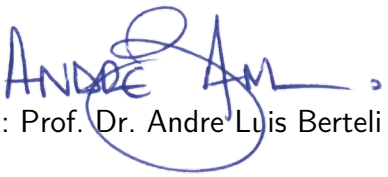
Beneficiário: Mateus Piovezan Otto



Responsável: Prof. Dr. Andre Luis Berteli Ambrosio

Análise multiparamétrica do problema das fases em cristalografia de proteínas por aprendizagem profunda.

Relatório Científico Final relativo ao período de 10/07/2019 à 31/01/2020 do projeto 2018/23675-7 na modalidade Iniciação Científica fomentado pela Fundação de Amparo à Pesquisa do Estado de São Paulo, FAPESP

Grupo de Cristalografia

Departamento de Física e Ciência Interdisciplinar

Instituto de Física de São Carlos

Universidade de São Paulo

São Carlos, SP

Informações Gerais

- **Título do projeto:**

  Análise multiparamétrica do problema das fases em cristalografia de proteínas por aprendizagem profunda.

- **Nome do Pesquisador Responsável:**

  Mateus Piovezan Otto

- **Instituição Sede do projeto:**

  Grupo de Cristalografia
  Departamento de Física e Ciência Interdisciplinar
  Instituto de Física de São Carlos
  Universidade de São Paulo

- **Equipe de pesquisa:**

  Orientador: Dr. Andre Luis Bertelli Ambrosio

- **Número do Processo FAPESP:**

  FAPESP 2018/23675-7

- **Período de Vigência do projeto:**

  01/02/2019 a 31/01/2020

- **Período coberto pelo Relatório Científico em questão:**

  10/07/2019 à 31/01/2019

- **Assinatura do Pesquisador Responsável:**

Mateus Piovezan Otto

Resumo

O problema das fases é notório na cristalografia de proteínas por difração de raios X. Fundamentalmente, limitações tecnológicas inerentes aos sistemas de detecção dessa radiação resultam na perda de informações sobre as fases das ondas espalhadas construtivamente pelos componentes do cristal. Como consequência, o cálculo direto da função de distribuição de densidade eletrônica na cela unitária é impossibilitado. Atualmente, dois métodos experimentais podem ser aplicados para se contornar esse problema: (i) substituição parcial do solvente aquoso ordenado por íons elétron-densos (metálicos ou halogênicos) ou (ii) quantificação seletiva do componente dispersivo (lambda-dependente) do fator de espalhamento atômico. Alternativamente, informações prévias, na forma de estruturas cristalinas conhecidas que são funcionalmente relacionadas ou homólogas a componentes no cristal, podem servir como fonte de um conjunto inicial de fases. Apesar de desafiadoras, quando viáveis, as aplicações desses diferentes métodos já possibilitaram a determinação de mais de uma centena de milhares de modelos atômicos, para as mais diversas proteínas (e seus complexos). Neste projeto, com base nesta coleção de informações estruturais já disponibilizadas no banco de dados *Protein Data Bank*, propomos uma análise multiparamétrica do problema das fases, com base em aprendizagem de máquina profunda. Nossa hipótese é a de que o mapeamento estatístico extensivo de observações sobre distribuições de fases conhecidas, como um modelo preditivo, pode permitir conclusões sobre o valor alvo de fases em conjuntos de dados ainda não resolvidos, eliminando assim a necessidade de experimentos adicionais ou de estruturas previamente conhecidas. Avanços recentes na área de aprendizado de máquinas, listados nesta proposta, têm permitido soluções de problemas antes considerados intransponíveis e, portanto, substanciam nossa proposta.

Realizações no período

1.1 A cristalografia e o problema das fases

A cristalografia, como técnica fundamental de estudo da estrutura da matéria, surge no início do século XX com a análise do primeiro padrão de difração de raios X por Walter Fridrich e Paul Knipping. Embora incipiente, a ferramenta permitiu confirmar a constituição atômica da matéria e a existência de interações e ligações entre os átomos, fornecendo evidências essenciais para o desenvolvimento das nascentes teorias atômica e quântica (RUPP; KANTARDJIEFF, 2010).

Apesar das demonstrações iniciais de sua efetividade, a cristalografia só foi aplicada com sucesso na determinação estrutural de uma macromolécula em 1958, quando John Kendrew resolveu a estrutura da mioglobina. O hiato temporal entre os sucessos de Fridrich e Kendrew está associado à necessidade de desenvolver técnicas especiais que permitissem o *faseamento* do padrão de difração de moléculas biológicas.

Essencialmente, o faseamento de um padrão de difração consiste em recuperar as fases dos fatores de estrutura, que são inevitavelmente perdidas no processo de detecção dos raios X devido à promediação da radiação por um intervalo de medida ($\sim 10^{-3}$ s) muito maior que o seu período característico ($\sim 10^{-18}$ s). A ausência desta informação impossibilita a reconstrução espacial da densidade eletrônica da molécula analisada; sua recuperação, portanto, é essencial para o sucesso de um experimento cristalográfico.

Para moléculas pequenas, objetos do estudo de Fridrich e Knipping, métodos probabilísticos (ou diretos) que exploram relações matemáticas entre distintas reflexões e a existência de espalhamentos de alta intensidade permitiam o faseamento (FORTIER, 2010). Para macromoléculas, no entanto, a baixa resolução de coleta de dados impede a utilização desses métodos, de modo que o faseamento depende da realização de experimentos acessórios, além da difração do cristal proteico nativo, ou do emprego de propriedades de espalhamento anômalo intrínsecas aos átomos da rede cristalina.

Esses requisitos adicionais, ao delongarem prazos e impedirem a resolução imediata da macromolécula, atuam como objeções ao desenvolvimento eficiente da pesquisa fundamental e de tecnologias substanciadas na biologia estrutural, como o desenvolvimento de fármacos baseados na estrutura do ligante.

