VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

VYUŽITÍ VARIAČNÍCH AUTOENKODÉRŮ PRO ANCESTRÁLNÍ REKONSTRUKCI SEKVENCÍ

THE APPLICATION OF VARIATIONAL AUTOENCODERS FOR ANCESTRAL SEQUENCE RECONSTRUCTION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. PAVEL KOHOUT**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MILOŠ MUSIL, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání diplomové práce

Student:	**Kohout Pavel, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Bioinformatika a biocomputing
Název:	**Využití variačních autoenkodérů pro ancestrální rekonstrukci sekvencí** **The Application of Variational Autoencoders for Ancestral Sequence Reconstruction**
Kategorie:	Bioinformatika

Zadání:

1. Nastudujte teorii strojového učení bez učitele, variační enkodéry, vícenásobného zarovnání a ancestrální rekonstrukce.
2. Vyzkoušejte různé protokoly VAE na Pfam rodině ABhydroláz.
3. Navrhněte vícenásobné zarovnání pro referenční sekvence haloalkane dehalogenáz.
4. Vytvořte latentní prostor s využitím vícenásobného zarovnání a navrhněte ancestrální sekvence.
5. Validujte robustnost metody pomocí několika kol náhodné inicializace.
6. Vyhodnoťte výsledky a porovnejte navržené kandidátní sekvence s daty publikovanými v literatuře.

Literatura:

- Ding X, Zou Z, Brooks Iii CL. Deciphering protein evolution and fitness landscapes with latent space models. Nature communications. 2019; 10(1):1-3.
- Babkova P, Sebestova E, Brezovsky J, Chaloupkova R, Damborsky J. Ancestral haloalkane dehalogenases show robustness and unique substrate specificity. ChemBioChem. 2017; 18(14):1448-56.
- Hon J, Borko S, Stourac J, Prokop Z, Zendulka J, Bednar D, Martinek T, Damborsky J. EnzymeMiner: automated mining of soluble enzymes with diverse structures, catalytic properties and stabilities. Nucleic Acids Research. 2020.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Musil Miloš, Ing., Ph.D.**
Konzultant:	Mazurenko Stanislav, Ph.D., LL
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1. listopadu 2021
Datum odevzdání:	18. května 2022
Datum schválení:	22. října 2021

Abstrakt

Proteinové inženýrství je interdisciplinární vědní obor zabývající se návrhem vylepšených proteinů. Úspěšnou metodou využívanou pro návrh stabilnějších a aktivnějších proteinů je ancestrální rekonstrukce sekvencí. Tato metoda zkoumá evoluční vztahy mezi existujícími proteiny a za pomoci fylogenetických stromů vytváří jejich evoluční předchůdce, které kýžené vylepšené vlastnosti často vykazují. Proto nové a robustnější metody využívající matematické modely společně s obrovským množstvím sekvenčních dat by se mohly stát mocným nástrojem proteinového inženýrství. Tato diplomové práce se zabývá použitím variačních autoenkodérů jako alternativního přístupu k návrhu ancestrálních sekvencí oproti konvenčním metodám využívajícím fylogenetické stromy. V práci byly provedeny experimenty pro optimalizaci architektury a navrženy statistické metody pro evaluaci kvality modelů a generovaných sekvencí. Současně byly provedeny zkoušky robustnosti celé metody a navrhnuty a implementovány strategie pro generaci sekvence předků.

Abstract

Protein engineering is an interdisciplinary science concerned with the design of improved proteins. A successful method used to design more stable and active proteins is ancestral sequence reconstruction. This method explores the evolutionary relationships between existing proteins and uses phylogenetic trees to generate their evolutionary ancestors, which often exhibit the desired improved properties. Therefore, new and more robust methods using mathematical models together with huge amounts of sequence data could become a powerful tool for protein engineering. This thesis explores the use of variational autoencoders as an alternative approach to ancestral sequence design compared to conventional methods using phylogenetic trees. Experiments were performed to optimize the architecture and statistical methods were proposed to evaluate the quality of the models and the sequences generated. At the same time, robustness tests of the whole method were performed and strategies for ancestral sequence generation were proposed and implemented.

Klíčová slova

bioinformatika, strojové učení, generativní modely, variační autoenkodéry, proteinové inženýrství, ancestrální rekonstrukce, optimalizace proteinů

Keywords

bioinformatics, machine learning, generative models, variational autoencoders, protein engineering, ancestral reconstruction, protein optimization

Citace

KOHOUT, Pavel. *Využití variačních autoenkodérů pro ancestrální rekonstrukci sekvencí.* Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Miloš Musil, Ph.D.

Využití variačních autoenkodérů pro ancestrální rekonstrukci sekvencí

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Miloše Musila Ph.D. Další informace mi poskytli Dr. Stanislav Mazurenko a prof. Jiří Damborský. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Pavel Kohout
16. května 2022

Poděkování

Chtěl bych poděkovat konzultantovi Dr. Stanislavu Mazurenkovi za osobní, aktivní a nadšený přístup a matematický přehled během konzultací k diplomové práci. Dále bych rád poděkoval panu prof. Jiřímu Damborskému za poskytnutí skvělého zázemí pro práci. Také bych rád poděkoval všem členům Loschmidtových laboratoří za příjemnou atmosféru, inspirativní rady a prostředí. Velké díky jde i rodině za podporu a trpělivost během tvorby této diplomové práce.

Obsah

Kapitola 1

Úvod

Veškeré známé živé organismy na naší planetě jsou složeny z proteinů. Proteiny neboli bílkoviny jsou jednou z nejdůležitějších přírodních makromolekul zastávající nejrůznější funkce v živých organismech. Funkcionalita proteinů je velice pestrá, počínaje funkcí základní stavební jednotky v buňkách, transportem malých molekul do oblasti jejich působnosti až po enzymy katalyzující biologické procesy v lidském těle. Díky široké oblasti aplikací ve vědeckém i průmyslovém prostředí je studiu funkcí a návrhu nových proteinů s vylepšenými vlastnostmi věnována značná pozornost.

V minulých dekádách se i přes velké úsilí mnoha vědeckých týmů po celém světě nepodařilo odhalit a porozumět všem biologickým procesům, proteinům a jejich funkcím v živých organismech. Proto tato oblast stále zůstává, společně s návrhem optimálních proteinů, nevyřešenou problematikou. Neexistence systematického přístupu k návrhu a studiu velmi komplexních biologických procesů vedl ke vzniku vědního oboru proteinového inženýrství. Proteinové inženýrství se snaží na základě známých informací o proteinové struktuře nebo funkci vytvořit unikátní proteiny s požadovanými vlastnostmi. K racionálnímu návrhu nových proteinů využívá metody počítačového designu a metod řízené evoluce, kdy pomocí vhodných mutací proteinu na určitých pozicích aminokyselinové sekvence zavádí nové, či vylepšuje stávající, vlastnosti již existujících proteinů.

Jednou ze současných a robustních aplikací racionálního návrhu proteinů je metoda rekonstrukce ancestrálních sekvencí využívána především při tvorbě stabilnější a aktivnějších enzymů. Většina existujících přístupů tvorby ancestrálních sekvencí je založena na sestavení fylogenetických stromů, které jsou odvozeny pomocí statistických či bayesovských metod z mnohonásobného zarovnání relevantních sekvencí. Avšak i v oblasti rekonstrukce ancestrálních sekvencí se začíná experimentovat s použitím umělé inteligence a neuronových sítí. Jedním z přístupů je využití variančních autoenkodérů. Ve článku [14] bylo ukázáno, že latentní prostor variačních autoenkodérů je schopný zachytit evoluční schéma vstupních sekvencí a je tedy možnou alternativní metodou pro návrh ancestrálních sekvencí bez nutnosti tvorby fylogenetického stromu.

Cílem této práce je navrhnout, implementovat, otestovat a vyhodnotit program pro ancestrální rekonstrukci sekvencí za využití variančních autoenkodérů. Dále je analyzována robustnost celé metody, jelikož v dostupné literatuře ji doposud nebyla věnována dostatečná pozornost, a provedena detailní studie na rodině proteinů $\alpha\beta$hydroláz.

Práce je rozdělena do kapitol. Kapitola 2 je věnována úvodu do molekulární biologie a poskytuje základní potřebné informace o složení a struktuře proteinů společně s vysvětlením principů proteinového inženýrství. V kapitole 3 je poskytnut popis metod pro jednoduché i vícenásobné zarovnání sekvencí společně s uvedením nejznámější metod používaných

v praxi. Kapitola 4 je věnována problematice ancestrální rekonstrukce sekvencí. Nadále je vysvětlena problematika fylogenetických stromů s odvozováním ancestrálních sekvencí společně s vkládáním mezer. V kapitole 5 je věnována pozornost variačním autoenkodérům včetně motivace jejich použití v doméně ancestrální rekonstrukce. Kapitola 6 je zaměřena na uvedení datových sad a kapitola 7 prezentuje metody pro evaluaci kvality modelu. Dále jsou popsány navržené strategie pro dolování sekvencí z latentního prostoru a programová implementace řešení. V kapitole 8 je nejdříve věnována pozornost replikaci původního protokolu s původní architekturou na naši proteinovou rodinu *pfam* PF00561. Následně je uveden proces optimalizace architektury neuronové sítě společně s vyhodnocením experimentů optimalizované datové sady. Kapitola 9 obsahuje shrnutí dosažených výsledků s uvedením možného budoucího vývoje projektu.

Kapitola 2

Proteiny

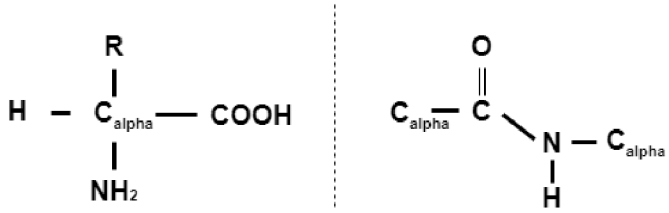
Proteiny neboli bílkoviny jsou součástí veškerých živých organismů na naší planetě. Jsou základními stavební kameny organické hmoty, hrají důležitou roli v řadě biologických procesů a svoji přítomností určují funkce jednotlivých buněk. Jejich studium je důležité pro pochopení fungování živých organismů a umožňuje jejich aplikaci v oblasti zdravotnictví, potravinářství, zemědělství a průmyslu využívající potenciál upravených proteinů.

Tato kapitola slouží jako úvod do biologické problematiky se zaměřením na složení, strukturu a syntézu proteinů. Kapitola vychází z knihy [1]. Dále budou popsány metody aplikované v proteinovém inženýrství.

2.1 Aminokyseliny

Aminokyselinou je v organické chemii označována každá molekula obsahující aminovou $-NH_2$ a karboxylovou $-COOH$ skupinu. Ve volné přírodě se vyskytuje více jak 300 aminokyselin, avšak pouze 20 (navíc sem lze zařadit i pyrolysin, selenocystein a N-formylmethionin) se vyskytuje jako základní stavební jednotka v proteinech a označují se jako biogenní aminokyseliny. Oproti organickým biogenní aminokyseliny mají aminovou a karboxylovou skupinu navázanou vždy na centrální α uhlíku, jak je ukázáno na obrázku 2.1 vlevo. Postranní řetězce R určují chemické vlastnosti jednotlivých aminokyselin jako polaritu, afinitu k vodě, kyselý nebo zásaditý charakter a další. Mají také zásadní vliv na strukturní uspořádání výsledných proteinů a na jejich stabilitu [63, 5].

Aminokyseliny jsou do polypeptidového řetězce spojeny pomocí peptidické vazby, která vzniká mezi aminovou skupinou první aminokyseliny a karboxylovou skupinou druhé při současném odštěpení molekuly vody. Schéma peptidické vazby je zachyceno na obrázku 2.1 vpravo. Kostra vytvořeného řetězce je tedy tvořena pouze aminovými, karboxylovými skupinami a alfa uhlíky jednotlivých aminokyselin s postranními řetězci směřujícími do prostoru. Úhly mezi rovinami kostry řetězce se nazývají torzní úhly.



Obrázek 2.1: Schéma aminokyseliny (vlevo) a peptidické vazby (vpravo)

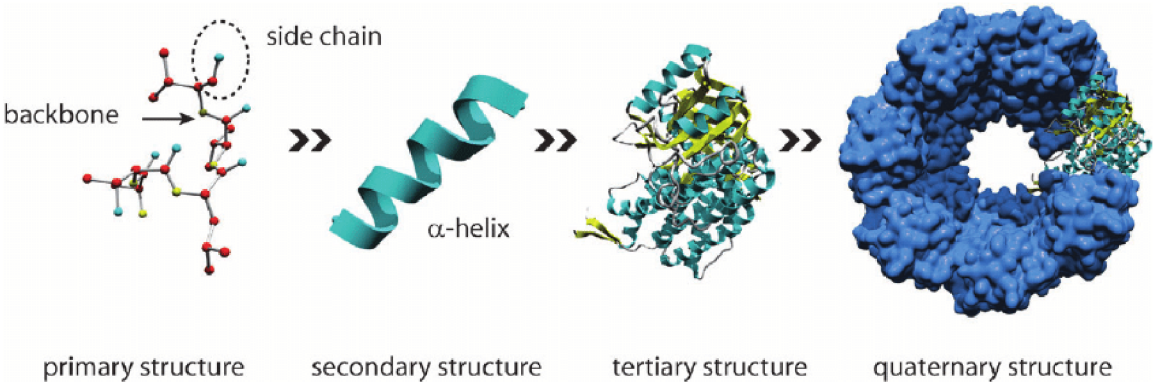
2.2 Struktura proteinů

Proteiny jsou makromolekulární organické látky složené z řetězce/ů aminokyselin typicky s délkou větší než 50 aminokyselin. Proteiny zaujímají v přirozeném prostředí energeticky nejvýhodnější prostorovou konformací přesně danou sekvencí aminokyselin v peptidickém řetězci.

Při popisu struktury proteinu rozlišujeme 4 základní úrovně organizace struktur [69], které jsou zachyceny na obrázku 2.2:

- **Primární struktura:** je dána lineárním pořadím aminokyselin v polypeptidovém řetězci a všechny následující úrovně na ní závisí. Každý protein má tuto sekvenci originální a jednoznačně ho definuje.
- **Sekundární struktura:** určuje lokální prostorové uspořádání polypeptidového řetězce. K nejčastějším zástupcům prostorovým konformací patří: **α-helix** a **β-list**. α-helix vzniká, když se polypeptidový řetězec ovijí kolem sebe sama. Vznikají tak stabilizující vodíkové můstky mezi aminokyselinami ležícími ve šroubovici nad sebou. Takto vzniklá šroubovice je tvořena pravidelnými otáčkami s velikostí 3,6 aminokyselinových zbytků. β-list je konformace tvořena dvěma řetězci či pouze jeho částmi běžícími prostorově vedle sebe. Stabilizován je vodíkovými můstky mezi aminovou skupinou prvního řetězce a karboxylovou skupinou druhého a naopak.
- **Terciální struktura:** charakterizuje prostorové uspořádání polypeptidového řetězce proteinu. Celková struktura je ovlivněna intramolekulárními vazebními interakcemi jako jsou disulfidické můstky, iontové vazby, van der Waalsovy síly, chemickými vlastnostmi postranních řetězců sdružující hydrofobní aminokyseliny ve vnitřních částech proteinu a sekvencí sekundárních struktur α-helixů a β-listů.
- **Kvartérní struktura:** je třírozměrné uspořádání proteinů skládajících se z více polypeptidových řetězců (podjednotek) spojenými nekovalentními vazbami, které dohromady tvoří jeden funkční protein zvaný oligomer nebo oligomerní protein. Výsledný oligomer utváří prostorové uspořádání podle stejných principů, jako je zmíněno v utváření terciálních struktur, vlivem Van der Waalsových sil, disulfidických a vodíkových můstků mezi proteinovými podjednotkami.

Prostorová struktura proteinů je jednou z nejdůležitější vlastností, která je determinována posloupností aminokyselin v peptidickém řetězci. Společně s chemickými vlastnostmi jednotlivých postranních řetězců určují funkci proteinu, kdy i jediná změna v primární struktuře může zcela zničit jeho funkci. Tento fakt, společně s obrovským prohledávaným prostorem, dělá z problematiky vylepšování vlastností proteinů intelektuálně i výpočetně náročnou operaci. Proto je nutné při návrhu nových metod vhodně integrovat a interpretovat veškeré dostupné informace o cílové rodině proteinů, společně s použitím vhodných heuristik pro dosažení maximální účinnosti. S komplexitou strukturní informace roste i náročnost procesu, kterým je tato informace získána. Proto databáze obsahující primární struktury mívají milióny záznamů, zatímco strukturní databáze obsahují pouze stovky tisíc. Situace je ještě horší v případě anotovaných dat, kde databáze dosahují v lepších případech tisíců záznamů. Pro mou práci jsem se zaměřil na početně nejbohatší primární strukturu sekvencí a jejich zdroje uvedu v sekci 3.1.



Obrázek 2.2: Grafické znázornění organizace struktury proteinu. Převzato z [24].

2.3 Syntéza proteinů

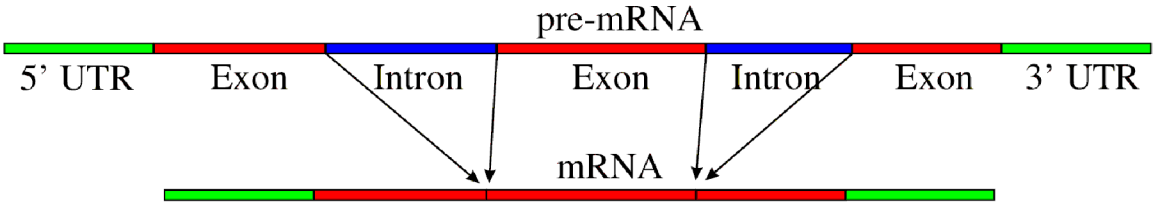
Proteiny vznikají v živých buňkách při procesu zvaném *proteosyntéza*. Děj *proteosyntézy* má počátek v jádře buňky v molekule DNA, kdy za pomoci řady menších biologických procesů je genetická informace překódována až do peptidického řetězce, který se následně složí do žádaného proteinu. Nejvýznamnějšími částmi proteosyntézy jsou děje transkripce a translace, doplněné o některé dílčí kroky, v závislosti zda se vyskytujeme v prokaryotickém či eukaryotickém organismu.

Transkripce je inicializační děj *proteosyntézy*, kdy dochází k přepisu informace z DNA do RNA a začíná rozvolněním krátkého úseku dvojšroubovice. Tak se uvolní úsek DNA pro přístup transkripčních molekul. Nejvýznamnější molekulou je RNA polymeráza, která nasedá na vlákno DNA v blízkosti oblasti promotoru. Promotor je úsek DNA sekvence, který je obvykle umístěn na začátku přepisovaného genu. Je složen ze specifických posloupností nukleotidů, jako je například TATA box u eukaryot či Pribnowova sekvence u prokaryot. RNA polymeráza se pohybuje po vlákně DNA a krok po kroku syntetizuje komplementární vlákno mediátorové RNA (*mRNA*). Je tak provedeno na základě komplementarity nukleových bází, kdy se páruje cytosin s guaninem a adenin s uracilem, který v RNA nahrazuje poslední základní nukleovou bázi thymin. Syntéza *mRNA* probíhá až po nalezení terminální sekvence, která práci RNA polymerázy ukončí. Tato sekvence se může vyskytovat jak v přepisovaném tak i vytvářeném řetězci [11].

Transkripce je v eukaryotních organismech doprovázena i dalšími procesy. To je dáno faktem, že v eukaryotickém genu rozlišujeme úseky tzv. exonů a intronů. Exony jsou segmenty překládané sekvence, které nesou informaci pro stavbu proteinu, zatímco introny žádnou takovou informaci nenesou. Proto se vzniklé vlákno nazývá prekurzorovou mediátorovou RNA (pre–mRNA). Následně je na pre–mRNA aplikován **sestřih**, čímž dojde k odstranění nekódujících částí sekvence za vzniku mRNA. Ve většině případů sestřih nemusí proběhnout jen jedním způsobem. Takto může z jednoho genu vznikat více proteinů a zvyšuje se bohatost eukaryotického proteomu. Vizualizace procesu sestřihu je vidět na obrázku 2.3.

K vláknu mRNA bývá v eukaryotickém systému přidávaná **5'čepička**, která ho chrání při transportu z jádra buňky a usnadňuje zahájení translace.

