Gerador de proteínas para o tratamento da Hemofilia tipo B baseado em *Deep Reinforcement Learning*

Arthur Moriggi Pimenta

Dissertação apresentada ao
Instituto de Matemática e Estatística
da Universidade de São Paulo
para obtenção do título de
Mestre em Ciências

Programa: Ciência da Computação
Orientador: Prof. Dr. Roberto Hirata Junior

São Paulo
Novembro de 2023

Gerador de proteínas para o tratamento da Hemofilia tipo B baseado em *Deep Reinforcement Learning*

Arthur Moriggi Pimenta

Esta é a versão original da dissertação elaborada pelo candidato Arthur Moriggi Pimenta, tal como submetida à Comissão Julgadora.

Lista de abreviaturas

MSA	Alinhamento múltiplo de sequências (*Multiple Sequence Alignment*)
IS	Idêntidade de Sequência
TM-Score	Escores de Modelagem de Template (*Template Modeling Score*)
LLM	Modelo Grande de Linguagem (*Large Language Model*)
PPO	Otimização de Política Proxima (*Proximal Policy Optimization*)
IME	Instituto de Matemática e Estatística
USP	Universidade de São Paulo

Lista de figuras

Lista de tabelas

Sumário

Introdução

A hemofilia tipo B é uma doença hereditária rara, caracterizada por uma deficiência no fator IX de coagulação sanguínea. Os pacientes com a doença enfrentam um risco maior de sofrer com hemorragias graves, tanto internas quanto externas, podendo levar a complicações debilitantes e até mesmo à morte PM, 2020.

Apesar dos avanços notáveis no tratamento da hemofilia nas últimas décadas, muitos desafios ainda persistem. O tratamento tradicional da hemofilia tipo B envolve a infusão de fatores de coagulação recombinantes ou derivados do plasma sanguíneo Gouw, 2013. Embora eficazes na prevenção de hemorragias, esses tratamentos apresentam limitações significativas, como a necessidade de infusões frequentes devido à curta meia-vida do fator IX na circulação, a possibilidade de desenvolvimento de inibidores anticoagulantes e os elevados custos associados Mancuso, 2005. Nesse contexto, o projeto de sequências proteicas surge como uma abordagem com potencial de reduzir os custos associados ao tratamento da doença, ao projetar proteínas sob medida que desempenham a mesma função, porém, mais estáveis, exigindo uma quantidade menor de infusões.

0.1 Projeto de Sequências

O projeto de sequências refere-se ao processo de criação ou otimização de uma sequência de aminoácidos para produzir uma proteína cuja estrutura tridimensional possua propriedades desejáveis. Essa tarefa é extremamente desafiadora, pois o número de permutações possíveis para uma sequência de aminoácidos cresce exponencialmente com seu tamanho, tornando inviável a busca por força bruta Wenze Ding e Gong, 2022.

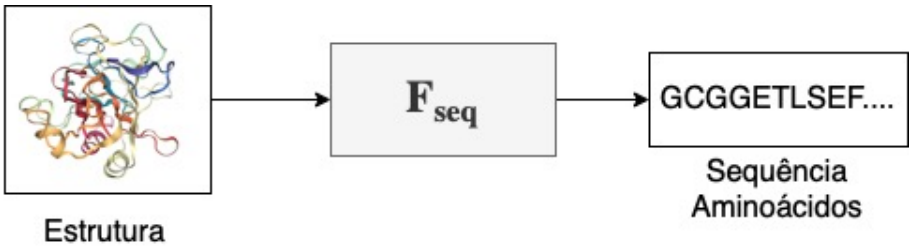


Figura 1: *O projeto de sequências se resume a desenvolver uma função F_{seq} que mapeia uma estrutura proteica tridimensional em uma sequência de aminoácidos.*

0.1.1 Baseado em Busca

No projeto de sequências baseado em busca, a função F_{seq} é definida por um processo iterativo composto por etapas bem estruturadas. Inicialmente, é gerada uma sequência de aminoácidos baseada em uma heurística inicial H. A sequência resultante é avaliada por uma função objetivo F_{obj}, que calcula a métrica a ser otimizada. Se a sequência atual for melhor que a sequência anterior de acordo com a métrica definida, o Agente A a adota como nova sequência final e realiza mutações na sequência atual. Esse processo é repetido até que um critério de parada seja atingido, como um número máximo de iterações ou a convergência em F_{obj}.

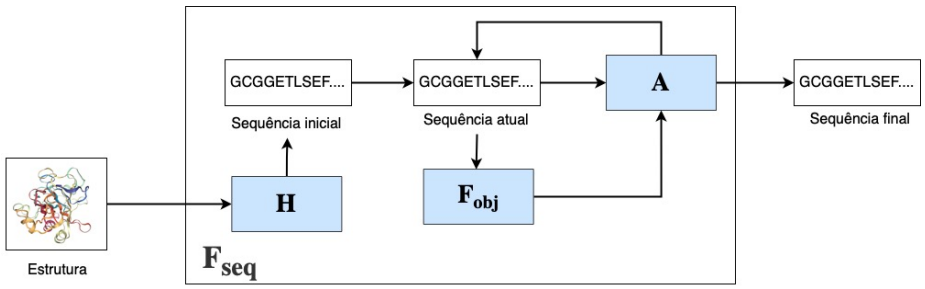


Figura 2: *O projeto de sequências baseado em busca define F_{seq} como um processo iterativo que combina uma heurística inicial H, a avaliação por F_{obj} e as modificações realizadas pelo Agente A. O processo é repetido até atingir um limiar calculado por F_{obj}.*

O *Rosetta* B, 2019 se baseia em um pipeline como este, no qual a heurística inicial H utiliza a sequência de uma proteína estruturalmente semelhante como ponto de partida. A função objetivo F_{obj} avalia a energia livre da proteína, considerando fatores como interações de Van der Waals, interações eletrostáticas e ligações de hidrogênio. O Agente A utiliza o Método de Monte Carlo (MMC) para realizar mutações e adotar configurações com menor energia livre. Abordagens alternativas podem usar a similaridade estrutural como função objetivo, sendo uma métrica amplamente utilizada o *Template Modeling Score* (TMScore) Zhang, 2004.

0.1.2 Baseado em Aprendizado Profundo

O projeto de sequências baseado em aprendizado profundo define a função F_{seq} a partir de redes neurais artificiais. O J. D. e. al, 2022 por exemplo, determina a F_{seq} como uma rede neural profunda, denominada *ProteinMPNN*, que mapeia de forma direta a estrutura alvo à sequência de aminoácidos. A rede é construída através de uma *Message Passing Neural Network* (MPNN) composta por uma arquitetura *encoder-decoder* que se baseia nas características da estrutura como distância e orientação dos átomos no espaço, para fazer predições J. D. e. al, 2022.

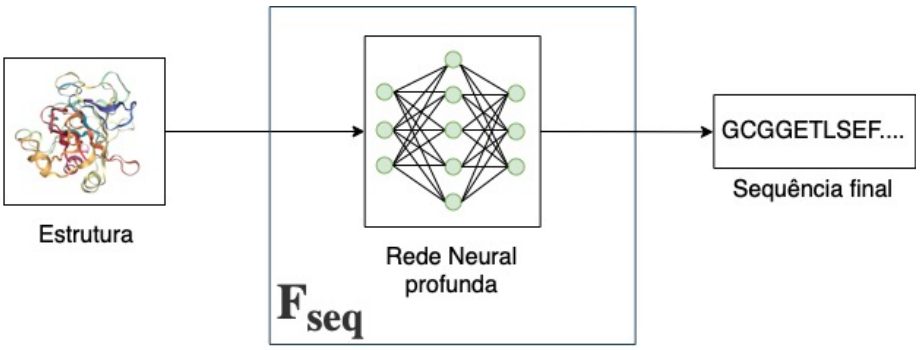


Figura 3: *Projeto de Sequências baseado em aprendizado profundo.*

0.2 Proposta

Além de suscitar uma possível resposta imunológica, o tratamento da hemofilia tipo B baseado na infusão de fator IX é custoso devido ao pouco tempo de vida da proteína na corrente sanguínea, demandando infusões frequentes. Neste sentido, este trabalho busca responder à seguinte pergunta científica: É possível, através de projeto de sequências, projetar uma proteína que desempenhe a mesma função do fator IX, mas com maior estabilidade, menor imunogenicidade e menos necessidade de infusões?

Para responder a esta questão, formulamos a hipótese de que, se existir uma proteína alternativa superior ao Fator IX, ela deverá ser estruturalmente similar. Assim, o objetivo será identificar proteínas estruturalmente semelhantes que atendam aos critérios estabelecidos. Para simplificar o problema, focaremos no projeto do domínio protease, reconhecido como o mais relevante para a função do fator IX.

Nossa proposta consiste no desenvolvimento de um *pipeline* que combina estratégias de busca com aprendizado profundo para otimização de sequências proteicas. Esse *pipeline* será estruturado em três módulos principais: Condições Iniciais, Treinamento e Geração de Sequências.

O módulo de Condições Iniciais tem como objetivo obter a sequência inicial de aminoácidos por meio da heurística H, além de calcular o erro inicial associado a essa sequência utilizando a função objetivo F_{obj}. Para isso, empregaremos o *ProteinMPNN* como heurística H e o TMScore como métrica de avaliação estrutural F_{obj}.

Diferente do uso de agentes que realizam mutações aleatórias, como no *Rosetta*, no módulo de Treinamento treinaremos uma rede neural profunda, o *GenSeq*, para atuar como o Agente A, sendo capaz de realizar mutações que otimizem a similaridade com a estrutura alvo, isto é, o Fator IX.

Por fim, no módulo de Geração de Sequências, o agente treinado será utilizado para produzir um conjunto variantes proteicas estruturalmente semelhantes ao Fator IX, por meio de um processo de mutação orientado. As sequências geradas serão então avaliadas para identificar possíveis candidatas viáveis que possam substituir o Fator IX no tratamento da hemofilia tipo B.

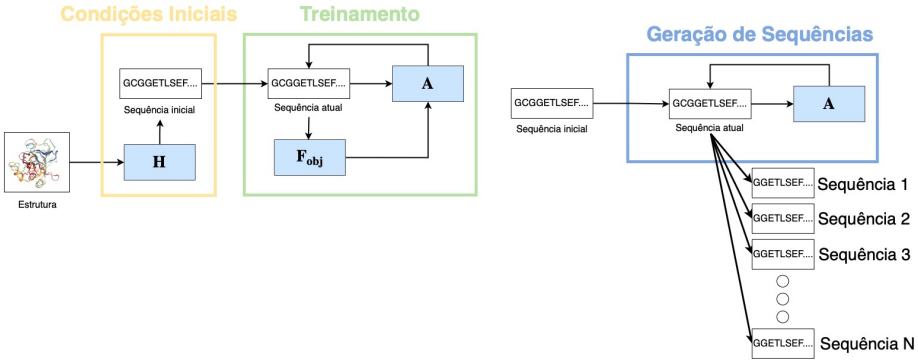


Figura 4: *Pipeline proposto para o projeto de sequências.*

0.3 Objetivos

1. Obter um conjunto de sequências de aminoácidos que mimetizem a estrutura do fator de coagulação IX baseado na métrica TMScore.
2. Avaliar, dentro do conjunto de proteínas obtidas em 1, quais possuem potencial de substituir o Fator IX de modo tornar o tratamento de Hemofilia tipo B mais acessível e eficiente.
3. Desenvolver um *pipeline* genérico que produza sequências que mimetizem uma estrutura qualquer.

Capítulo 1

Revisão Bibliográfica

O objetivo deste capítulo é situar o presente trabalho no contexto das contribuições existentes na área de pesquisa. Para isso, analisamos diversas publicações da literatura que estão alinhadas com os objetivos deste estudo.

1.1 Trabalhos relacionados

O tratamento atual para hemofilia B baseia-se na reposição do FIX por meio de proteínas recombinantes ou derivadas do plasma humano. A introdução de fatores de coagulação recombinantes na prática clínica representou um marco no manejo da doença, proporcionando maior segurança e eficácia em comparação com derivados do plasma, reduzindo significativamente os riscos de transmissão de patógenos Cohen e Kessler, 1995. No entanto, as terapias baseadas na reposição de FIX ainda apresentam desafios substanciais. Devido à curta meia-vida do FIX na circulação, pacientes geralmente necessitam de infusões frequentes para manter níveis terapêuticos adequados, o que aumenta os custos e a complexidade do tratamento Mancuso, 2005. Além disso, os pacientes podem desenvolver inibidores, que são anticorpos neutralizantes dirigidos contra o FIX, comprometendo a eficácia do tratamento Mancuso ME, 2017. Avanços recentes, como o desenvolvimento de variantes do FIX de longa duração como o rFIX-Fc, estenderam a meia-vida da proteína, reduzindo a frequência de infusões al., 2013. Apesar disso, essas terapias continuam a ser custosas e, em alguns casos, limitadas pela resposta imunológica do paciente.

O design de proteínas surge como uma alternativa com potencial de desenvolver tratamentos mais eficientes. Trata-se de um campo emergente que busca criar ou modificar proteínas para otimizar suas funções biológicas, estabilidade estrutural ou características terapêuticas. Os avanços nessa área têm sido impulsionados pelo aprimoramento de tecnologias de modelagem molecular e pelo uso de ferramentas computacionais especializadas. O software Rosetta B, 2019 por exemplo, foi uma das primeiras ferramentas amplamente utilizadas para predição de estruturas proteicas. Seu funcionamento baseia-se na minimização da energia livre de uma proteína, modelando interações moleculares fundamentais como interações de Van der Waals, forças eletrostáticas, ligações de hidrogênio e efeitos hidrofóbicos. A predição estrutural utilizando o Rosetta fundamenta-se no princípio de que a conformação nativa de uma proteína corresponde a conformação de menor energia

livre. O processo é iterativo e inicia-se com a geração de modelos estruturais iniciais, frequentemente por meio da montagem de fragmentos, em que pequenos segmentos estruturais de proteínas conhecidas são recombinados para gerar conformações aproximadas da proteína alvo. Para cada modelo gerado é estimado a energia livre da conformação. A partir dessas estimativas, o Rosetta aplica métodos estocásticos, como Monte Carlo Simulated Annealing, para explorar diferentes conformações e minimizar a função de energia, conduzindo a um refinamento progressivo da estrutura modelada. Os modelos que apresentam menores valores são considerados candidatos mais prováveis para representar a estrutura nativa da proteína em estudo.

Jumper, 2021 introduziu o *AlphaFold*, um modelo que representou um avanço significativo na predição estrutural de proteínas, ao introduzir uma abordagem baseada em *deep learning* que superou métodos computacionais tradicionais. O problema da predição estrutural pode ser formalmente descrito como a minimização da função de energia livre associada à conformação da proteína. Diferentemente dos métodos tradicionais que exploram o espaço conformacional por meio de amostragem estocástica, o *AlphaFold* aprende diretamente um mapa de distâncias e ângulos que minimiza essa função de energia. O modelo utiliza redes neurais treinadas em bancos de dados contendo estruturas proteicas determinadas experimentalmente, como as obtidas por cristalografia de raios X, espectroscopia por ressonância magnética nuclear (RMN) e criomicroscopia eletrônica. A base teórica do *AlphaFold* fundamenta-se na modelagem das interações residuais a partir de alinhamentos múltiplos de sequências (MSA, Multiple Sequence Alignments) e do aprendizado das representações espaciais dos resíduos por meio de camadas de atenção baseadas em grafos. O modelo recebe como entrada uma sequência de aminoácidos e gera um conjunto de representações internas (*embeddings*) que capturam informações sobre a proximidade e orientação relativa dos resíduos na estrutura tridimensional da proteína. O modelo foi amplamente validado por meio de sua participação na edição 14 do *CASP (Critical Assessment of Structure Prediction)*, uma avaliação que compara métodos de predição de estrutura proteica com dados experimentais inéditos. No CASP14, o *AlphaFold* demonstrou um desempenho quase equivalente ao de técnicas experimentais. Esse avanço revolucionou o campo da biologia estrutural, reduzindo a dependência exclusiva de métodos experimentais e abrindo novas possibilidades para a engenharia de proteínas e a compreensão de doenças associadas a dobramentos proteicos anômalos.

Z. L. e. al, 2022 apresentaram o *ESMFold*, um modelo baseado em *transformers* capaz de realizar a predição de estruturas proteicas diretamente a partir de sequências de aminoácidos. Diferentemente de modelos como o *AlphaFold*, que fazem uso extensivo de alinhamentos múltiplos de sequências (*MSA, Multiple Sequence Alignments*) e atenção baseada em grafos para inferir interações residuais, o *ESMFold* prioriza a velocidade de processamento, permitindo uma predição rápida de estruturas com precisão moderada. A arquitetura do *ESMFold* baseia-se no modelo de linguagem proteica *ESM-2*, um *transformer* pré-treinado em grandes bancos de dados de sequências biológicas. Esse modelo aprende representações internas que capturam informações evolutivas e estruturais implícitas, sem depender explicitamente de *MSAs*. A predição estrutural ocorre por meio da inferência direta da topologia tridimensional da proteína a partir dessas representações, minimizando a necessidade de computação intensiva e permitindo um tempo de inferência reduzido. Embora o *ESMFold* não alcance a precisão estrutural do *AlphaFold*, sua abordagem é

extremamente vantajosa para aplicações que exigem triagem de alto desempenho, como análises em larga escala e estudos preliminares de modelagem proteica. Sua rapidez permite que estruturas sejam geradas em segundos, enquanto o *AlphaFold* pode demandar minutos a horas, dependendo da complexidade da proteína analisada.