1.2 Proposta de solução

Nossa proposta concerne à utilização de ferramentas de aprendizado de máquina para a predição das fases dos fatores de estrutura baseando-se apenas em informações disponíveis antes do faseamento. Efetivamente, supomos que a grande quantidade de estruturas resolvidas

depositadas no *Protein Data Bank* (PDB) contenham relações desconhecidas entre, por exemplo, as simetrias cristalinas e as fases dos fatores de estrutura, que possam ser captadas por modelos de aprendizagem supervisionada. Caso apresentem resultados satisfatórios, esses modelos poderiam ser disponibilizados à comunidade científica sob o formato de uma biblioteca aberta do *Python*, visando tanto sua utilização na resolução de estruturas quanto o seu escrutínio e sucessivo aperfeiçoamento.

1.3 Padronização do código, versionamento e reproducibilidade

A grande quantidade de dados não processados, a manutenção de um ambiente de programação organizado e a construção de modelos de aprendizagem com bons resultados e transparentes exigiram uma formalização da metodologia empregada. Ao longo do trabalho, adotamos medidas que visam consolidar os resultados sob as bases do rigor empírico, buscando evitar o descompasso apontado em (SCULLEY et al., 2018) entre a taxa de progresso e a de consolidação da pesquisa em aprendizado de máquina.

Primeiramente, adotamos um padrão de sintaxe e documentação baseados nos *Python Enhacement Proposals* PEP 8 e PEP 257. A adoção dessas convenções garante consistência ao código, facilitando o trabalho conjunto e garantindo que o usuário final tenha acesso a informações detalhadas sobre a execução das funções implementadas pela biblioteca.

Além disso, para facilitar o trabalho concomitante em diferentes seções do projeto, utilizamos o sistema de versionamento *Git*, hospedando a biblioteca na plataforma *Github*. Desse modo, quando do lançamento de versões estáveis da biblioteca, os usuários podem incluir novas funcionalidades, reportar erros e contribuir com sua solução. Na mesma direção, importantes colaborações científicas, como o Observatório CTA e o colisor de partículas ALICE, disponibilizam suas ferramentas de *software* à comunidade.

Ainda nesse quesito, ressaltamos que o fluxo de trabalho em projetos de aprendizado de máquina não envolve a consecução de etapas encadeadas. Na realidade, ele requer a modificação de um certo conjunto de modelos e atributos (*e.g.* grupos de simetria e parâmetros da célula unitária) que apresentam bons resultados preliminares, para que sejam progressivamente aperfeiçoados na execução da tarefa. Naturalmente, é importante que adições experimentais não sejam permanentes e ocorram separadamente da versão consolidada. Nesse sentido, explorando a divisão em *branches* do *Git*, empregamos a metodologia *Feature Branch Workflow*, representada na Figura 1; nela, as etapas de exploração e readequação ocorrem separadas do código principal (`master`), evitando a introdução de mudanças deletérias.

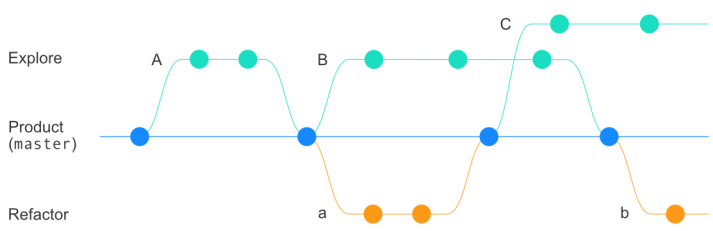


Figura 1 – Desenvolvimento segundo o *Feature Branch Workflow*.

Quanto à reproducibilidade dos resultados, como a execução do código envolve a geração de número aleatórios, sempre que possível fixamos a semente aleatória. Isso garante que as amostras testadas em diferentes estações de trabalho sejam idênticas, de modo que os resultados possam ser posteriormente comparados.

1.4 Coleta de dados e pré-processamento

Como mencionamos, a principal fonte das informações empregadas neste trabalho é a base de dados PDB, que contém os resultados de mais de 10^5 experimentos cristalográficos de estruturas macromoleculares. Esses resultados estão armazenados sob a forma de arquivos `.mtz` e `.pdb`, disponibilizados no ato de deposição da estrutura.

O download desses arquivos é feito por uma suíte de comandos baseados na biblioteca *urllib*, que automatiza tanto os *queries* de comunicação com o servidor do PDB quanto a conversão, através do programa Phenix (LIEBSCHNER et al., 2019), dos `.mtz` originais em arquivos de informação cristalográfica `.cif`, que contém texto interpretável.

Por sua vez, explorando as convenções de construção dos arquivos `.cif` e `.pdb`, desenvolvemos duas classes, `CIFParser` e `PDBParser`, que se baseiam na coincidência de expressões regulares para extrair as informações disponíveis. Essas classes permitem o armazenamento de dados em formato tabular, a sua compactação visando reduzir o uso de memória e, mais importante, a conversão para o formato de *Data Frame* da biblioteca *pandas*. Esse formato é extensamente utilizado na comunidade de aprendizado de máquina pela conveniência que oferece na visualização dos dados, detecção de valores faltantes e pela profícua interação com outras bibliotecas importantes na área, como *numpy* e *scickit-learn*; por essas razões, foi adotado como formato base desta pesquisa.

Naturalmente, ao empregarmos essa representação, estamos definindo que a unidade básica de treinamento dos algoritmos de aprendizagem são as reflexões cristalográficas de uma determinada estrutura. Todavia, essa pode não ser a representação mais adequada, de modo que a especificação de outras unidades básicas de treinamento, como a conversão de dados tabulares em imagens (SHARMA et al., 2019), é um possível aperfeiçoamento do estado corrente da implementação.