Translace je posledním dějem *proteosyntézy* a dochází při něm k překladu genetické informace na posloupnost aminokyselin spojených do peptidického řetězce. Překlad probíhá na buněčných organelách, zvaných ribozomy, za pomoci genetického kódu daného vždy tro-



Obrázek 2.3: Aplikace sestřihu na prekurzorovou mediátorovou RNA sekvenci obsahující introny (zelené části jsou součástí exonů). Převzato z [70].

jicí nukleotidů jdoucích za sebou (tzv. kodony). Genetický kód je zachycen na obrázku 2.4. Kodon kóduje vždy jednu z dvaceti aminokyselin, které jsou k ribozomu přinášeny pomocí transferové RNA (tRNA) obsahující antikodon. Antikodon je složený z trojce komplementárních bází k těm, které kódují nesenou aminokyselinu. Komplementarita bází kodonu a antikodonu zajistí, že se aminokyselina na tRNA naváže na správné místo v peptidickém řetězci. Trojce bází může tvořit až $4 \times 4 \times 4 = 64$ unikátních kombinací, které ale kódují pouze 20 aminokyselin. Proto jsou některé aminokyseliny kódovány pomocí více kodonů (kód je tzv. degenerovaný). Tato vlastnost se odráží i ve složení tRNA, která v antikodonu pro aminokyseliny s více charakteristickými kodony obsahuje speciální bázi inosin, schopnou se párovat s uracilem, cytosinem i adeninem (tzv. kolísavé párování bází).

báze v **druhé** poloze

báze v **první** poloze	U	C	A	G	báze v **třetí** poloze
U	UUU Phe UUC Phe UUA Leu UUG Leu	UCU Ser UCC Ser UCA Ser UCG Ser	UAU Tyr UAC Tyr UAA Stop UAG Stop	UGU Cys UGC Cys UGA Stop UGG Trp	U C A G
C	CUU Leu CUC Leu CUA Leu CUG Leu	CCU Pro CCC Pro CCA Pro CCG Pro	CAU His CAC His CAA Gln CAG Gln	CGU Arg CGC Arg CGA Arg CGG Arg	U C A G
A	AUU Ile AUC Ile AUA Ile AUG Met	ACU Thr ACC Thr ACA Thr ACG Thr	AAU Asn AAC Asn AAA Lys AAG Lys	AGU Ser AGC Ser AGA Arg AGG Arg	U C A G
G	GUU Val GUC Val GUA Val GUG Val	GCU Ala GCC Ala GCA Ala GCG Ala	GAU Asp GAC Asp GAA Glu GAG Glu	GGU Gly GGC Gly GGA Gly GGG Gly	U C A G

Obrázek 2.4: Tabulka pravidel pro přepis nukleotidové sekvence do peptidického řetězce (genetický kód). Aminokyseliny jsou popsané pomocí třípísmenných zkratek. Označení stop značí kodon pro ukončení translace. Převzato z [36].

Proces převedení genetické informace na cílový produkt se nazývá exprese genu. Exprese všech genů není vždy stejná a je závislá na buňce, ve které k ní dochází. Díky tomu v organismu dochází k diferenciaci buněk, které mohou efektivně vykonávat svoji činnost.

2.4 Proteinové inženýrství

Proteinové inženýrství je relativně mladou vědní oblastí, která si klade za cíl studium proteinů současně s návrhem jejich modifikací s požadovanými a vylepšenými vlastnostmi. Aplikace postupů proteinového inženýrství není omezena pouze na akademické prostředí, ale velká pozornost je ji věnována i v oblasti průmyslu a aplikací v medicíně [35, 57]. Mezi dva hlavní přístupy, používané v proteinovém inženýrství, patří racionální návrh a metoda řízené evoluce.

Racionální návrh je starší z přístupů k návrhu nových mutantů. Základem je využívání rozsáhlých znalostí obsažených v biologických databázích společně s pečlivou analýzou struktury za účelem vytypovaní vhodných úprav pro dosažení požadovaných vlastností proteinu. Typickým příkladem může být návrh mutace proteinu záměnou původní aminokyseliny za jinou. Pro získání vhodných mutací se využívá bioinformatických nástrojů, které jsou s kvalitními vstupními daty esenciální složkou racionální návrhu. Konkrétními nástroji lze například zlepšit termostabilitu [47] či predikovat škodlivost aplikovaných mutací. Bioinformatické nástroje a algoritmy jsou založeny na heuristikách a mechanismech motivovaných přírodními procesy. Jedním z aplikovaných mechanismů, zejména při návrhu stabilnější proteinů, je ancestrální rekonstrukce sekvencí, která se snaží z příbuzných sekvencí sestavit často stabilnější evoluční předchůdce (viz kapitola 4). S neustále rostoucím počtem vysoce výkonných experimentálních metod přibývá i počet dostupný dat, což má za důsledek častější aplikace metod strojového učení v proteinovém inženýrství. Ty jsou v obrovském množství dat schopny rozeznávat důležité vzory, které mohou být použity při případné následné analýze.

Druhým přístupem je *metoda řízené evoluce*. Metoda je postavená na simulování přirozené evoluce proteinů aplikací náhodných mutací, při kterých dojde k výměně jedné aminokyseliny za druhou. Náhodných změn může být dosaženo například umělým zvýšením chybovosti při replikaci DNA. Modifikované proteiny s nejlepšími vlastnostmi jsou vybrány pro další kolo evoluce, přičemž se tento proces opakuje, dokud není dosaženo přijatelných výsledků. Metoda *řízené evoluce* nepoužívá bioinformatické nástroje a je nezávislá na kvalitě vstupních dat. Je však velice náročná na laboratorní práci a finanční prostředky. Laboratoře praktikující metody řízené evoluce jsou často vybaveny roboty, které tento proces zcela automatizují.

Ve skutečném světě se kombinují oba dva přístupy. Pomocí racionálního návrhu se vytypují pozice aminokyselin, které budou podrobeny metodě řízené evoluci. Tím se možný počet modifikovaných proteinů výrazně sníží, a proto je tento přístup finančně i časově méně náročný.

Kapitola 3

Databáze a zarovnání sekvencí

V předchozí kapitole 2 jsme se dozvěděli o složení a syntéze proteinů. Také jsem se zmínil o přístupu pro návrh jejich vylepšených variant zvaný proteinové inženýrství. V této kapitole blíže popíši databáze shromažďující biologické sekvence a nástroje pro jejich následné vyhledávání. Nadále zmíním základní úkon bioinformatiky, zarovnání sekvencí, umožňující identifikovat podobné regiony v DNA, RNA nebo proteinových sekvencích, které mohou způsobovat funkcionální, strukturální či evoluční vztahy mezi zkoumanými sekvencemi. Vícenásobné zarovnání sekvencí je důležité pro následné zkoumání evolučních závislostí mezi jednotlivými sekvencemi a bude vysvětleno na konci této kapitoly.

3.1 Databáze biologických sekvencí

Jedna z klíčových úloh bioinformatiky je spravování databází biologických dat, vytváření nástrojů pro následnou analýzu záznamů a integrace již existujících nástrojů pro komplexní vyhodnocení dat z více dostupných databází s přívětivou grafickou vizualizací pro koncového uživatele.

UniProt

Databáze *UniProt*[12] je jednou z nejvýznamnějších databází proteinových sekvencí. Je volně dostupná a sdružuje velké množství informací o biologických funkcích. *Uniprot* nabízí následné databáze: *UniProtKB*, *UniParc* a *UniRef*.

UniProtKB (*UniProt* KnowledgeBase) se skládá z dvou dílčích databází, které se od sebe liší kvalitou, jakou byly jednotlivé záznamy v databázích anotovány. První z nich se nazývá *UniProtKB/Swiss-Prot*. Tato databáze obsahuje neredundantní, manuálně anotované a ověřené záznamy. Data obsažená ve *Swiss-Prot* jsou získaná z vědecké literatury a jsou pečlivě analyzována. Anotace záznamů jsou pravidelně ověřovány, aby byla zaručena koherence s aktuálními vědeckými poznatky. Druhou databází *UniProtKB* je *UniProtKB/TrEMBL*, která obsahuje vysoce kvalitně výpočetně analyzované záznamy rozšířené o automaticky dodané anotace. Tato databáze byla zavedena jako odpověď na rychle rostoucí počet biologických sekvencí, které nebyla *Swiss-Prot* schopna začleňovat mezi své záznamy, kvůli požadované kvalitě anotací vyžadující značné pracovní úsilí.

UniParc je databáze neredundantních sekvencí. Záznamy jsou extrahovány z více databází a identické sekvence jsou sloučeny do jedné, bez ohledu na organismus jejich původu. Záznamy neobsahují anotace, ale odkazy na původní zdroje, kde je možné tyto informace

dohledat. Pokud se data v původní databázi změní, *UniParc* tyto změny reflektuje a navíc si udržuje historii o změnách daného záznamu.

UniRef (*UniProt* Reference Clusters) je složena ze tří databází obsahující množiny sekvencí z *UniProtKB* a *UniParc* shlukované do jednoho záznamů dle dané úrovně identity sekvencí. *UniRef100* shlukuje identické sekvence a sekvenční fragmenty do jednoho *UniRef* záznamu. Tyto sekvence jsou nadále shlukovány do množin sdílející 90% a 50% podobnost v databázích *UniRef90*, respektive *UniRef50*.

Pro vyhledávání v *uniprot* databázi se využívá **BLAST**. BLAST (Basic Local Alignment Search Tool) je heuristický algoritmus umožňující porovnání sekvencí v databázích a navíc rozpoznat obdobné sekvence mající podobnost nad předem definovanou úroveň. Algoritmus byl publikován již v roce 1990 [2] a od té doby je jedním ze základních nástrojů bioinformatiky.

Algoritmus rozdělí vstupní sekvenci, oproti které hledáme v mnohem vetší databázi podobné sekvence, do slov o velikosti W (pro proteiny typicky 3, DNA 11). Pro každé slovo je vytvořena množina všech alternativních slov s jednou záměnou znaku ohodnocenou podle podobnosti oproti původnímu slovu například pomocí matice BLOSUM62 (viz. 3.2.1) a pouze slova mající míru podobnosti větší něž určitý práh jsou ponechána v tzv. tabulce sousednosti. V druhé části algoritmus detekuje výskyt slov z tabulky sousednosti v porovnávané sekvenci. V případě nalezení slova je tento úsek z obou stran rozšiřován a jakmile skóre přesáhne minimální mez, je tato sekvence reportována na výstupu algoritmu.

Pfam

Pfam[4] je databáze proteinových rodin, která obsahuje jejich anotace a mnohonásobné zarovnání sekvencí vygenerované pomocí statistického modelu zvaného Skrytý Markovův model. Databáze umožňuje vyhledávat a rozdělovat proteiny podle jejich funkčních úseků, běžně zvaných jako domény. V současné době *Pfam* databáze verze 35.0 disponuje 19632 záznamy a umožňuje generovat vysokoúrovňové shlukování do tzv. klanů. Klan je sbírka položek *Pfam*, které jsou příbuzné podobností sekvence, struktury nebo profilu-HMM.

Každá rodina Pfam se skládá z pečlivě sestaveného zarovnání obsahujícího malou sadu reprezentativních členů rodiny (tzv. *seed* sekvence), profilových skrytých Markovových modelů (profilových HMM) sestavených ze zárodečného zarovnání a automaticky generovaného úplného zarovnání. To obsahuje všechny detekovatelné proteinové sekvence patřící do rodiny, jak byly definovány na základě prohledávání profilových HMM v databázích primárních sekvencí.

Klíčovým nástrojem pro tvorbu *pfam* rodin a zařazování jednotlivých sekvencí k těmto rodinám je HMMER. Tento nástroj je volně dostupný a běžně využívaný balíček pro sekvenční analýzu. Je používán při identifikaci homologních sekvencí a tvorbě vícenásobného zarovnání. HMMER detekuje homologní sekvence porovnáváním tzv. HMM-profilů oproti jedné sekvenci či celé databázi. HMM-profil je speciální varianta statistického modelu zvaného Skrytý Markovův model (Hidden Markov state) navržený pro doménu biologických sekvencí.

3.2 Zarovnání sekvencí

Zarovnání řetězců je jednou ze základních úloh bioinformatiky. Poskytuje nám informaci o tom, jak se od sebe jednotlivé organismy liší, zda-li se daný gen nachází i u jiných organismů a pomáhá určit funkce jednotlivých genů. V rámci evolučního procesu se mohou od

sebe odlišit dvě sekvence aminokyselin či nukleotidů mající původ ve společném předkovi nejčastěji kvůli:

- **Mutaci**: mutace je záměna znaku jednoho za druhý a dochází k ní během evoluce poměrně často. Zapříčiněna bývá chybou při duplikování DNA, či vlivem prostředí (záření atd.).
- **Vložení/odstranění znaku**: dochází k němu méně často. Mění se délka řetězce a při zarovnání je nutné uvažovat vložení mezery.

Účelem zarovnání sekvencí je vyjádřit pomocí skórovací metriky podobnost dvou sekvencí a přiřadit znaky první posloupnosti k druhé. Na základě charakteru řešené úlohy a postihu za vkládání mezer rozlišuje 3 přístupy:

- **Globální zarovnání**: je porovnání dvou sekvencí, které jsou vnímány jako celek a každé vložení mezery je penalizováno. Lze použít například u porovnání dvou genů ze stejných organismů.
- **Semi–globální zarovnání**: ignoruje penalizování mezer na začátku a na konci řetězce. Využívá se v případech, kdy hledáme výskyt kratší sekvence v delší a využívá se například k hledání genu v celém genomu daného organismu.
- **Lokální zarovnání**: slouží k zarovnání výrazně kratší sekvence k delší a snaží se najít všechny výskyty kratšího podřetězce. K penalizaci za mezery přidává i penalizaci za změnu znaku čímž se liší od předchozích dvou přístupů.

Zarovnání sekvencí je složitý, a při větším počtu sekvencí, výpočetně náročný problém. Výpočetní složitost naivních algoritmů dosahuje exponenciální třídy, což je pro delší sekvence nepřijatelné. Proto jsou jednotlivé algoritmy postaveny na heuristikách snižujících výpočetní náročnost. Navíc pro dosažení co nejlepších výsledků blízkým reálnému světu je nutné uvažovat rozdílné pravděpodobnosti mutací jednotlivých nukleotidů či aminokyselin. Problémy zmíněné výše se snaží řešit například skórovací matice a dynamické programování.

3.2.1 Skórovací matice

V reálných případech zarovnání nukleotidových či aminokyselinových sekvencí musíme kromě vložení a odebrání uvažovat i se záměnou jednotlivých znaků za jiné. Při naivním přístupu jsou pokuty pro všechny záměny shodné. Tento přístup není vhodný, jelikož neodráží prostorové ani fyzikální vztahy mezi jednotlivými aminokyselinami či nukleotidy, pro které je pravděpodobnější záměna členů uvnitř skupin pyrimidinů (cytosin, thymin) či purinů (adenin, guanin) než nahrazení prvku první skupiny prvkem z druhé. Stejná situace je i v případě aminokyselin, kdy je změna aminokyseliny na jinou se stejnými fyzikálními a chemickými vlastnostmi pravděpodobnější něž záměna za zcela odlišnou. Typickým příkladem by mohlo být nahrazení hydrofobní aminokyseliny za kyselinu s nábojem (např. methionin za kyselinu asparagovou), která má pro celkovou povahu proteinu a jeho vlastnosti mnohem vetší důsledky než záměna jedné hydrofobní aminokyseliny za druhou (např. methionin za valin). Podobně je to i s počty mutací, kdy jednobodové mutace kodonů jsou pravděpodobnější než vícebodové.

Aplikací znalostí pravděpodobností změn aminokyselin mezi sebou lze sestavit takzvanou skórovací matici. Ta v jednotlivých polích obsahuje číselné ohodnocení pravděpodobností pro zachování, případně přeměnu, dané aminokyseliny na jinou. Hodnocení zarovnání

je vypočítáno jako suma ohodnocení všech pozic dle skórovací matice společně s penalizací za mezery, které se nadále rozlišují podle své délky (z pohledu evoluce je mnohem pravděpodobnější vznik menšího počtu delších mezer, než velkého počtu krátkých mezer).

Sestavení obecné skórovací matice je velmi komplikované, proto se často využívá přístup, kdy jsou hodnoty určeny experimentálně na základě rychlosti přeměny mezi sekvencemi, u kterých je známa příbuznost (sekvence jedné proteinové rodiny). Příkladem skórovacích matic můžeme uvést PAM(Point Accepted Mutation) a BLOSUM-XX matice, kde XX značí procentuální míru podobnosti sekvencí. Matice BLOSUM-62 pro sekvence s 62 procentní podobností je zachycena na obrázku 3.1.

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	**9**	-1	-1	-3	0	-3	-3	-3	-4	-3	-3	-3	-3	-1	-1	-1	-1	-2	-2	-2
S	-1	**4**	1	-1	1	0	1	0	0	0	-1	-1	0	-1	-2	-2	-2	-2	-2	-3
T	-1	1	**4**	1	-1	1	0	1	0	0	0	-1	0	-1	-2	-2	-2	-2	-2	-3
P	-3	-1	1	**7**	-1	-2	-1	-1	-1	-1	-2	-2	-1	-2	-3	-3	-2	-4	-3	-4
A	0	1	-1	-1	**4**	0	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1	-2	-2	-2	-3
G	-3	0	1	-2	0	**6**	-2	-1	-2	-2	-2	-2	-2	-3	-4	-4	0	-3	-3	-2
N	-3	1	0	-2	-2	0	**6**	1	0	0	-1	0	0	-2	-3	-3	-3	-3	-2	-4
D	-3	0	1	-1	-2	-1	1	**6**	2	0	-1	-2	-1	-3	-3	-4	-3	-3	-3	-4
E	-4	0	0	-1	-1	-2	0	2	**5**	2	0	0	1	-2	-3	-3	-3	-3	-2	-3
Q	-3	0	0	-1	-1	-2	0	0	2	**5**	0	1	1	0	-3	-2	-2	-3	-1	-2
H	-3	-1	0	-2	-2	-2	1	1	0	0	**8**	0	-1	-2	-3	-3	-2	-1	2	-2
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	**5**	2	-1	-3	-2	-3	-3	-2	-3
K	-3	0	0	-1	-1	-2	0	-1	1	1	-1	2	**5**	-1	-3	-2	-3	-3	-2	-3
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	**5**	1	2	-2	0	-1	-1
I	-1	-2	-2	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	**4**	2	1	0	-1	-3
L	-1	-2	-2	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	**4**	3	0	-1	-2
V	-1	-2	-2	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	**4**	-1	-1	-3
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	**6**	3	1
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	**7**	2
W	-2	-3	-3	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	**11**

Obrázek 3.1: Příklad skórovací matice BLOSUM-62 pro sekvence mající 62 procentní podobnost. Celkové skóre je vypočteno jako suma ohodnocení přes jednotlivé pozice zarovnání. Převzato z [44].

3.2.2 Dynamické programování

V případě existence skórovací matice pro zkoumané sekvence lze využít hrubý přístup, kdy sestavíme všechny kombinace zarovnání, ohodnotíme je a to s nejvyšším skóre prohlásíme za výsledek. Bohužel i pro relativně krátké sekvence například o délce 95 a 100 znaků je počet všech možných zarovnání roven přibližně 55 miliónům. S delšími sekvencemi se tento přístup stává nepoužitelný, a proto je nutné využít nápaditější řešení. Příkladem je aplikace tzv. dynamického programování, kdy je problém rozdělen na řady menších podproblémů. Algoritmus na zarovnání dvou sekvencí byl sestaven v roce 1970 Saulem B. Needlemanem a Christianem D. Wunschem [48] a stal se základním kamenem algoritmů v bioinformatice.

V problematice zarovnání dvou sekvencí máme v každém kroku tři možnosti. První možností je nevložit mezeru do žádného ze dvou řetězců a první dva znaky zarovnáme, nebo vložíme mezeru do první či druhé sekvence. Přičemž jednotlivé kroky můžeme pro příklad ohodnotit následovně:

- 1 za shodu
- 0 za neshodu

- -1 za vložení mezery

Tak se nám řešený problém rozdělí na tři dílčí podproblémy, které v sobě zahrnují ohodnocení jednoho kroku výpočtu dle jednotlivých operací. Celkové skóre je ovšem dáno tím, jak dopadne zbytek celého zarovnání, a proto krok výpočtu aplikujeme na každý z nově vygenerovaných stavů. Tímto způsobem bychom pokračovali dokud nezpracujeme všechny znaky.