Ferramentas como o *ProteinMPNN* J. D. e. al, 2022 utilizam arquiteturas baseadas em *Message Passing Neural Networks* (MPNNs) para prever sequências de aminoácidos capazes de adotar conformações tridimensionais específicas. Diferentemente dos métodos tradicionais de *design* de proteínas, que frequentemente utilizam abordagens estocásticas ou heurísticas para otimizar sequências, como o *Rosetta*, o *ProteinMPNN* emprega *deep learning* para explorar de maneira eficiente o espaço de sequências e gerar variantes estruturalmente viáveis. O modelo opera sobre a representação estrutural de uma proteína como um grafo, onde os resíduos de aminoácidos correspondem a nós e as interações entre eles são representadas por arestas ponderadas. A passagem de mensagens ocorre entre os nós vizinhos, permitindo que o modelo aprenda padrões de interações residuais que favorecem a estabilidade estrutural da proteína. O modelo consiste de 3 camadas de *encoder* e 3 de *decoder* seguidos de uma camada escondida de 128 neurônios prevendo a sequência de aminoácidos de modo auto regressivo, do terminal N ao C.

Angermueller C, 2019 apresentaram o *DyNA PPO*, que consiste em uma abordagem de *desing* de sequências proteicas baseada em aprendizado por reforço profundo (*Deep Reinforcement Learning*, DRL), atualizando seus parâmetros através do algorítimo *Proximal Policy Optimization*, PPO. O *DyNA PPO* modela o problema do design de proteínas como um processo de decisão de Markov, onde um agente gera sequências de aminoácidos de forma autoregressiva, adicionando resíduos um por um. A função de recompensa é baseada em métricas de estabilidade estrutural e funcionalidade da proteína, sendo calculada apenas ao final da geração da sequência. Esse mecanismo permite que o modelo aprenda estratégias eficazes para explorar o espaço de sequências de forma eficiente. O modelo é treinado iterativamente, ajustando sua política para maximizar a recompensa acumulada. Durante o treinamento, um conjunto de redes neurais auxiliares avalia a qualidade das sequências geradas, fornecendo sinais de aprendizado ao agente. Embora o *DyNA PPO* tenha demonstrado resultados promissores em benchmarks computacionais, sua validação experimental ainda está em aberto.

A análise da literatura existente evidencia que o design computacional de proteínas tem avançado significativamente, impulsionado por metodologias como modelagem molecular baseada em física, aprendizado profundo e aprendizado por reforço. Modelos como *AlphaFold*, *Rosetta*, *ESMFold* e *ProteinMPNN* têm contribuído de forma expressiva para a predição estrutural e o *design* de sequências proteicas. Além disso, técnicas de aprendizado por reforço, como *DyNA PPO*, demonstraram grande potencial para a otimização de sequências proteicas, explorando o espaço conformacional de maneira adaptativa e eficiente.

No entanto, não foram encontrados estudos que apliquem aprendizado por reforço especificamente para a otimização do FIX no tratamento da hemofilia B. A pesquisa realizada não identificou publicações que abordem a aplicação dessa abordagem para modificar ou aprimorar a sequência do FIX, de modo a aumentar sua estabilidade, prolongar sua meia-vida ou reduzir sua imunogenicidade como proposto pelo presente trabalho.

Capítulo 2

Fundamentação teórica

Neste capítulo vamos apresentar os fundamentos teóricos utilizados neste trabalho acerca de proteínas e aprendizado por reforço profundo.

2.1 Proteína

Uma proteína é uma macromolécula biológica composta por cadeias de aminoácidos. Constituem a maior parte da massa seca de uma célula, podendo desempenhar funções enzimáticas, estruturais, imunológicas, de transporte, entre outras B. Alberts, 2002.

Existem milhares de proteínas diferentes em uma célula, onde cada uma é composta por uma sequência distinta dos 20 tipos de aminoácidos encontrados na natureza, unidos através de ligações peptídicas. Por conta disto, as proteínas são também conhecidas como polipeptídeos B. Alberts, 2002.

A cadeia polipeptídica é composta por três componentes: a cadeia principal, as cadeias laterais e as ligações peptídicas. A cadeia principal é também referida como o esqueleto da proteína e é definida como uma série repetitiva e encadeada de átomos de carbono, nitrogénio e oxigénio. Anexadas a cadeia principal, encontram-se as cadeias laterais. Estas não estão envolvidas nas ligações peptídicas e são responsáveis por caracterizar as principais propriedades da proteína B. Alberts, 2002.

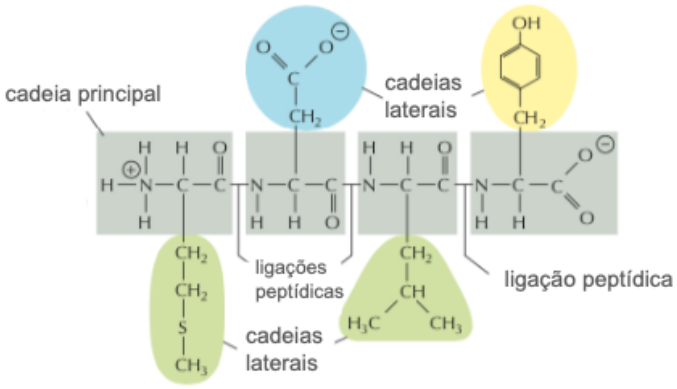


Figura 2.1: *Componentes de uma proteína*

Aminoácido			Cadeia lateral	Aminoácido			Cadeia lateral
Aspartic acid	Asp	D	negative	Alanine	Ala	A	nonpolar
Glutamic acid	Glu	E	negative	Glycine	Gly	G	nonpolar
Arginine	Arg	R	positive	Valine	Val	V	nonpolar
Lysine	Lys	K	positive	Leucine	Leu	L	nonpolar
Histidine	His	H	positive	Isoleucine	Ile	I	nonpolar
Asparagine	Asn	N	uncharged polar	Proline	Pro	P	nonpolar
Glutamine	Gln	Q	uncharged polar	Phenylalanine	Phe	F	nonpolar
Serine	Ser	S	uncharged polar	Methionine	Met	M	nonpolar
Threonine	Thr	T	uncharged polar	Tryptophan	Trp	W	nonpolar
Tyrosine	Tyr	Y	uncharged polar	Cysteine	Cys	C	nonpolar
Aminoácidos polares				Aminoácidos apolares			

Figura 2.2: *Cada aminoácido tem uma abreviação de uma ou três letras - figura 3-2 adaptada B. Alberts, 2002.*

2.1.1 Estrutura proteíca

A função de uma proteína está intimamente relacionada à sua estrutura tridimensional, que é formada principalmente a partir das interações entre as cadeias laterais e móleculas encontradas no meio físico ao qual a proteína está inserida. Devido a essas interações, a maioria das proteínas assume uma estrutura tridimensional única, que é determinada pela ordem dos aminoácidos na sequência. A conformação final tende a ser aquela que possui a menor energia livre B. Alberts, 2002. Por energia livre entende-se como à energia potencial total do sistema molecular que inclui a proteína e seu meio circundante. Essa energia é uma medida termodinâmica que combina componentes de energia entálpica (relacionada a ligações químicas e interações eletrostáticas) e entropia (relacionada à desordem ou ao número de configurações possíveis das moléculas).

B. Alberts, 2002 explica que a polaridade das cadeias laterais desempenham um papel importante na predição estrutural da proteína. As cadeias não polares (hidrofóbicas) tendem a se agrupar no interior da molécula, evitando contato com a água. Já as polares tendem a se dispor perto da superfície da molécula, onde podem formar ligações de hidrogênio com a água e outras moléculas polares.

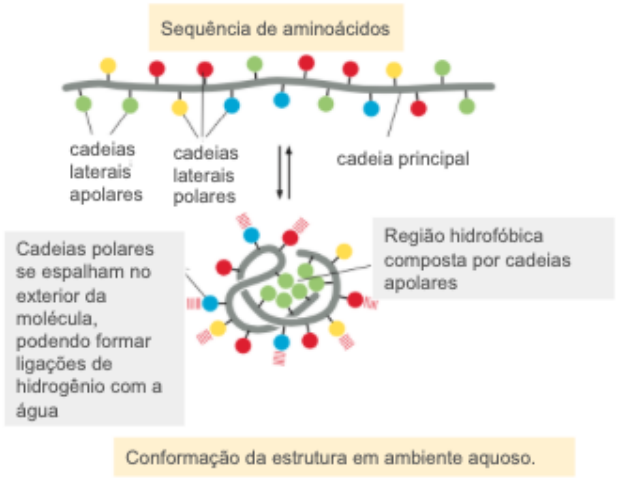


Figura 2.3: *Exemplo de conformação protéica B. Alberts, 2002 - adaptado*

2.1.2 O Fator IX

O Fator IX (FIX) é uma proteína essencial para o processo de coagulação. Quando ativado (FIXa), atua em conjunto com o Fator VIIIa (FVIIIa) para formar um complexo conhecido como tenase intrínseco, que é responsável por ativar o Fator X. Este processo é fundamental para a geração de trombina e formação de coágulos M. H. B. e. al, 2024. A ausência ou disfunção do FIX compromete o processo de ativação do FX, impossibilitando uma coagulação saúdavel, o que configura a hemofilia tipo B.

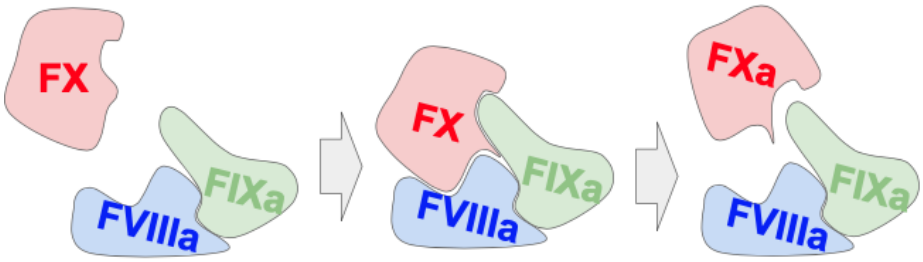


Figura 2.4: *Atuação do FIXa na ativação do FX*

O FIXa é composta por diferentes domínios, cada um desempenhando um papel essencial nas interações moleculares que permitem a formação do complexo tenase intrínseco. Na região N-terminal, o domínio Gla (gamma-carboxiglutâmico) é responsável por ligar o FIXa a superfícies de membranas fosfolipídicas, uma interação dependente de íons de cálcio, fundamental para posicionar o complexo de coagulação no local da lesão vascular M. H. B. e. al, 2024. Os domínios de fator de crescimento epidérmico (EGF1 e EGF2) desempenham funções complementares: o EGF1 facilita o reconhecimento de cofatores e substratos, enquanto o EGF2 interage com o domínio A3 do Fator VIIIa, contribuindo para a montagem do complexo tenase M. H. B. e. al, 2024. O papel mais crítico para a estabilidade e a funcionalidade do complexo tenase reside no domínio protease, cuja interação com

o domínio A2 do FVIIIa permite a ativação conformacional necessária para uma catálise eficiente do FX. M. H. B. e. AL, 2024 explica que variantes do FIXa, como o FIX-Padua, que modificam o domínio protease, apresentam maior eficiência catalítica, destacando seu papel central no projeto de proteínas otimizadas para terapia de hemofilia.

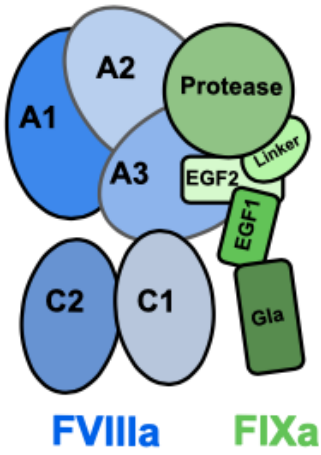


Figura 2.5: *Interação entre FIXa e FVIIIa, formando o complexo tenase. Adaptação da figura 3 de M. H. B. e. AL, 2024*

A estrutura do domínio protease é a estrutura alvo do projeto de sequências proposto por este trabalho. É composta por 235 resíduos. Logo, o espaço de possíveis novas sequências com este tamanho, considerando 20 diferentes aminoácidos, é de 235^20.

Figura 2.6: *Estrutura alvo - Domínio protease do Fator IXa*

As proteínas projetadas neste estudo serão desenvolvidas para mimetizar a estrutura do domínio protease. O objetivo é identificar, entre as variantes geradas, aquelas capazes de estabelecer interações mais estáveis com o FVIIIa, aprimorando a estabilidade do complexo tenase intrínseco. Adicionalmente, espera-se que as novas estruturas apresentem um perfil

imunogênico reduzido em comparação ao FIXa nativo, contribuindo para maior segurança e eficácia terapêutica.

2.1.3 Docking

A análise de docking é uma técnica computacional amplamente utilizada no estudo de interações biomoleculares, particularmente para prever a orientação e a afinidade entre duas moléculas, como proteínas. Nesse sentido, é essencial para avaliar como uma proteína substituta do FIXa interage com o FVIIIa. O docking molecular simula o reconhecimento entre uma molécula receptora e outra ligante. No caso de interações proteína-proteína, o receptor e o ligante são ambos macromoléculas, cujas superfícies e conformações influenciam a formação de um complexo estável. O objetivo do docking é identificar as poses mais favoráveis termodinamicamente, ou seja, aquelas que minimizam a energia livre do sistema.

Durante o docking, diversas poses do ligante em relação ao receptor são geradas. Ferramentas como RosettaDock ou AutoDock realizam essa busca, explorando translações, rotações e flexibilidade local. Cada pose é avaliada com base em métricas energéticas, como a *Contact Molecular Surface* (CMS), *Interface Buried SASA* (IBSASA), *Delta Gibbs Free Energy of Binding* (DDG) e *Surface Area Potential Score* (SAP Score).

A CMS é uma medida desenvolvida para superar as limitações de métricas tradicionais, como Shape Complementarity (SC) e Delta Solvent Accessible Surface Area (Delta SASA) Coventry, 2021. Enquanto SC analisa a complementaridade geométrica entre superfícies moleculares e Delta SASA calcula a área de superfície acessível ao solvente antes e depois da complexação, ambas possuem limitações na identificação de interfaces bem empacotadas. Em contraste, o CMS combina a área de contato com um fator de penalização baseado na distância entre triângulos de superfície, oferecendo uma representação mais precisa da qualidade do empacotamento na interface Coventry, 2021.

A IBSASA é frequentemente usada para estimar a extensão de área de superfície enterrada durante a formação de um complexo proteico, correlacionando-se com a estabilidade de ligação. No entanto, sozinha, essa métrica pode ser enganosa em interfaces mal organizadas, onde grandes áreas de contato podem não resultar em interações eficazes Coventry, 2021.

Por outro lado, a DDG fornece uma medida direta do quão favorável energéticamente é uma interação, com valores mais negativos indicando maior estabilidade termodinâmica Coventry, 2021. A DDG, calculada frequentemente com o software Rosetta, permite a otimização de sequências e conformações para maximizar a energia de ligação.

O SAP Score avalia a hidrofobicidade com base na superfície acessível ao solvente, sendo uma ferramenta crucial para prever a propensão à agregação e a estabilidade de complexos Coventry, 2021. Alterações no SAP Score após a ligação, conhecidas como Delta SAP Score, fornecem insights sobre a contribuição hidrofóbica para a força motriz da interação.

2.1.4 Resposta Imunológica

A resposta imune desencadeada por uma proteína terapêutica é um fator crítico que pode influenciar diretamente sua segurança e eficácia. No contexto do projeto de uma proteína substituta do FIXa, a análise dessa resposta imune é especialmente relevante para evitar complicações associadas à formação de inibidores, uma limitação comum no tratamento de hemofilia. Esses inibidores são anticorpos neutralizantes que se ligam à proteína administrada, reduzindo sua eficácia e exigindo estratégias de tratamento alternativas mais caras e menos seguras.

A imunogenicidade está intrinsicamente relacionada à presença de epítopos imunogênicos na sequência da proteína. Esses epítopos são regiões específicas que se ligam a moléculas de MHC classe II em células que apresentam antígenos (APCs), facilitando a ativação de linfócitos T auxiliares Bjoern Peters e Sette, 2020. A afinidade de ligação é um fator determinante na indução da resposta imune. Altas afinidades entre peptídeos e MHC-II aumentam a probabilidade de reconhecimento imune e subsequente ativação de células T Bjoern Peters e Sette, 2020.

A predição de afinidade de ligação entre epítopos e MHC-II emprega ferramentas computacionais baseadas em aprendizado de máquina, como o NetMHCIIpan, que simulam interações entre peptídeos derivados da proteína e uma variedade de alelos de MHC. A sequência da proteína é fragmentada em peptídeos sobrepostos com comprimentos típicos de 8 a 15 aminoácidos, e a afinidade de cada peptídeo é quantificada por meio do valor de IC_{50} (concentração inibitória de 50%). Peptídeos com valores de IC_{50} baixos (($IC_{50} < 50$ nM)) indicam maior afinidade de ligação e, portanto, são mais propensos a serem reconhecidos como epítopos imunogênicos Bjoern Peters e Sette, 2020.