Abaixo, na Figura 2, mostramos as 5 primeiras linhas do *Data Frame* resultante da extração de informações de 5 estruturas cristalográficas que passaram pelas etapas de processamento descritas na Seção 1.5.

index_h	index_k	index_l	FOBS	SIGFOBS	UOBS	SIGUOBS	FC	PHI	FOM	RESOL
-31.0	13.0	1.0	34.62	5.125	0.001816	0.000269	33.34	129.1	0.8276	1.901
-31.0	13.0	2.0	58.16	3.523	0.003056	0.0001851	53.75	-30.02	0.948	1.9
-31.0	14.0	1.0	49.56	3.625	0.002575	0.0001882	41.3	-60.1	0.8667	1.906
-31.0	14.0	2.0	20.12	7.33	0.001047	0.0003815	12.39	-69.1	0.4016	1.905
-31.0	14.0	3.0	58.72	3.49	0.003065	0.0001822	57.97	105.06	0.962	1.903

Figura 2 – Um `DataFrame` contendo informações extraídas através das ferramentas de *parsing*.

1.5 Processamento de dados

Mudanças de simetria

Resultados preliminares indicavam que a codificação dos grupos espaciais através de *one-hot encoding*, isto é, incluindo variáveis booleanas para cada estrutura indicando à que grupo de simetrias esta pertencia, não era efetiva, apresentando baixa importância nos algoritmos que empregam CART (*Classification and Regression Trees*).

Desse modo, optamos por transformar todas as estruturas para o grupo espacial P_1 e evitar a inclusão das operações de simetria através de variáveis categóricas. Nesse caso, as simetrias estão implícitas através da similaridade de atributos existente entre reflexões simetricamente relacionadas.

Para realizar a conversão, utilizamos a ferramenta SFTOOLS da suíte de programas para cristalografia CCP4. Importante notar que a execução dessa etapa, ao aumentar os arquivos `.cif` pela inclusão de novas reflexões, exige maiores tempos de carregamento dos dados e treinamento dos modelos, dificuldades que suplantamos parcialmente pela utilização de recursos de computação de alta performance disponibilizados pela Universidade de São Paulo (USP).

Unitarização dos fatores de estrutura

Um aspecto central no processamento de dados antes da sua aplicação em aprendizado de máquina é a normalização dos atributos presentes no conjunto de treinamento. Esse procedimento garante que recursos distintos, pertencentes a intervalos de valores dissemelhantes, estejam na mesma escala numérica, evitando a introdução de viés não planejado e acelerando o treinamento do algoritmo.

Ao normalizarmos os valores de um atributo estamos assumindo que as entradas associadas a ele podem ser comparadas, ou melhor, que o conteúdo de uma entrada, ao ser determinado por um processo físico comum, é fisicamente equivalente ao de outras. No entanto, observação atenta revela que os módulos dos fatores de estrutura não satisfazem a esse pressuposto básico.

Para verificarmos isso, observe que o fator de estrutura:

$$\mathbf{F}(\mathbf{S}) = \sum_j f_j \exp\left[2\pi i(\mathbf{r_j} \cdot \mathbf{S})\right] \tag{1.1}$$

além de decrescer com o $|S|$, o módulo do vetor de espalhamento, devido à sua dependência com o fator de forma atômico (f_j), está vinculado ao fator de temperatura (B). Uma vez que tanto f_j quanto B estão intrinsecamente relacionados à estrutura cristalina que se analisa, não assumindo os mesmos valores para diferentes cristais, então $|\mathbf{F}(\mathbf{S})|$ não preserva a mesma escala entre diferentes estruturas.

Este fato sugere que uma estandardização dos módulos dos fatores de estrutura tem de eliminar a influência dessas dependências. A alternativa principal, como sugerida em (DRENTH,

2006), é introduzir o fator de estrutura unitário:

$$\mathbf{U}(\mathbf{S}) = \frac{\mathbf{F}(\mathbf{S})_{pt}}{\sum_j Z_j} \tag{1.2}$$

onde Z_j é o número atômico do j-ésimo átomo da célula unitária e $\mathbf{F}(\mathbf{S})_{pt}$ é o fator de estrutura assumindo que os espalhadores são pontuais, suprimindo a dependência em $|S|$ e no movimento térmico. De maneira geral:

$$\mathbf{F}(\mathbf{S})_{pt} = \frac{\sum_j Z_j \exp\left[B_j \left(\sin^2\theta/\lambda^2\right)\right]\mathbf{F}_{obs}}{\sum_j f_j} \tag{1.3}$$

onde $j = 1, \ldots, N$ indexa os átomos da célula unitária e $\mathbf{F}_{obs}$ é o fator de estrutura na *escala absoluta*, cujos módulos são medidos no experimento. Estimando B_j pela disposição isotrópica média quadrática das posições atômicas em torno do equilíbrio, teremos:

$$\mathbf{U}(\mathbf{S}) = \frac{\exp\left[B \left(\sin^2\theta/\lambda^2\right)\right]\mathbf{F}_{obs}}{\sum_j f_j} \tag{1.4}$$

Que goza da propriedade de ser independente dos coeficientes de escala existentes entre conjuntos distintos de reflexões, permitindo a normalização fisicamente consistente dos fatores de estrutura.

Assim, com auxílio dos desenvolvedores do CCP4, implementamos, através da Computational Crystallography Toolbox (CCTBX) a conversão dos módulos dos fatores de estrutura na escala absoluta em fatores de estrutura unitários. Nesse caso, os fatores de temperatura (B) da fórmula de Debye-Weller foram estimados pela disposição quadrática média, suposta isotrópica, dos átomos da célula unitária:

$$B = 8\pi^2\langle u_{iso}^2\rangle \tag{1.5}$$

No entanto, a Equação (1.5) envolve informações apenas disponíveis em um modelo da estrutura, contradizendo o requisito de utilizar apenas dados disponíveis *a priori*. Entendemos que uma possível maneira de saná-lo é estimar B através do *plot* de Wilson. Contudo, como discutido em (DIEDERICHS, 2010), sabe-se que o fator de temperatura assim estimado é sistematicamente menor que o valor médio da distribuição real de B_j.