Při důkladnějším studiu vznikajících kombinací v jednotlivých větvích výpočtu zjistíme, že se určité kombinace opakují, a proto není nutné počítat zarovnání ve všech větvích stromu opakovaně. Navíc bylo dokázáno, že všechny možné kombinace ze stromu výpočtu lze převést do grafu, jehož uzly jsou organizovány do 2D matice (obrázek 3.2 vlevo). Díky tomu lze problém zarovnání sekvencí transformovat na problém nalezení cesty v grafu. Celý výpočet můžeme reprezentovat jako tabulku, kde sloupce reprezentují jednu sekvenci a řádky druhou. Současně se první sloupec a řádek tabulky nastaví na hodnoty zvyšující se penalizace za vkládání mezer do první respektive druhé sekvence (obrázek 3.2 vpravo). Hodnoty vnitřních buněk tabulky jsou dány jako maximum ze tří možností (obrázek 3.2 uprostřed):

- převzetí hodnoty ze shora s aplikací pokuty za vložení mezery
- převzetí hodnoty zleva s přičtením pokuty za vložení mezery
- převzetí hodnoty od levého horního souseda s přičtením hodnoty za shodu/neshodu znaků

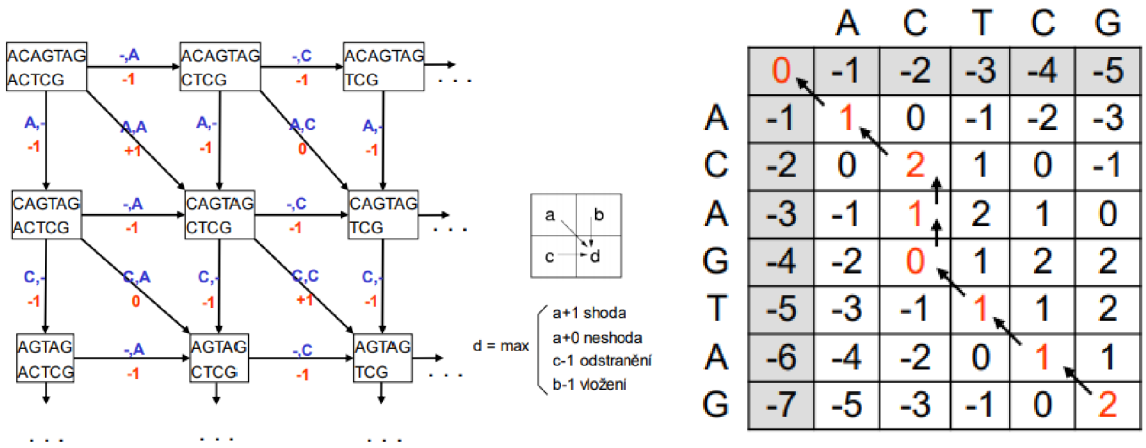
V pravém dolním rohu vypočtené tabulky se nachází ohodnocení optimální varianty zarovnání. Nyní musíme získat tvar výsledného zarovnání, který získáme průchodem grafu z levého dolní rohu do pravého horního. Rekonstrukce cesty probíhá vždy pohybem do sousední levé, horní či diagonální levé horní buňky, ze které byla vypočtena hodnota aktuální buňky (viz obrázek 3.2 vpravo). V určitých případech může existovat více cest symbolizující stejně kvalitní řešení zarovnání.

Výše uvedená varianta algoritmu slouží pro provedení globálního zarovnání. Existují i upravené varianty, které slouží k výpočtu semi–globálního případně lokálního zarovnání. Například při vytváření semi–globálního zarovnání je první řádek a sloupec inicializován na nuly, díky čemuž ignorujeme mezery na začátku sekvence. Navíc v posledním řádku nepenalizujeme vložení mezery.

3.3 Vícenásobné zarovnání

Bohužel při získávání znalostí o rozsáhlých proteinových rodinách za účelem detekce rysů, které stojí za jejími vlastnostmi, si nevystačíme pouze s porovnáním dvou sekvencí. Stejně tak tomu je u evoluční rekonstrukce sekvencí, kde zarovnání celé skupiny sekvencí je esenciální složkou kvalitní ancestrální rekonstrukce [67].

Při uvažování vícenásobného zarovnání narážíme, oproti porovnání dvou sekvencí, na několik problémů. První logickou otázkou může být hodnocení kvality zarovnání. U dvou sekvencí je situace jednoduchá, kdy zvyšujeme skóre za shodné znaky a v případě vložení mezery či neshody jej penalizujeme, ale se třemi a více sekvencemi se situace komplikuje. Prvním nabízeným řešením je zavedení podmínky, kdy zvyšujeme skóre jen v případě rovnosti všech znaků na dané pozici v každé sekvenci. Tento přístup funguje dobře jen při



Obrázek 3.2: Dynamické programování aplikované na problematiku zarovnání sekvencí konkrétně na sekvence ACAGTAG a ACTCG. Vlevo: ukázka převodu stromu výpočtu na graf s uzly organizovanými do 2D matice. Uprostřed: grafické znázornění výpočtu vnitřních buněk tabulky. Vpravo: výsledné optimální zarovnání odpovídá cestě v grafu z pravého dolního do levého horního rohu. Převzato z [44].

zarovnávání velmi podobných sekvencí. Lepší možností se jeví metoda **sumy párů** postavená na myšlence, kdy větší skóre zarovnání značí vysokou podobnost mezi jednotlivými dvojicemi sekvencí. Pro každý sloupec zarovnání je vypočteno skóre všech dvojic znaků a výsledek sloupce je dán součtem dílčích ohodnocení. Problémem je nutnost porovnat vysoký počet párů znaků.

Jednou z možností, jak vytvořit vícenásobné zarovnání, je využití obecnější aplikace dynamického programování pro více sekvencí. V případě výskytu n sekvencí budeme uvažovat n–rozměrnou matici grafu. Tento přístup je možný pouze pro malý počet relativně krátkých sekvencí, jelikož časová složitost algoritmu roste exponenciálně s počtem sekvencí.

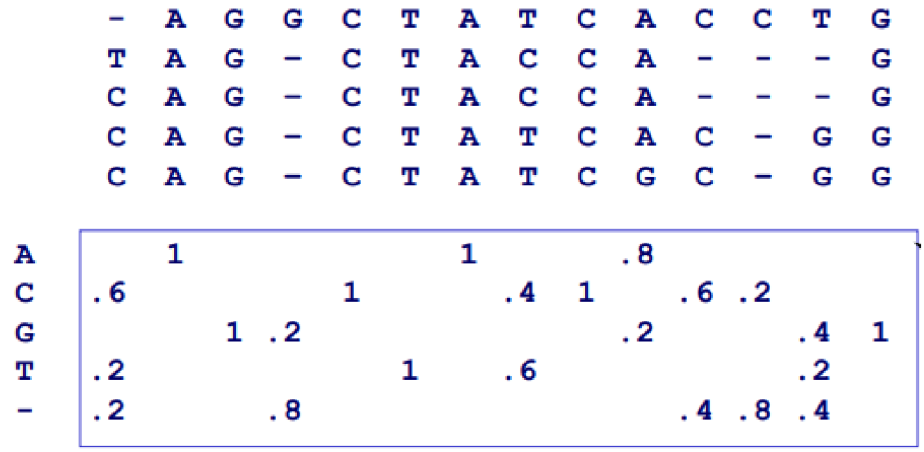
Pro efektivní řešení bylo nutné vymyslet odlišný přístup, kterými jsou progresivní metody.

3.3.1 Progresivní metody

Progresivní metody vycházejí z myšlenky, kdy zarovnáním dvou řetězců z množiny k sekvencí vytvoříme množinu *k-1* sekvencí a tak postupujeme, dokud nejsou zarovnány všechny řetězce. S redukcí počtu sekvencí se postupem výpočtu dostaneme k problému, jak zarovnat sekvenci k zarovnání nebo zarovnání k zarovnání a jak jej ohodnotit.

Hodnocení vícenásobného zarovnání lze reprezentovat za pomoci tzv. profilu. Profil je matice, která značí procentuální zastoupení znaků ve sloupcích (obrázek 3.3). Profily dvou zarovnání můžeme použít i pro ohodnocení jejich vzájemného zarovnání. To je vypočteno vždy ve dvojicích odpovídajících sloupců jednotlivých profilů, jako suma křížově vynásobených procentuálních hodnot zastoupení znaků v profilech s hodnotou skórovací matice pro tuto dvojici, případně s penalizací za vložení mezery.

Při postupném přidávání sekvencí do zarovnání upravujeme vždy jen vkládané sekvence nikoliv již vytvořené zarovnání. Proto mezivýsledek, který je jednou vložen, nelze v následujících krocích výpočtu již upravovat. To znamená, že pořadí výběru dvojic výrazně ovlivňuje kvalitu výsledného zarovnání. Kvalitní výsledky dostaneme vždy, když budeme zarovnávat nejvíce si podobné dvojice sekvencí. To lze provést díky pomocnému stromu tzv. *Guided*



Obrázek 3.3: Ve spodním obdélníku je profil zarovnaných sekvencí vyjadřující procentuální zastoupení znaků ve sloupcích. Převzato z [43].

tree založeného na matici vzdálenosti. Tvorba *Guided tree* podle metody UPGMA je blíže popsána v sekci 4.2.1.

Základním problém progresivních metod je jejich výpočetní náročnost a dlouhá doba běhu. Nejnáročnější částí algoritmu je porovnání všech dvojic sekvencí a tvorba matice vzdáleností. Pokud vyžadujeme přesné zarovnání každé dvojice sekvencí, použijeme metodu dynamického programování. Při 100 sekvencích délky N řešení vede na 4950 (100 $\times$ 99 / 2) porovnání sekvencí s N^2 položek v matici pro jednu dvojici. Často se však spokojíme s hodnotami pouze ukazujícími míru podobnosti sekvencí. Proto lze využít i méně výpočetně náročné metody počítání *k-tic*, kdy pro každou sekvenci sestavíme vektor značící počet výskytů podřetězců v nich obsažených. Podřetězce jsou tvořeny všemi možnými kombinacemi *k-tic* dané abecedy. Míra podobnosti je pak určena euklidovskou vzdáleností mezi dvěma vektory četností podřetězců. Přičemž platí, že dvě sekvence jsou si podobnější, když je vzdálenost nižší.

Dalším problémem progresivních metod je nemožnost měnit již utvořené zarovnání, protože ani nejlepší možný výběr pořadí sekvencí nemusí vést k optimálnímu řešení.

3.3.2 Iterativní metody

Fakt, že i nejlepší možné pořadí postupného výběru sekvencí nemusí vést optimálnímu řešení, vyžaduje přístup, který umožní vytvořené zarovnání nějakým způsobem nadále upravovat. Proto vznikly tzv. iterativní metody a jeden z možných postupů výpočtu lze popsat následovně:

- Na vstupu je zadán pomocný strom a výsledek vícenásobného zarovnání.
- Pomocný strom se rozstřihne na dvě části a z každé se sestaví vícenásobné zarovnání. Jako návod pro sestavení se použijí vzniklé podstromy.
- Z obou zarovnání sestavíme výsledné zarovnání a ohodnotíme jeho skóre.
- Pokud skóre nově sestaveného zarovnání je lepší než stávající, nahraď jej a pokračuj ve výpočtu dokud je dosahováno lepších výsledků nebo nejsou prozkoumány všechny možnosti rozdělení stromu. Strom je postupně systematicky dělen od kořene k jednotlivým listům stromu.

3.4 Zástupci progresivní a iterativních metod

CLUSTAL

Techniky popsané v sekci progresivních metod jsou základem jednoho z nejrozšířenějších algoritmů CLUSTAL [26]. Dnes je již CLUSTAL celá rodin algoritmů pro zarovnání sekvencí, která pod sebou sdružuje řadu variací zlepšující přesnost a výkon. Základní postup společný pro všechny varianty je dán následovně:

- Porovnání všech dvojic sekvencí mezi sebou a sestavení tabulky podobnosti
- Sestavení pomocného stromu
- Postupné zarovnání dvojic sekvencí podle pomocného stromu

CLUSTAL používá při sestavování tabulky podobnosti princip BLAST/FASTA. Strom je postupně sestavován dvojicemi s největší mírou podobností stejně, jak je tomu u metody UPGMA (sekce 4.2.1). Při výpočtu se bere v úvahu procentuální zastoupení znaků (profil) a jako skórovací matice se používají pouze vybrané verze BLOSUM a PAM.

Vylepšená verze **CLUSTALW** [64] používá pro sestavení tabulky podobnosti princip dynamického programování a metodu Neighbour-Joining (sekce 4.2.1). Při výpočtu ohodnocení zarovnání je navíc každá sekvence váhována podle své významnosti s vylepšenou penalizací za vložení mezery. Skórovací matice BLOSUM-XX jsou vybírány dynamicky podle podobnosti sekvencí.

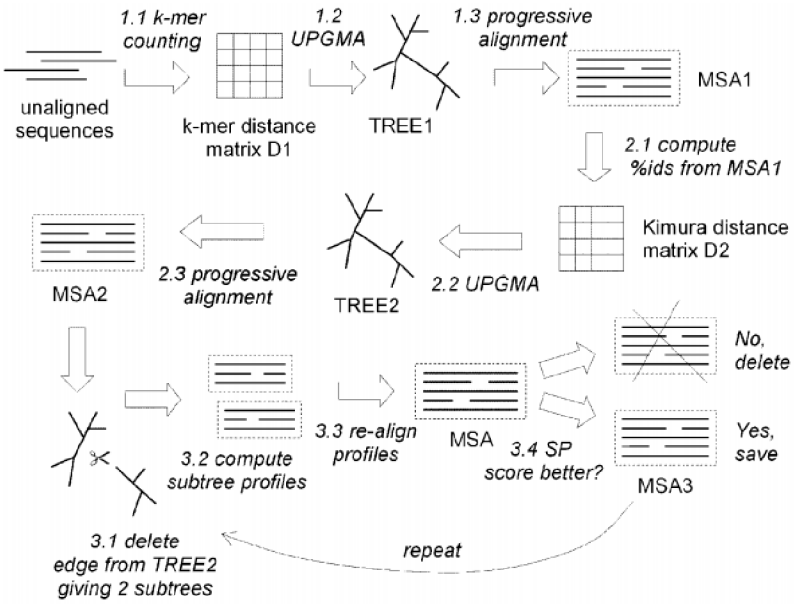
Nejnovější verzí je **CLUSTALΩ** [59], který pro tvorbu pomocného stromu používá metodu mBed [6] s časovou složitostí O($NlogN$) oproti O(N^2) u CLUSTALW při zachování stejné přesnosti. Sekvence jsou reprezentovány vektory, které značí vzdálenost od referenční sekvence, proto mohou být extrémně rychle shlukovány pomocí standardních metod jako K-means či UPGMA. Samotné zarovnání je provedeno za pomoci skrytých markovových modelů [60].

MUSCLE

MUSCLE (multiple sequence comparison by log-expectation) je zástupce iterativních metod. Prvně byl publikovaný Robertem C. Edgarem v roce 2004 [16] a od té doby se stal oblíbeným nástrojem pro tvorbu vícenásobného zarovnání. Princip jeho fungování lze rozdělit na tři části:

- **Výpočet předběžného zarovnání:** vícenásobné zarovnání je zkonstruováno na základě matice podobností vytvořené počítáním k-tic a pomocného stromu metodou UPGMA.
- **Přepočet zarovnané sekvence:** ze zarovnání z prvního kroku se vypočte nová matice vzdáleností skrze přesnější Kimurovu vzdálenost. Poté se sestaví druhý pomocný strom a vytvoří se nové zarovnání.
- **Aplikace iterační metody:** na strom a zarovnání z předchozího kroku.

Grafické znázornění algoritmu MUSCLE lze vidět na obrázku 3.4.



Obrázek 3.4: Grafické znázornění výpočtu metody MUSCLE. Převzato z [43].

Kapitola 4

Ancestrální rekonstrukce proteinových sekvencí

V předchozí kapitole 3 byly uvedeny metody zarovnání sekvencí nezbytné pro porovnání biologických sekvencí společně s metodou vícenásobného zarovnání sekvencí, které nabízí nástroj na extrakci vzorů z velkých proteinových rodin a je kritickým vstupem pro ancestrální rekonstrukci sekvencí. V této kapitole bude uvedena motivace a metoda ancestrální rekonstrukce sekvencí s konkrétními pozitivními příklady aplikace. Nadále budou popsány důležité kroky pro ancestrální rekonstrukci sekvencí, kterými je tvorba kořenových fylogenetických stromů a odvození sekvence předků.

4.1 Metoda ancestrální rekonstrukce sekvencí

V současné době jsou známy velké proteinové rodiny složené z jedinců mající velikou odlišnost v aminokyselinové sekvenci. Analýza vícenásobného zarovnání takovýchto rodin umožňuje identifikaci konzervovaných klíčových aminokyselin, například zbytků aktivních míst, které jsou charakteristické pro celou rodinu proteinů. Taková analýza však zřídka odhalí soubor reziduí, které jsou zodpovědné za funkční rozmanitost pozorovanou u velkých proteinových rodin. Důvodem je skutečnost, že během evoluce, trvající mnoho miliónů let, se v zkoumaných sekvencích nahromadilo velké množství neutrálních i epistatických mutací. Zejména neutrální mutace vytváří sekvenční šum, který znemožňuje identifikaci klíčových mutací pro diverzifikaci v rámci proteinové rodiny. Proto není vždy možné studovat vliv historických mutací na vznik nových funkcí pouze na základě existujících sekvencí [46].

Z evolučního hlediska jsou současné sekvence listy fylogenetického stromu (viz sekce 4.2) a představují variace zkoumané v jeden časový okamžik. Přístup porovnání současných sekvencí lze proto nazvat „horizontálním přístupem“. Tzv. „vertikální přístup“ bere navíc v úvahu evoluční historii proteinů a je schopný detekovat zásadní, ale i jemné rozdíly v aminokyselinách [46].

S myšlenkou ancestrální rekonstrukce aminokyselinových sekvencí, založenou na srovnání existujících sekvencí, přišel v roce 1963 E. Zuckerkandl a L. Pauling [51]. Ancestrální rekonstrukce sekvencí je tedy pravděpodobnostní metoda, která se snaží zkoumat hluboké evoluční vztahy mezi homologními sekvencemi za účelem znovu sestavení proteinů nacházejících se na konkrétní evoluční trajektorii [28]. V současnosti se metodika rekonstrukce ancestrální sekvencí používá i v proteinovém inženýrství. Tato výkoná a robustní strategie je využívána především pro konstrukci vysoce aktivních a stabilních katalyzátorů [50].

Přístup navíc umožňuje v určitých případech navrhnout mutace, které dokáží změnit specificitu proteinu, čehož je velmi obtížné dosáhnout při použití pouze racionálního návrhu [3]. Při návrhu evolučních předků metoda vychází nejen z vícenásobného zarovnání relevantních homologních sekvencí, ale také z fylogenetického stromu (viz sekce 4.2). V současnosti jsou známy dva hlavní algoritmické přístupy, přístup **maximální věrohodnosti** (maximal likelihood) [71] a **Bayesovi inference** [30].

Jednou z úspěšných aplikací metody ancestrální rekonstrukce lze jmenovat studium genu RNázy, kdy se podařilo rekonstruovat společného předchůdce tohoto genu z prasete, velblouda, jelena, ovce a vola. Rekonstruovaný předek žil před 40 milióny let a aktivita rekonstruované RNázy byla pětinásobně vyšší než u současných jedinců [32].

4.2 Fylogenetické stromy

Fylogenetické stromy jsou větvící se diagramy ukazující evoluční vztahy mezi různými biologickými organismy na základě podobností a rozdílů v jejich fyzikálních nebo genetických vlastnostech. Pomocí fylogenetických stromů lze zkoumat konzervované a evolučně specifické oblasti proteinu. Náhled do evoluční dynamiky také umožňuje predikovat pozice vhodné pro mutace či interakce mezi proteiny. Evoluční podobnost je i nedílnou součástí predikce struktury proteinu na základě šablony [18].

Samotný fylogenetický strom se skládá z několika dílčích částí:

- **Listové uzly** jsou vstupní sekvence do analýzy. Obvykle se jedná o zarovnané sekvence zkoumaných proteinů či genů.
- **Vnitřní uzly** reprezentují společného předka sekvencí, které se nacházejí ve stromě pod ním. Tyto sekvence nemusely nikdy reálně existovat, jedná se pouze o nejpravděpodobnější podobu dle použité metodologie.
- **Kořen** je společný předek všech sekvencí. Fylogenetický strom nemusí mít vždy kořen (tzv. nekořenový strom). Nekořenový strom lze převést na kořenový pomocí vložení sekvence, která je evolučně velmi vzdálená všem sekvencím v listových uzlech (tzv.outgroup rooting).
- **Hrany** ukazují vztahy mezi předkem a potomkem. Délka hrany signalizuje tzv. evoluční vzdálenost, která vyjadřuje počet mutací a nebo čas uplynulý od oddělení od společného předka.

Pro konstrukci fylogenetických stromů je nezbytné mít vybranou vhodnou skupinu vstupních sekvencí. Stromy jsou konstruovány na základě změn mezi vstupními sekvencemi za využití několika přístupů založených na editační vzdálenosti, pravděpodobnosti či na znacích. V následujících sekcích uvedu některé přístupy blíže s příkladem konkrétních metod.

4.2.1 Metody založené na vzdálenosti

Metody založené na vzdálenosti zakládají tvorbu evolučních stromů na skutečnosti, že délka hran stromu reflektuje evoluční vzdálenost sekvencí (označovanou D_{ij}) danou počtem substitucí znaků v jednotlivých sekvencích nebo časem vývoje od jednoho organismu k druhému. Proto nás při konstrukci stromů zajímá, aby délka jakékoliv cesty mezi dvěma uzly na cestě i a j odpovídala počtu mutací mezi organismy. Stromová vzdálenost mezi dvěma

uzly i a j se označuje jako $d_{ij}(T)$. Jak ale sestrojit takový evoluční strom, aby odpovídal editační vzdálenosti?

K sestavení takového stromu nám mohou pomoci tzv. aditivní matice. Aditivní nazýváme matici, pro kterou lze sestrojit evoluční strom respektující editační vzdálenosti $d_{ij}(T) = D_{ij}$. Pokud matice není aditivní, hledáme takový strom, který nejlépe aproximuje hodnoty D v matici vzdáleností např. pomocí minimalizace chyby čtverce.