Portanto, o objetivo ao projetar a proteína substituta do FIXa é não apenas otimizar a estabilidade de ligação ao FVIIIa, mas também evitar a apresentação de novos epítopos imunogênicos que poderiam levar à formação de inibidores. A redução da imunogenicidade melhora a adesão ao tratamento e reduz complicações, consolidando a importância da integração de predições imunológicas no pipeline de projeto de proteínas terapêuticas.

2.1.5 Mutação

Uma mutação no contexto de proteínas refere-se a uma modificação na sequência de aminoácidos. Por menor que seja esta modificação, a mutação pode acarretar em uma alteração na estrutura tridimensional da proteína, uma vez que a ordem que os aminoácidos estão dispostos na sequência influencia diretamente na conformação da estrutura. Neste trabalho, vamos assumir que uma mutação consiste na substituição de apenas um aminoácido por outro na sequência.

2.1.6 Similariedade entre estruturas

Para projetar proteínas semelhantes ao FIXa, é necessário definir uma métrica de similaridade entre estruturas. Duas métricas amplamente utilizadas são o *Root-Mean-Square Deviation* (RMSD) e o *Template Modeling Score* (TM-score) Zhang, 2004, cada uma com características próprias que influenciam sua adequação a diferentes cenários de

comparação estrutural.

O **RMSD** é definido como a raiz quadrada da média dos quadrados das distâncias entre pares de átomos correspondentes em duas estruturas alinhadas:

$$\text{RMSD} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} d_i^2}, \tag{2.1}$$

onde N representa o número de pares de átomos alinhados e d_i é a distância entre os i-ésimos átomos nas estruturas comparadas. Embora seja uma métrica intuitiva, o RMSD é altamente sensível a desalinhamentos locais e não é normalizado pelo comprimento da proteína, o que limita sua eficácia ao comparar proteínas de diferentes tamanhos.

O **TMScore**, por sua vez, mede a similaridade estrutural global, superando limitações do RMSD. Zhang, 2004 a formulou da seguinte maneira:

$$\text{TM-score} = \frac{1}{L_{\text{ref}}}\sum_{i=1}^{L_{\text{model}}} \frac{1}{1+\left(\frac{d_i}{d_0(L_{\text{ref}})}\right)^2}, \tag{2.2}$$

onde L_{ref} é o número de resíduos na estrutura de referência, L_{model} é o número de resíduos no modelo comparado, e d_i é a distância entre os átomos C_α alinhados.

Zhang, 2004 define d_0 em função de L_{ref} da seguinte maneira:

$$d_0(L_{\text{ref}}) = 1.24 \times (L_{\text{ref}} - 15)^{1/3} - 1.8. \tag{2.3}$$

Este parâmetro adapta a escala de distâncias consideradas significativas para o alinhamento estrutural, ajustando-se ao comprimento da proteína de referência, L_{ref}. O termo $(L_{\text{ref}}-15)^{1/3}$ reflete uma relação empírica entre o tamanho da proteína e as distâncias típicas entre resíduos alinhados, modelando o crescimento da distância permissível conforme a proteína aumenta de tamanho. A constante multiplicativa 1.24 e o deslocamento -1.8 também foram determinados empiricamente para fornecer um valor de d_0 que otimiza a correlação entre o TM-score e a percepção biológica de similaridade estrutural. Essa função adapta o limiar de normalização das distâncias entre pares de resíduos alinhados, garantindo que proteínas de diferentes comprimentos sejam comparadas de forma justa. Proteínas maiores toleram distâncias absolutas maiores entre resíduos alinhados sem que o TM-score seja severamente penalizado Zhang, 2004.

O TMScore varia entre 0 e 1, onde valores próximos a 1 indicam alta similaridade topológica Zhang, 2004. Se a distância entre cada par alinhado de átomos C_α for igual a 0, então o TMScore é igual a 1, indicando sobreposição completa entre as macromoléculas. Em contra partida, quanto maiores são as distâncias, menor é o TMScore (mais próximo de 0).

Devido à sua robustez na avaliação da similaridade estrutural global e à sua interpretabilidade, com valores limitados entre 0 e 1, adotaremos o TM-score como a métrica principal

a ser otimizada neste projeto. O cálculo desta medida será feito a partir da plataforma desenvolvida por Zhang, 2022.

2.1.7 Conservation Score

O *Conservation Score* (CS) é uma métrica que estima a importância ou contribuição individual de cada resíduo da sequência na caracterização da estrutura da proteína. Resíduos cruciais na estrutura da proteína tendem a ser conservados, visto que alterações neles podem impactar significativamente a função proteíca. O score é obtido levando-se em conta a frequência com que certas combinações de aminoácidos ocorrem em proteínas homólogas ao longo da evolução das espécies Eddy, 2004. Para este trabalho, utilizamos o CS calculado pelo servidor ConsurfDB Ben Chorin, 2020, tendo a estrutura do FIXa de coagulação como entrada.

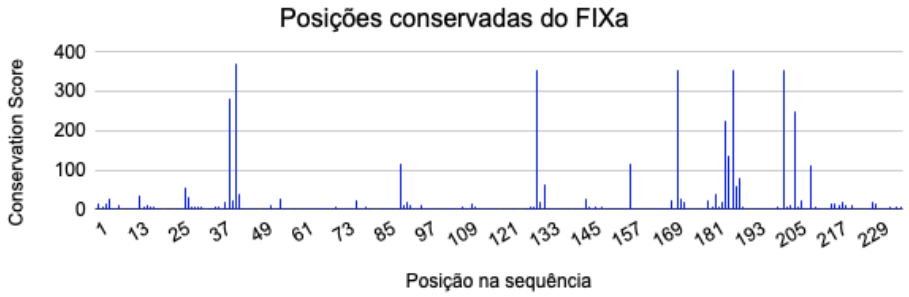


Figura 2.7: *A maior parte dos resíduos do FIXa possuem um baixo CS. Os maiores valores estão concentrados em cerca de 12 resíduos espalhados ao longo da sequência.*

A identificação de posições conservadas é um aspecto fundamental ao projetar novas proteínas, pois fornece informações essenciais para evitar mutações em regiões críticas, preservando assim a estabilidade estrutural e a função biológica da molécula.

2.1.8 Similariedade entre aminoácidos

Da mesma forma que regiões com altos score de conservação devem ser evitadas, convém evitar mutações que substituem aminoácidos muito diferentes. Nesse sentido, para mensurar o nível de similaridade entre os aminoácidos, utilizamos uma métrica de distância fundamentada no trabalho de Lopes Tiago J. S., 2021, que propõe uma abordagem vetorial para representar as propriedades de cada aminoácido. Nesse estudo, cada aminoácido foi descrito por um vetor de 544 dimensões, incorporando uma ampla gama de características físico-químicas e estruturais. A construção desses vetores foi realizada com o auxílio do pacote seqinR Charif, 2007, uma ferramenta especializada em análise de sequências biológicas. O pacote seqinR utiliza uma base de dados abrangente contendo propriedades como peso molecular, hidrofobicidade, polaridade, acessibilidade à superfície, entre outras, para derivar descritores numéricos que caracterizam as propriedades de cada aminoácido.

A partir dessa representação, Lopes Tiago J. S., 2021 introduziu uma matriz de distâncias, construída a partir das distâncias euclidianas de cada par de vetores, fornecendo uma

medida quantitativa da dissimilaridade entre aminoácidos.

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
A	0	1.5	1.57	1.51	1.52	1.59	1.51	1.52	1.52	1.46	1.48	1.51	1.49	1.47	1.6	1.38	1.49	1.44	1.55	1.59
C	1.5	0	1.6	1.54	1.48	1.68	1.51	1.46	1.61	1.49	1.48	1.59	1.34	1.5	1.61	1.53	1.54	1.42	1.57	1.56
D	1.57	1.6	0	1.48	1.65	1.61	1.55	1.62	1.59	1.67	1.65	1.51	1.46	1.53	1.63	1.52	1.58	1.6	1.72	1.62
E	1.51	1.54	1.48	0	1.62	1.63	1.53	1.6	1.58	1.63	1.58	1.5	1.51	1.47	1.61	1.56	1.58	1.57	1.7	1.63
F	1.52	1.48	1.65	1.62	0	1.63	1.48	1.39	1.61	1.45	1.48	1.56	1.6	1.54	1.61	1.58	1.52	1.42	1.53	1.49
G	1.59	1.68	1.61	1.63	1.63	0	1.58	1.6	1.57	1.65	1.68	1.53	1.47	1.57	1.59	1.54	1.62	1.57	1.77	1.59
H	1.51	1.51	1.55	1.53	1.48	1.58	0	1.49	1.51	1.52	1.46	1.48	1.44	1.46	1.52	1.51	1.54	1.47	1.62	1.5
I	1.52	1.46	1.62	1.6	1.39	1.6	1.49	0	1.58	1.42	1.42	1.5	1.58	1.51	1.57	1.57	1.53	1.14	1.55	1.45
K	1.52	1.61	1.59	1.58	1.61	1.57	1.51	1.58	0	1.59	1.61	1.48	1.54	1.45	1.49	1.44	1.49	1.55	1.65	1.56
L	1.46	1.49	1.67	1.63	1.45	1.65	1.52	1.42	1.59	0	1.5	1.58	1.64	1.56	1.62	1.47	1.52	1.43	1.53	1.54
M	1.48	1.48	1.65	1.58	1.48	1.68	1.46	1.42	1.61	1.5	0	1.62	1.55	1.52	1.65	1.6	1.6	1.48	1.57	1.57
N	1.51	1.59	1.51	1.5	1.56	1.53	1.48	1.5	1.48	1.58	1.62	0	1.4	1.45	1.52	1.35	1.49	1.56	1.62	1.54
P	1.49	1.34	1.46	1.51	1.6	1.47	1.44	1.58	1.54	1.64	1.55	1.4	0	1.45	1.57	1.38	1.46	1.55	1.65	1.58
Q	1.47	1.5	1.53	1.47	1.54	1.57	1.46	1.51	1.45	1.56	1.52	1.45	1.45	0	1.52	1.43	1.44	1.49	1.59	1.53
R	1.6	1.61	1.63	1.61	1.61	1.59	1.52	1.57	1.49	1.62	1.65	1.52	1.57	1.52	0	1.53	1.56	1.51	1.73	1.55
S	1.38	1.53	1.52	1.56	1.58	1.54	1.51	1.57	1.44	1.47	1.6	1.35	1.38	1.43	1.53	0	1.21	1.49	1.55	1.5
T	1.49	1.54	1.58	1.58	1.52	1.62	1.54	1.53	1.49	1.52	1.6	1.49	1.46	1.44	1.56	1.21	0	1.43	1.53	1.56
V	1.44	1.42	1.6	1.57	1.42	1.57	1.47	1.14	1.55	1.43	1.48	1.56	1.55	1.49	1.51	1.49	1.43	0	1.59	1.4
W	1.55	1.57	1.72	1.7	1.53	1.77	1.62	1.55	1.65	1.53	1.57	1.62	1.65	1.59	1.73	1.55	1.53	1.59	0	1.59
Y	1.59	1.56	1.62	1.63	1.49	1.59	1.5	1.45	1.56	1.54	1.57	1.54	1.58	1.53	1.55	1.5	1.56	1.4	1.59	0

Figura 2.8: *Distâncias entre pares de aminoácidos* Lopes Tiago J. S., 2021

2.2 Aprendizado por reforço profundo

Nesta seção vamos introduzir os conceitos envolvidos no aprendizado por reforço profundo, que formam a base teórica da técnica que iremos utilizar: Proximal Policy Optimization (PPO). Para tal, será fornecido um contexto sobre Processo de Decisão de Markov (MDP), Redes Neurais (NN) e o método Ator-Crítico (AC).

2.2.1 Processo de Decisão de Markov

O Processo de Decisão de Markov (MDP) é utilizado para modelar processos de forma probabilística. Ele é chamado de markoviano pois a distribuição de probabilidade de um estado depende apenas do estado anterior e da ação selecionada. Uma ação dá origem a um novo estado ao passo que promove alterações no estado atual. Um Agente seleciona cada ação com base em uma política (π) que mapeia estados a ações. Em um MDP, o processo evolui em etapas, ou *steps*, à medida que o agente toma ações. A sequência de estados, desde o inicial até o final, é chamada de episódio. A qualidade de cada ação é avaliada com base no quão benéfico é o estado alcançado. De acordo com Pellegrini, 2007, o MDP pode ser definido como uma tupla <S,A,T,R> onde:

- S é o conjunto de possíveis estados;
- A é o conjunto de ações que interferem no processo;
- T :S×A×S→[0,1] é uma função que quantifica a probabilidade de mudança do estado s ∈ S para o estado s' ∈ S, dado a ação a ∈ A selecionada. É representado por T (s' |s, a);

- r : S × A→r é uma função que mede a recompensa por selecionar a ação a ∈ A quando o processo está no estado s ∈ S.

De modo a encontrar um bom π, é necessário definir uma maneira de comparar diferentes políticas. Uma técnica comum é comparar a esperança da recompensa acumulada descontada, ou Valor de Retorno (R), que cada política produziu em um mesmo episódio.

$$R = E\left[\sum_{k=1}^{z} \gamma^{k-1} r_k\right] \tag{2.4}$$

Sendo z o tamanho do episódio, k o índice do *step* e o r_k a recompensa obtida em k. A constante γ é um valor entre 0 e 1, chamado de fator de desconto. Ele é responsável por atribuir um peso maior a recompensas obtidas no início do episódio.

Além disso, existem outras duas métricas importantes para se avaliar a qualidade de uma política: a Função de Valor (V) e a função de Valor da Ação (Q). A primeira estima a vantagem de estar em um estado s, calculando a esperança de R dado o estado inicial s0 = s.

$$V_\pi(s) = E[R|s_0 = s] \tag{2.5}$$

A segunda estima a vantagem de selecionar a ação a estando no estado s. Em outras palavras, é definida como a esperança de R dado o estado inicial e a ação inicial:

$$Q_\pi(s, a) = E[R|s_0 = s, a_0 = a] \tag{2.6}$$

Neste trabalho vamos modelar tanto a política π quanto a Função de Valor V utilizando redes neurais profundas.

2.2.2 Redes Neurais

Inspirado no sistema neural humano, uma rede neural artificial consiste em um conjunto de nós, chamados de neurônios, interconectados por arestas Bishop, 2006. Em geral, é organizada em camadas, que incluem três tipos: a camada de entrada, a camada oculta e a camada de saída, conforme ilustrado na figura 2.9.

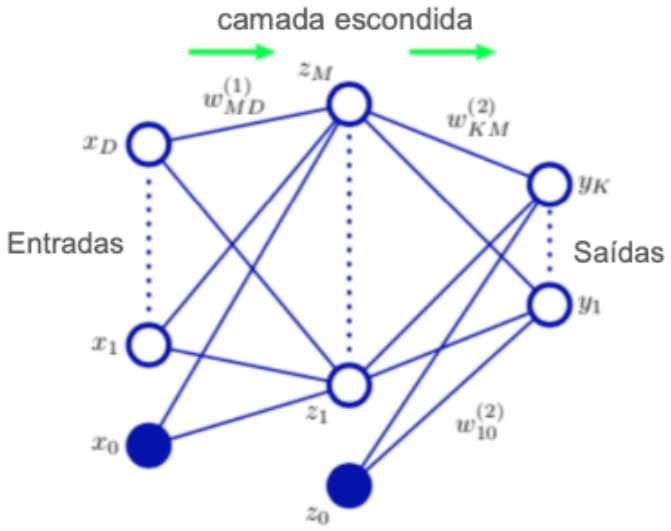


Figura 2.9: *Arquitetura de uma rede neural - Bishop, 2006 adaptado*

Sinais fluem entre os neurônios através das arestas que estão associadas a um peso (w) que quantifica a sua importância. Nos neurônios é feito o processamento dos diversos sinais recebidos que, em seguida, aplica-se uma função, conhecida como função de ativação. Esta pode ser simplesmente uma combinação linear das entradas ponderadas pelos seus respectivos pesos, como pode ser função não linear como sigmoid, tangente hiperbólica, entre outras Bishop, 2006.

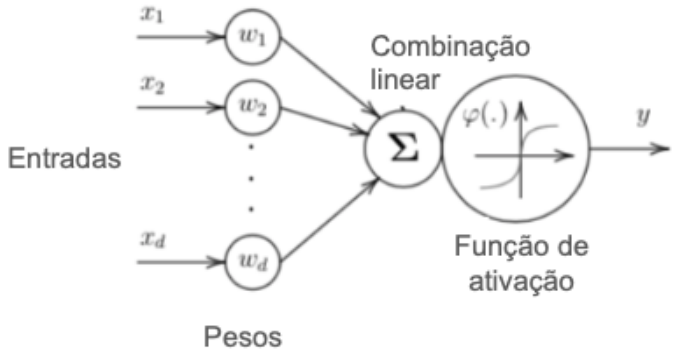


Figura 2.10: *Neurônio processando sinal Figueiredo, 2018*

Os pesos das arestas são ajustados durante a fase de treinamento, de modo a orientar a rede a produzir a saída esperada para uma entrada específica. Nesta fase, é aplicado uma função de perda baseado na saída da rede. Existem diversas técnicas possíveis para minimizar a função de perda. Dentre elas, uma das mais tradicionais é a baseada no gradiente descendente, que atualiza os pesos conforme a equação a seguir.

$$w(t+1) = w(t) - n\nabla L(w(t)) \tag{2.7}$$

Onde w é o vetor de pesos, n é uma constante positiva conhecida como taxa de aprendizagem, que quantifica o tamanho do ajuste dos parâmetros a cada iteração e L é a função de perda. Um dos algoritmo mais utilizados para calcular o gradiente da função de perda e atualizar os parâmetros da rede é o *Backpropagation* Bishop, 2006.