Codificação dos índices de Miller

Como verificamos no relatório parcial, os índices de Miller assumiam importância significativa na determinação das predições do modelo de floresta aleatória, vide Tabela 1. Todavia, mesmo na condição de recurso relevante, os índices não podem, a exemplo dos já discutidos fatores de estrutura, serem normalizados. É certo, por exemplo, que reflexões indexadas pela mesma tripla (h, k, l) não representam, para cristais com diferentes parâmetros da célula unitária, reflexões numa mesma direção ou associadas à mesma quantidade de matéria espalhadora.

Tabela 1 – Importância das características (*feature importances*) na Floresta Aleatória ordenadas decrescentemente.

Característica	Importância (%)
`FOBS`	0.218
`RESOL`	0.217
`SIGFOBS`	0.213
`index_k`	0.114
`index_l`	0.109
`index_h`	0.105
⋮	⋮

Para garantir que a informação de direcionalidade e do volume da célula unitária sejam preservados na normalização, empregamos dois esquemas de codificação, que intitulamos *planar* ou *angular*, como representados na Figura 3.

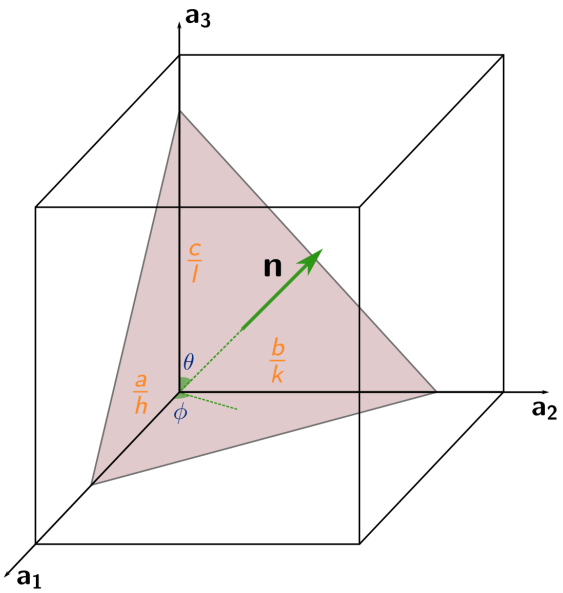


Figura 3 – Representação dos esquemas de codificação planar e angular.

No esquema *planar*, os índices de Miller são convertidos para distâncias através dos seus recíprocos, exceto quando são nulos, isto é, quando o plano é paralelo ao eixo indexado. Assim, se tomarmos $i \in \{h, k, l\}$ e a_i o comprimento do eixo correspondente, então:

$$i \longrightarrow \begin{cases} \dfrac{a_i}{i} & \text{se } i \neq 0 \\ 0 & \text{se } i = 0 \end{cases} \tag{1.6}$$

Essa codificação foi aplicada em (YEO DONGHUN KIM, 2019), no qual é relatado o seu uso para processamento de dados num problema de análise de componentes principais. Os autores reportam sucesso na abordagem, suplantando a necessidade de longos cálculos computacionais

em Teoria do Funcional da Densidade para obtenção da densidade de estados eletrônica, uma quantidade importante na determinação de propriedades de metais.

No esquema *angular*, por sua vez, optamos por preservar a informação da direcionalidade da reflexão. Nesse caso, convertemos os índices de Miller em ângulos esféricos (ϕ, θ) que especificam o vetor normal ao plano cristalino indexado por (h, k, l). Em suma, realizamos o seguinte procedimento:

1. Transformamos índices $i \in \{h, k, l\}$ nulos numa representação do infinito $i = \texttt{inf}$ com a propriedade que $1/\texttt{inf} = 0$.

2. Obtemos o vetor normal unitário ao plano indexado por (h, k, l):

$$\mathbf{n} = \frac{1}{\sqrt{\left(\frac{ab}{hk}\right)^2 + \left(\frac{ac}{hl}\right)^2 + \left(\frac{bc}{kl}\right)^2}} \left(\frac{bc}{kl}, \frac{ac}{hl}, \frac{ab}{hk}\right) \tag{1.7}$$

3. Convertemos o vetor $\mathbf{n} = (n_x, n_y, n_z)$ nos ângulos esféricos (ϕ, θ):

$$(\phi, \theta) = \left[\arctan\left(\frac{n_y}{n_x}\right), \arccos n_z\right] \tag{1.8}$$

4. Transformamos $(h, k, l) \longrightarrow (\phi, \theta)$

Codificação da fase

A terceira etapa de processamento consiste em readequar as fases dos fatores de estrutura do conjunto *objetivo* pela modificação de duas propriedades: o diâmetro do conjunto ou a representação computacional dos seus valores. Para clarificar, entendemos esses conceitos como:

a) *Diâmetro do conjunto*: a distância entre os elementos máximo e mínimo do conjunto. Se $\mathcal{Y}$ é o conjunto objetivo, então $\operatorname{diam}(\mathcal{Y}) = \operatorname{máx} \mathcal{Y} - \min \mathcal{Y}$.

b) *Representação computacional*: é um mapa $f : \mathcal{Y} \to X$ inversível e, portanto, que preserva a informação contida em $\mathcal{Y}$, permitindo decodificar X ao final do treinamento para reobter os elementos do conjunto objetivo original. Evidentemente, se fizermos $\forall y \in \mathcal{Y}, f(y) = y$, teremos a representação convencional.

Primeiramente, para modificação do diâmetro do conjunto utilizamos um esquema de escalonamento linear, no intuito de limitar os valores do conjunto objetivo ao intervalo $[-1, 1]$.

Em segundo lugar, observe que as fases dos fatores de estrutura, na condição de ângulos, são variáveis de resposta circular, isto é $\texttt{PHI} \in [-180°, 180°]$, mas $-180°$ e $180°$ representam o mesmo ponto. É desse fato que surge a necessidade de uma mudança na representação computacional de `PHI`.