Řešení s dostupnou aditivní maticí lze nalézt pomocí sekvence následujících kroků:

- Nalezení sousedních uzlů I, J se společným předkem K
- Odstranění sekvence I, J z matice vzdálenosti s vložením K
- Vypočtení nových vzdáleností všech zbylých sekvencí od K

Problém výše zmíněného algoritmu spočívá ve vhodném výběru sousedních uzlů. Řešení může skýtat například metoda Neighbour-Joining nebo UPGMA.

Neighbour-Joining

Metoda Neighbour-Joining [56] řeší problematiku výběru sousedních uzlů A a B výběrem dvou nejbližších uzlů, které jsou současně co nejvíce vzdálené od ostatních. Postup výpočtu algoritmu lze rozdělit na 5 kroků:

- Vypočti matici všech vzdáleností mezi uzly a vytvoří se strom hvězdicového tvaru se společným středem a uzly odpovídající vstupním sekvencím.
- Nalezni uzly A a B, které odpovídají minimu z funkce $min[D(a,b) - u(a) - u(b)]$ a spoj je do jednoho uzlu U. Vzdálenost uzlů $D(a,b)$ lze získat přímo z matice vzdáleností. Hodnota $u(a)$ odpovídá vzdálenosti uzlu A od všech ostatních je vypočtena za pomoci vztahu:
$$u(a) = \frac{1}{N-2} \sum D(a,i),$$
kde I je množina všech uzlů o mocnosti N.
- K nově vzniklým hranám mezi uzlem U a dvěma původními A, B přiřaď délku dle vztahů $L(a,u) = (D(a,b) + u(a) - u(b))/2$ a $L(b,u) = (D(a,b) + u(b) - u(a))/2$.
- Vypočti vzdálenost vzniklého uzlu U od všech ostatních uzlů vztahem $D(u,i) = (D(a,i) + D(b,i) - D(a,b))/2$.
- Opakuj od druhého bodu dokud nezůstane jediný uzel.

Metoda Neighbour-Joining generuje obecně nekořenové stromy a dosahuje relativně dobrých výsledků i pro neaditivní matice.

UPGMA

Unweighted Pair Group Method with Arithmetic mean (dále UPGMA) je jednoduchá metoda tvorby stromů, která při konstrukci předpokládá konstantní běh biologických hodin ve všech větvích stromu. Proces výpočtu lze rozdělit na sekvenci následujících kroků:

- Každou sekvenci v matici vzdáleností nahraď shluky o jedné položce.

- Vyber z matice vzdáleností dvojici shluků A a B mající mezi sebou nejmenší vzdálenost.
- Dvojici A, B spoj do jednoho shluku a přepočítej vzdálenosti nového shluku od všech ostatních dle vztahu $d_{A\cup B,C} = (|A| \times d_{A,C} + |B| \times d_{B,C})/(|A| + |B|)$
- Spojení shluků zakresli do struktury stromu
- Pokračuj od bodu 2 dokud nejsou spojeny všechny dvojce

4.2.2 Metody založené na věrohodnosti

Metody založené na věrohodnosti (tzv. maximum likelihood) jsou obecným řešením používaným v řadě oblastí. Myšlenka tohoto přístupu je založena na maximalizaci pravděpodobnosti, že známá vstupní data byla vygenerována zkoumaným modelem. V případě tvorby fylogenetického stromu je modelem kořenový strom s parametry reprezentující jeho strukturu a délky jednotlivých větví. Pozorovanými daty jsou pak vícenásobné zarovnání sekvencí. Problém tvorby stromu lze rozdělit na tři podproblémy tzv. maličký, malý a velký likelihood problém.

Maličký likelihood problém

Maličký likelihood problém má za úkol vypočítat celkové skóre věrohodnosti stromu společně s ohodnocením vnitřních uzlů. Na vstupu vyžaduje vícenásobné zarovnání vstupních sekvencí současně se stromem, který má ohodnocené listové uzly a jsou pro něj známy délky hran.

Řešení maličkého likelihood problému poskytuje *Felsensteivnův algoritmus* [17] založený na principu dynamického programování. Pro každý sloupec vícenásobného zarovnání je vyčleněn jeden listový uzel. Všechny uzly poté obsahují vektor o velikosti zkoumané abecedy (pro abecedu nukleotidů je délka vektoru rovna 4), jehož položky reprezentují hodnotu věrohodnosti pro jednotlivé znaky. Chod algoritmu lze rozdělit na dvě hlavní části.

V prvním kroku se jedná o průchod stromem od listů ke kořeni a dochází k vyhodnocení hodnoty věrohodnosti pro jednotlivé znaky ve všech uzlech dle následujících pravidel:

- **Listové uzly**: ve vektoru listových uzlů je hodnota položky nastavena na jedničku v případě, že znak odpovídá znaku tohoto políčka, jinak se vloží nula.
- **Vnitřní uzly**: hodnota věrohodnosti k-té položky z vektoru vnitřního uzlu s_k je vypočtena na základě vztahu $L_{s_k}(k) = \left[\sum_{s_i} P_{s_k s_i}(t_i) L_{s_i}(i)\right] \times \left[\sum_{s_j} P_{s_k s_j}(t_j) L_{s_j}(j)\right]$, kde $P_{s_k s_i}$ značí pravděpodobnost přechodu mezi *k-tou* položkou vektoru uzlu s_k a *i-tým* znakem potomka. $L_{s_i}(i)$ symbolizuje aktuální hodnotu věrohodnosti *i-tého* znaku potomka. Indexy i a j slouží pro odlišení pravého a levého potomka.
- **Kořenový uzel**: ohodnocení celého stromu je dáno vztahem $L = \sum_{s_0} \pi_{s_0} L_{s_0}(0)$, kde π_{s_0} značí pravděpodobnost počátečního znaku na dané v sekvenci. $L_{s_0}(0)$ reprezentuje hodnotu věrohodnosti jednotlivých znaků ve vektoru kořene, která je vypočtena stejně jako v předchozím bodě.

V druhém kroku dochází k průchodu od kořene k listům se současným ohodnocením vnitřních uzlů. Pro každý vnitřní uzel s_k stačí pouze dosadit do vzorce pro výpočet $L_{s_k}(k)$ a vybrat maximum.

Malý likelihood problém

Vstupem malého likelihood problému je vícenásobné zarovnání sekvencí a tvar stromu s ohodnocením listových uzlů. Algoritmus vypočítává ohodnocení vnitřních uzlů a délky hran stromu, které odpovídají nejvěrohodnějšímu uspořádání stromu.

V [17] byl společně s algoritmem pro řešení maličkého likelihood problému odvozen iterační vzorec, který je schopný pro konkretní hranu stromu upravovat její délku tak, aby docházelo pouze ke zvyšování věrohodnosti stromu. Algoritmus se skládá ze tří základních částí:

- Vyber počáteční délky hran $t_1 \dots t_n$.
- Pro jednotlivé hrany $t_1 \dots t_n$ aplikuj výpočet dle iteračního vzorce na úpravu délek hran, dokud dochází k zvyšování hodnoty věrohodnosti stromu.
- Opakuj krok 2, dokud jsou nalézány nové délky hran zvyšující věrohodnost celého stromu.

Velký likelihood problém

Jedná se o nejvyšší vrstvu maximum likelihood algoritmů zastřešující celý výpočet evolučního stromu. Na základě vstupního vícenásobného zarovnání je vytvořen tvar stromu s ohodnocením listových a vnitřních uzlů společně s délkami hran $t_1 \dots t_n$, které odpovídají maximální hodnotě věrohodnosti pro daný strom.

Tvar stromu je nalezen pomocí mnohonásobné aplikace vkládání nových větví do struktury stromu, kdy pro strom mající n-1 listů je počet možností vložení nové větve roven 2n-5. Po každém přidání nové větve je proveden přepočet délek hran (viz, malý likelihood problém), přičemž po určitém počtu vložení nových větví dojde k prostříhání a reorganizaci stromu, aby bylo možné opustit oblast případného lokálního minima.

4.3 Převod na kořenový fylogenetický strom

Při ancestrální rekonstrukci sekvencí a ve fylogenetice vycházíme z klíčového předpokladu existence běžného společného předchůdce všech homologních sekvencí ve fylogenetickém stromu. V případě předchozích metod pouze algoritmus UPGMA vytváří stromy s kořenem. V ostatních případech společný předchůdce pro všechny sekvence chybí, a proto je před použitím daného stromu pro ancestrální rekonstrukci vyžadováno jej převést na strom s kořenem. Řešení nabízí hned několik přístupů mající své specifické požadavky a nedostatky. V následujících odstavcích si zběžně přiblížíme několik z nich.

Jednou z metod používaných pro vytvoření kořenu do stromu bez kořene je přístup **Outgroup rooting** [42]. Metoda je založena na znalosti studovaného systému, která je použita ke správnému umístění kořenu stromu. To se provádí ručním výběrem tzv. outgroup – sekvence nebo malá skupiny sekvencí, o kterých je známo, že jsou více vzdáleně příbuzné než všechny ostatní sekvence ve fylogenetickém stromu. Kořen stromu je poté umístěn mezi tuto outgroup a zbytek sekvencí stromu.

Oproti předchozímu přístupu metoda **Midpoint rooting** nevyžaduje žádné expertní znalosti studovaného systému. V případě tohoto algoritmu je kořen umístěn do prostředního bodu stromu. Prostřední bod je vypočítán jako střed nejdelší vzdálenosti ze všech párů terminálních uzlů. Metoda je zejména vhodná pro stromy mající konstantní evoluci, ale v případě nevyvážených stromů nevykazuje metoda dobré výsledky.

Posledním zmíněným zástupcem metod pro vytváření fylogenetického stromu s kořenem je metoda **Minimal ancestor deviation** [65]. Metoda se snaží vyhodnotit kritérium odchylky kořene pro všechny možné pozice kořene a všechny páry terminálních uzlů v daném stromu bez kořene. Jako pozice kořene je brána každá větev stromu. Obecně řečeno algoritmus se snaží umístit kořen do pozice, kde je nejmenší odchylka mezi kořenem a středem dané větve pro všechny větve studovaného stromu.

4.4 Rekonstrukce ancestrálních sekvencí

Predikce sekvencí ancestrálních proteinů z vícenásobných sekvenčních zarovnání je užitečná pro mnoho bioinformatických analýz. Předpovídání sekvencí předků není jednoduchý úkol a spoléhá se na přesné zarovnání a fylogenetický strom s dobře umístěným kořenem. Sekvence předků jsou, v případě použití metody maximální věrohodnosti, již produktem metody na odvození tvaru stromu, kde rekonstrukce předků je součástí tzv. maličkého likelihood problému (viz 4.2.2). Dalším přístupem lze jmenovat Bayesovu inferenci, která je, obdobně jako přístup maximální věrohodnosti, pravděpodobnostní metoda kombinující apriorní pravděpodobnost stromu společně s věrohodností dat pro vytvoření aposteriorní pravděpodobnostní distribuce daného stromu.

Současným velkým problémem je rekonstrukce ancestrálních mezer, které mohou představovat události inzerce/delece během evoluce. Mezery nemohou být vyhodnocovány stejným přístupem jako jednotlivé znaky aminokyselinových residuí. Tento problém ještě nebyl vyřešen robustním přístupem a většinou vyžaduje značné úsilí při manuální opravě.

Většina současných algoritmů je založeno na přístupu uvedeném v [19], kdy na začátku algoritmus přiřadí vektory délky podle sekvencí ve vstupním zarovnání do všech terminálových a vnitřních uzlů. Pole ve vektoru koncových uzlů jsou pak vyplněna hodnotami 0 nebo 1, což znamená mezeru v dané poloze zarovnání sekvence pro odpovídající terminálový uzel. Algoritmus se poté přesouvá z terminálních uzlů do kořene a pole ve vektorech vnitřních uzlů získají své hodnoty dle následujících pravidel:

- Pokud hodnoty pole pravého a levého potomka jsou 0, výsledek je 0.
- Pokud hodnoty pole pravého a levého potomka jsou 1, výsledek je 1.
- Pokud hodnoty pole levého potomka je 0 a hodnota v pravém je 1, výsledek je X.
- Pokud hodnoty pouze jednoho z potomků je X, výsledek je hodnota druhého potomka.
- Pokud hodnoty pole levého potomka a pravého je X, výsledek je X.

Po aplikaci těchto pravidel se algoritmus nachází v kořenovém uzlu. Hodnoty v jeho poli jsou aktualizovány podle stejných pravidel uvedených výše. Pole obsahující X jsou porovnány s jejich aposteriorními pravděpodobnostmi. Pokud je její hodnota vyšší než určitý zvolený práh, pak je toto pole nastaveno na hodnotu 0, v opačném případě je pole označeno jako mezera. Ve zpětném průchodu algoritmu od kořene k listovým uzlům jsou odstraněny všechny nejasnosti v mezerách nahrazením za hodnotu z rodičovského uzlu nacházející se na stejné pozici v zarovnání.

Rekonstrukce předků je i motivací mé práce. V nadcházejících kapitolách popíši moji metodu inference ancestrální sekvencí za pomocí variačních autoenkodérů, která by mohla přinést alternativu k předešlým a výpočetně náročným metodám. Jak si ukážeme v dalších kapitolách, výhodou mnou zkoumaného přístupu je i generace ancestrálních mezer bez nutnosti jakéhokoliv explicitního algoritmu pro jejich výpočet.

Kapitola 5

Variační autoenkodéry

V předešlých kapitolách byly představeny základní bioinformatické nástroje pro porovnávání proteinů, charakterizace významných míst v proteinových rodinách a pro studium evolučního vývoje proteinů od společného předka za pomocí fylogenetických stromů. Nadále byla uvedena metoda ancestrální rekonstrukce sekvencí pro návrh proteinů se zvýšenou stabilitou a aktivitou. V této kapitole budou uvedeny variační autoenkodéry metoda strojového učení bez učitele. Nadále budou představeny vlastnosti generovaného latentního prostoru pro data biologických sekvencí a motivace pro použití variačních autoenkodérů právě pro ancestrální rekonstrukci sekvencí, která by mohla přinést alternativu k předešlým výpočetně náročným metodám.

5.1 Koncepty strojového učení

S neustále rostoucím počtem veřejně dostupných dat vzniká požadavek na jejich účinné zpracovávání a extrahovaní užitečných informací obsazených v těchto rozličných typech záznamů. Jedním z efektivních nástrojů pro systematické zpracování velkého množství dat jsou i metody nabízené oblastí zvanou umělá inteligence. Zejména podmnožina umělé inteligence tzv. strojové učení zaujala v současnosti, díky zvyšování výpočetního výkonu dostupných strojů, pozornost velkého množství výzkumných skupin i soukromých podniků.

5.1.1 Učení s učitelem

Učení s učitelem je úkol učící funkci, která mapuje vstup na výstup na základě trénovacích dat ve formě párových hodnot vstup–výstup. Přístup učení s učitelem analyzuje trénovací data a vyvozuje funkci, která může být následně použita pro nové vzorky. Před trénovacím procesem je potřeba shromáždit vstupní data, ke kterým je nutno přiřadit odpovídající značky. Tvorba takové označkované datové sady je časově a často i finančně náročná činnost. Zástupci algoritmů této kategorie jsou například rozhodovací stromy, algoritmus k-nejbližších sousedů nebo support vector machines.

5.1.2 Učení bez učitele

Učení bez učitele je typ algoritmu, který se učí vzory z neoznačených dat. Je založen na myšlence, kdy je prostřednictvím napodobování stroj nucen vytvořit kompaktní vnitřní reprezentaci zdrojových dat. Na rozdíl od přístupu učení s učitelem, kdy k vstupním datům

jsou známy i příslušné výsledky pro danou úlohu, vykazují metody bez učitele samoorganizaci, která zachycuje vzory jako hustotu pravděpodobnosti.

V biologii slibuje očekávaný nárůst sekvenování bezprecedentní údaje o rozmanitosti přírodních sekvencí. Navíc v kombinaci s kapacitou modelů, strojové učení bez učitele umožní významný pokrok v oblasti učení reprezentací a statistického generování v rámci biologických sekvencí. Učení přirozeného rozložení evoluční variability proteinových sekvencí je logickým krokem k prediktivnímu a generativnímu modelování pro biologii [55], které je využito i v mé práci, jak si představíme v dalších sekcích.

5.1.3 Diskriminativní a generativní modely

V oblasti strojového učení rozlišujeme modely do dvou kategorií na základě mechanismu, kterým modelují danou úlohu. První skupinou jsou **diskriminativní** modely, které se učí pravděpodobnost $p(y|x)$, aby mohly předpovídat značky y vzhledem ke vstupům x. Typickým příkladem diskrimitativních modelů jsou klasifikátory. Na rozdíl od diskriminačních modelů, **generativní modely** se učí generovat příklady, které nejsou v trénovací množin a to tak, že se učí rozložení $p(x)$ trénovacích dat. Generativní modely lze také využít pro diskrimitativní úkoly, když se naučí složenou distribuci $p(x, y)$ a poté použijí definici $p(y|x) = p(x, y) / p(x)$. Mezi zástupce generativních modelů patří sítě typu GAN (Generative Adversial Networks) a právě i variační autoenkodéry, které budou detailně vysvětleny v sekci 5.2.

5.1.4 Redukce dimensionality

Ve strojovém učení je redukce dimensionality procesem snižování počtu rysů, které popisují určitá data. Tato redukce se provádí buď výběrem (zachovávají se pouze některé existující rysy), nebo extrakcí (na základě starých rysů se vytvoří snížený počet nových rysů) a může být užitečná v mnoha situacích, které vyžadují nízkou dimenzi dat (vizualizace dat, ukládání dat, náročné výpočty). Přestože existuje mnoho různých metod redukce dimensionality, můžeme stanovit globální rámec, kterému odpovídá většina z těchto metod.

Nejprve nazvěme kodérem proces, který vytváří reprezentaci „nových rysů“ z reprezentace „starých rysů“ (výběrem nebo extrakcí), a dekodérem proces opačný. Redukci dimensionality pak lze interpretovat jako kompresi dat, kdy kodér data komprimuje z původního prostoru do zakódovaného prostoru, nazývaného také **latentní prostor**, zatímco dekodér je dekomprimuje. V závislosti na počátečním rozložení dat, dimenzi latentního prostoru a definici kodéru může být tato komprese samozřejmě ztrátová, což znamená, že část informací se během procesu kódování ztratí a při dekódování je nelze obnovit.

Hlavním účelem metody redukce dimensionality je najít nejlepší pár kodér/dekodér z dané rodiny. Jinými slovy, pro danou množinu možných kodérů a dekodérů hledáme dvojici, která při kódování zachovává maximum informace a při dekódování má tedy minimální chybu rekonstrukce. Označíme-li E a D rodiny kodérů a dekodérů, které uvažujeme, pak lze problém redukce dimensionality zapsat takto

$$(e^*, d^*) = \underset{(e,d)\in E\times D}{\arg\max}\, \epsilon\left(x, d\left(e\left(x\right)\right)\right) \tag{5.1}$$

kde

$$\epsilon\left(x, d\left(e\left(x\right)\right)\right) \tag{5.2}$$

značí rekonstrukční chybu mezi vstupními daty a jejich variantami, které vznikly po zakódování a dekódování pomocí dané dvojce enkodér/dekodér.

Principal component analysis

Jeden z nejznámějších zástupců metod redukce dimensionality je metoda zvaná principal component analysis (dále PCA). Myšlenkou metody PCA je vystavět N_{lat} nových nezávislých rysů, které jsou lineární kombinací N_{dim} původních rysů. Projekce těchto dat na podprostor definovaný novými rysy je co nejblíže původním datům ve smyslu euklidovské vzdálenosti. Jinými slovy PCA se snaží najít lineární podprostor původního prostou (daný ortogonálními bázemi nových rysů), tak aby chyba vzniklá aproximací dat jejich projekcí na tento podprostor byla co nejmenší.

Když se vrátíme k původní myšlence redukce dimensionality, pak hledáme takový enkodér, který je definovaný $N_{lat} \times N_{dim}$ maticí s ortonormálními vektory v řádcích zajišťující nezávislost. Stejně tak i dekodér je opět matice s rozměry $N_{dim} \times N_{lat}$ s ortonormálními řádky. V případě PCA lze matici sestavit z N_{lat} jednotkových vlastních vektorů odpovídajících N_{lat} největších vlastních hodnot kovarianční matice vstupních rysů. Navíc poté je dekodér definován jako transponovaná matice enkodéru.

Autoenkodér

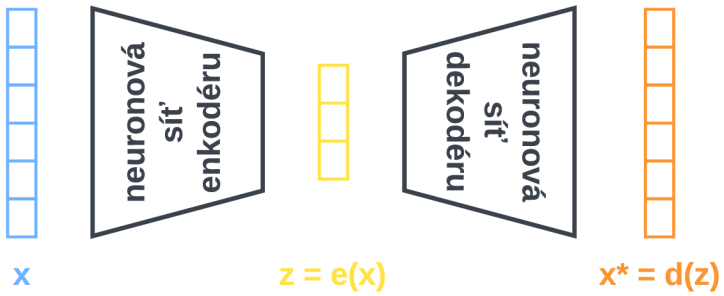
Dalším zástupcem metod pro redukci dimensionality jsou autoenkodéry, které představují jedno ze základních paradigmat pro strojové učení bez učitele. Autoenkodéry byly poprvé představeny v 80. letech 20. století Hintonem [27], aby vyřešily problém zpětné propagace bez cílových značek, na které měla být síť naučena. Jako cílové značky byla použita data samotná.