No contexto deste trabalho, as funções de perda estarão fortemente relacionadas ao valor de Retorno R e a função de Valor V apresentadas na seção anterior.

2.2.3 Método Actor-Critic

O método Ator-Crítico (AC) é um procedimento que tem como objetivo otimizar a política de um MDP Konda e Tsitsiklis, 2001. Ele é composto por dois componentes principais: o Ator e o Crítico. O Ator é aquele que seleciona as ações baseado nos estados, i.e, desempenha a função da política em um MDP. O Crítico, como o nome sugere, avalia a ação selecionada pelo Ator, estimando o vantagem de estar no estado alcançado. Assim, o Crítico é representado pela função de Valor do MDP. Tanto o Ator quanto o Crítico serão estimados por uma rede neural profunda cujos parâmetros serão ajustados utilizando a técnica *Proximal Policy Optimization* J. S. e. al, 2017.

2.2.4 PPO

A técnica *Proximal Policy Optimization* é um modelo de aprendizado por reforço profundo que define funções de perda para ajustar os pesos das redes neurais pertencentes à arquitetura do método AC J. S. e. al, 2017.

Uma vez que o Crítico objetiva estimar o valor de V, o ajuste dos seus parâmetros (W) é feito através de um gradiente descendente a fim de minimizar o quadrado da diferença entre a saída da rede ($\hat{V}(W)$) e o V observado (V^{obs}) J. S. e. al, 2017:

$$L_t^{VF}(W) = (\hat{V}_t(W) - {V_t}^{obs})^2 \tag{2.8}$$

Uma dos principais motivações do PPO é, durante o processo de otimização do MDP, evitar atualizações de política muito grandes. Para isto, J. S. e. al, 2017 força com que a razão entre a política atual e a política anterior ($r_t(\Theta)$) seja limitada a um intervalo conveniente.

$$r_t(\Theta) = \frac{\pi_\Theta}{\pi_{\Theta old}} \tag{2.9}$$

Desta forma, para ajustar os parâmetros (Θ) do Ator, J. S. e. al, 2017 define a seguinte função objetivo a ser maximizada por um gradiente ascendente:

$$L^{CLIP}(\Theta) = \hat{E}_t[min(r_t(\Theta)\hat{A}_t, clip(r_t(\Theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{2.10}$$

Onde ϵ é uma constante que o J. S. e. al, 2017 sugere ser igual a 0.2. $\hat{A}_t$ é definido como a estimativa da função de vantagem, que corresponde ao benefício de selecionar uma ação a no estado s, em comparação com uma seleção aleatória de uma ação no estado s. Em outras palavras, é a diferença entre Q e V:

$$A = Q(s,a) - V(s) \tag{2.11}$$

A função objetivo $L^{CLIP}(\Theta)$ é a esperança do mínimo de dois termos: $r_t(\Theta)\hat{A}_t$ e $clip(r_t(\Theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$. O primeiro direciona a política a selecionar ações que maximizam a função de vantagem. O segundo é uma versão truncada do primeiro, onde é aplicado um *clip* no $r_t(\Theta)$ de modo a garantir que a nova política não se distancie muito da política anterior.

Uma vez que o aprendizado das redes do AC são dependentes entre si, J. S. e. AL, 2017 sugere uma função objetivo única para treinar ambos:

$$L_t^{PPO}(\Theta) = \hat{E}_t[L_t^{CLIP}(\Theta) - c_1 L_t^{VF}(\Theta) + c_2 S(s_t)] \tag{2.12}$$

Onde c_1 e c_2 são constantes e S é definido por J. S. e. AL, 2017 como um bônus de entropia que garante que o agente explore suficientemente o ambiente durante o treinamento.

2.2.5 Escalonamento Multidimensional (MDS)

O uso de técnicas de redução de dimensionalidade, como o *Metric Multidimensional Scaling* (MDS), é motivado pela necessidade de construir representações vetoriais de objetos em um espaço de menor dimensão, preservando as distâncias definidas em uma matriz de similaridade ou dissimilaridade. No contexto deste trabalho, utilizamos o MDS para gerar uma representação vetorial para cada aminoácido, respeitando as distâncias previamente definidas na matriz proposta por LOPES TIAGO J. S., 2021. Esta abordagem permite incorporar relações complexas de similaridade em uma forma compatível com métodos de aprendizado de máquina e análise estrutural.

Dado um conjunto de n objetos e uma matriz de distâncias $D = (d_{ij})$, onde cada elemento d_{ij} representa a distância entre o par de objetos i e j, o objetivo do MDS é encontrar uma configuração vetorial $X_i = [x_{i1}, x_{i2}, ..., x_{im}]$ para cada objeto i em um espaço de dimensão m, de modo que a distância euclidiana entre X_i e X_j aproxime d_{ij}. O MDS busca minimizar as distâncias reais e as distâncias no espaço embutido, utilizando a função de perda conhecida como *stress*:

$$Stress(X_1, X_2, ..., X_n) = \sqrt{\sum_{i \neq j=1}^{n} (d_{ij} - \|X_i - X_j\|)^2}. \tag{2.13}$$

A minimização desta função é geralmente realizada pelo algoritmo SMACOF (*Scaling by MAjorizing a Complicated Function*), que iterativamente melhora a configuração de pontos em direção a uma solução de menor *stress*, garantindo uma melhor correspondência entre as distâncias calculadas e as observadas KRUSKAL, 1964. Embora outros métodos, como o gradiente descendente, possam ser utilizados, o SMACOF é amplamente preferido devido à sua eficiência e estabilidade.

Neste trabalho, a implementação do MDS foi realizada utilizando a biblioteca `sklearn`,

que fornece uma solução eficiente para problemas de escala multidimensional, baseada nas técnicas originalmente propostas por Kruskal, 1964.

Capítulo 3

Metodologia

Neste capítulo será detalhado os três módulos propostos na figura 4: Condições Iniciais, Treinamento e Geração de Sequências. Além disso, será apresentado os métodos de avaliação das proteínas geradas, bem como as especificações de hardware e software utilizados.

3.1 Condições iniciais

O objetivo do primeiro módulo (figura 3.1) é obter a sequência (S_{ini}) e o erro (Er_0) iniciais. A primeira é obtida ao se passar a estrutura alvo como entrada do *ProteinMPNN*. Já o erro inicial é definido pela seguinte expressão:

$$Er_0 = 1 - TMScore(E_{FIX}, E_{ini}) \tag{3.1}$$

Onde E_{FIX} é a estrutura alvo, i.e, a estrutura do fator IX de coagulação e E_{ini} é a estrutura vinculada a sequência inicial, predita pelo *ESMFold* - descrito em 1. O *TMScore* consiste em uma medida de similariedade entre estruturas, detalhada em 2.1.6

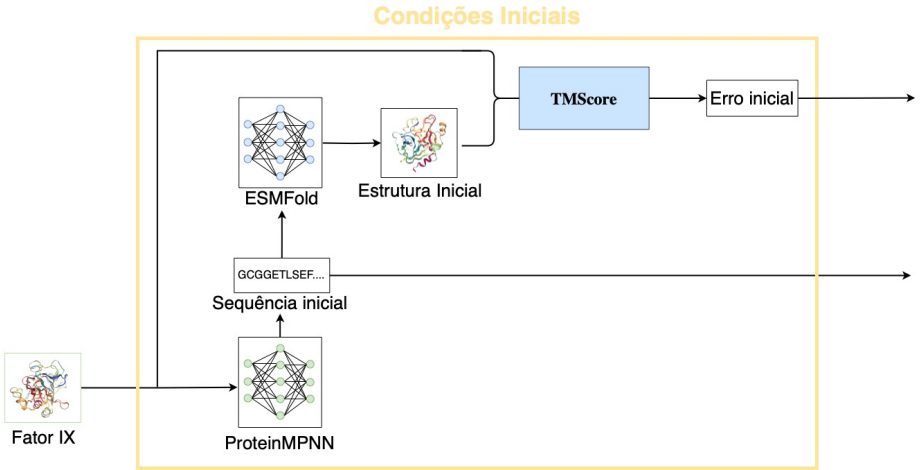


Figura 3.1: *Obtenção das condições iniciais*

Algorithm 1 Obtenção das Condições Iniciais

Require: Estrutura alvo E_{FIX}
Ensure: Sequência inicial S_{ini} e erro inicial Er_0
1: $S_{ini} \leftarrow ProteinMPNN(E_{FIX})$
2: $E_{ini} \leftarrow ESMFold(S_{ini})$
3: $Er_0 \leftarrow 1 - TMScore(E_{FIX}, E_{ini})$
4: **return** S_{ini}, Er_0

Esta implementação[1], bem como seu comando para execução[2] estão disponíveis. As parametrizações e os comandos para execução utilizadas no *ProteinMPNN*, *ESMFold* e *TMScore* são apresentadas a seguir:

Comando para execução - *ProteinMPNN*

```
python src/ProteinMPNN/protein_mpnn_run.py \
   --pdb_path {structure_path} \
   --pdb_path_chains {chains_to_design} \
   --out_folder tmp \
   --num_seq_per_target 1 \
   --sampling_temp "0.5" \
   --seed 37 \
   --batch_size 1
```

Parametrização - *ProteinMPNN*

1. **pdb_path**: `data/05_model_input/fixa_human_SP_renumber.pdb`. Caminho até o arquivo da estrutura do FIX.
2. **pdb_path_chains**: `"H"`. Cadeia proteica a ser predita.
3. **out_folder**: `"tmp"`. Nome do diretório temporário.
4. **num_seq_per_target**: `1`. Quantidade de sequências geradas.
5. **sampling_temp**: `0.5`. Temperatura de amostragem. Quanto menor, mais conservadora a predição.
6. **seed**: `37`. Semente aleatória para reprodutibilidade da predição.
7. **batch_size**: `1`. Tamanho do batch utilizado para inferência.

Comando para execução - *ESMFold*

```
curl -X POST --data "{initial_seq}" https://api.esmatlas.com/foldSequence/v1/
    pdb/ > tmp/pred_initial_structure.pdb
```

Parametrização - *ESMFold*

1. **data**: String contendo a sequência de aminoácidos. Ex: "AACYA...".

[1] Disponível em: https://github.com/ArthurPimenta0306/Hallucination/blob/main/hpd/src/hpd/pipelines/Modeling/nodes.py

[2] `kedro run mo_get_initial_conditions -nodes get_initial_conditions_B_node`

Comando para execução - *TMScore*

```
./USalign -outfmt 2 {path_structure_1} {path_structure_2}
```

Parametrização - *TMScore*

1. **path_structure_1**: Caminho até o arquivo .pdb da estrutura protéica referência.
2. **path_structure_2**: Caminho até o arquivo .pdb da segunda estrutura protéica a ser comparada.

3.2 Treinamento

O módulo de Treinamento foi construído em duas etapas: Estágios I e II, detalhados em 3.2.3 e 3.2.4 respectivamente. Para ambos os estágios, o treinamento se dá através de interações entre dois componentes: O Ambiente de aprendizado e o Agente.

3.2.1 Ambiente de aprendizado

O ambiente de aprendizado é modelado como um MDP e, portanto, é responsável por definir o espaço de ações, o espaço de estados, a função de transição entre estados dado uma ação e a função de recompensa.

Espaço de ações

Definimos uma ação como um vetor bi-dimensional composto pelo índice p de uma posição na sequência e um aminoácido r. A ação será utilizada para realizar uma mutação na proteína, inserindo o r na posição p. De modo a incentivar o Agente a explorar diferentes ações, definimos, para cada iteração, um conjunto de posições e aminoácidos válidos: P_{val} e R_{val} respectivamente. P_{val} é composto por todas as possíveis posições na sequência, com exceção das posições previamente selecionadas dentro do mesmo episódio. Definimos R_{val} sendo igual aos 20 possíveis aminoácidos, com exceção dos aminoácidos que foram selecionados mais do que N_{rep} vezes. Introduzimos N_{rep} como um hiperparâmetro que define a quantidade máxima permitida que um mesmo resíduo pode ser selecionado dentro de um episódio. Desta forma, o espaço de ações, para cada iteração i, é definido por:

$$A_i = \left\{(p, r) | p \in P_{val_i}, r \in R_{val_i}\right\} \tag{3.2}$$

Espaço de estados

Definimos o espaço de estados como o conjunto de todas as possíveis sequências de n aminoácidos, onde n é o tamanho da sequência inicial:

$$S = \{(r_1, r_2, r_3, ..., r_n) | r_i \in R_{encoded} \forall i \in [1, n]\} \tag{3.3}$$

Onde $R_{encoded}$ é o conjunto dos 20 possíveis aminoácidos, cada um representado por um vetor codificado pelo *Encoder*. O *Encoder* consiste em um dicionário que mapeia cada

aminoácido para uma representação vetorial que conserva as distâncias entre cada par de aminoácidos estabelecidas por Lopes Tiago J. S., 2021. Este mapeamento foi construído através de um MDS[3] utilizando 21 dimensões.

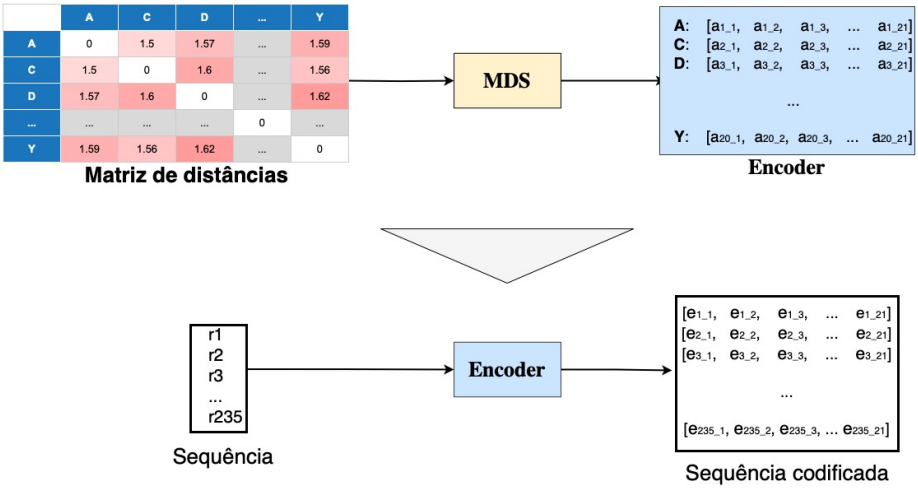


Figura 3.2: *Construção e uso do Encoder.*

Para escolher a quantidade de dimensões a se utilizar no MDS, definimos o ponto a partir do qual a curva[4] entre *Stress* e dimensionalidade tende a se aproximar assintóticamente do eixo x do plano cartesiano. Este ponto, conforme a figura 3.3, é o que possui 21 dimensões.

[3] Implementação do MDS construída pelo pacote scikit learn, disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html

[4] Disponível na seção "Encoding Aminoácidos"em https://github.com/ArthurPimenta0306/Hallucination/blob/main/hpd/notebooks/GeneralAnalysis.ipynb

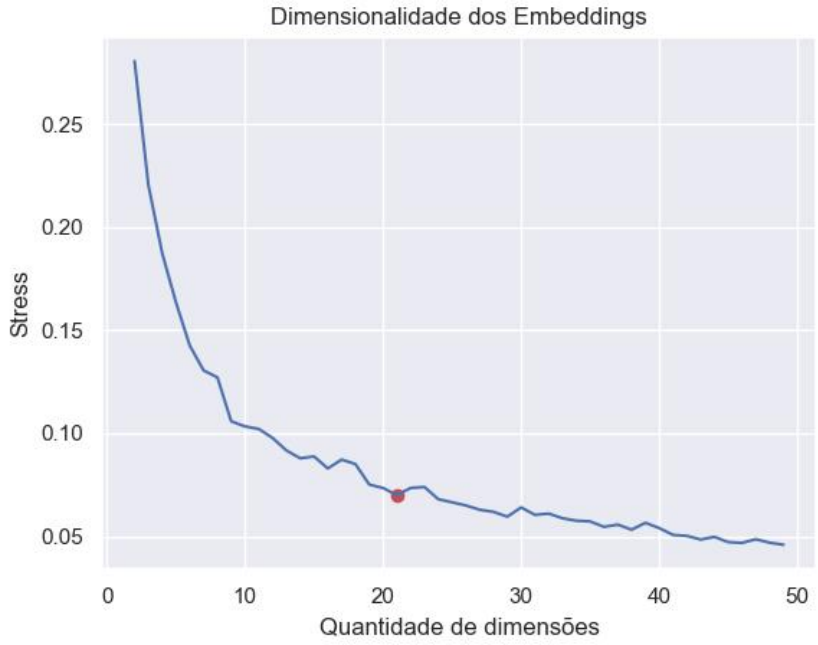


Figura 3.3: *Escolha da dimensionalidade dos Embeddings*

Função de transição de estados

A função de transição de estados é caracterizada pela geração de uma nova sequência, resultante da mutação definida pelo Agente. Em outras palavras, é uma função que tem como entrada uma ação e o estado atual e como saída o estado resultante da transição.