Contudo, é importante notar que não encontramos nenhuma abordagem na literatura que indicasse a viabilidade de uma ou outra codificação e apontasse os impactos na performance de

modelos de aprendizado de máquina. De fato, a atenção da comunidade de estatística para os problemas com variáveis de entrada e resposta circulares é recente (PEWSEY; NEUHÄUSER; RUXTON, 2013).

No entanto, seguindo indicações empíricas da sua eficiência, como discutido na comunidade *Cross Validated* e explorando a possibilidade de redes neurais produzirem resultados em 2 ou mais dimensões, utilizamos a seguinte representação:

$$f(\varphi) = [\cos(\varphi), \sin(\varphi)] \tag{1.9}$$

onde φ é a fase do fator de estrutura. A essa codificação damos o nome de `sin-cos`.

1.6 Ferramentas de visualização

A exploração visual dos dados disponíveis é etapa mandatória em problemas de aprendizado de máquina (GÉRON, 2017). Tipicamente, as variáveis de entrada e resposta possuem correlações que podem ser aproveitadas para subsidiar a escolha do algoritmo de aprendizagem. De fato, há uma pletora de modelos sob o nome de *Manifold Learning* (aprendizado em variedades) que visa explorar estruturas geométricas existentes no espaço dos atributos para reduzir a dimensionalidade das informações ou facilitar a regressão à variável de resposta.

Nesse sentido, buscamos verificar se essas relações no espaço de atributos do nosso problema poderia sugerir a adoção de alguma estratégia de aprendizagem. Alguns resultados dessa busca estão representadas nas Figuras 4 e 5 abaixo.

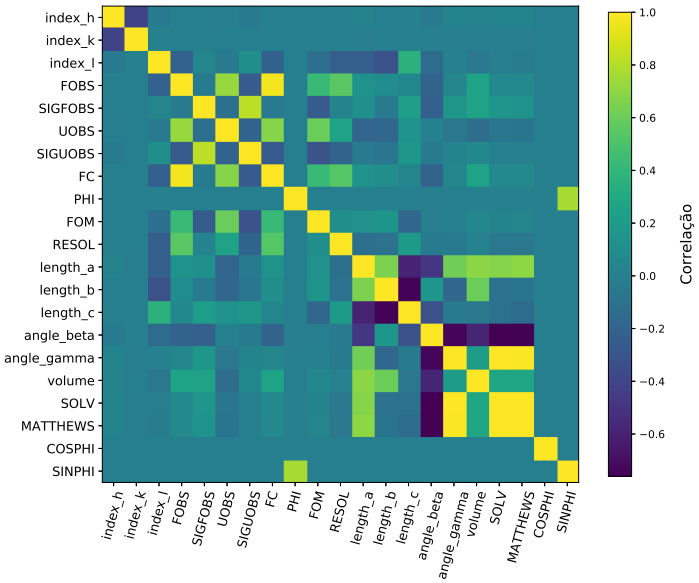


Figura 4 – Correlação entre atributos do conjunto de dados.

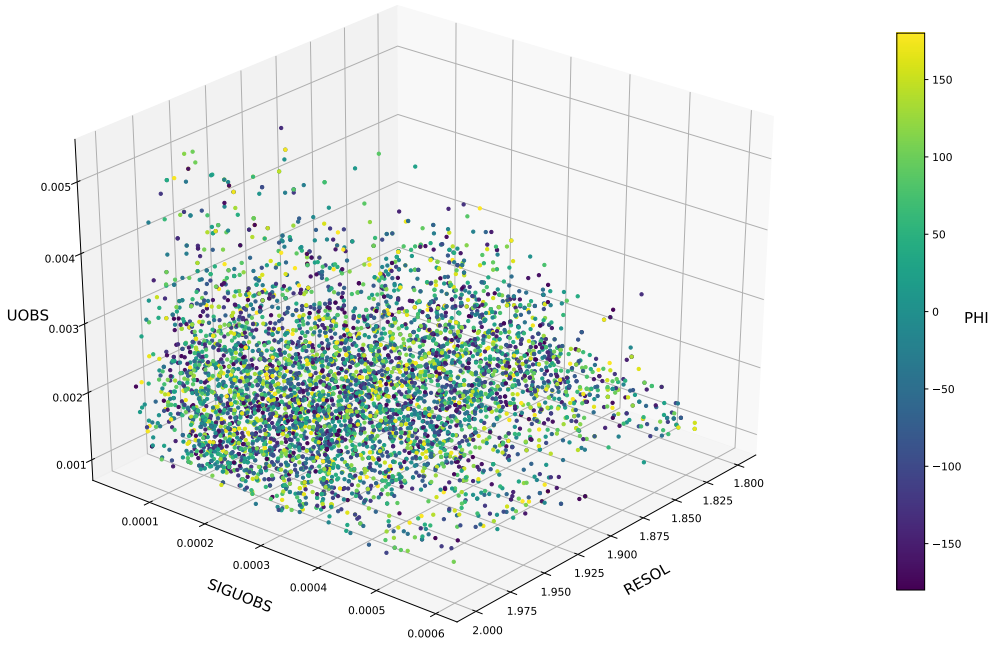


Figura 5 – Relação entre `UOBS`, `SIGUOBS`, `RESOL` no espaço de atributos.

Essas figuras, no entanto, não sugerem nenhuma relação geométrica explorável entre os recursos e a variável de resposta (`PHI`). Note, todavia, que isso não representa que essas relações não existam, mas que possam possuir, por exemplo, leis físicas subjacentes intrincadas impossíveis de serem visualizadas em espaços de baixa dimensão ou, até mesmo, que a representação dos dados não é adequada.