Obecná myšlenka autoenkodérů je poměrně jednoduchá a spočívá v nastavení kodéru a dekodéru jako neuronových sítí a v naučení nejlepšího schématu kódování a dekódování pomocí iteračního optimalizačního procesu. V každé iteraci tedy propagujeme přes architekturu autoenkodéru (kodér a následně dekodér) data, porovnáme zakódovaný-dekódovaný výstup s počátečními hodnotami a zpětně šíříme chybu přes architekturu, abychom aktualizovali váhy sítí.

Celková architektura autoenkodéru je tvořena enkodérem a dekodérem, které jsou propojeny úzkým hrdlem, jak je zachyceno na obrázku 5.1. Tím je zajištěno, že pouze hlavní strukturní vzory dat mohou projít skrz hrdlo a následně být na výstupu rekonstruovány. Enkodér a dekodér jsou definovány pomocí vlastních architektur realizovaných, v případě strojového učení, pomocí neuronových sítí. Hledaní enkodéru a dekodéru, které minimalizují chybu při rekonstrukci je provedeno pomocí metody gradientního sestupu přes parametry těchto sítí.

Kódovací a dekódovací matice získané pomocí PCA přirozeně definují jedno z řešení, ke kterému bychom se rádi dostali pomocí gradientního sestupu, ale není jediné. Lze totiž zvolit několik bází, které popisují stejný optimální podprostor, a tak může několik dvojic kodér/dekodér poskytnout optimální chybu rekonstrukce. Navíc u lineárních autoenkodérů, na rozdíl od PCA, nemusí být nové rysy na sobě nezávislé, protože v neuronových sítích neexistují žádná omezení ortogonality.

S dostatečně velkým množstvím parametrů sítí bychom byli schopni provést kompresi dat za nulové ztráty informace. Avšak při návrhu architektury musíme myslet na to, že cena za nulovou chybovost při rekonstrukci ze zakódované latentní reprezentace jde v ruku v ruce se ztrátou pravidelnosti v latentním prostoru. Dále také platí, že při redukci dimensiona-



Obrázek 5.1: Schéma autoenkodéru

lity si nepřejeme pouze zakódovat data, ale chceme je zakódovat při zachování hlavní části strukturních informací v nich obsažených. Proto dimensionalita latentního prostoru, společně se strukturou jednotlivých částí autoenkodéru, musí být pečlivě přizpůsobena k dané problémové doméně.

5.2 Variační autoenkodéry

V předchozích sekcích jsme diskutovali základní metody pro redukci dimensionality. Tyto metody jsou vhodné pro kompresi dat do méně dimenzionálního prostoru, který může sloužit následně jako vstup pro další, například klasifikační, úlohy. Avšak klasický autoenkodér není vhodný jako generativní model. Mohlo by se zdát, že při předpokladu dostatečné pravidelnosti latentního prostoru, jsme schopni náhodně vybrat jakýkoliv zakódovaný bod, následně jej dekódovat a obdrželi bychom smysluplný výstup podobný trénovacím datům. Bohužel zaručit pravidelnost latentního prostoru je obtížný úkol kvůli distribuci vstupních dat, dimensionalitě latentního prostoru a zvolené architektuře použitého enkodéru. Jednou z možností, jak docílit pravidelnosti latentního prostoru, je přidání regularizačního prvku do trénovací funkce, protože autoenkodér je trénován výhradně na kódování a dekódování vstupních dat s co nejmenšími ztrátami bez ohledu na to, jak je latentní prostor uspořádán. Právě řešení problému regularizace latentního prostoru nabízejí **variační autoenkodéry**.

5.2.1 Variační inference

Variační autoenkodéry (dále jen VAEs) a proces generace nových dat z latentního prostoru je postaven na matematickém pravděpodobnostním rámci Bayesovské inference, která ke svému řešení využívá přístup zvaný variační inference [37].

Předpokládejme, že X označuje proměnnou, která reprezentuje naše data, a předpokládáme, že X je generována z latentní proměnné Z (zakódované reprezentace), která není přímo pozorována. Latentní model definuje složenou pravděpodobnost X a Z jako $p(X, Z) = p_\theta(Z)\, p_\theta(X|Z)$, kde θ reprezentuje parametry složené distribuce. Proces generace z $p(X, Z)$ se tak skládá ze dvou kroků:

1. Hodnota Z je generována z určité apriorní distribuce pravděpodobnosti $p_\theta(Z)$.

2. Hodnota x je generována z nějaké podmíněné pravděpodobnostní distribuce dané $p_\theta(X|Z)$.

Podmíněná pravděpodobnost $p_\theta(X|Z)$ muže být viděna jako dekodér, který dekóduje hodnotu Z na X. Vzhledem k pozorovaným datům pro proměnnou X učení se parametrů θ, které popisují generativní proces pomocí přístupu maximální věrohodnosti, je náročný úkol. Jedním z důvodů obtížnosti je to, že marginální pravděpodobnost pozorované proměnné X daná vzorcem

$$p(X) = \int p_\theta(X, Z)\, dZ \tag{5.3}$$

není analyticky řešitelná a je drahá na výpočet, když podmíněná pravděpodobnost $p_\theta(X|Z)$ je komplexní. Dalším důvodem obtížnosti je případ, kdy podmíněná pravděpodobnost $p_\theta(X|Z)$ je komplexní, například je parametrizována pomocí umělé neuronové sítě a aposteriorní distribuce $p_\theta(Z|X)$ se stává analyticky neřešitelná [37]. Navíc může být velmi obtížné získávat nezávislé vzorky z $p_\theta(Z|X)$, což dělá maximalizační algoritmy [15] nevhodné pro řešení maximalizace marginální pravděpodobnosti $p_\theta(X)$. Jedna z efektivních možností pro učení parametrů θ je použití aproximační metody známé jako variační interference [7][34].

Při variačním odvozování se k vyřešení potíže s aposteriorním rozdělením $p_\theta(Z|X)$ používá rodina aproximativních rozdělení, $q_\phi(Z|X)$, parametrizovaných pomocí ϕ, která je zavedena k aproximaci aposteriorního rozdělení $p_\theta(Z|X)$. Namísto optimalizace mezní pravděpodobnosti pozorovaného rozdělení $p_\theta(Z)$ optimalizuje variační odvozování alternativní cílovou funkci, která se nazývá *evidence lower bound objective function* (dále ELBO) [37][7][38]. Pro jakoukoliv volbu inferenčního modelu $q_\phi(Z|X)$, včetně volby variačních parametrů ϕ, platí:

$$\begin{aligned}
\log p_\theta(X) &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(X)\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(X,Z)}{p_\theta(Z|X)}\right]\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(X,Z)}{q_\theta(Z|X)}\frac{q_\phi(Z|X)}{p_\theta(Z|X)}\right]\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{p_\theta(X,Z)}{q_\theta(Z|X)}\right]\right]}_{\mathcal{L}_{\phi,\theta}(X)} + \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[\log\left[\frac{q_\phi(Z|X)}{p_\theta(Z|X)}\right]\right]}_{D_{KL}}
\end{aligned} \tag{5.4}$$

kde druhá část v poslední rovnici v 5.4 označuje Kullback-Leiblerovu divergenci (KL divergence) mezi aproximačním rozložením $q_\phi(Z|X)$ a apriorním rozložením $p_\theta(Z|X)$ a její hodnota je nezáporná

$$D_{KL}\left(q_\phi(Z|X)\,||p_\theta(Z|X)\right) \geq 0 \tag{5.5}$$

a nulová pouze tehdy, když $q_\phi(z|x)$ se rovná pravé aposteriorní pravděpodobnosti.

První část v poslední rovnici v 5.4 je ELBO

$$\mathcal{L}_{\phi,\theta}(X) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(X,Z) - \log q_\theta(Z|X)\right] \tag{5.6}$$

Vzhledem k nezápornosti KL divergence je ELBO nižší než mezní logaritmická pravděpodobnost dat.

$$\begin{aligned}
\mathcal{L}_{\phi,\theta}(X) &= \log p_\theta(X) - D_{KL}\left(q_\phi(Z|X)\,||p_\theta(Z|X)\right) \\
&\leq \log p_\theta(X)
\end{aligned} \tag{5.7}$$

Z rovnice 5.7 je zřejmé, že maximalizace ELBO $\mathcal{L}_{\phi,\theta}(X)$ v závislosti na parametrech θ a ϕ, bude současně optimalizovat dvě věci, které nás zajímají:

1. Bude přibližně maximalizovat mezní pravděpodobnost $p_\theta(X)$. To znamená, že náš generativní model bude lepší.

2. Minimalizuje aproximaci KL divergenci $q_\phi(Z|X)$ od skutečné aposteriorní pravděpodobnosti $p_\theta(Z|X)$, takže $q_\phi(Z|X)$ bude lepší.

5.2.2 Trik reparametrizace

V předešlé sekci byl uveden pravděpodobnostní rámec, který tvoří matematický základ variačních autoenkodérů. Variační inference umožňuje odvodit vztahy, které je možné efektivně realizovat ve výpočetním prostředí. Podstatou je stanovit parametrizovanou rodinu rozdělení a hledat nejlepší aproximaci našeho cílového rozdělení $p_\theta(Z|X)$ mezi touto rodinou. Nejlepší prvek v rodině je takový, který minimalizuje danou chybu měření aproximace v podobě KL divergence a je nalezen gradientním sestupem nad parametry, které popisují tuto rodinu rozdělení. Příkladem takové parametrizovatelné rodiny rozdělení, která je i běžně používána v rámci variačních autoenkodérů, je Gaussovo rozdělení s parametry střední hodnoty a kovariance.

V našem případě je zvoleno apriorní rozdělení ve formě m-dimenzionálního Gaussova rozdělení s diagonální maticí kovariance s předpokladem nezávislosti proměnných. Za těchto podmínek jsou oba parametry, střední hodnoty a kovariance, vektory se stejnou mocností v případě kovariance obsahující pouze diagonální prvky. Takto zvolené předpoklady mohou mít za následek zmenšení přesnosti aproximace původní distribuce.

Jednotlivé parametry by mohly být aproximovány pomocí samostatných neuronových sítí, ale v praxi jsou váhy pro střední hodnotu i kovarianci sdíleny, jak lze vidět na obrázku 5.2. Celková architektura variačního autoenkodéru je obdobně jako u autoenkodéru složena z enkodéru a dekodéru. Proces vzorkování, který byl popsán v předchozí sekci, se skládá ze dvou dílčích kroků, přičemž v prvním kroku dochází k náhodné generaci latentní proměnné. Nyní však musíme dbát na to, jakým způsobem vzorkujeme data, která jsou vrácena enkodérem během trénování. Vzorkovací proces musí být vystavěn takovým způsobem, aby umožňoval zpětné propagování chyby skrz celý model a právě náhodný proces vzorkování z distribuce $p(Z)$ tento postup komplikuje.

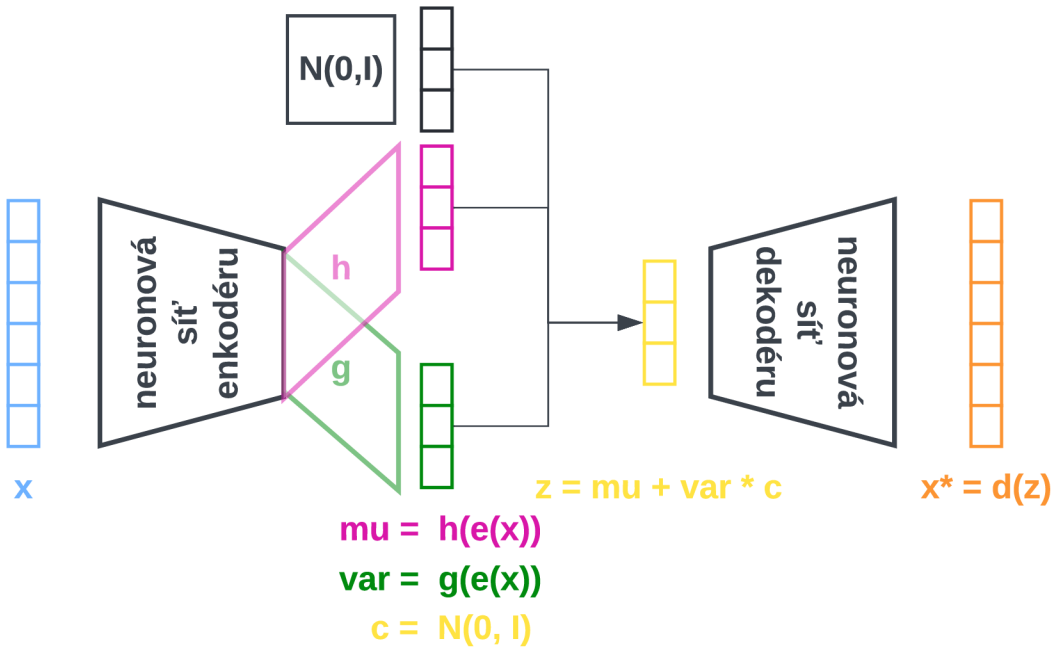
Řešení nabízí tzv. trik reparametrizace, který umožňuje využití gradientního sestupu, i přestože uprostřed naší architektury dochází k náhodnému procesu. Trik je založen na skutečnosti, že když náhodné vzorkování pochází z Gaussovi distribuce se střední hodnotou $h(e(X))$ a hodnotou kovariance $g(e(X))$, poté lze tento proces vyjádřit jako

$$z = g(e(X))\, N(0, I) + h(e(X)) \tag{5.8}$$

Graficky je trik reparametrizace zachycen na obrázku 5.2. Díky umístění náhodného procesu mimo parametry modelu, lze propagovat diferenciaci parametrů dle chyby i do sítě enkodéru a účinně tak trénovat variační autoenkodéry [38].

5.3 Aplikace variačních autoenkodérů

Variační autoenkodéry zaujali pozornost v oblasti strojového učení díky jejich přívětivým vlastnostem, zejména interpretaci rozložení latentního prostoru a dobrého trénování, které je



Obrázek 5.2: Schéma variačního autoenkodéru a procesu vzorkování dat z latentního prostoru za pomoci triku reparametrizace. Vzorkovací proces je držen mimo datový proud v modelu. Díky tomu je umožněno využití přístupu učení neuronových sítí pomocí zpětné propagace chyby.

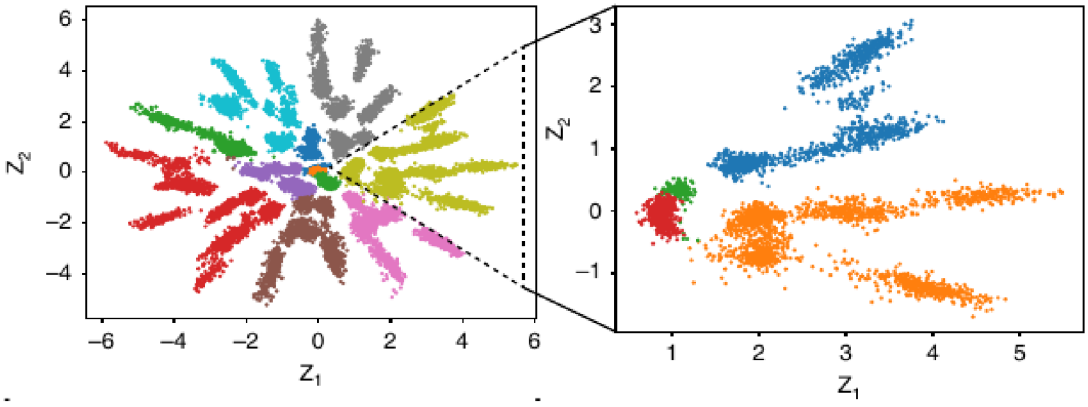
jednodušší než v případě sítí typu GAN [68]. Latentní modely učené za pomocí variačních autoenkodérů byly široce použity v mnoha problémech strojového učení a dosáhly velmi dobrých výsledků v oblasti zpracování obrazu a řeči [9][53].

5.3.1 Variační autoenkodéry pro evoluci proteinů

Díky pokroku v oblasti stochastické variační inference, kterými jsou i variační autoenkodéry, mohou být spojité latentní modely trénovány na velkém množství datových sekvencí. To umožňuje použít jako trénovací sady velké proteinové rodiny a zkoumat vztahy mezi jednotlivými proteinovými sekvencemi podle umístění jejich zakódované reprezentace v latentním prostoru.

Ve článku [14], který byl hlavním iniciátorem mé práce, autoři objevili, že latentní prostor generován variačními autoenkodéry, zachycuje evoluční vztahy mezi proteiny. Autoři trénovali modely latentního prostoru pomocí VAEs na mnohonásobném zarovnání sekvencí ze tří proteinových rodin: doména fibronektinu typu III (Pfam přístupové id: PF00041, cytochromu P450 (kód PF00067) a genu stafylokokové nukleázy (PF00565). Počet unikátních sekvencí použitých pro trénování modelů latentního prostoru byl 46 498, 31 062 a 7448. Pro účely vizualizace byl použit dvourozměrný latentní prostor, jak lze vidět na obrázku 5.3. Jak je z obrázku patrné, latentní prostor má tvar hvězdy, jejíž ramena směřují z centra do okrajových oblastí latentního prostoru vždy podél specifického směru pro každou větev evoluce. Autoři dále ověřili, že tento tvar je opravdu dán výskytem evolučních souvislostí v proteinových sekvencí, protože natrénovali shodný model na 10 000 náhodně vygenerova-

ných sekvencí, z distribuce LG evolučního modelu [41], a oproti modelům natrénovaným na přírodních proteinových rodinách jsou sekvence náhodně zakódovány do latentního prostoru a vzor hvězdy již nebyl více pozorován. V obrázku 5.3 lze vidět i různé barevné reprezentace zakódovaných sekvencí. Barvy jsou odvozeny na základě výskytu proteinové sekvence ve fylogenetickém stromu v určitý čas evoluce a body jedné barvy odpovídají proteinům vyskytujícím se ve stejné pod větvi za tímto časovým řezem v odpovídající hloubce stromu. Vyobrazená reprezentace odkazuje na schopnost VAEs sdružovat do shluků sekvence podle výskytu ve fylogenetickém stromu. Navíc tato schopnost se nesnižuje ani při detailnějším pohledu na vývoj ve vybrané podvětvi fylogenetického stromu, jak je ukázáno na obrázku 5.3 v pravé části.



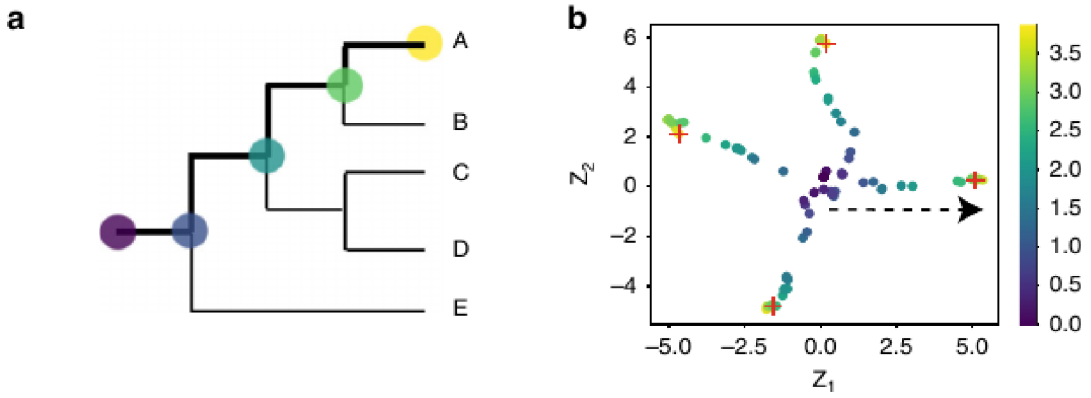
Obrázek 5.3: Latentní prostor zachycuje evoluční vztahy v proteinových rodinách. Barevně shodné body odpovídají sekvencím patřícím do stejné podvětve fylogenetického stromu. Řez stromu je proveden v určité hloubce odpovídající zkoumanému času evoluce. **Vpravo**: variační autoenkodér je schopný zachytit evoluční vztahy i na detailnějším podprostoru fylogenetického stromu. Převzato z [14].