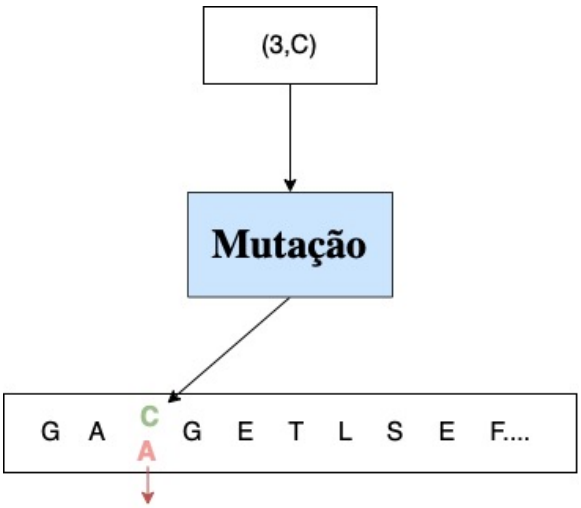


Figura 3.4: *Exemplo de mutação. O aminoácido A na posição 3 está sendo substituído pelo o aminoácido C*

Função de recompensa

Já a função de recompensa calcula a qualidade da ação selecionada pelo Agente. Nas seções 3.2.3 e 3.2.4 especificamos a função de recompensa de cada estágio de treinamento.

3.2.2 Agente

O Agente é modelado por uma rede neural, batizada de *GenSeq*, cujos parâmetros são otimizados através da técnica *Proximal Policy Optimization* (PPO)[5], sendo responsável por decidir qual a melhor mutação a se fazer, dada a sequência de entrada. Em outras palavras, o Agente deve escolher, dentro do espaço de ações, a ação que maximiza a recompensa acumulada. Os hiperparâmetros utilizados no Agente são apresentados na tabela a seguir.

Hiperparâmetro	Valor
Número de camadas escodidas	2
Número de neurônios por camada escondida	64
Função de ativação	tangente hiperbólica
Otimizador	Adam
Taxa de aprendizagem	3e-4
Número de épocas	10
Fator de desconto (γ)	0.99

Tabela 3.1: *Hiperparâmetros do PPO*

Após ser processada pelo *Encoder*, a sequência passa pela camada de entrada do *GenSeq* e segue para duas camadas escondidas de 64 dimensões cada. A camada de saída, por sua vez, possui 255 unidades, organizadas em duas partes:

1. As primeiras 235 unidades da camadas estão associadas ao primeiro atributo da ação: A posição p da sequência de entrada que deve sofrer mutação. A posição predita pelo Agente corresponde ao índice da unidade de saída com maior valor entre as 235 primeiras.

2. Já as 20 últimas unidades de saída estão associadas ao segundo atributo da ação: O índice do aminoácido r que o Agente sugere inserir na posição p. Este valor corresponde ao índice da unidade de saída com maior valor entre as 20 últimas unidades, subtraído de 235.

Por exemplo, suponha que, entre as 235 primeiras unidades, a de maior valor é a terceira, e que, entre as 20 útltimas, o maior valor esteja na unidade 240. Nesse caso, a ação sugerida pelo Agente é fazer uma mutação na posição 3, inserindo o aminoácido de índice 5.

[5] Implementação do PPO construída pelo pacote stable baselines, disponível em: https://github.com/Stable-Baselines-Team/stable-baselines3-contrib/tree/master/sb3_contrib/ppo_mask

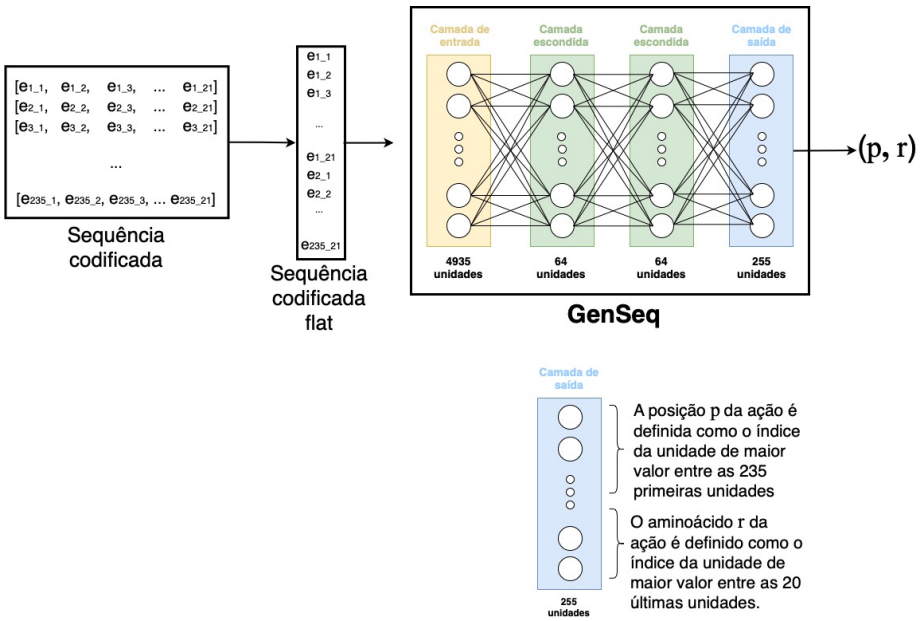


Figura 3.5: *Arquitetura da rede neural do Agente (GenSeq)*

3.2.3 Treinamento - Estágio I

Tendo em vista o vasto número de ações possíveis, desenvolvemos o Estágio I de treinamento com o objetivo de estimular o Agente a encontrar um subespaço menor de ações para atuar. Neste estágio, construímos o ambiente de aprendizado *EnvI*, onde atribuímos recompensas negativas (ou penalizações) para ações em posições da sequência cujo o *Conservation Score* 2.1.7 é alto, bem como penalizamos ações onde o aminoácido substituído é muito diferente do que fora inserido. Neste sentido, definimos a seguinte função de recompensa:

$$R_I = -(CS(p) + AminoDist(r, r_{prev})) \tag{3.4}$$

Onde p e r são a posição e o aminoácido selecionados pelo Agente, respectivamente. O r_{prev} é o aminoácido que estava na posição p antes da mutação. $CS(p)$ é o *Conservation Score* normalizado da posição p e $AminoDist(r, r_{prev})$ é a medida de similariedade 2.1.8 entre os aminoácidos r e r_{prev}.

O ambiente *EnvI* foi construído seguindo a padronização proposta pelo pacote *gym*[6] que garante compatibilidade com algoritmos de aprendizado por reforço. Como entrada, o ambiente recebe uma sequência inicial S_{ini}, o *Encoder*, a matriz de similaridade entre aminoácidos *AminoDist*, a função de score de conservação CS, e os hiperparâmetros M_{max} e N_{rep}, que determinam, respectivamente, o número máximo de mutações permitidas por episódio e o número máximo de repetições para um mesmo aminoácido dentro do episódio.

Durante a inicialização, o estado atual S recebe a sequência inicial S_{ini}. Além disso, é

[6] GYM, API padrão para ambientes de aprendizado por reforço, disponível em: https://www.gymlibrary.dev/

estabelecido o conjunto de posições válidas P_{val}, que inicialmente contém todas as posições da sequência, e o conjunto de aminoácidos válidos R_{val}, que se inicia com todos os 20 aminoácidos possíveis. Também é inicializado um contador $step_{id}$, que rastreia o número de iterações realizadas. A padronização do *gym* prevê a implementação de dois principais métodos: *Reset* e *Step*. O método *Reset* é responsável por restaurar o ambiente ao seu estado inicial, garantindo que cada novo episódio de aprendizado comece a partir de uma condição padronizada. A interação com o ambiente ocorre por meio do método *Step*, que recebe como entrada uma ação (p, r), onde p representa a posição da mutação na sequência e r corresponde ao aminoácido que será inserido nessa posição. O primeiro passo da função consiste em armazenar o aminoácido atual de S na posição p, denotado como r_{prev}. Em seguida, a sequência S é modificada com a inserção de r na posição p e convertida em uma representação vetorial por meio da função $Encoder(S)$. A recompensa R_I é calculada e os conjuntos P_{val} e R_{val} são ajustados. Caso a posição p já tenha sido utilizada, ela é removida de P_{val}. De maneira similar, se o aminoácido r já foi escolhido N_{rep} vezes, ele é removido de R_{val}. Isto é feito de modo a evitar com que o modelo vicie em fazer um número limitado de mutações e não explore devidamente o espaço. Para determinar se o episódio deve ser encerrado, a função verifica se o número total de mutações atingiu M_{max}. Se esse limite for alcançado, a variável de controle $done$ é definida como verdadeira, sinalizando o fim do episódio. Por fim, a função retorna a sequência codificada S_{enc}, a recompensa R_I e a variável de controle $done$.

Algorithm 2 Ambiente de aprendizado - Estágio 1

Require: Classe $gym.ENV$,
sequência inicial S_{ini},
encoder $Encoder$,
matriz de similaridade entre aminoácidos *AminoDist*,
função de score de conservação CS,
número máximo de mutações por episódio M_{max},
número máximo de repetições do mesmo aminoácido por episódio N_{rep}
Ensure: Classe *EnvI*
class *EnvI* **herda** $gym.ENV$
    $S \leftarrow S_{ini}$
    $P_{val} \leftarrow$ *Todas as posições na sequência*
    $R_{val} \leftarrow$ *Todos os aminoácidos*
    $step_{id} \leftarrow 0$
function RESET()
    $S \leftarrow S_{ini}$
    $P_{val} \leftarrow$ *Todas as posições na sequência*
    $R_{val} \leftarrow$ *Todos os aminoácidos*
    $step_{id} \leftarrow 0$
end function
function STEP((p, r))
    $r_{prev} \leftarrow$ aminoácido na posição p da sequência S
    $S \leftarrow$ inserir r na posição p de S
    $S_{enc} \leftarrow Encoder(S)$
    $R_I \leftarrow -(CS(p) + AminoDist(r, r_{prev}))$
    if $p \in P_{val}$ **then**
        Remover p de P_{val}
    end if
    if r já foi escolhido N_{rep} vezes **then**
        Remover r de R_{val}
    end if
    $done \leftarrow False$
    if $step_{id} \geq M_{max}$ **then**
        $done \leftarrow True$
    end if
    $step_{id} \leftarrow step_{id} + 1$
    return $S_{enc}, R_I, done$
end function

O *loop* de pré treino é construído baseado na interação entre o agente *GenSeq* e o ambiente *EnvI*. Esse primeiro estágio começa inicializando o ambiente *EnvI* com a sequência inicial S_{ini}, o *GenSeq* com pesos aleatórios e o histórico de recompensas R_{hist} como uma lista vazia. Obtemos S_{enc} codificando S_{ini} através do $Encoder$. S_{enc} consiste na sequência S_{ini} em um formato compatível com *GenSeq*. O *loop* é então inicializado com o *GenSeq* obtendo uma ação (p, r) baseado em S_{enc}. Com a ação (p, r) é realizado um *step*

no ambiente, resultando em uma nova sequência S_{enc}, a recompensa R_I associada a essa mutação e a variável *done* que indica se o episódio atingiu o número máximo de mutações. R_I é adicionado ao R_{hist} e o ambiente $EnvI$ é reiniciado caso *done* seja verdadeiro. Por fim, os pesos do $GenSeq$ são atualizados com base em R_{hist} através do PPO, caso o *batch* tenha sido finalizado. O *loop* se repete até que seja atingido N_{iter} iterações.

Algorithm 3 Treinamento - Estágio I

Require: Número total de iterações N_{iter},
número máximo de mutações por episódio M_{max},
sequência inicial S_{ini},
encoder $Encoder$,
matriz de similaridade entre aminoácidos $AminoDist$,
função de score de conservação CS,
número máximo de repetições do mesmo aminoácido por episódio N_{rep},
tamanho do batch $batch_size$
Ensure: Agente pré-treinado $GenSeq$
$env \leftarrow EnvI(S_{ini}, Encoder, AminoDist, CS, M_{max}, N_{rep})$
$GenSeq \leftarrow$ Iniciar com pesos aleatórios
$S_{enc} \leftarrow Encoder(S_{ini})$
$R_{hist} \leftarrow []$
for $i = 1$ to N_{iter} **do**
    $(p, r) \leftarrow GenSeq(S_{enc})$
    $(S_{enc}, R_I, done) \leftarrow env.step(p, r)$
    Adiciona R_I ao R_{hist}
    if done = True **then**
        $env.reset()$
    end if
    if i mod batch_size = 0 **then**
        Atualizar pesos do GenSeq baseado em R_{hist} utilizando PPO
        $R_{hist} \leftarrow []$
    end if
end for
return $GenSeq$

A implementação do $EnvI$[7] bem como do primeiro estágio de treinamento[8] estão disponíveis. A figura a seguir ilustra uma visão geral da relação entre o ambiente de aprendizagem e o agente durante o primeiro estágio de treinamento.

[7] Disponível em: https://github.com/ArthurPimenta0306/Hallucination/blob/main/hpd/src/hpd/pipelines/RLenvs/nodes.py

[8] Disponível em: https://github.com/ArthurPimenta0306/Hallucination/blob/main/hpd/src/hpd/pipelines/Modeling/nodes.py

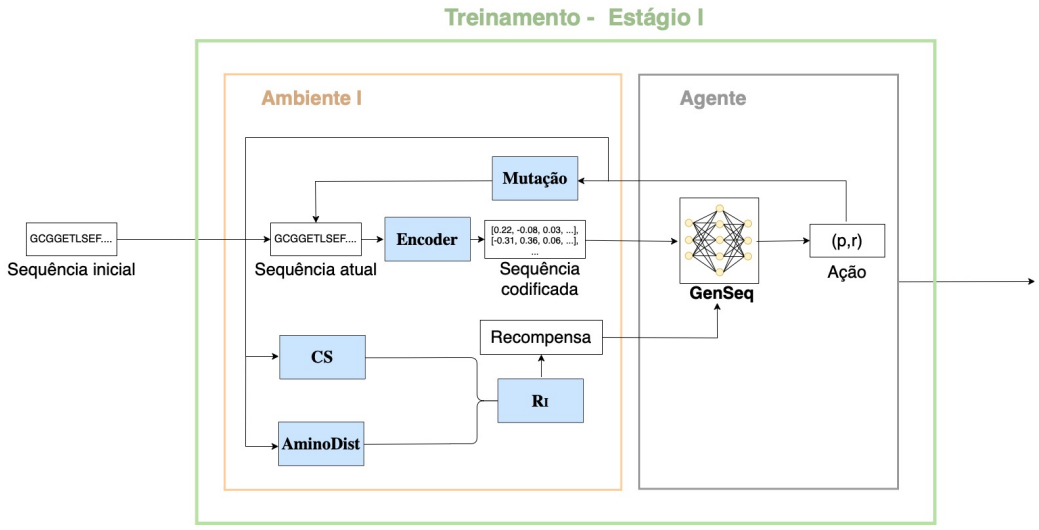


Figura 3.6: *Primeiro estágio de treinamento*

Neste estágio, o Agente foi treinado com um total de 2 milhões de iterações, podendo executar até 100 mutações por episódio.

Parâmetro	Valor
Número máximo de mutações por episódio	100
Número de iterações	2.000.000
Tamanho do *batch*	200
N_{rep}	7

Tabela 3.2: *Parâmetros preliminares*

Após finalizado o primeiro estágio, comparamos o desempenho do *GenSeq* com o desempenho de uma política que define ações aleatoriamente, onde observamos que a mediana da distribuição de recompensas do *GenSeq* é consideravelmente superior ao do Agente aleatório.

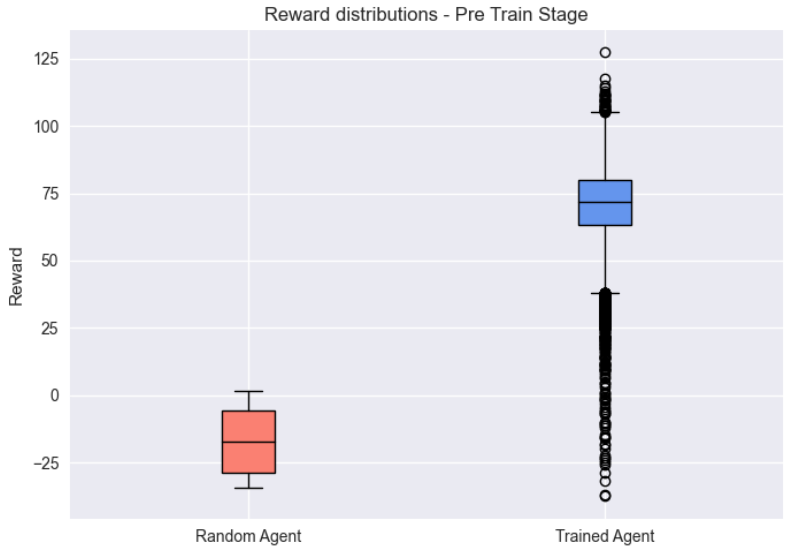


Figura 3.7: *Comparação do Agente pré treinado com um Agente aleatório*

Observa-se que o *GenSeq* atingiu um platô de recompensas neste ambiente com cerca de 2500 episódios, como ilustrado em 3.8.