1.7 Modelos de aprendizado

Pela versatilidade e resultados positivos em outros problemas de aprendizado de máquina, utilizamos dois modelos de aprendizagem:

a) XGBoost (ou XGB, do inglês *Extreme Gradient Boosting*): algoritmo desenvolvido por (CHEN; GUESTRIN, 2016) baseado na técnica de *tree boosting* na qual uma sequência de estimadores CART (*Classification and Regression Trees*) é treinada para ajustar-se ao erro preditivo do estimador precedente.

b) DNN (*Deep Neural Network*): definida por (GÉRON, 2017) como uma rede neural com duas ou mais camadas ocultas. Em particular, empregamos duas camadas ocultas pela sensibilidade do problema à *overfitting*, *i.e.* por apresentar razoável performance no conjunto de treinamento não correspondida por resultados satisfatórios no conjunto de testes. Esses modelos foram construídos pela biblioteca de baixo-nível *tensorflow*, sendo, portanto, bastante flexíveis.

Particularmente, construímos 5 modelos, cuja princípios de construção descrevemos a seguir:

1. `RPR` : amostragem aleatória de fases da distribuição do conjunto objetivo utilizado no treinamento. Fornece uma performance de referência para estimar a qualidade dos regressores construídos.

2. `XGB-NFS` : regressão via XGB sem seleção de atributos (*feature selection*). Fases não escalonadas, *i.e.* mantidas no intervalo $[-180°, 180°]$. Hiperparâmetros selecionados, vide Tabela 2, através de validação cruzada aleatória com 10 iterações em Pipeline da biblioteca *scikit-learn*.

`miller_indexes_encoder__encoding`	`'both'`
`feature_scaling`	`RobustScaler`
`xgb__min_child_weight`	`0.5`
`xgb__max_depth`	`20`
`xgb__n_estimators`	`100`
`xgb__alpha`	`10`

Tabela 2 – Hiperparâmetros selecionados para o modelo `XGB-NFS` .

3. `XGB-FS` : regressão via XGB com seleção de atributos usando as importâncias dos atributos (*feature importances*) obtidas no modelo `XGB-NFS` . Hiperparâmetros, vide Tabela 3, selecionados via validação cruzada aleatória com 15 iterações.

`miller_indexes_encoder__encoding`	`'both'`
`feature_scaling`	`RobustScaler`
`feature_selection__k`	`5`
`xgb__min_child_weight`	`0.5`
`xgb__max_depth`	`20`
`xgb__n_estimators`	`80`
`xgb__alpha`	`10`

Tabela 3 – Hiperparâmetros selecionados para o modelo `XGB-FS` .

4. `DNN` : regressão via rede neural com 2 camadas ocultas de 500 e 200 na primeira e segunda camada, respectivamente. As camadas são completamente conectadas. Não houve seleção de atributos. Fases linearmente escalonadas para o intervalo $[-1.0, 1.0]$. Função objetivo como erro quadrático médio. Hiperparâmetros selecionados manualmente, vide Tabela 4, sem procedimento rigoroso de determinação.

`n_hidden1`	`500`
`n_hidden2`	`200`
`layer_type`	`fully_connected`
`optimizer`	`AdagradOptimizer`
`optimizer__learning_rate`	`0.2`
`n_epochs`	`2000`
`batch_size`	`1000`

Tabela 4 – Hiperparâmetros do modelo `DNN`.

5. `DNN-2`: regressão via rede neural com a mesma configuração da rede anterior. Fases codificadas pelo esquema `sin-cos`. Função objetivo como erro quadrático médio.

1.8 Avaliação dos modelos

Descrição da amostra utilizada

Para treinamento e avaliação dos modelos, empregamos a amostra descrita na Tabela 5. Na sua construção foram utilizados 5 deposições do PDB, sob os índices 4YGG, 3R3B, 4HR5, 3IOY e 3HG9. Ressalta-se que, mesmo incluindo pequeno número de estruturas, a amostra ocupa, sem procedimentos de redução da representação numérica, cerca de 150M de memória temporária.

Nome	Tipo	N	f
`X_train, y_train`	Treinamento	437 018	80%
`X_test, y_test`	Teste	109 255	20%

Tabela 5 – Propriedades da amostra.

1.8.1 `RPR`

Este regressor serve apenas como referência para compreensão dos modelos que apresentaremos a seguir: quão melhor é o desempenho destes relativamente àquele pode nos indicar se as hipotéticas relações entre o conjunto de treinamento e o objetivo estão sendo apreendidas. A Figura 6 e o sumário de resultados da Tabela 6 mostram as predições no conjunto de testes e a raiz quadrada do erro quadrático médio nos conjuntos de treinamento e testes, respectivamente.

Conjunto	$\sqrt{\text{EQM}}$
Treinamento	147.7 °
Teste	147.9 °

Tabela 6 – Raiz quadrada do erro quadrático médio (EQM) para cada conjunto utilizando o modelo `RPR`.

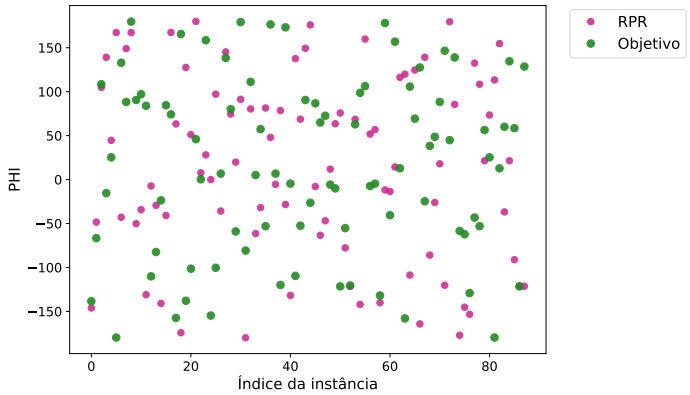


Figura 6 – Predições para o modelo `RPR` no conjunto de testes.

1.8.2 `XGB-NFS`

Neste primeiro modelo efetivo, apresentamos os resultados da predição tanto no conjunto de treinamento quanto no conjunto de testes, devido à manifestação de fenômeno comum a todos os modelos treinados: a tendência à aglutinação das predições no valor médio do conjunto objetivo e o desempenho significativamente melhor durante o treinamento, o que é sintomático de *overfitting*.

Os gráficos dispostos nas Figuras 7 e 8 mostram, após a redução da densidade de pontos, o valor real da fase (•) e o valor predito (•) para cada reflexão.