Obrázek 5.3 ukázal, že VAEs jsou schopné zachytit evoluční souvislosti mezi proteiny. Ze studia latentní reprezentace proteinů bylo nadále zjištěno, že body kořenových uzlů fylogenetických stromů mají tendenci být umístěny kolem středu latentního prostoru. Toto zjištění vedlo k hypotéze existence ancestrálních vztahů v latentní reprezentaci proteinových rodin. Proto autoři rekonstruovali ancestrální sekvence navržené na základě fylogenetických stromů a namapovali je do latentního prostoru (viz obrázek 5.4). Body v trajektorii odpovídají ancestrálním sekvencím od jednoho listového uzlu až ke kořenu fylogenetického stromu (obrázek 5.4a). Jak ukazuje obrázek 5.4b body v latentním prostoru reprezentující předchůdce dané listové sekvence vytváří stabilní trajektorie směřující do středu.

V článku [14] byla ukázána schopnost variačních autoenkodérů mapovat vytvořené ancestrální sekvence do latentního prostoru s předvidatelnými trajektoriemi. Nebyla však věnována pozornost generativnímu procesu nových sekvencí, díky kterému by variační autoenkodéry mohly tvořit alternativní metodu pro návrh ancestrálních sekvencí. To je předmětem mé práce a tato problematika bude řešena v následujících kapitolách.

5.4 Limitace současného řešení

V předchozí sekci bylo představeno použití variačních autoenkodérů v biologii. Konkrétně mapování evolučních závislostí do latentního prostoru modelu. Tyto mapovací schopnosti



Obrázek 5.4: Latentní prostor zachycuje ancestrální vztahy mezi sekvencemi. **a** Do latentního prostoru byly namapovány sekvence reprezentující evoluční předchůdce trénovacích sekvencí z fylogenetického stromu od listového uzlu po kořen. **b** Výsledek namapování ancestrálních sekvencí vytváří stabilní trajektorie směřující do centra latentního prostoru. Převzato z [14].

zaujaly moji pozornost a rozhodl jsem se zkoumat možnosti generovaní ancestrálních sekvencí pomocí právě variačních autoenkodérů. Referenční [14] a další studie však ukázaly své limity nejvíce týkající se oblasti generování nových kandidátních sekvencí. Sekvence generované modelem v [14] byly velmi vzdálené od referenční (maximálně 60% podobnost), což pro případ optimalizace jednoho konkrétního proteinu nebyl vhodný výchozí bod. Navíc v literatuře nebyla popsána žádná pokročilejší strategie pro exploraci latentního prostoru společně s chybějící analýzou robustnosti daných modelů vůči náhodnému nastavení počátečních vah neuronové sítě. Také jsme byli zaujati aplikací na menším množství sekvencí sdílející specifická katalytická residua oproti těm v celé *pfam* rodině.

Optimalizace modelu a celého pracovního procesu společně s řešením předchozích nedostatků byla předmětem mé práce. Cílem bylo, aby navržené sekvence co s největší pravděpodobností úspěšně potvrdili vylepšené vlastnosti v následných laboratorních testech. Navíc jsem se zaměřil na vylepšení bázového rámce VAEs pro zajištění větší kontroly nad generovanými vzorky a jejich vlastnostmi. Jedním ze zástupců modelů snažících se zvyšovat kontrolu nad vlastnostmi vygenerovaných vzorků jsou i tzv. **podmíněné variační autoenkodéry** [61] (známe jako conditional autoencoders, dále CVAE).

Generativní proces VAEs se skládá z kroků generace náhodné proměnné z apriorní distribuce $p(z)$ a následné generace dat x z generativní distribuce $p_\omega(x|z)$, kde parametry jsou odvozeny na základě stochastického gradientního variačního rámce s využitím objektivní funkce ELBO (rovnice 5.4).

Rámec VAE lze jednoduše upravit tak, aby modelovala rozdělení dat podmíněných pomocnými proměnnými c, podmíněním sítí enkodéru a dekodéru těmito proměnnými. Potom objektivní funkce je následující:

$$\mathcal{L}_{\phi,\theta}(X) = \mathbb{E}_{q_\phi(z|x,c)}\left[\log p_\theta(x|z,c) - D_{KL}\left(\log q_\phi(z|x,c)\,||p(z|c)\right)\right] \tag{5.9}$$

Podmíněním na další proměnné c lze do modelu a jeho generativních schopností vnést další informaci o právě zpracovávaných datových vstupech. Tento přístup byl využit i při generaci proteinových sekvencí [23]. Jedním z často objevujících se problémů u navržených proteinů je jejich nízká rozpustnost, která brání následné charakteristice v laboratoři, jak

bylo ukázáno i v případě našich kandidátů (viz kapitola 8). Proto můžeme ke vstupním sekvencím do modelu přidat i hodnoty rozpustnosti, ať už predikované, nebo experimentálně ověřené, a naučit náš model reflektovat tyto souvislosti. Hodnoty rozpustnosti mohou být rozděleny například do 3 kategorií: *LOW*, *MEDIUM* a *HIGH* dle zvolených procentuálních prahů rozpustnosti. Nastavením příznaku *HIGH* při generování sekvencí pak nutíme dekodér do sekvence zakomponovat znaky vyskytující se v dobře rozpustných proteinech a zvýšit tak pravděpodobnost úspěchu následných laboratorních testů. Navržené sekvence však mohou být v některých případech porušeny a zničena jejich specifická aktivita.

Kapitola 6

Data

V této části bude blíže popsána rodina haloalkan dehalogenáz jakožto cílová proteinová rodina mých experimentů. Dále budou uvedeny datové sady, s které byly použity při experimentech. Jednotlivé datové sady, jak jsou uvedeny, byly v průběhu projektu optimalizovány pro dosažení návrhu biologicky relevantnějších sekvencí.

6.1 Rodina proteinů haloalkan dehalogenáz

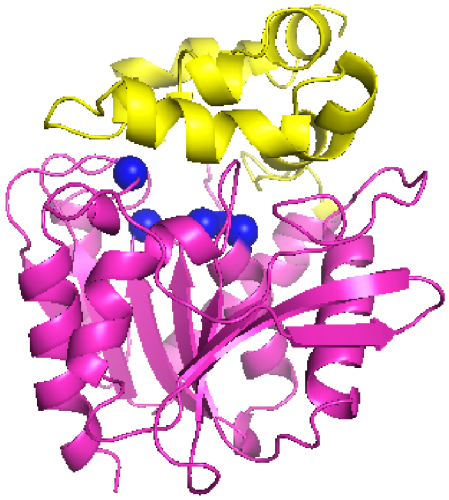
Haloalkan dehalogenázy jsou mikrobiální enzymy, které katalyzují přeměnu halogenovaných uhlovodíků na odpovídající alkohol, halogenový aniont a proton [31]. Přeměna je provedena za pomoci hydrolytického štěpení vazby mezi halogenem a uhlíkem za současného vzniku nové vazby C–OH [39].

Strukturně haloalkan dehalogenázy patří do nadrodiny proteinů α/β–hydroláz. I přes fakt, že členi této nadrodiny vycházejí ze stejného předchůdce, tak se jednotlivé proteiny nevyznačují vysokou sekvenční podobností. Jsou si podobní především v prostorovém uspořádaní, které je tvořeno z dvou domén. První je generačně konzervovaná hlavní doména α/β–hydroláz z větší části složena z osmi paralelně vedených β–listů lemovaných několika α–helixy. Druhá doména, též nazývána *víčková doména*, bývá proměnná z hlediska počtu a uspořádání sekundárních prvků. Je tvořena výhradně α–helixy spojenými smyčkami a výrazně se svým složením a uspořádáním podílí na substrátové specifitě enzymu s aktivním místem umístěným mezi víčkovou a hlavní doménou [39]. Zástupce rodiny haloalkan dehalogenáz můžete vidět na obrázku 6.1.

Díky dobře prostudovanému katalytickému ději, schopnosti rozkládat uhlovodíky obsahující halogeny a katalyzovat chemické reakce bez přítomnosti kofaktorů, jsou haloalkan dehalogenázy atraktivním cílem pro akademických výzkumné týmy i průmyslové aplikace. Schopnost odbourávat řadu toxických látek nebezpečných pro životní prostředí nabízí velký potenciál v oblasti jeho dekontaminace, například od bojového plynu jako je yperit, či bioremediace. Dalším průmyslovou oblastí aplikace se nabízí biokatalýza, předpříprava substrátů pro další průmyslové využití či detekce halogenovaných sloučenin v životním prostředí [40].

6.2 Datové sady

Vstupem do VAEs, jak bude v následujících sekcích ukázáno, je vícenásobné zarovnání sekvencí a jeho kvalita společně s relevantností sekvencí v nich obsažených, je důležitá pro generativní schopnost modelu vůči našemu požadavku rekonstrukce ancestrálních sekvencí.



Obrázek 6.1: Zástupce rodiny haloalkan dehalogenáz protein 1bn6 [49]. Hlavní doména je zvýrazněna růžově, víčková žlutě. Modré kuličky značí katalytické residua proteinu.

Proto byly vybrány takové zdroje dat disponující zarovnáními sekvencí. Pro studování vlastností variačních autoenkodérů, a jejich schopnosti rekonstrukce ancestrálních sekvencí, jsme vybrali celkem 3 datové sady, které obsahují zástupce proteinů rodiny haloalkan dehalogenáz. První z nich byla obecná sada proteinů cílené *Pfam* rodiny, zatímco zbylé dvě sady byly vystavěny na základě sofistikovanějších nástrojů pro získání opravdu relevantních proteinů pro naši cílovou doménu.

Datová sada *Pfam*

Vlastnosti latentního prostoru variačních autoenkodéru byly zkoumány na celistvých datových rodinách [14][13] získaných z databáze Pfam [4]. Proto jsme se na začátku práce s VAEs zaměřili na otestování jeho mapovacích vlastností pro naši rodinu proteinů haloalkan dehalogenáz, které jsou členy *Pfam* rodiny s označením PF00561.

Proteiny této rodiny mají společnou alfa/beta hydrolázovou strukturu pro řadu hydrolytických enzymů s velmi rozdílným fylogenetickým původem a katalytickou funkcí. Jádrem každého enzymu je alfa/beta list, který obsahuje 8 vláken spojených šroubovicemi. Předpokládá se, že enzymy se oddělily od společného předka, přičemž se zachovalo uspořádání katalytických residuí. Všechny mají katalytickou triádu, jejíž prvky jsou neseny na smyčkách, které jsou nejlépe konzervovanými strukturními prvky rodiny. Tato katalytická doména se vyskytuje u velmi širokého spektra enzymů a dohromady se skládá ze **104788** sekvencí nalezených pomocí HMM profilu poskytovaných ve formátu *stockholm*. Rodina byla vystavěna na profilu zarovnání **48** *seed* sekvencí.

6.2.1 Přizpůsobené datové sady z nástroje EnzymeMiner

Mým úkolem bylo navrhnou ancestrální sekvence pro cílový protein haloalkan dehalogenázy z genu *DhaA*. Proto jsem potřeboval z velmi široké *Pfam* rodiny vyčlenit pouze ty sek-

vence, které jsou svojí funkcionalitou, vlastnostmi a strukturou podobné k mému cílovému proteinu. Dolování relevantních sekvencí z proteinových databází je široká problematika, která je předmětem aktivního výzkumu. K jejímu řešení, pomáhají široké vědecké komunitě, veřejně dostupné vyhledávací nástroje dostupné ve formě webových rozhraní. Jedním z takovýchto nástrojů je i ***EnzymeMiner*** [29].

EnzymeMiner je volně dostupný vyhledávací nástroj pro vyhledávaní sekvencí upřednostňující takové kandidáty, u nichž je pravděpodobnější, že si zachovají katalytickou aktivitu a jsou heterologně exprimovatelné v rozpustné formě v Escherichia coli. Nástroj poskytuje řadu výstupních souborů, s nejrůznějšími statistickými a biologickými daty, včetně souboru s vícenásobným zarovnáním nalezených sekvencí. Vyhledávání *EnzymeMineru* se skládá ze tří částí:

1. vyhledání homologních sekvencí
2. filtrování sekvencí obsahující esenciální residua
3. sběr anotací sekvencí

Jako vstup nástroj vyžaduje zadat *query* sekvenci a šablonu s esenciálními residui. *Query* sekvence slouží jako základ pro počáteční homologické vyhledávání. Šablony esenciálních zbytků, definované jako dvojice proteinové sekvence a sady esenciálních zbytků v této sekvenci, umožňují serveru upřednostnit shody, které s větší pravděpodobností zachycují funkci enzymu. Esenciálními zbytky mohou být katalytické residua a residua vázající ligandy nebo kofaktory, které jsou nezbytné pro správnou katalytickou funkci.

V první fázi je možné poskytnout i více sekvencí pro homologní vyhledávání a tak rozšířit výslednou množinu. Společně s přidáním dalších sekvencí je zapotřebí v nich identifikovat i esenciální residua. V druhém kroku jsou pro jednotlivé *query* sekvence, a sekvence jim homologní, vystavěny mnohonásobná zarovnání sekvencí, kde dochází ke kontrole esenciálních residuí. Ve třetím kroku dochází k vytváření anotací z různých datových zdrojů.

Datová sada *HLDs*

Datová sada *HDLs* byla vytvořena na základě požadavku vytvoření vícenásobného zarovnání blízkého k zkoumaným proteinů halonalkan dehalogenáz, zejméně genu *DhaA*. Proteinové sekvence byly vybrány za využití nástroje *EnzymeMiner*. Jako vstup byla vybrána sekvence 3.8.1.5 – Haloalkane dehalogenase, pro kterou bylo použito všech **33** navržených *query* sekvencí společně s esenciálními residui za účelem pokud možno co největšího zvětšení výsledné datové sady.

I se zachováním všech navržených *query* sekvencí nebyla mocnost výsledného zarovnání oproti datové sadě *Pfam* příliš velká (kolem 8895 záznamů). Proto bylo experimentováno s nastavení vyhledávajících parametrů. Všechny volby pokročilého nastavení byly ponechány v původním nastavení kromě nastavení parametru *maximálního počtu nalezených sekvencí pomocí PSI-BLAST*. jeho hodnota byla zvýšena z původních maximálně 10000 analyzovaných záznamů na 50000. Tím bylo nástroji *EnzymeMiner* umožněno analyzovat více nalezených sekvencí, což mělo za následek vyšší počet výsledných sekvencí na výstupu. Výsledná datová sada byla složena z **22567** sekvencí.

Datová sada *HLD I-II*

Datová sada *hts–projekt* byla vytvořena na základě výsledků experimentů s předchozí sadou *HLDs*, která projevila tendenci ke generování nerozpustných proteinů (viz sekce výsledků

8.2), a práce v týmu Loschmidtových laboratoří [66]. Práce byla zaměřena na integraci bioinformatiky a mikrofluidiky při vysoce výkonné těžbě nových haloalkan dehalogenáz. Vícenásobné zarovnání sekvencí *EnzymeMineru* v [66] bylo na základě sekvenční podobnosti a kompozici katalytických pentád shlukováno do čtyř podrodin HLD-I, HLD-II, HLD-III, a HLD-IV.

Pro účely zvýšení rozpustnosti proteinů jsme po konzultaci výsledků z ostatních experimentů a diskuzi v týmy vyloučili z vyhledávání *query* sekvence, které byly zástupci podrodin HLD-III a HLD-IV vykazující nižší rozpustnost. Proto bylo provedeno vyhledání nástrojem EnzymeMiner za pomocí pouze dvou sekvencí **DhlA** a **LinB** v vstupní mu *custom sequences*. Jako *další známé sekvence* byly použity sekvence známých a experimentálně ověřených zástupců haloalkan dehalogenáz. Celkově bylo takto nalezeno **4053** sekvencí.

Kapitola 7

Implementace

V této části bude blíže popsány metody použité při evaluaci modelů a strategie výběru kandidátů z latentního prostoru pro případné laboratorní experimenty. Popíši také implementaci experimentů, včetně struktury programu a postupu vyhodnocení kandidátů. Konkrétní výsledky pro jednotlivé metody jsou uvedeny v kapitole 8.

7.1 Předzpracování a reprezentace vstupních dat

7.1.1 Předzpracování dat

Schopnost variačních autoenkodérů zachycovat evoluční souvislosti je podmíněna poskytováním vstupních sekvencí ve formě vícenásobného zarovnání. V hrubé formě takové zarovnání obsahuje tisíce sekvencí se šířkou daleko přesahující skutečnou délku naší sekvence. V případě datových sad uvedených v sekci 6.2 bylo běžnou praxí, kdy zarovnání pro sekvence délky kolem 300 aminokyselin mělo délku i 2000 znaků! Tento fakt je dán tím, že sekvence při zarovnání mohou být doplněny o mezery, které symbolizují například deleci/inzerci daného znaku během evoluce.

V případě, kdy budeme sekvence a aminokyseliny v nich obsažené reprezentovat jako čísla, lze vstupní zarovnání vnímat jako velmi řídkou matici. Pro zvýšení efektivnosti modelu se často uvádí předzpracování vstupních dat. Tento krok je velmi žádaný i v případě vstupu ve formě vícenásobného zarovnání sekvencí, kdy velká část modelu by byla vyplýtvána na učení se nepodstatných informací ve vstupní matici.

Předzpracování v mé implementaci probíhá vůči jedné sekvenci tzv. *query* sekvenci. Ta se může shodovat s *query* sekvencí použitou při hledání nástrojem *EnzymeMiner*, ale i nemusí. Výběr *query* sekvence v předzpracování je velmi podstatná událost pro výsledný vzhled a vlastnosti latentního prostoru. Sekvencí je vhodné volit tu, která je předmětem našeho zájmu. Kroky filtrace zarovnání totiž podporují pozice v ni obsažené a výsledky mají mírné zkreslení vůči této sekvenci. Předzpracování probíhá v po sobě jdoucích krocích zachycených na obrázku 7.1. Zprvu jsou načtena data ze zvoleného datového zdroje, kterým může být databáze *Pfam* či zarovnání z nástroje *EnzymeMiner*. Šířka zarovnání je výrazně redukována v druhém kroku předzpracování, kdy jsou pozice obsahující v *query* sekvenci mezery vyňaty z matice. Výjimkou jsou ty sloupce, které, i přes existenci mezery v *query* sekvenci na této pozici, jsou okupovány aminokyselinovými symboly pro více než 80 % sekvencí v zarovnání a jsou tak vyhodnoceny jako informačně a evolučně zajímavé (obrázek 7.1-2). Opětovné redukci šířky zarovnání dochází, za cílem zvýšení variability ve vstupní datové sadě a efektivnosti modelu, odstraňováním sloupců obsahující mezery na více než 20 %

řádcích. Tímto dochází k odstraňování residuí i z *query* proteinu, jehož varianty se snažíme rekonstruovat z latentního prostoru a takový zásah do struktury by mohl znamenat značné poškození struktury a funkčnosti proteinu. Proto jsou tyto sloupce odstraněny, ale jejich pozice a obsah uchován pro následné přidání do sekvence prezentované na výstupu procesu generovaní ancestrálních sekvencí ve zprávě pro uživatele. Čtvrtý krok odstraňuje sekvence bohatší na mezery než zvolený práh. Práh byl zvolen opět na hodnotu 20 %. Posledním krokem předzpracování je, v případě volby, shluková filtrace sekvencí na základě identity a váhování sekvencí (7.1-5). Při shlukování sekvencí jsou vytvořeny shluky se sekvencemi, které sdílí 90% identitu. Z každého shluku je vybrána pouze jedna sekvence pro trénování, přičemž je zajištěno, že zvolená *query* sekvence se mezi nimi vyskytuje.