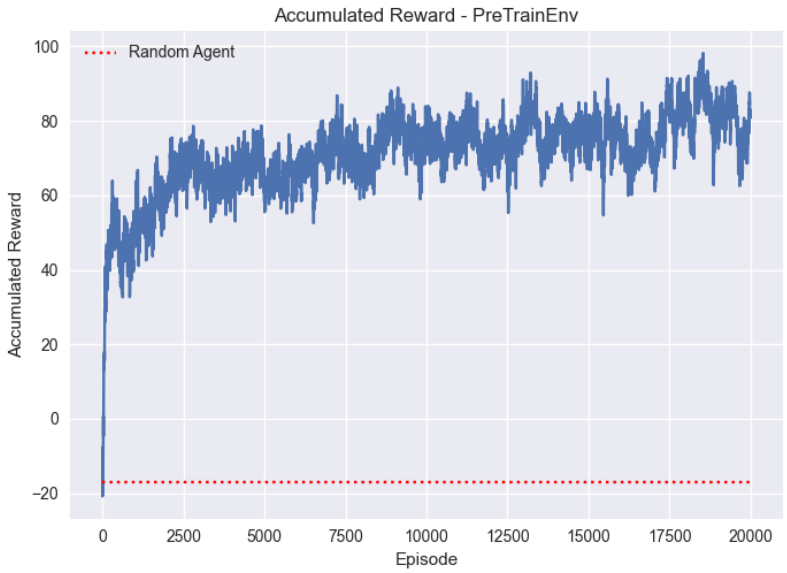


Figura 3.8: *Média móvel das recompensas por episódio - treino*

3.2.4 Treinamento - Estágio II

O objetivo do segundo estágio de treinamento é fazer com que o agente pré treinado aprenda a escolher ações que maximizem o *TM-Score* entre a estrutura gerada e a alvo,

i.e, minimizem o erro $1 - TMScore$. Para isto, definimos a função de recompensa como a soma de 3 termos:

$$R_{II} = -aEr_i + b(Er_0 - Er_i) + c(Er_{i-1} - Er_i) \tag{3.5}$$

Onde Er_i e Er_{i-1} são os erros da iteração i e i-1 respectivamente, e a, b e c são constantes. O primeiro termo, $-aEr_i$, é a penalização pelo erro atual. O segundo determina uma penalização se o erro atual for maior que o inicial e uma recompensa caso contrário. Análogamente, o terceiro termo penaliza se o erro atual for maior que o anterior e recompensa se for menor. Como o objetivo principal é encontrar sequências com erro menores que o inicial, atribuímos um peso maior ao segundo termo, i.e, $b > a$ e $b > c$. Além disso, consideramos $c > a$ para valorizar reduções do erro ao longo do episódio. Introduzimos também o conceito de vitória e derrota a cada episódio. Consideramos vitória caso o erro final seja menor que o inicial e derrota caso contrário. De modo a valorizar a vitória e penalizar a derrota, ao final de cada episódio é adicionado uma recompensa adicional dada por $2500(Er_0 - Er_f)$, onde Er_f é o erro final, i.e, da última iteração do episódio. Cada episódio é finalizado ao se atingir um número máximo de iterações, ou um limiar de erro. Foi definido um limite superior para o erro Er_{max} de modo a evitar que o Agente atinja erros muito grandes em um mesmo episódio, e um limite inferior Er_{min} para valorizar episódios em que o Agente foi capaz de encontrar "atalhos", i.e, conseguiu reduzir o erro considerávelmente em poucas iterações.

O pior cenário em termos de recompensa para o agente é o caso em que limiar de erro máximo Er_{max} ocorre antes de atingir o número máximo de iterações. Nesse caso, a recompensa adicional é dada pela sequinte expressão:

$$R_{II} \leftarrow R_{II} + 2500(Er_0 - Er_i) + R_{II}(M_{max} - step_{id}) \tag{3.6}$$

Ou seja, um termo adicional de penalização, $R_{II}(M_{max} - step_{id})$, é incluído na recompensa final. Esse termo representa uma estimativa pessimista da recompensa total caso o agente fosse autorizado a completar todas as M_{max} iterações. A suposição subjacente é que, nas iterações restantes $M_{max} - step_{id}$, o agente manteria o desempenho atual, ou seja, continuaria acumulando a mesma recompensa R_{II} associada a um erro superior ao limite máximo Er_{max}. Isso desestimula fortemente trajetórias em que o erro excede Er_{max} precocemente.

A inicialização do segundo ambiente $EnvII$ é feita da mesma forma que $EnvI$, porém com a inclusão da definição de Er_i e Er_{i-1} que iniciam iguais ao erro inicial Er_0. O ambiente é dependente de parâmetros como S_{ini}, a estrutura do FIX (E_{FIX}), o erro inicial (E_r0), os limiares de erro Er_{max} e Er_{min}, o *Encoder*, M_{max}, N_{rep} e os coeficientes da função de recompensa (a, b, c).

O método *step* do $EnvII$, assim como no primeiro ambiente, recebe como entrada uma ação (p, r) que é utilizada para realizar a mutação na sequência S. Posteriormente, a sequência é transformada em uma representação codificada pelo $Encoder$, originando a S_{enc}. É feita então a predição da estrutura tridimensional E_S associada a sequência S utilizando o modelo *ESMFold*. O erro da iteração anterior recebe o erro atual, que por sua

vez, é atualizado com o resultado de $1 - TMScore(E_{FIX}, E_S)$. Tendo Er_i e Er_{i-1}, é calculado a recompensa R_{II} seguindo a equação 3.5. A atualização de P_{val} e R_{val} é realizada com a mesma lógica utilizada em $EnvII$. Já o método *reset* pouco difere do método implementado em $EnvI$. A única alteração consiste na atribuição do erro inicial para Er_i e Er_{i-1}.

Algorithm 4 Ambiente de Aprendizado - Estágio II

Require: Classe $gym.ENV$,
Condições iniciais S_{ini} e Er_0
Estrutura do FIX E_{FIX},
Limiares de erro Er_{min} e Er_{max},
encoder $Encoder$,
número máximo de mutações por episódio M_{max},
número máximo de repetições do mesmo aminoácido por episódio N_{rep},
coeficientes da função de recompensa (a, b, c)
Ensure: Classe $EnvII$
class $EnvII$ **herda** $gym.ENV$
    $S \leftarrow S_{ini}$
    $Er_i, Er_{i-1} \leftarrow Er_0$
    $P_{val} \leftarrow$ *Todas as posições na sequência*
    $R_{val} \leftarrow$ *Todos os aminoácidos*
    $step_{id} \leftarrow 0$
function RESET()
    $S \leftarrow S_{ini}$
    $Er_i, Er_{i-1} \leftarrow Er_0$
    $P_{val} \leftarrow$ *Todas as posições na sequência*
    $R_{val} \leftarrow$ *Todos os aminoácidos*
    $step_{id} \leftarrow 0$
end function
function STEP((p, r))
    $S \leftarrow$ inserir r na posição p de S
    $S_{enc} \leftarrow encoder(S)$
    $E_S \leftarrow ESMFold(S)$
    $Er_{i-1} \leftarrow Er_i$
    $Er_i \leftarrow 1 - TMScore(E_{FIX}, E_S)$
    $R_{II} \leftarrow -aEr_i + b(Er_0 - Er_i) + c(Er_{i-1} - Er_i)$
    if $p \in P_{val}$ **then**
        Remover p de P_{val}
    end if
    if r já foi escolhido N_{rep} vezes **then**
        Remover r de R_{val}
    end if
    $step_{id} \leftarrow step_{id} + 1$
    $done \leftarrow False$
    if $step_{id} \geq M_{max}$ ou $Er_i \leq Er_{min}$ ou $Er_i \geq Er_{max}$ **then**
        $done \leftarrow True$
        if $step_{id} = M_{max}$ **then**
            $R_{II} \leftarrow R_{II} + 2500(Er_0 - Er_i)$
        end if
        if $Er_i \geq Er_{max}$ **then**
            $R_{II} \leftarrow R_{II} + 2500(Er_0 - Er_i) + R_{II}(M_{max} - step_{id})$
        end if
    end if
    return $S_{enc}, R_{II}, done$
end function

O *loop* de treinamento segue a mesma lógica do primeiro estágio, alterando apenas o ambiente de aprendizado de $EnvI$ para $EnvII$ e suas respectivas dependências. Além disso, o treinamento se inicia com os pesos do *GenSeq* pré treinados, obtidos através do primeiro estágio.

Algorithm 5 Treinamento - Estágio II

Require: Número total de iterações N_{iter},
número máximo de mutações por episódio M_{max},
Condições iniciais S_{ini} e Er_0
encoder $Encoder$,
tamanho do batch $batch_size$,
coeficientes da função de recompensa (a, b, c),
número máximo de repetições do mesmo aminoácido por episódio N_{rep}
Estrutura do FIX E_{FIX},
Limiares de erro Er_{min} e Er_{max},
Agente pré treinado *GenSeq*
Ensure: Agente treinado *GenSeq*
$env \leftarrow EnvII(S_{ini}, Er_0, Encoder, (a, b, c), E_{FIX}, Er_{min}, Er_{max}, M_{max}, N_{rep})$
$S_{enc} \leftarrow Encoder(S_{ini})$
$R_{hist} \leftarrow []$
for $i = 1$ to N_{iter} **do**
    $(p, r) \leftarrow GenSeq(S_{enc})$
    $(S_{enc}, R_{II}, done) \leftarrow env.step(p, r)$
    Adiciona R_{II} ao R_{hist}
    if done = True **then**
        $env.reset()$
    end if
    if i mod batch_size = 0 **then**
        Atualizar pesos do GenSeq baseado em R_{hist} utilizando PPO
        $R_{hist} \leftarrow []$
    end if
end for
return *GenSeq*

A implementação do $EnvII$[9] bem como do segundo estágio de treinamento[10] estão disponíveis. A figura a seguir ilustra uma visão geral da relação entre o segundo ambiente de aprendizagem e o agente durante o segundo estágio de treinamento.

[9] Disponível em: https://github.com/ArthurPimenta0306/Hallucination/blob/main/hpd/src/hpd/pipelines/RLenvs/nodes.py

[10] Disponível em: https://github.com/ArthurPimenta0306/Hallucination/blob/main/hpd/src/hpd/pipelines/Modeling/nodes.py

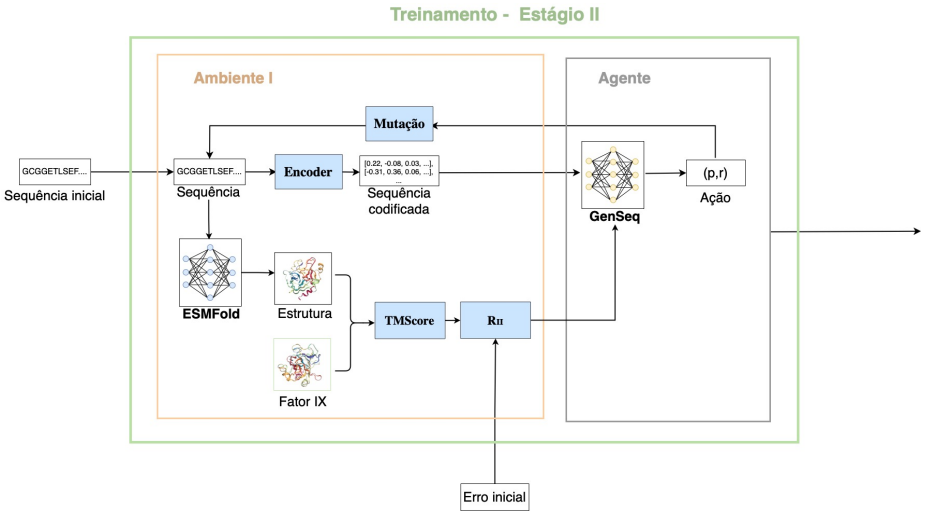


Figura 3.9: *Segundo estágio - Treino*

Neste estágio, o Agente foi treinado com um total de 60 mil de iterações, podendo executar até 45 mutações por episódio.

Parâmetro	Valor
Número máximo de mutações por episódio	45
Erro mínimo no episódio	0.05
Erro máximo no episódio	0.083
Número de iterações	60.000
a	12
b	200
c	50
N_{rep}	7

Tabela 3.3: *Parâmetros preliminares*

Assim como no pré treino, a distribuição das recompensas do Agente ao longo do treinamento é bem superior às recompensas de um Agente aleatório.

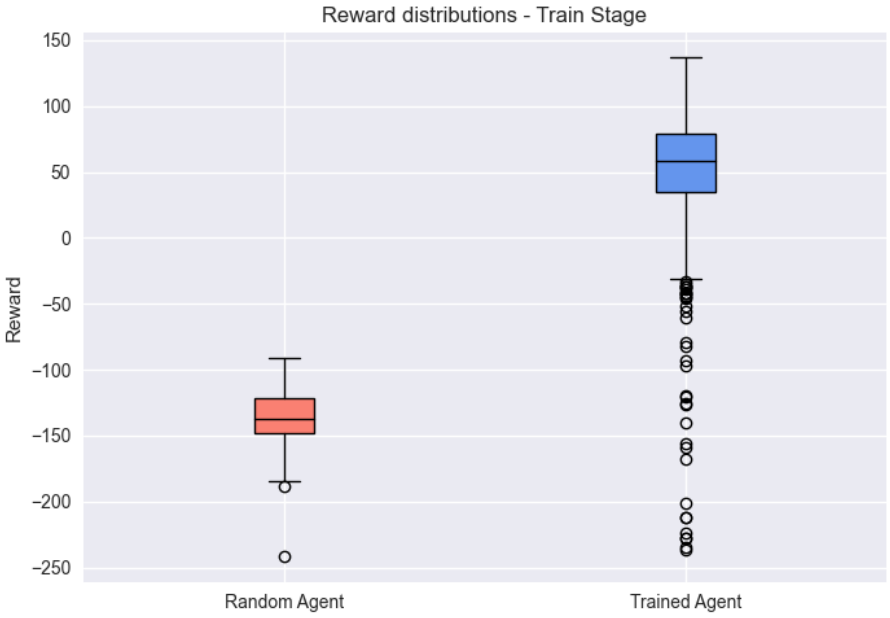


Figura 3.10: *Comparação do Agente treinado com um Agente aleatório*

Repare que, devido ao pré treino, as recompensas médias do Agente são consideravelmente superiores à recompensa média do Agente aleatório desde os primeiros episódios.

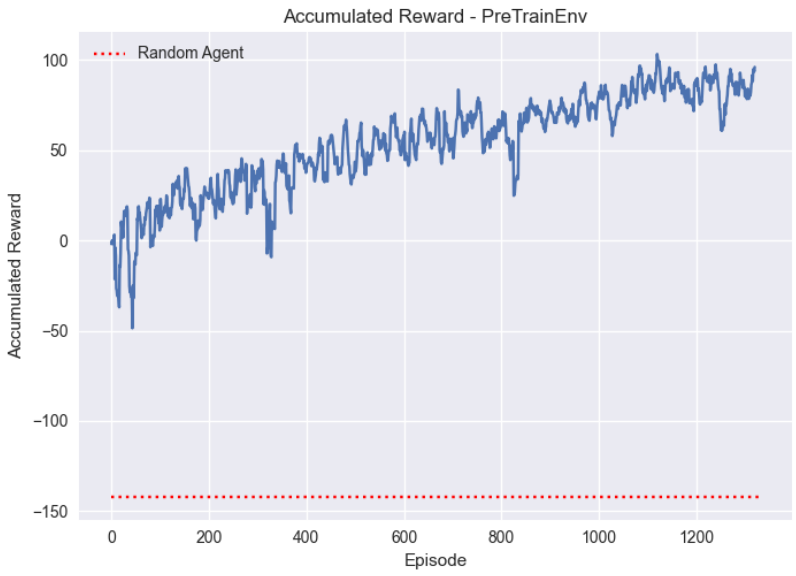


Figura 3.11: *Média móvel das recompensas por episódio - treino*

3.3 Geração de sequências

Por fim, o módulo de Geração de Sequências utiliza o agente treinado para gerar novas sequências que sejam capazes de mimetizar a estrutura alvo melhor do que a sequência inicial, em termos de *TMScore*. Para isso, o ambiente $EnvII$ é inicializado com as mesmas condições utilizadas no segundo estágio de treinamento, incluindo a sequência inicial S_{ini}, o erro inicial Er_0, o $Encoder$, os coeficientes (a, b, c) e a estrutura alvo E_{FIX}. Para maximizar a diversidade de sequências geradas, as restrições impostas pelos limiares de erro Er_{min} e Er_{max} são removidas, sendo definidos como 0 e 1, respectivamente. Além disso, a geração de sequências é configurada para produzir um total de 235 novas variantes, fixando os parâmetros N_{iter} e M_{max} em 235. O processo de geração consiste em iterativamente obter uma ação com o *GenSeq* e com ela executar o método *step* no $EnvII$.

Algorithm 6 Geração de sequências

Require: Número total de iterações N_{iter},
número máximo de mutações por episódio M_{max},
Condições iniciais S_{ini} e Er_0
encoder $Encoder$,
coeficientes da função de recompensa (a, b, c),
número máximo de repetições do mesmo aminoácido por episódio N_{rep}
Estrutura do FIX E_{FIX},
Limiares de erro Er_{min} e Er_{max},
Agente treinado *GenSeq*
Ensure: Novas sequências $Seqs$
$env \leftarrow EnvII(S_{ini}, Er_0, Encoder, (a, b, c), E_{FIX}, Er_{min}, Er_{max}, M_{max}, N_{rep})$
$S_{enc} \leftarrow Encoder(S_{ini})$
$Seqs \leftarrow []$
for $i = 1$ to N_{iter} **do**
    $(p, r) \leftarrow GenSeq(S_{enc})$
    $(S_{enc}, R_{II}, done) \leftarrow env.step(p, r)$
    Adiciona $env.S$ ao $Seqs$
end for
return $Seqs$

A implementação da geração de sequências está disponível[11]. A figura a seguir representa um esquemático do *loop* de geração de sequências descrito.