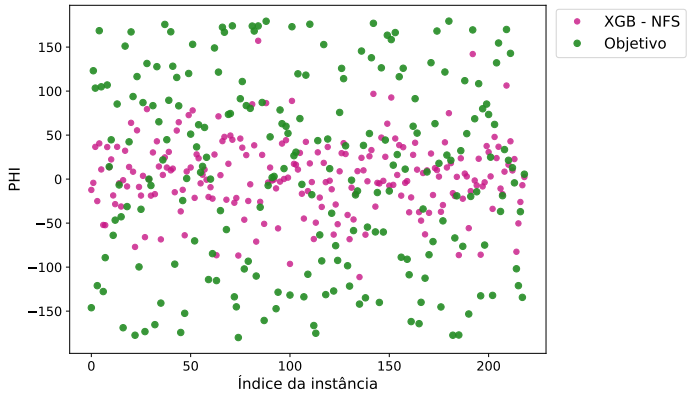


Figura 7 – Predições para o modelo `XGB-NFS` no conjunto de treinamento.

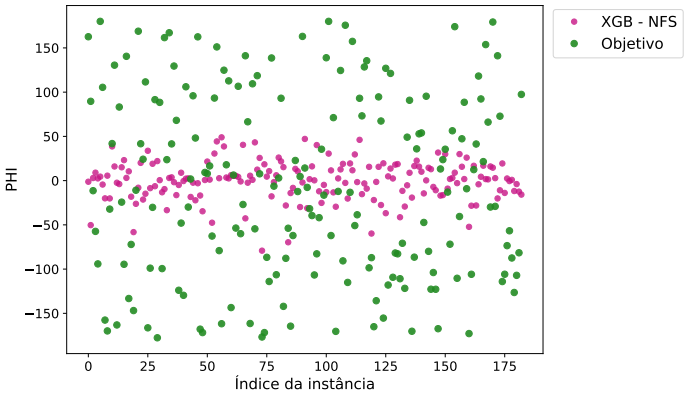


Figura 8 – Predições para o modelo `XGB-NFS` no conjunto de testes.

Um sumário dos resultados desse modelo se encontra na Tabela 7, onde empregamos a raiz quadrada do erro quadrático médio (EQM) como métrica de performance. Note que, devido à natureza circular do objetivo (•), essa métrica pode não ser adequada e precisa ser substituída pela correspondente em estatística circular (FISHER, 1995).

Conjunto	$\sqrt{\text{EQM}}$
Treinamento	72.2°
Teste	105.3°

Tabela 7 – Raiz quadrada do erro quadrático médio (EQM) para cada conjunto utilizando o modelo `XGB-NFS`.

1.8.3 `XGB-FS`

Utilizando o modelo anterior, obtivemos as importâncias de cada atributo, como dispostas na Figura 9. Os atributos de importância nula podem ter sido selecionados pela aplicação da regularização ℓ_1 via o parâmetro `xgb__alpha`, que gera, tipicamente, modelos esparsos.

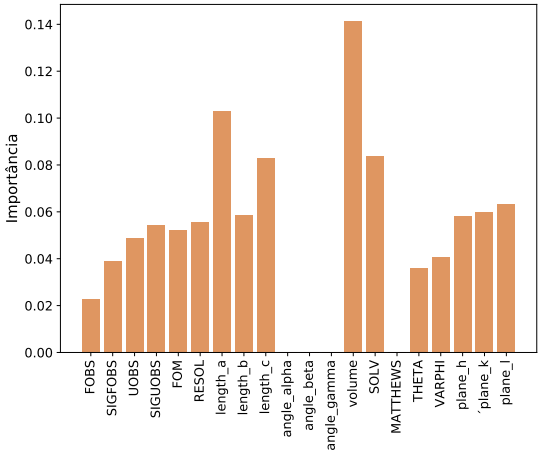


Figura 9 – Importâncias dos atributos conforme o modelo `XGB-NFS`.

O modelo resultante da aplicação da seleção de atributos teve melhor performance com os

$k = 5$ atributos de maior importância, gerando as predições da Figura 10.

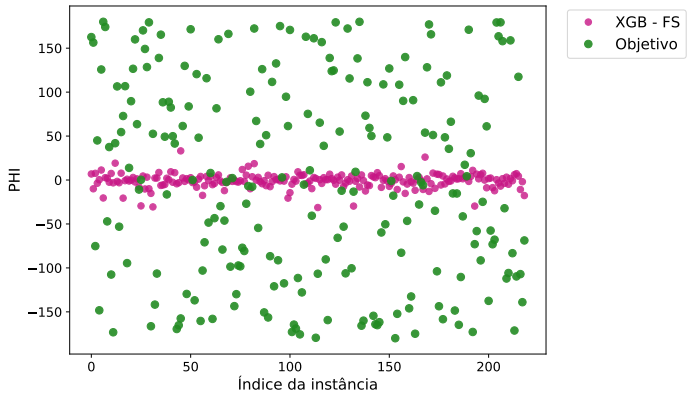


Figura 10 – Predições para o modelo `XGB-FS` no conjunto de testes.

Novamente, os resultados da Figura 8 sob a luz do sumário de performance disposto na Tabela 8 evidenciam que os modelos foram incapazes de capturar possíveis dependências entre os atributos e as fases dos fatores de estrutura, mesmo que os resultados sejam efetivamente melhores que a predição aleatória do modelo `RPR`.

Conjunto	$\sqrt{\text{EQM}}$
Treinamento	104.0 °
Teste	104.7°

Tabela 8 – Raiz quadrada do erro quadrático médio (EQM) para cada conjunto utilizando o modelo `XGB-FS`.

1.8.4 `DNN`

Agora, utilizando, pela primeira vez, uma arquitetura de rede neural, obtivemos as predições evidenciadas na Figura 11. Mais uma vez, o sumário da Tabela 9 mostra que obtivemos resultados similares aos anteriores, com erro de $\approx 110°$.