Aby se snížila redundance a zdůraznila rozmanitost datové sady jsou sekvence v zarovnání ještě váhované. Vážení sekvencí může také snížit zkreslení v distribuci sekvencí druhů přítomných v zarovnání, protože u genomů některých druhů je větší pravděpodobnost, že budou sekvenovány více než jiné. Mnou použitá metoda váhování [25] je založena na přiřazování vah dle pozic a je dána následovně. Pokud označíme šířku zarovnání L a počet řádků jako N, pak znak s_j^n reprezentuje typ aminokyseliny n-té sekvence na j-té pozici. V této metodě je váha sekvence dána jako suma všech dílčích vah pro jednotlivé pozice v sekvenci. Abychom mohli určit váhu sekvence, nejprve musíme vypočítat matici $\left\{w_j^n : n = 1 \dots N, j = 1 \dots L\right\}$, kde w_j^n je příspěvek k váze n-té sekvence od j-té pozice a celková váha je vypočítán následovně:

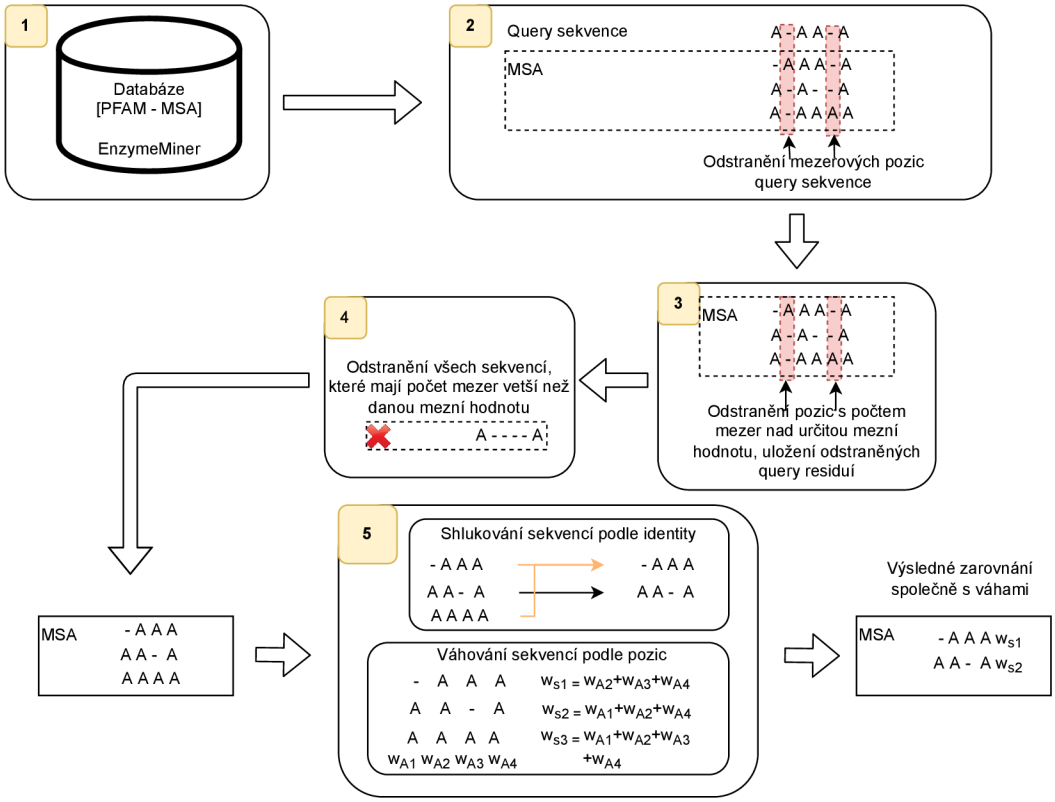
$$w_j^n = \frac{1}{C_j} \times \frac{1}{C_j^n} \tag{7.1}$$

kde C_j je počet unikátních aminokyselin v daném sloupci a C_j^n je množství sekvencí v zarovnání, které mají na j-té pozici stejnou aminokyselinu jako n-tá sekvence. Váha sekvence je nakonec vypočtena jako suma všech vah $\sum_{j=1}^{L} w_j^n$. Všechny váhy jsou normovány $\widetilde{w^n} = w^n / \sum_{j=1}^{L} w_j^n$ tak, aby suma všech vah byla rovna jedné.

7.1.2 Reprezentace vstupních dat

Mnoho modelů pro strojové učení vyžaduje na vstupu číselnou reprezentaci vstupních dat. V našem případě je na vstupu vždy jedna aminokyselinová sekvence z vícenásobného zarovnání, která na jednotlivých pozicích reprezentuje aminokyseliny jako abecední symboly. V souvislosti s variačními autoenkodéry lze vnímat učení se rekonstruovat sekvence z nízkodimenzionální reprezentace zpět na původní sekvenci jako klasifikační problém s kategorickými daty, kde se pro každý symbol snažíme predikovat jeho typ/kategorii. Proto by aminokyseliny mohly být reprezentovány jako číselné hodnoty, avšak použití tohoto kódování a umožnění modelu předpokládat přirozené uspořádání mezi kategoriemi může vést ke špatnému výkonu nebo neočekávaným výsledkům (předpovědi na půli cesty mezi kategoriemi). Používaným řešením je tzv. one-hot kódování.

One-hot kódování proteinové sekvence $S = (s_1, s_2, s_3, \dots, s_L,)$ ze zarovnání s L pozicemi je reprezentováno jako binární $21 \times L$ matice $\mathbf{X}$, pro kterou $X_{ij} = 1$ pokud platí $s_j = i$ jinak $X_{ij} = 0$. s_j odpovídá aminokyselině na j-té pozici v proteinu a jednotlivé typy aminokyselin jsou označeny čísly od 0 do 20, kde 0 reprezentuje mezeru v zarovnání a čísla od 1 do 20 odpovídají 20 esenciálním aminokyselinovým typům.



Obrázek 7.1: Schéma předzpracování vstupního vícenásobného zarovnání. 1) Data jsou načtena z *Pfam* databáze nebo dodaného *.fasta* souboru, 2) mezery v *query* sekvenci a příslušné pozice v zarovnání jsou odstraněny, nebo ponechány v případě kdy více jak 80 % sekvencí má na této pozici nějakou aminokyselinu, 3) odstranění těch sloupců zarovnání, kde více jak 20 % sekvencí má mezeru (pozice odstraněné z *query* jsou uchovány pro potřeby rekonstrukce) 4) sekvence s počtem mezer větší než stanovený práh jsou odstraněny, 5) shlukování sekvencí podle 90% podobnosti a náhodné vybrání jedné sekvence ze shluku, váhování sekvencí podle unikátnosti residuí v nich obsažených. Výsledek je zarovnání s redukovaným počtem mezer a s pozicemi s vysokou informační hodnotou.

7.2 Evaluace kvality modelů

Vyhodnocení kvality výstupu generativního modelu je netriviální úkol. U modelů, které generují text napodobující lidskou tvorbu, je zlatým standardem to, zda lidský čtenář dokáže rozlišit, zda byl text napsán algoritmem, nebo člověkem. Mezi další, lépe škálovatelné metriky patří statistická analýza modelem vyprodukovaný vzorků a měření perplexity generovaných sekvencí vypočtené modelem s různou architekturou. Pro proteinové sekvence uvažujeme za zlatý standard to, zda lze vygenerovaný protein vytvořit a v biologickém systému je schopný se sbalit do funkční konformace. To je však časově a finančně náročné a pro praktické aplikace nepoužitelné. Proto lze na vyhodnocování kvality modelu pohlížet jako na hodnocení „generativní kapacity“. Tento termín znamená schopnost modelu generovat nové sekvence tažené z naučené distribuce p(S), které jsou statisticky nerozlišitelné od těch z cílové proteinové rodiny. U takového statistického rozložení se nesnažíme o absolutní shodu, ale pouze do míry do jaké cílové statistické rozdělení popisuje část sekvenčního pro-

storu, které má určité požadované vlastnosti *in vivo*. Hranici statistické shody tedy musíme volit v závislosti na cílových vlastnostech generovaných proteinů [33].

7.2.1 Statistiky sekvence

V mém projektu jsem si kladl za cíl generovat mutanty sdílející původní proteinovou funkci zadané *query* sekvence a přitom zvyšovat jejich stabilitu. Proto jednou z důležitých vlastností modelu, kterou jsem se rozhodl sledovat, byla jeho schopnost **rekonstruovat přesné bodové zakódování** *query* sekvence zpět do původní reprezentace. V experimentech jsem upřednostňoval modely mající absolutní identitu. Sekvenční podobnost jsem se rozhodl měřit i pro ostatních sekvence v trénovací datové sadě. S obecně vyšší identitou rekonstrukce pro sekvence bylo očekáváno, že vygenerované kandidátní řešení bude s větší pravděpodobností biologicky relevantní. Také jednou z pozorovaných hodnot je marginální pravděpodobnost $p_\theta(X)$, kterou náš model vidí vygenerovanou sekvenci. Pro danou sekvenci $\mathbf{X}$ je marginální pravděpodobnost $p_\theta(X)$ určena jako integrál $\int p_\theta(X, Z)\, dZ$, který může být vypočítán pomocí Monte Carlo aproximace vzorkování důležitosti:

$$\begin{aligned} p_\theta(X) &= \int p_\theta(X, Z)\, dZ = \int q_\phi(Z|X)\, \frac{p_\theta(X, Z)}{q_\phi(Z|X))} dZ \\ &= \mathop{\mathbb{E}}_{Z \sim q_\phi(Z|X)} \left[\frac{p_\theta(X, Z)}{q_\phi(Z|X)} \right] = \frac{1}{N} \sum_{i=1}^{n} \left[\frac{p_\theta\left(X, Z^i\right)}{q_\phi\left(Z^i|X\right)} \right] \end{aligned} \tag{7.2}$$

kde Z^i značí nezávislé vzorky z distribuce $q_\phi(Z|X)$ a N odpovídá počtu vzorků. V případě této práce byla použita hodnota $N = 5 \times 10^3$. Vzorkování důležitosti bylo použito pro rychlou evaluaci generalizace modelu na validační datové sadě.

Další sledovanou statistikou je **průměrná rekonstrukční shoda**. Ta je určena vzorkováním, obdobně jako v předchozím případě, z distribuce dané střední hodnotou a variací naučenou během trénovací fáze. Následně vzorky z dané distribuce dekódujeme a vypočítáme průměrnou sekvenční podobnost mezi rekonstruovanými vzorky a původní sekvencí. Pro hlubší pochopení vlastností modelu pro různé skupiny sekvencí jsem před procesem trénování a evaluace kvality rozdělil data do následných stejně velkých disjunktních množin:

- *Trénovací kontrola*: je složena ze sekvencí vyskytujících se v trénovací sadě. Průměrná rekonstrukční shoda by pro tuto skupinu měla být relativně vysoká a slouží pro referenci vůči ostatním množinám.
- *Positivní kontrola*: skupina sekvencí, která byla náhodně odebrána z trénovací sady a odpovídá jejich 5 procentům. Tyto sekvence jsou původní a jejich statistická distribuce rekonstrukční shoda by měla stínovat trénovací množinu.
- *Negativní kontrola*: sekvence, které byly náhodně vygenerovány podle profilu celé předzpracované datové sady vícenásobného zarovnání. Slouží k determinaci hodnoty statistické energie pro sekvence vygenerované náhodným procesem. Distribuce hodnot negativní kontroly by měla být zřetelně separovaná a menší než ostatní distribuce.

7.2.2 Statistika prvního a druhého řádu

Vytvářením statistik prvního a druhého řádu se snažíme odhadnout kvalitu generovaných vzorků vůči původní datové sadě na základě frekvenční analýzy aminokyselinových skupin v daných pozicích sekvenčního zarovnání.

Statistika prvního řádu je porovnání četnosti výskytu každé aminokyseliny na každé pozici mezi dvěma soubory dat. V příkladě, kdy na určité pozici generovaného a vstupního zarovnání získáme hodnoty relativních četností znaku (0,5,0,5), by indikovalo, že pro danou pozici existuje aminokyselina, která se vyskytuje v 50 % případů generovaných vzorků a 50 % referenčních sekvencí.

Statistiky druhého řádu jsou podobné, ale zahrnují dvě pozice místo jedné. Hodnota četností v bodě (0,5,0,3) by poté značila, že 30 % generovaných sekvencí obsahuje v daném páru sloupců X, Y současný výskyt určitého páru aminokyselin *A1* na pozici X a *A2* na pozici Y, zatímco ve vstupní datové sadě se daný uspořádaný pár aminokyselin vyskytuje v totožných sloupcích v 50 % případů.

Pro získání většího vhledu do distribuce znaků v generované datové sadě je ještě počítáno párové kovarianční skóre daného následujícím vzorcem:

$$C^{ij}_{\alpha\beta} = f^{ij}_{\alpha\beta} - f^{i}_{a} f^{b}_{\beta} \tag{7.3}$$

kde $f^{ij}_{\alpha\beta}$ a f^{i}_{a}, f^{b}_{β} jsou statistiky druhého respektive prvního řádu pro sloupce *i, j* zarovnání. Každý kovarianční člen měří rozdíl mezi společnou frekvencí pro dvojice aminokyselin a součinem frekvencí reziduí na jednotlivých místech, tj. očekávané počty při hypotéze statistické nezávislosti. Pokud $C^{ij}_{\alpha\beta}$ se rovná 0 pro všechny $\alpha\beta$, poté pozice *i, j* nejsou v kovarianci. Spolu se vyvíjející aminokyseliny jsou důležitým aspektem sekvenční variability v proteinových zarovnání a schopnost generativního modelu reprodukovat párové kovarianční skóre trénovacího souboru dat byla v minulosti použita jako základní, netriviální míra schopnosti modelu modelovat kovariance proteinových sekvencí. Takto můžeme detekovat, jak dobře model zachycuje interakce mezi vzdálenými aminokyselinami, což je důležitý ukazatel pro pravděpodobnou stabilitu a funkci generovaných proteinů [45].

Pro každý model porovnávám párové kovarianční skóre pro všechny pozice a rezidua v generovaném ($\widehat{C}^{ij}_{\alpha\beta}$) a vstupním ($C^{ij}_{\alpha\beta}$) zarovnání pomocí Pearsonova korelačního koeficientu $\rho\left(\left\{C^{ij}_{\alpha\beta}\right\}, \left\{\widehat{C}^{ij}_{\alpha\beta}\right\}\right)$. V případě mých modelů byly statistiky vypočítány pro vstupních a synteticky generovaných datových sad majících 3000 náhodně vybraných vzorků. Syntetická data byla rekonstruována z bodů latentního prostoru, které byly vzorkovány dle zvolené apriorní distribuce, tj. *Gaussovi* distribuce se střední hodnotou rovnou 0 a variací 6.

7.2.3 Zachycení evolučních vztahů v prostoru

Pro prohledávací strategie latentního prostoru byla potřeba zajistit schopnost reflektovat v jeho struktuře evoluční závislosti. V původní studii [14] (obrázek 5.4) bylo ukázáno stabilní mapování trajektorie evolučního vývoje směrem do středu latentního prostoru v případě modelu používajícího jednoduchou jednovrstvou architekturu a *pfam* rodinu jako vstupní datovou sadu. Pro účely dosažení generování ancestrální sekvencí blízkých *query* proteinu bylo zapotřebí změnit datovou sadu a architekturu modelu. Proto jsem při vyhodnocování modelu bral v potaz i jeho schopnost mapovat trajektorie směrem do středu, aby prohledávací strategie mohly brát vzdálenost od středu latentního prostoru jako metriku potencionální kvality sekvence.

Pro měření kvality mapování evolučních závislostí do latentního prostoru bylo zapotřebí vytvořit fylogenetický strom s odvozenými ancestrálními sekvencemi. Pro evaluaci bylo vytvořeno celkem 9 fylogenetický stromů za pomocí nástroje FireProt–Asr [47]. Pro každý strom bylo použito celkem 100 náhodně vybraných sekvencí z trénovací datové sady. Počet listových uzlů stromu byl zvolen s ohledem na omezení velikosti vstupního zarovnání nástroje FireProt–Asr. Vytvořená stromová struktura s odvozenými sekvencemi byla poté namapována do latentního prostoru. Na základě namapování bylo vyhodnoceno vícero statistik, které byly vždy prezentovány ve vizuální formě histogramu.

V první statistice jsem sledoval korelaci mezi evoluční vzdáleností v dané větvi fylogenetického stromu od kořenového uzlu a vzdáleností v latentním prostoru, jak je ukázáno na obrázku 7.2. Listové uzly jsou očekávány blíže okraji prostoru, zatímco evolučně mladší sekvence by se měly nacházet u středu. Optimalizovaná trénovací datová sada nemusí nevytvářet rovnoměrný evoluční strom a tak gravitační bod trajektorií může být posun dále od středu. Proto jsem také kontroloval korelaci evoluční vzdálenosti s pozicemi kolem první komponenty směru v latentním prostoru (zelená šipka na obrázku 7.2). Deviace směrů prvních komponent byly nadále porovnány s optimálními trajektoriemi směřující přímo do centru (fialová šipka na obrázku 7.2). Výpočet deviace byl proveden jako skalární součin unárních vektorů optimální a odvozené trajektorie evoluce, kde hodnota 1 značí ideální směr do centra prostoru.

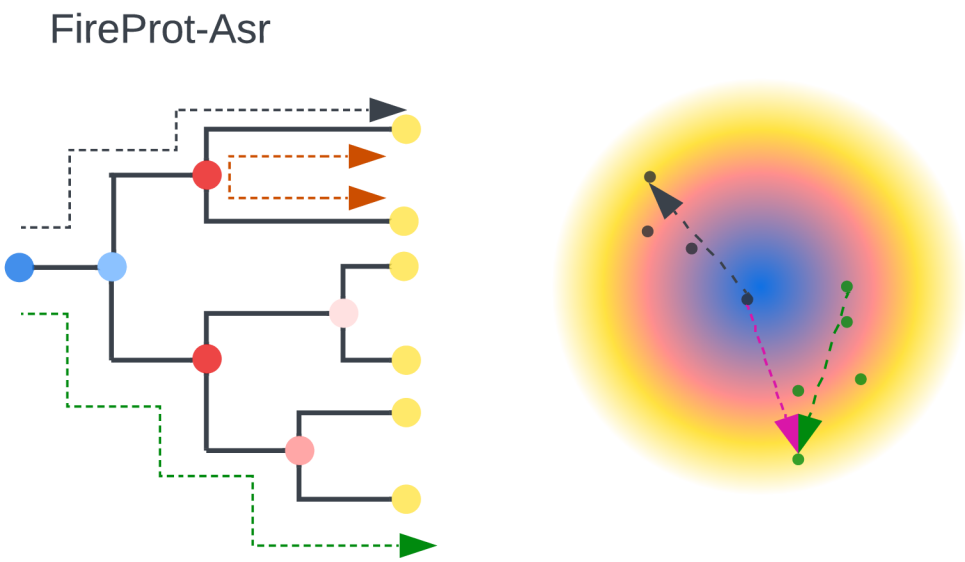
Popsaný proces mapování ancestrální sekvencí navržených pomocí nástroje FireProt–Asr musel být upraven, protože navržené sekvence obsahovaly více pozic než kolik bylo obsaženo v původních parametrech architektury sítě. Proto jsem se nejdříve snažil vytvořit zarovnání pomocí profilů s celou vstupní datovou sadou a poté zachovat pouze stejné pozice jako v případě původního předzpracování. Tento přístup však ukazoval špatné výsledky, kdy i listové sekvence byly namapovány daleko od originálního zakódování. Jelikož profilové zarovnání dvou množin sekvencí ztrácí pro VAEs důležité informace, rozhodl jsem se použít párové zarovnání s některou z původních sekvencí obsažených ve vstupním zarovnání pomocí nástroje *ClustalΩ* [58]. Původní sekvence byla zvolena mezi listovými uzly podstromu s kořenem tvořeným právě zarovnávanou ancestrální sekvencí (hnědý podstrom na obrázku 7.2 vlevo) a byla vybrána ta sdílející nejvyšší sekvenční podobnost.

7.3 Strategie pro dolování sekvencí

V mé práci jsem se zaměřil na hlubší studium vlastností latentního prostoru, které jsem následně využíval pro generování nových proteinových kandidátů. Proto jsem pro své experimenty zavedl, oproti předchozím studiím používajících pro generování proteinů náhodné vzorkování kolem zakódování šablonové sekvence, pokročilejší techniky prozkoumávání latentního prostoru, které budou popsány v následujících sekcích. Pro všechny strategie jsou generovány komplexní strategické ukazatele ve formátu tabulky a grafu, jak bude uvedeno v sekci 7.4.2 a konkrétní výsledky jsou uvedeny v kapitole 8.

7.3.1 Simulace metody řízené evoluce

Simulace metody řízené evoluce byla inspirována metodou proteinového inženýrství řízené evoluce, kdy jsou proteiny modifikovány a nejlepší varianty proteinu následně slouží jako šablona pro mutace v dalším kole. V experimentu dochází k náhodným mutacím záměnou aminokyselin na úrovni textové reprezentace proteinu. Program umožňuje i mutovat více pozic současně. Protein je poté zakódován do latentního prostoru. V závislosti na zkou-



Obrázek 7.2: Proces evaluace mapování evolučních vztahů do latentního prostoru. Fylogenetické stromy jsou mapovány po větvích. První metrika vyhodnocuje korelaci mezi evoluční a prostorovou vzdáleností (černá šipka). Nadále je porovnávána korelace pozic kolem první komponenty trajektorie v latentním prostoru (zelená šipka) společně s uhlovým rozdílem od optimální trajektorie směřující do centra prostoru (fialová). Hnědá šipka znázorňuje podstrom daného předchůdce pro výběr původní sekvence ke které bude provedeno párové zarovnání.

maných vlastnostech latentního prostoru a snahu navrhnou stabilnější ancestrální sekvence bylo v průběhu experimentů navrženo více variant *fitness* funkce, jak vybírat nejlepší kandidáty pro následné kolo evoluce.

Funkce nejbližší sekvence byla vytvořena na předpokladu schopnosti mapování ancestrální sekvencí blíže středu latentního prostoru, kdy jsem stanovil jako kritérium kvality sekvence její vzdálenost od centra prostoru (viz rovnice 7.4). Proto sekvence zvolená pro další krok evoluce byla zvolena ta mající nejmenší euklidovskou vzdálenost od středu prostoru.

$$fitness = ||z|| \tag{7.4}$$

S předchozí *fitness* funkcí docházelo k velkým krokům v latentním prostoru zavedením mutací, které nebyly typické pro vstupní zarovnáni. S velkou pravděpodobností tak mohlo docházet k implementaci destruktivní mutací pro funkci proteinu. Proto jsem se rozhodl do *fitness* funkce, nazvané **funkce biologicky stabilní sekvence**, zavést i další rozhodovací parametr na základě marginální pravděpodobnosti modelu $p_\theta(X)$ mutovaných sekvencí (viz rovnice 7.2) zvyšující pravděpodobnost správné funkce proteinu. Fitness funkce byla dána následovně:

$$fitness = (1 - p_\theta(X_i)) + ||z_{rel}|| \tag{7.5}$$

kde $z_{rel} = X_i / max\left(\{X_0, \ldots, X_{n-1}, X_n\}\right)$ značí relativní vzdálenost od centra latentního prostoru vůči nejvzdálenějšímu zakódování.

