *https://keras.io/*, 2019.

[13] COMMUNITY, K. "Keras", *https://scikit-learn.org/stable/*, 2019.

[14] PYMC3. "PyMC3 - Mean and Covariance Functions", *https://docs.pymc.io/notebooks/GP-MeansAndCovs.html*, 2018.

[15] RASMUSSEN, C., WILLIAMS, C. K. I. *Gaussian Processes for Machine Learning*. 1 ed. New York, Thomas Dietterich, Editor, 2004.

[16] 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.

[17] BAILEY, K. "Gaussian Processes for Dummies", *http://katbailey.github.io/post/gaussian-processes-for-dummies/*, v. 1, n. 1, pp. 1, jun. 2016.

[18] 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.

[19] DANI GAMERMAN, H. F. L. *Markov Chain Monte Carlo Stochastic Simulation for Bayesian Inference*. 2 ed. New York, Chapman and Hall/CRC, 2006.

[20] WIKIPEDIA. "Rede neural convolucional", *Rede neural convolucional*, jun. 2018.

[21] WIKIPEDIA. "Perceptron Multicamadas", *https://pt.wikipedia.org/wiki/Perceptron_multicamadas*, jun. 2019.

[22] WIKIPEDIA. "Pseudoaleatoriedade", *https://pt.wikipedia.org/wiki/Pseudoaleatoriedade*, 2019.

[23] MURPHY, K. P. *Machine Learning A Probabilistic Perspective*. 1 ed. Massachussetts Institute of Technology, MIT, 2012.

[24] CELSO, A. "Notas de Aula de Sinais e Sistemas", *UFRJ - COE350 - Sinais e Sistemas - ECA*, jun. 2014.

[25] VUOLO, J. H. *Fundamentos da Teoria de Erros*. Editora Edgard Blücher LTDA, 1996.

[26] BIB, G. "Documentação biblioteca *George*", *https://george.readthedocs.io/en/latest/*, 2012.

[27] WIKIPEDIA. "Wikipedia, probabilidade marginal", *https://en.wikipedia.org/wiki/Marginal_likelihood*, jun. 2019.

[28] MARINA SEIKEL, CHRIS CLARKSON, M. S. "Reconstruction of dark energy and expansion dynamics using Gaussian processes", *JCAP06*, v. 2, n. 2, pp. 20, 2012.

[29] LAB, D. D. "Fitting Gaussian Process Models in Python", *https://blog.dominodatalab.com/fitting-gaussian-process-models-python/*, mar. 2017.

[30] PYMC3. "PyMC3 - Example", *https://docs.pymc.io/notebooks/GP-MaunaLoa.html*, 2018.

[31] PYMC3. "PyMC3 - Getting Started", *https://docs.pymc.io/notebooks/getting_started.html*, 2018.

[32] PYMC3. "PyMC3 - Continuos functions", *https://docs.pymc.io/api/distributions/continuous.html*, 2018.

[33] PYMC DEVS. "memory leak issues when running pymc3 model in a loop", *https://github.com/pymc-devs/pymc3/issues/1825*, mar. 2017.

[34] K. SOOKNUNAN, M. L. "Classification of Multiwavelength Transients with Machine Learning", v. 1, pp. 16, nov. 2018.

[35] PYMC DEVS. "PyMC3 extra variables, and issues with theano." *https://github.com/pymc-devs/pymc3/issues/772*, jun. 2015.

[36] PYMC DEVS. "memory leak issue when running the pymc3 in AWS machine", *https://github.com/pymc-devs/pymc3/issues/1959*, 2018.

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.

Apêndice B

Avaliação das Interpolações

Dentro do repositório citado no apêndice A, um *notebook* em especial possui as diversas interpolações feitas para avaliar de maneira empírica quais métodos e quais *random seeds* se mostraram mais eficazes. O arquivo é o https://github.com/FelipeMFO/supernova_identification/blob/master/src_and_notebooks/processing/GP_evaluation.ipynb.