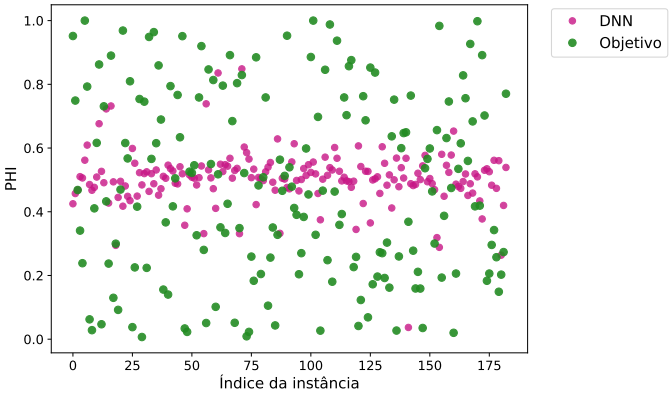


Figura 11 – Predições para o modelo `DNN` no conjunto de testes.

Conjunto	$\sqrt{\text{EQM}}$
Treinamento	98.8 °
Teste	110.3 °

Tabela 9 – Raiz quadrada do erro quadrático médio (EQM) para cada conjunto utilizando o modelo `DNN`.

1.8.5 `DNN-2`

Por último, empregando o esquema de codificação `sin-cos` e uma rede neural com duas saídas, obtivemos os resultados mostrados na Figura 12 com a performance sintetizada na Tabela 10. Note que, embora mais disperso e isento do fenômeno de aglutinação das predições na região central, o modelo possui erro similar ao regressor aleatório `RPR`.

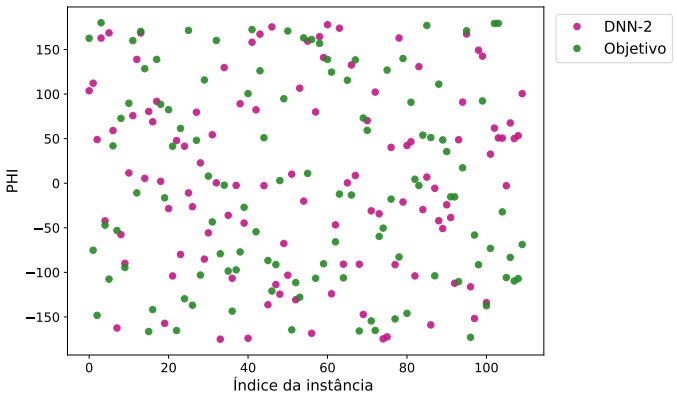


Figura 12 – Predições para o modelo `DNN-2` no conjunto de testes.

Conjunto	$\sqrt{\text{EQM}}$
Treinamento	134.15 °
Teste	147.5213 °

Tabela 10 – Raiz quadrada do erro quadrático médio (EQM) para cada conjunto utilizando o modelo `DNN-2`.

1.9 Discussão final

No geral, os modelos não apresentaram efetividade suficiente para sua aplicação no faseamento de estruturas inéditas. Todavia, alcançamos importantes marcos de desenvolvimento de código e delimitamos seções prioritárias de estudo e aperfeiçoamento, dando mais ênfase aos fundamentos matemáticos e estatísticos que subjazem a esse projeto.

Efetivamente, versões aprimoradas deste trabalho devem assentar-se sobre modelos estatisticamente bem delineados e matematicamente mais complexos, a exemplo da rede neural `DNN-2`; melhorias na escolha de parâmetros, treinamento mais extenso e processamento fisicamente consistente, podem aperfeiçoar o desempenho ou apontar novas direções de pesquisa.

Avaliação do apoio institucional

Este projeto desenvolveu-se em ambiente de discussão, construção de hipóteses e trabalho que, certamente, não seria possível sem o apoio da FAPESP.

Agradeço, portanto, pela confiança depositada na capacidade de executarmos este projeto e pelo apoio financeiro dispensado via o processo 2018/23675-7.

Referências

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 785–794. ISBN 9781450342322. Citado na página 12.

DIEDERICHS, K. *Wilson plot*. 2010. <https://strucbio.biologie.uni-konstanz.de/ccp4wiki/index.php/Wilson_plot>. Citado na página 8.

DRENTH, J. *Principles of Protein X-ray Crystallography*. [S.l.]: Springer New York, 2006. Citado na página 8.

FISHER, N. *Statistical Analysis of Circular Data*. Cambridge University Press, 1995. (Statistical Analysis of Circular Data). ISBN 9780521568906. Disponível em: <https://books.google.com.br/books?id=wGPj3EoFdJwC>. Citado na página 16.

FORTIER, S. (Ed.). *Direct Method for Solving Macromolecular Structure*. [S.l.]: Kluwer Academic Publishers, 2010. Citado na página 4.

GÉRON, A. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. [S.l.]: O'Reilly Media, 2017. Citado 2 vezes nas páginas 11 e 12.

LIEBSCHNER, D. et al. Macromolecular structure determination using x-rays, neutrons and electrons: recent developments in phenix. *Acta Cryst.*, 2019. Citado na página 6.

PEWSEY, A.; NEUHÄUSER, M.; RUXTON, G. *Circular Statistics in R*. [S.l.]: OUP Oxford, 2013. (Ebrary online). Citado na página 11.

RUPP, B.; KANTARDJIEFF, K. *Biomolecular Crystallography: Principles, Practice, and Application to Structural Biology*. [S.l.]: Garland Science, 2010. Citado na página 4.

SCULLEY, D. et al. *Winner's Curse? On Pace, Progress, and Empirical Rigor*. 2018. Citado na página 5.

SHARMA, A. et al. Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture. *Scientific Reports*, v. 9, n. 1, p. 11399, 2019. Citado na página 6.

YEO DONGHUN KIM, C. K. . S. S. H. B. C. Pattern learning electronic density of states. *Nature Scientific Reports*, April 2019. Citado na página 9.