[11] Disponível em: https://github.com/ArthurPimenta0306/Hallucination/blob/main/hpd/src/hpd/pipelines/Evaluation/nodes.py

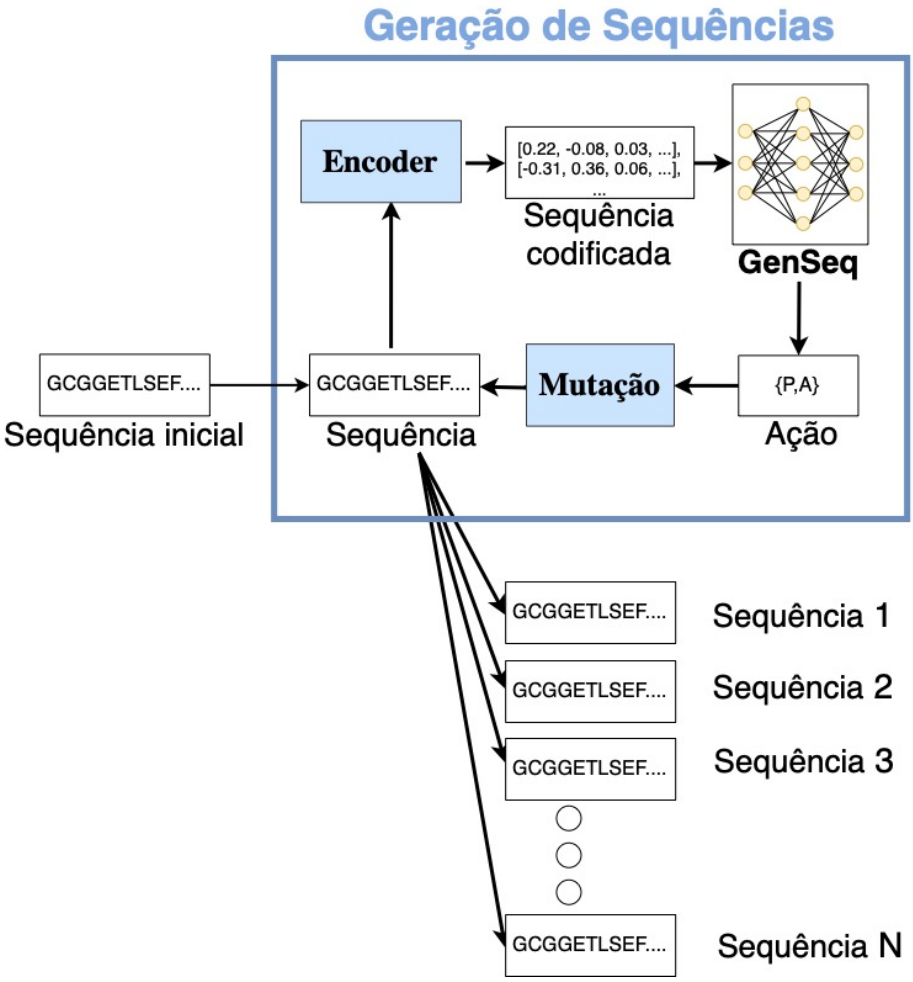


Figura 3.12: *Gerando sequência otimizada*

3.4 Métodos de avaliação

Em andamento

A avaliação das proteínas geradas pelo agente será realizada em três etapas: análise estrutural, interação molecular por Docking e imunogenicidade. Cada etapa é essencial para verificar se as proteínas geradas possuem características estruturais, funcionais e imunológicas adequadas para substituir o Fator IX (FIX) no tratamento da hemofilia B.

3.4.1 Análise Estrutural

A primeira etapa consiste em medir a similaridade estrutural entre cada proteína gerada e o FIX por meio do *TMScore*. Para isso, foi implementado um algoritmo que recebe como entrada o conjunto de sequências geradas S_{gen}, a estrutura alvo E_{FIX} e o modelo de predição estrutural $ESMFold$. Para cada sequência gerada, é feito a predição da estrutura tridimensional E_S e o cálculo do *TMScore* em relação à estrutura alvo E_{FIX}.

Algorithm 7 Avaliação Estrutural via TMScore

Require: Conjunto de sequências geradas S_{gen},
estrutura alvo E_{FIX},
modelo de predição estrutural $ESMFold$
Ensure: Lista de TMScore para cada sequência gerada
$TMScore\_list \leftarrow []$
for S em S_{gen} **do**
    $E_S \leftarrow ESMFold(S)$
    $TM_S \leftarrow TMScore(E_{FIX}, E_S)$
    Adiciona TM_S a $TMScore\_list$
end for
return $TMScore\_list$

3.4.2 Avaliação de Interação com o Fator VIII (Docking Molecular)

Além da similaridade estrutural, é fundamental avaliar a capacidade das proteínas geradas de interagir corretamente com o Fator VIII (FVIII), replicando a função do FIX nativo. Para isso, realizamos análises de *docking molecular* entre cada estrutura gerada e o FVIII, utilizando métricas como *Contact Molecular Surface* (CMS), *Interface Buried SASA* (IBSASA), *Delta Gibbs Free Energy of Binding* (DDG) e *Surface Area Potential Score* (SAP Score).

Antes da realização do *docking*, é necessário alinhar estruturalmente a proteína gerada com o FVIII para garantir que a proteína mutante ocupe a mesma posição relativa do FIX no complexo original. Isso é feito utilizando *ChimeraX* Elaine C. Meng, 2023 por meio do comando `matchmaker`, que alinha a estrutura gerada ao FIX presente no complexo tenase. O procedimento segue os seguintes passos: (i) carregamento do PDB do complexo tenase, (ii) carregamento do PDB da estrutura gerada, (iii) alinhamento da estrutura gerada sobrepondo-a ao FIX do complexo tenase e (iv) remoção do FIX original, mantendo apenas a estrutura gerada alinhada ao FVIII. Está implementação[12] está disponível no repositório do projeto.

Algorithm 8 Alinhamento Estrutural com ChimeraX

Require: Arquivo PDB do complexo tenase, arquivo PDB da estrutura gerada
Ensure: Estrutura gerada alinhada ao FVIII
Carregar PDB do complexo tenase
Carregar PDB da estrutura gerada
Alinhar a estrutura gerada ao FIX usando *matchmaker*
Remover o FIX do complexo
return Estrutura gerada alinhada ao FVIII

Imagem do passo a passo do alinhamento com prints do ChimeraX

[12] Disponível em: https://github.com/ArthurPimenta0306/Hallucination/blob/main/hpd/src/docking/ChimeraX_run_script.py

Após o alinhamento estrutural utilizando *ChimeraX*, realizamos o *docking molecular* por meio do protocolo *Rosetta Docking*, que permite prever a orientação e estabilidade do complexo proteína-proteína gerado. O *Rosetta Docking* consiste em um procedimento iterativo que busca a conformação de menor energia livre do complexo. O protocolo inicia com a geração de múltiplas poses, onde a proteína mutante é rotacionada e transladada em relação ao FVIII. Em seguida, as conformações passam por um refinamento energético que inclui otimização da orientação relativa (*rigid-body minimization*) e ajuste das cadeias laterais na interface (*side-chain minimization*). O protocolo utiliza a função de energia do *Rosetta* para estimar a estabilidade do complexo, considerando fatores como interações de *Van der Waals*, ligações de hidrogênio, energia eletrostática e solubilidade. Baseado na função de energia e da posição relativa dos átomos no complexo, é calculada as métricas *CMS*, *IBSASA*, *DDG* e *SAP Score*. O comando para execução do docking, bem como sua parametrização estão disponíveis[13] e ilustrados a seguir.

Comando para execução - *Rosetta Docking*

```
../rosetta.source.release-371/main/source/bin/docking_protocol.default.
    macosclangrelease -ignore_unrecognized_res @flag_file -out:prefix
    process_1_
```

Parametrização - *Rosetta Docking*

1. **ignore_unrecognized_res**: Atributo utilizado para ignorar resíduos não reconhecidos pelo *Rosetta*.
2. **flag_file**: Arquivo .flag contendo os parâmetros de execução do *Rosetta Docking*.
3. **out:prefix**: Prefixo para os arquivos de saída gerados pelo *Rosetta Docking*.

Parametrização - Arquivo .flag utilizado como *input* do *Rosetta Docking*

1. **in:file:s**: Especifica o caminho para o arquivo PDB do complexo de entrada (FVIII junto de uma proteína gerada), que será utilizada no processo de docking. Ex: `complex_step_1.pdb`
2. **nstruct**: `1000`. Define o número de modelos (poses de encaixe entre as duas proteínas) a serem gerados no processo de *docking*.
3. **partners**: `A_H`. Especifica quais cadeias da estrutura PDB interagem no *docking*.
4. **dock_pert**: `3 8`. Aplica uma perturbação inicial na orientação da proteína antes do docking. O primeiro valor (`3`) representa um deslocamento translacional em Ångströms, enquanto o segundo (`8`) define uma rotação em graus.
5. **out:path:all**: Define o diretório onde todos os arquivos de saída do docking serão armazenados. Ex: `output_files/`

[13] Disponível em: https://github.com/ArthurPimenta0306/Hallucination/blob/main/hpd/src/docking/docking_protein_pipeline.sh e https://github.com/ArthurPimenta0306/Hallucination/blob/main/hpd/src/docking/main.py

A partir dos resultados do docking, implementamos[14] o processo que calcula as métricas responsáveis por quantificar a qualidade da interação entre as proteínas geradas e o FVIII utilizando o *PyRosetta*.

3.4.3 Avaliação da Imunogenicidade das Proteínas Geradas

Uma proteína substituta do FIX precisa apresentar baixa imunogenicidade para evitar respostas imunológicas adversas em pacientes. Para isso, realizamos uma análise comparativa entre os epítopos imunodominantes das proteínas geradas e do FIX nativo, utilizando o *NetMHCIIpan-4.0* K. K. Jensen, 2018, um modelo de aprendizado de máquina para predição de afinidade entre peptídeos e MHC.

O comando para execução do *NetMHCIIpan-4.0* é:

Corrigir o comando

```
netMHCIIpan -f generated_sequences.fasta -a DRB1_0101 -rank_threshold 2.0 >
    netmhc_results.txt
```

A partir dos resultados, calculamos a imunogenicidade relativa de cada sequência.

Algorithm 9 Cálculo da Diferença de Imunogenicidade

Require: Saída do *NetMHCIIpan-4.0*, sequência do FIX nativo
Ensure: Valores comparativos de imunogenicidade
1: Analisar afinidades de epítopos da sequência gerada
2: Comparar com os epítopos da sequência do FIX
3: Calcular a diferença média de afinidade
4: **return** Diferença de imunogenicidade

Dessa forma, ao combinar essas três análises, conseguimos avaliar a adequação estrutural, funcional e imunológica das proteínas geradas pelo agente.

[14] Disponível em: https://github.com/ArthurPimenta0306/Hallucination/blob/main/hpd/src/docking/get_interface_metrics_small.py

Capítulo 4

Resultados e Discussões

Em andamento

Com o Agente treinado, é possível gerar novas sequências seguindo a arquitetura ilustrada na figura 3.12. O Agente gerou mais de 200 proteínas diferentes a partir de mutações na sequência inicial. De modo a identificar cada proteína, vamos considerar a ordem com que foi gerada como seu respectivo ID. Filtramos as proteínas geradas baseado em um limiar de 92% de TMScore. Com isso, selecionamos as 63 primeiras sequências.

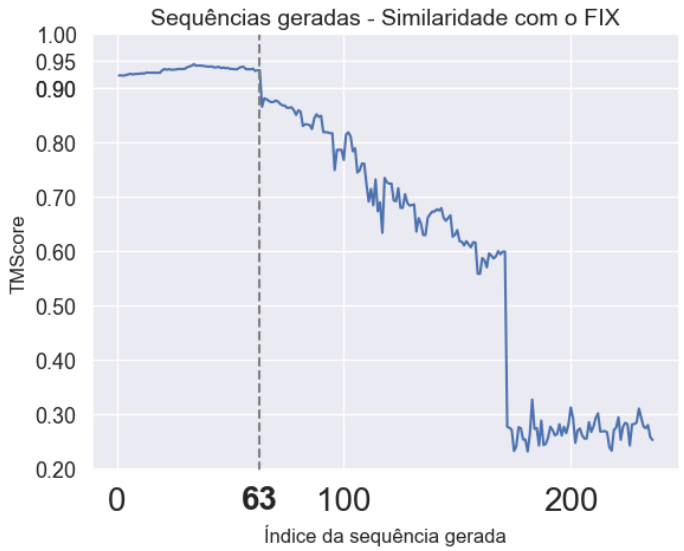


Figura 4.1: *Variação da similaridade entre as sequências geradas e o FIX*

A proteína gerada que é estruturalmente mais similar ao FIXa é a de ID 34, tendo TMScore igual a 94.48% e RMSD de 1.485 angstroms.

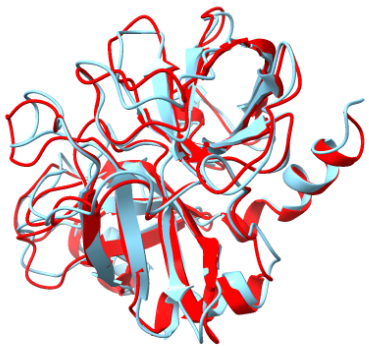


Figura 4.2: *Alinhamento entre FIX (em azul) e a proteína ID 34 (em vermelho)*

É interessante notar que, apesar da forte semelhança estrutural entre as proteínas, em termos de IS (Idêntidade de Sequência) são sequências muito diferentes, tendo em média apenas 30% de similaridade.

ID	TM-Score	RMSD	IS - seq. original
20	93.33%	1.75 Å	30.64%
30	93.62%	1.68 Å	30.21%
34	94.48%	1.48 Å	29.79%
63	93.36%	1.72 Å	26.81%

Tabela 4.1: *Generated proteins*

4.1 Resposta imunológica

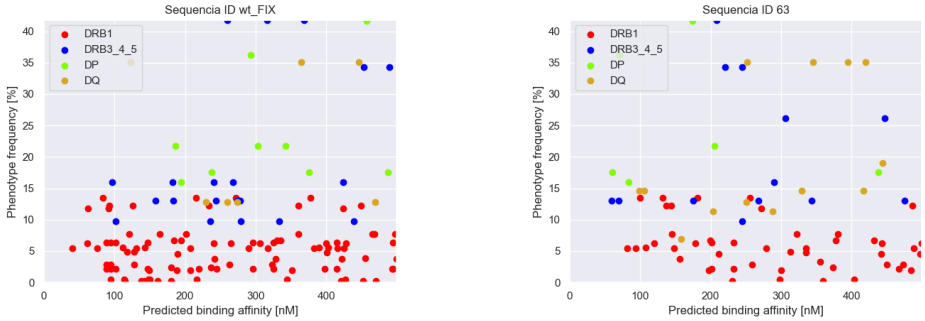
Para mensurar a propensão das proteínas geradas provocarem uma resposta imunológica no organismo, calculamos o *binding affinity* de cada proteína com os epítopos de quatro grupos diferentes de alelos apresentados na tabela 4.2

Grupo	Frequência em seres humanos
DP	25%
DQ	20%
DRB1	5%
DRB3_4_5	21%

Tabela 4.2: *Grupos de alelos considerados e a frequência com que ocorrem em seres humanos.*

Para esta análise, levamos em consideração apenas epítopos associados a valores de *binding affinity* menores que 500nm, indicando ligação forte. Nas figuras a seguir é possível visualizar a quantidade, *binding affinity* e frequência de ocorrência dos epítopos por cada grupo de alelo. Ao se comparar o resultado do FIX e da proteína de ID 63, é possível

notar uma redução significativa da quantidade de epítopos com ligação forte, influênciado principalmente pelo resultado do grupo DRB1.



(a) *Resposta imune FXI* **(b)** *Resposta imune ID 63*

A propensão a provocar uma resposta imune não está apenas relacionada a quantidade de epítopos, mas também é diretamente proporcional a frequência de ocorrência dos alelos e inversamente proporcional ao *binding affinity*. Neste sentido, para mensurar qual proteína induz a uma melhor resposta imune, propomos a métrica IM, que consiste na média da razão entre *binding affinity* (Af_i) e frequência (Fr_i), dividido pela quantidade de epítopos (N):

$$IM = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{N} \frac{Af_i}{Fr_i} \tag{4.1}$$

Desta forma, a proteína associada ao maior IM possui o menor risco de induzir a uma resposta imune, i.e, possui a resposta imune mais favorável. De modo geral, as 4 últimas sequências geradas possuem um IM superior ao do FIX: IDs 60, 61, 62 e 63.

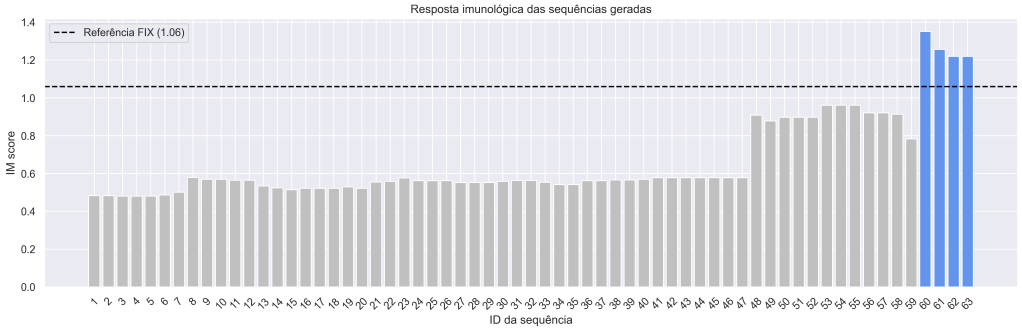


Figura 4.4: *Comparação do IM entre as proteínas geradas e o FIX*

Calculando o IM para cada grupo de alelo separadamente, observa-se que grande parte das sequências geradas possuem uma resposta imune mais favorável que o FIX para os grupos DRB3, DRB4 e DRB5. Para o grupo DRB1, as sequências a partir do ID 48 em diante obtiveram IM superior ao FIX. Já para os grupos DQ e DP, nenhuma sequência superou o IM do FIX.

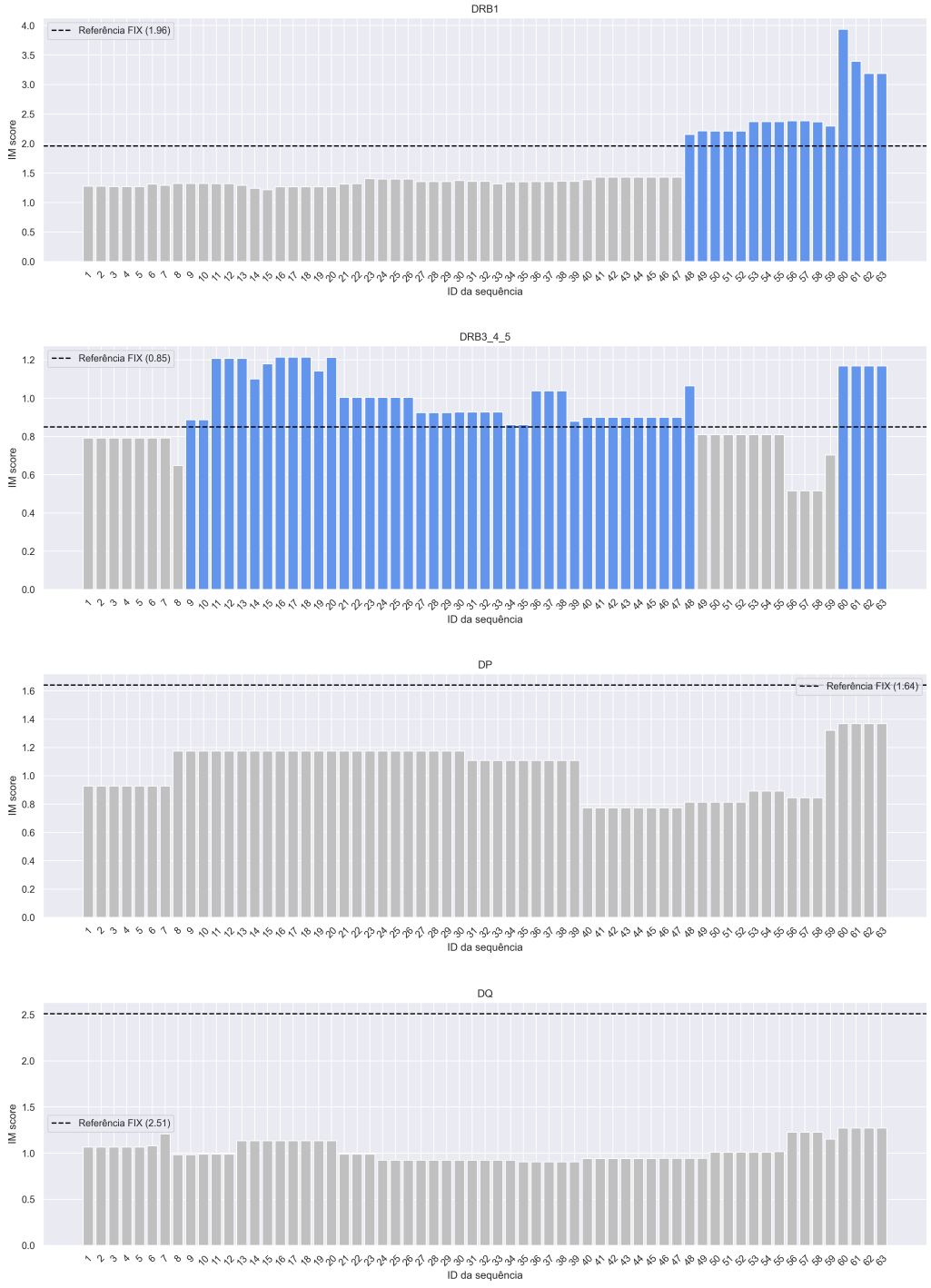


Figura 4.5: *IM calculado para cada grupo de alelo*

4.2 Docking

Ao projetar proteínas capazes de substituir o FIX no tratamento da Hemofilia B, um aspecto crítico é garantir que as proteínas geradas possuem uma interação eficaz ou até mais favorável com o FVIII se comparado ao FIX. Para isto, realizamos análises de *docking* molecular de cada sequência gerada com o FVIII, calculando métricas que capturam diferentes características da interface proteína-proteína.

- *Contact Molecular Surface*: obtivemos 56 proteínas (89% do total) superiores ao FIX em termos de CMS, sugerindo interfaces de ligação maiores para essas sequências.
- DDG: 46 proteínas (73% do total) atingiram um DDG mais favorável (menor) do que o FIX, indicando ligações mais estáveis.
- *Interface Buried Sasa*: 51 proteínas (81% do total) obtiveram um SASA superior ao FIX.
- *SAP Score Target*: apenas 3% das sequências projetadas atingiram valores superiores ao FIX em termos de SAP.

Discussão sobre como o pipeline foi capaz de otimizar metricas que não foi treinado

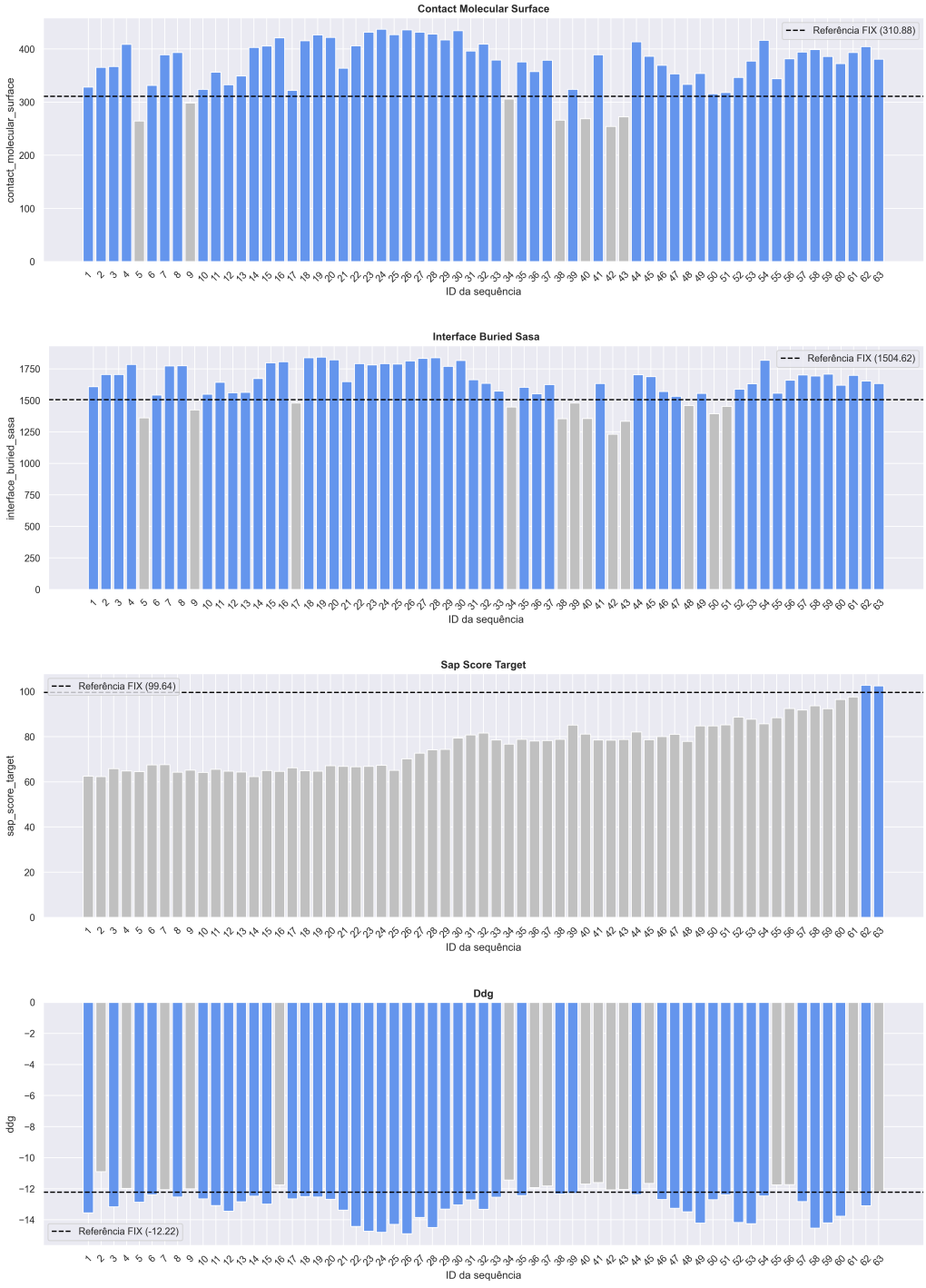


Figura 4.6: *Resultado do Docking entre as proteínas geradas e o fator VII*

Capítulo 5

Conclusão

Em andamento

Este trabalho apresentou um novo paradigma para o *design* de proteínas terapêuticas, aplicando aprendizado por reforço profundo para otimizar a sequência do fator IX (FIX), com o objetivo de melhorar sua estabilidade estrutural, reduzir a imunogenicidade e potencializar sua eficácia no tratamento da hemofilia tipo B.

A abordagem proposta combinou metodologias avançadas de modelagem molecular e inteligência artificial para explorar de maneira eficiente o vasto espaço de sequências proteicas. O uso do algoritmo *Proximal Policy Optimization* (PPO) permitiu a evolução de variantes do FIX por meio de um processo iterativo de seleção e refinamento, guiado por métricas estruturais e funcionais. A validação das proteínas geradas foi realizada por meio de análises computacionais, incluindo *Template Modeling Score* (TM-Score), *Contact Molecular Surface* (CMS) e *docking* molecular com o Fator VIII (FVIII), demonstrando que algumas variantes apresentam potencial terapêutico promissor.

A principal contribuição deste trabalho reside na aplicação inédita de aprendizado por reforço profundo para a otimização do FIX, um campo ainda pouco explorado na literatura científica. A inexistência de estudos anteriores que utilizem essa abordagem para o design de proteínas voltadas ao tratamento da hemofilia B reforça a originalidade e a relevância deste estudo.

Embora os resultados obtidos sejam promissores, este trabalho apresenta algumas limitações. A ausência de validação experimental (*wet lab*) impede a confirmação direta da funcionalidade das proteínas geradas, sendo necessário realizar testes laboratoriais para avaliar sua estabilidade e atividade biológica. Além disso, futuras melhorias podem incluir a incorporação de simulações físicas mais sofisticadas para refinar as predições estruturais e funcionais do modelo.

Em síntese, este trabalho demonstra o potencial do aprendizado por reforço profundo no design racional de proteínas terapêuticas. Os avanços obtidos aqui podem servir de base para novas pesquisas que busquem aprimorar o desenvolvimento de biofármacos personalizados, contribuindo para a inovação no tratamento da hemofilia tipo B e outras doenças genéticas.

Referências

[J. D. e. al 2022] J. Dauparas et al. "Robust deep learning–based protein sequence design using proteinmpnn". *Science* (2022) (citado nas pgs. 2, 7).

[J. S. e. al 2017] John Schulman et al. "Proximal policy optimization algorithms". *ArXiv* (2017) (citado nas pgs. 20, 21).

[M. H. B. e. al 2024] Mettine H.A. Bos et al. "Blood coagulation factor ix: structural insights impacting hemophilia b therapy". *American Society for Hematology Journal BLOOD* (2024) (citado nas pgs. 11, 12).

[Z. L. e. al 2022] Zeming Lin et al. "Language models of protein sequences at the scale of evolution enable accurate structure prediction". *bioRxiv* (2022) (citado na pg. 6).

[al. 2013] Massimo Franchini et al. "Outcome of clinical trials with new extended half-life fviii/ix concentrates." *Biologics: Targets and Therapy* (2013) (citado na pg. 5).

[Angermueller C 2019] et al Angermueller C. "Model-based reinforcement learning for biological sequence design". In: ed. por 8th International Conference on Learning Representations (ICLR 2020). 2019 (citado na pg. 7).

[B 2019] Kuhlman B. "Designing protein structures and complexes with the molecular modeling program rosetta". *J Biol Chem* (2019) (citado nas pgs. 2, 5).

[B. Alberts 2002] et al. B. Alberts. *Molecular Biology of the Cell*. Garland Science, 2002 (citado nas pgs. 9–11).

[Ben Chorin 2020] A. et al Ben Chorin. "Consurf-db: an accessible repository for the evolutionary conservation patterns of the majority of pdb proteins". *Protein Sci* (2020) (citado na pg. 16).

[Bishop 2006] M. Christopher Bishop. *Pattern Recognition and Machine Learning*. New York: Springer, 2006 (citado nas pgs. 18–20).

[Bjoern Peters e Sette 2020] Morten Nielsen Bjoern Peters e Alessandro Sette. "T cell epitope predictions". *Annual Reviews* (2020) (citado na pg. 14).

[Charif 2007] J. Charif D. Lobry. "R. in structural approaches to sequence evolution". *Springer* (2007) (citado na pg. 16).

[Cohen e Kessler 1995] Alice J. Cohen e Craig M. Kessler. "Treatment of inherited coagulation disorders". *The American Journal of Medicine* 99.6 (1995), pp. 675–682. ISSN: 0002-9343. DOI: https://doi.org/10.1016/S0002-9343(99)80256-3. URL: https://www.sciencedirect.com/science/article/pii/S0002934399802563 (citado na pg. 5).

[Coventry 2021] Brian Coventry. "Learning How to Make Mini-Proteins that Bind to Specific Target Proteins". Tese de dout. University of Washington, 2021 (citado na pg. 13).

[Eddy 2004] S. R. Eddy. "Where did the blosum62 alignment score matrix come from?" *Nature Biotechnology* (2004) (citado na pg. 16).

[Elaine C. Meng 2023] et al. Elaine C. Meng. "Ucsf chimerax: tools for structure building and analysis". *Protein Science* (2023) (citado na pg. 43).

[Figueiredo 2018] Jonathas Figueiredo. "Aplicação de algoritmos de aprendizagem por reforço para controle de navios em águas restritas". *Universidade de São Paulo* (2018) (citado na pg. 19).

[Gouw 2013] et al. Gouw S. C. "Factor viii products and inhibitor development in severe hemophilia a". *New England Journal of Medicine* (2013) (citado na pg. 1).

[Jumper 2021] et al. Jumper J. "Highly accurate protein structure prediction with alphafold". *Nature* (2021) (citado na pg. 6).

[K. K. Jensen 2018] et al. K. K. Jensen. "Improved methods for predicting peptide binding affinity to mhc class ii molecules". *Immunology* (2018) (citado na pg. 45).

[Konda e Tsitsiklis 2001] Vijay Konda e John Tsitsiklis. "Actor-critic algorithms". *Society for Industrial and Applied Mathematics* 42 (abr. de 2001) (citado na pg. 20).

[Kruskal 1964] J.B. Kruskal. "Nonmetric multidimensional scaling: a numerical method". *Psychometrika* (1964) (citado nas pgs. 21, 22).

[Lopes Tiago J. S. 2021] Rios Ricardo et al. Lopes Tiago J. S. Nogueira Tatiane. "Prediction of hemophilia a severity using a small-input machine-learning framework". *Nature* (2021) (citado nas pgs. 16, 17, 21, 26).

[Mancuso 2005] et al. Mancuso M. E. "Environmental risk factors for inhibitor development in children with hemophilia a: a case–control study". *Haematologica* (2005) (citado nas pgs. 1, 5).

[Mancuso ME 2017] Santagostino E. Mancuso ME. "Outcome of clinical trials with new extended half-life fviii/ix concentrates." *J Clin Med.* (2017) (citado na pg. 5).

[Pellegrini 2007] Jerônimo Pellegrini. *Processos de Decisão de Markov: um tutorial*. XIV. RITA, 2007 (citado na pg. 17).

[PM 2020] Mannucci PM. "Emophilia therapy: the future has begun." *Haematologica* (2020) (citado na pg. 1).

[Wenze Ding e Gong 2022] Kenta Nakai Wenze Ding e Haipeng Gong. "Protein design via deep learning". *Oxford Academic* (2022) (citado na pg. 1).

[Zhang 2022] et al. Zhang C. "Us-align: universal structure alignments of proteins, nucleic acids, and macromolecular complexes". *Nat Methods* (2022) (citado na pg. 16).

[Zhang 2004] et al. Zhang Y. "Scoring function for automated assessment of protein structure template quality". *Proteins: Structure, Function, and Bioinformatics* (2004) (citado nas pgs. 2, 14, 15).