Algoritmus řízené evoluce byl ukončen v případě, kdy vybraný kandidát byl v latentním prostoru umístěn blíže středu než stanovený práh. Pro všechny experimenty byl tento práh zvolen na vzdálenost 0.08.

7.3.2 Metoda přímé evoluce

V simulaci metody řízené evoluce jsem se zaměřil na optimalizaci na základě textové reprezentace sekvence společně se signály v podobě souřadnic latentního prostoru a případně pravděpodobnosti vidění naší sekvence modelem. Avšak nebyla využita, nebo pouze omezeně pro marginální pravděpodobnost, schopnost dekodéru navrhovat sekvence na základě bodů latentního prostoru, a proto jsem navrhl metodu přímé evoluce. Myšlenka přímé evoluce je založena na přístupu, kdy zakóduje *query* sekvenci do její latentní reprezentace a z ní poté vede vektor evoluce do středu prostoru kopírujíc myšlenku mapování ancestrální závislostí do latentního prostoru (viz obrázek 5.4). Vektor je rozdělen po rovnoměrných intervalech a body ležící na jejich hranicích jsou zvoleny pro rekonstrukci do textové podoby pomocí dekodéru. V experimentech byl počet rekonstruovaných bodů zvolen na 100.

Společně s generací ancestrální sekvencí lze tento přístup využít i k opačnému problému a to návrhu evolučních potomků. Za tímto účelem je evoluční vektor rozšířen i opačným směrem a je rekonstruováno prvních 30 kandidátů.

7.3.3 Evoluční strategie adaptace kovarianční matice

Prohledávání latentního prostoru za účelem maximalizace určité funkce je netriviální výpočetně náročný proces, zejména když zkoumaná funkce je komplexní, jako je tomu i v případě neuronových sítí a proteinové domény. Jedním ze způsobů takové optimalizace je zapojení evolučních optimalizačních strategií. Evoluční výpočet je rodina univerzálních optimalizačních technik, které potřebují pouze metodu pro reprezentaci a porovnávání řešení k nalezení optimálního řešení.

Jednou z evolučních strategií je evoluční strategie adaptace kovarianční matice (CMA-ES). Jedná se o robustní přístup, u kterého bylo prokázáno, že funguje na nelineárních a nekonvexních doménách, jako je optimalizace vlastností generovaného obrázku [8]. CMA-ES vzorkuje svou populaci z vícerozměrné normální matice rozdělení. Protože každé řešení je reprezentováno jako kombinace proměnných (v našem případě zakódovaná v latentní reprezentaci), udržuje CMA-ES kovarianční matici, která sleduje, jak jednotlivé proměnné ovlivňují fitness. V každé generaci strategie vytváří nový vzorek dat na základě informací, které jsou v kovarianční matici obsaženy. Matice se aktualizuje na základě fitness nových vzorků, což umožňuje algoritmu naučit se rozdělení úspěšných jedinců. Takto naučený model je poté aproximační model druhého řádu, díky čemuž je CMA-ES výkonnou strategií pro optimalizaci obtížné domény s reálnou hodnotou [22].

Díky komplexním vlastnostem CMA-ES jsem se tuto evoluční strategii rozhodl použít pro optimalizaci generovaných proteinů z latentního prostoru. Dle informací v [22] byla implementována CMA-ES strategie. Strategie nepracuje na úrovni textové reprezentace aminokyselinové sekvence, ale pomocí robustního přístupu vzorkování dle kovarianční matice prohledává redukovaný prostor VAEs. Algoritmus začíná optimalizační problém v souřadnicích odpovídajících zakódované *query* sekvenci, generuje body z latentního prostoru, dekóduje je a vytváří ohodnocení kandidátu na základě jejich textové a latentní reprezentace. Dle získaného ohodnocení je aktualizována kovarianční matice pro vzorkování v dalším kole. Tím CMA-ES využívá závislosti vstupních sekvencí naučené v enkodéru a hledá skrytá místa latentního prostoru, které optimalizují danou objektivní funkci.

V případě evolučních algoritmů nejdůležitější článkem bývá tzv. *fitness* funkce, která slouží pro výběr nejlepší kandidátů používaných v dalším kole iterace jako šablona pro následnou optimalizaci. V případě proteinových sekvencí takovou optimální *fitness* funkci, reflektující vlastnosti proteinů, není jednoduché navrhnout (zda–li nemožné). Proto ji musíme vystavět na příznacích aminokyselinových sekvencí potažmo hodnotách generativního modelu, které jsme schopni výpočetně rozhodovat. Po řadě experimentů bylo zvoleno, že *fitness* funkce bude složena z více dílčích optimalizovaných metrik, které byly vybrány následovně:

- **Vzdálenost od centra prostoru**: na základě pozorování vlastností latentního prostoru a cíle navrhnout stabilnější ancestrální sekvenci byla do funkce zahrnuta vzdálenost od středu prostoru mírně favorizující bližší body. Navíc funkce řídí proud evoluce směrem ke středu prostoru.
- **Cílová identita s *query* sekvencí**: udává žádanou procentuální sekvenční podobnost vygenerované sekvence s proteinem *Dhaa*. Umožňuje hlídat podobnost vygenerovaných sekvencí. Použita hodnota 92.5 %.
- **Logaritmická věrohodnost**: vrací věrohodnost generované sekvence vůči zbytku trénovací datové sady. Vyšší hodnoty značí pravděpodobnější náležitost do proteinové rodiny.
- **Tm hodnota**: do *fitness* funkce jsem zakomponoval i známé hodnoty stability (Tm) pro mutanty sekvencí blízkých *query* sekvenci získané z databáze FireProtDB [62]. Tyto hodnoty jsou interpolovány do latentního prostoru pomocí metody Gaussových procesů. Bohužel se jedná jen o malý počet mutantů a tak informace získaná z této interpolace není u všech kandidátů využívána, kvůli zvolené hranici akceptace nejistoty interpolované hodnoty.

Jedná se tedy o vícekriteriální optimalizaci, jejíž vyhodnocení lze provést více způsoby. Prvním je pomocí tzv. váhované podílu jednotlivých funkcí dávající dohromady jednu hodnotu. Jedná se o jednoduché řešení, které je však velmi obtížné nastavit tak, aby dávalo dobré výsledky. Proto byla pro výběr nejlepších kandidátů použita i tzv. *pareto* fronta. *Pareto* fronta vytváří množinu řešení, které nejsou dominovány žádným jiným. Dominované řešení je takové, které je ve všech sledovaných kritériích horší než jiné řešení. Kombinací prvků z této množiny je poté vytvořena nová střední hodnota pro další kolo evoluce.

Evoluční algoritmus běží po zvolený počet běhů. V každém kole je provedeno ohodnocení všech kandidátů a pokud některý splňuje cílové podmínky, může být prohledávání ukončeno i předčasně s dosaženým výsledkem. V mém případě je *fitness* funkce stavěna oproti optimalizaci sekvenční podobnosti s *query* sekvencí s mírou podobnosti 92.5 %.

Po experimentech a statistické analýze výsledků jsem stanovil počet generací na 50, velikost generované populace na 64, v algoritmu jsem kontroloval velikost kroku a dle hodnot matice ji upravoval. Počet běhů byl stanoven na 10. Počáteční krok algoritmu byl podroben důkladné statistické analýze a jeho hodnota byla mezi zkoumanými hodnotami v pravidelných intervale po 0.05 mezi 0.1 až 0.5 vyhodnocena jako 0.25.

7.4 Popis implementace a ovládání programu

Pro snadnější práci s generativním modelem byla implementována řada dílčích skriptů s možnou konfigurací v podobě textové reprezentace v souboru typu `json`. Program pro

ovládání vyvinuté aplikace je umístěn a spouštěn z podadresáře `/scripts/`. Kořenový adresář složky `/scripts/` obsahuje zdrojové soubory napsané převážně v jazyce Python rozdělené do balíčku dle jednotlivých úkolů. Zde je přítomen i modul implementující model VAEs s názvem `VAE_model.py`. Tento skript byl převzat z původní práce a modifikován pro naše účely.

Jazyk Python se v komunitě strojového učení stal určitým standardem a disponuje řadou frameworků velice ulehčujících práci s neuronovými sítěmi a vyhodnocením dat. Jedním z frameworků je volně dostupná knihovna PyTorch, která byla vybrána pro implementaci mé práce. Vizualizace výsledků je převážně provedena pomocí knihovny Matplotlib a Seaborn. Nadále byl použit i jazyk Shell pro automatizaci posloupnosti kroků nutných k natrénování a evaluaci modelu.

Převážná část práce byla vyvinuta za využití kapacit poskytnutým výpočetním gridem MetaCentrum[1]. Řada analýz byla provedena v interaktivním módu na výpočetních *cpu*, v případě trénování *gpu*, uzlech MetaCentra. Pro některé náročné úlohy bežící paralelně na mnoha výpočetních uzlech byly připraveny *pbs* skripty umístěné v adresáři `/pbs-scripts/`. Na *gpu* uzlech trvalo trénování modelu dle zvolené architektury a datové sady od **desítek minut až po jednotky hodin**. Skripty generující kandidátní sekvence běží v řádu minut. Při celkové práci na tomto projektu bylo spotřebováno **38 CPU dní**.

7.4.1 Ovládání programu

Program předpokládá spuštění z adresáře `/scripts/` a celý soubor operací potřebný pro práci s jednotlivý skripty pro dílčí kroky výpočtu je zapouzdřen prostřednictvím souboru `runner.py`. Soubor zpracuje příslušnou konfiguraci ve formátu `json` nacházející se ve stejném adresáři v podsložce `/model_configurations`. Aplikované nastavení experimentu se mezi konfigurace vybírá nastavením příznaku „status“ na hodnotu *on*. Konfigurační soubor může sloužit jako knihovna nastavení jednotlivých experimentů, avšak uživatel musí dbát na to, aby byl na hodnotu *on* nastaven nanejvýše jeden experiment. V opačném případě na to skript chybovým hlášením upozorní a ukončí se. Skript `runner.py` vyžaduje alespoň jeden parametr reprezentující název skriptu, který se má provést s aktuálně zapnutou konfigurací modelu. Pro spuštění komplexnějších úloh evaluace modelu a vytváření kandidátů byl vytvořen soubor `run_task.py`, který obsahuje řadu parametrů pro jednotlivé úlohy. Seznam podporovaných úloh je dostupný příkazem `python3 runner.py -h`. Všechny parametry lze nalézt v konfiguračním souboru a případně si je přenastavit.

Po nastavení parametrů modelu a protokolů v konfiguračním souboru, celý postup na předpřípravě dat, učení, evaluaci modelu a vytvoření kandidátu lze shrnout do série jednoduchých příkazů:

```
python3 runner.py msa_handlers/msa_preprocessor.py

python3 runner.py train.py

python3 runner.py run_task.py --run_generative_evaluation

python3 runner.py run_task.py --run_generative_evaluation_plot

python3 runner.py run_task.py --run_random_mutagenesis
```

[1] https://metavo.metacentrum.cz/

Soubory jsou vygenerovány do `/results/experimentDir/experimentName`, kde *experimentDir* je název složky shrnující všechny experimenty v konfiguračním souboru a lze jej nastavit v jeho horní části. Adresář *experimentName* poté odpovídá konkrétnímu názvu experimentu. Vygenerované výstupy příslušného experimentu uživatel nalezne v adresáři experimentu ve složce `/highlights/`.

Složka `pbs_scripts/` v kořenovém adresáři projektu sdružuje pomocné pbs skripty, které lze využít pro trénování a evaluaci modelu bez interaktivního využití výpočetních uzlů MetaCentra. Výsledky jsou v případě bezchybně ukončeného výpočtu přeneseny v kompresované formě z dočasného úložiště na výpočetním uzlu do složky `results/` v kořenovém adresáři. V případě chyby lze případné vygenerované soubory získat překopírováním z příslušného uzlu pomocí příkazu `scp`.

Instalace: lze provést za využití prostředí Anaconda a pomocí manuálu, který je popsán v souboru `README.md`. Před použitím pbs skriptů je potřeba upravit jednotlivé cesty dle aktuálního umístění programu uživatelem.

7.4.2 Výstup experimentu

Pro všechny strategie byl vytvořen jednotný reportovací modul `experiment_handler.py`. Ten pro navržené sekvence vytvoří sadu statistik vygenerovaných do *csv* souboru, spustěný automaticky po každé generační strategii. Skript lze za pomoci souboru `dual_axis.py` ve složce `support_scripts/` převést na graf zobrazující průběh sledovaných statistický parametrů přes jednotlivé navržené sekvence. Mezi sledované hodnoty patří průměrná rekonstrukční shoda, podobnost s původní sekvencí, sekvenční podobnost s nejbližší sekvencí v latentním prostoru z trénovací sady a počet residuí mající pravděpodobnost přiřazenou od VAEs větší než 90 % společně s počtem vložený/smazaných znaků z původní sekvence. Takto vynesené hodnoty jsou poté vizuálně zkontrolovány a body vykazující zajímavé statistické hodnoty mohou být vybrány pro následnou charakterizaci v laboratoři. Příklad výstupu je uveden v příloze A.1.

Kapitola 8

Experimenty

Tato kapitola je věnována sumarizaci výsledků experimentů s jednotlivými datovými sadami a strategiemi popsanými v předešlých kapitolách. Na závěr jsou porovnány mé výsledky s výsledky **dostupnými v literatuře**.

8.1 Reprodukce protokolu

Úvodní fázi projektu jsem se zaměřil na ověření původního protokolu uvedeného v referenčním článku [14] na naší proteinové rodině haloalkan dehalogenáz. Pro tento účel byla získána datová sada *Pfam* rodiny s označení PF00561 (viz 6.2). S touto datovou sadou jsem detailně zkoumal vlastnosti vygenerovaných latentních prostorů a jejich citlivost na nastavení parametrů při předzpracování sekvenčních dat. Původní pracovní protokol byl přepsán a více parametrizován, aby vyhovoval podmínkám testování pod různým nastavením parametrů.

Citlivost na volbu *query* sekvence

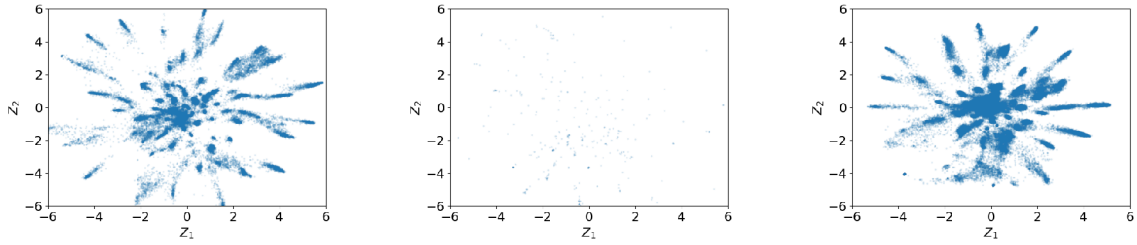
V předchozí sekci 7.1 byl popsán pracovní protokol předzpracování vstupního zarovnání. Jedním z parametrů je i tzv. *query* sekvence, nebo-li sekvence podle které probíhá filtrování ostatních sekvencí ze zarovnání. Pro ověření efektu volby dané sekvence na výsledný vzhled latentního prostoru a vlastnosti zpracovaného zarovnání jsem provedl řadu experimentů, které jsou krátce sumarizovány v tabulce 8.1. PIP_BREBN/26-275 a A0A1T1HAJ7_OCELI sekvence byly zvoleny jako *query*. V posledním scénáři jsem z protokolu předzpracování vyloučil druhý krok a vyzkoušel vzhled latentního prostoru bez použití filtrování vůči *query* sekvenci.

Identifikátor query sekvence	**Počet ponechaných sekvencí**	**Šířka zarovnání**
PIP_BREBN/26-275	40812	222
A0A1T1HAJ7_OCELI	331	151
Bez query sekvence	80132	97

Tabulka 8.1: Přehled velikosti výsledného trénovací datové sady po aplikaci rozdílných *query* sekvencí. V případě předzpracování bez volby *query* sekvence byl vynechán krok číslo 2 (viz obrázek 7.1).

V obrázku 8.1 jsou znázorněny latentní prostory pro všechny případy z tabulky 8.1. Z vybrané dotazovací sekvence PIP_BREBN/26-275 a z výše uvedené tabulky vyplývá,

že pro tréninkový postup bylo ponecháno 40812 sekvencí. Ostatní byly odstraněny. Šířka zbývajícího zarovnání je 222 aminokyselin. Latentní prostor je dobře shlukovaný (obrázek 8.1 vlevo). Experimenty prokázaly, že výběr dotazovací sekvence nemá žádný vliv na její polohu v latentním prostoru z hlediska vzdálenosti od středu latentního prostoru. Obrázek 8.1 uprostřed ukazuje, že postup filtrování a výsledný latentní prostor jsou citlivé na volbu *query* sekvence. Částí zpracování prošlo pouze 331 sekvencí a latentní prostor postrádá jakoukoli zjevnou strukturu. Vybraná dotazovací sekvence není ze zárodečných (seed) sekvencí. Nakonec jsem také otestoval, co se stane, pokud není zadána žádná dotazovací sekvence (obrázek 8.1 vpravo). I zde byly odstraněny pozice s více než 20 % mezer. Šířka zarovnání je rovna 97 aminokyselinám. Latentní prostor vykazuje strukturu, ale není shlukován tak zřetelně jako v případě dotazovací sekvence PIP_BREBN/26-275. Závěrem tohoto testu citlivosti protokolu předzpracování je, že jak zbývající sekvence v zarovnání, tak latentní prostor jsou velmi citlivé na referenční sekvenci a že výběr žádné sekvence stále zachovává shlukovou strukturu prostoru.



Obrázek 8.1: Vzhled latentního prostoru pro různé volby *query* sekvence. **Vlevo**: latentní prostor pro PIP_BREBN/26-275 je dobře strukturován, **uprostřed**: *query* sekvence byla zvolena náhodná sekvence A0A1T1HAJ7_OCELI nepatřící mezi seed sekvence *Pfam* rodiny PF00561 a protokolem předzpracování byla vyřazena většina sekvencí, **vpravo**: latentní prostor bez zvolené *query* sekvence stále zachovává shluky ne však tak ostré jako v prvním případě.

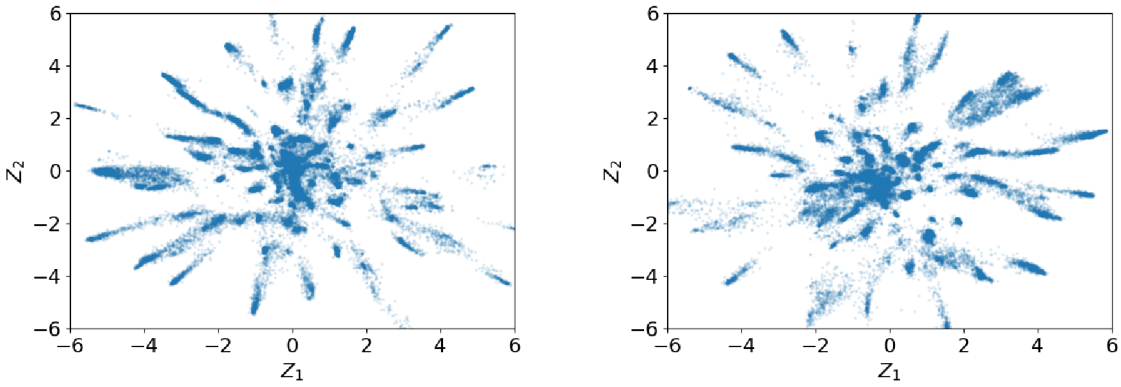
Citlivost na krok odstraňující mezery

Tento experiment demonstruje vliv odstranění pozic bohatých na mezery ze vstupní datové sady (viz schéma 7.1). Pro tento účel jsem upravil protokol předzpracování sekvencí, kdy byly ponechány všechny pozice, kde *query* sekvence měla aminokyselinový symbol, a odstraněny ty mající víc jak 20 % mezer. Má hypotéza byla, že množství sekvencí a skladba jednotlivých pozic v zarovnání může změnit rozložení latentního prostoru. Zpracování zarovnání v pracovním postupu s udržováním mezer bylo tedy následující:

- Protokol je proveden na datové sadě *Pfam* rodiny PF00561.
- *Seed* sekvence PIP_BREBN/26-275 je zvolena jako základní *query* sekvence.
- Podle zvolené *query* sekvence, jsou pozice s více než 20 % mezer na pozicích neobsazených aminokyselinou v *query* sekvenci odstraněny ze vstupního zarovnání.

Výsledná šířka zarovnání byla 250 a 222. Výsledné prostory jsou zachyceny na obrázku 8.2. Zatímco latentní prostor se mění v závislosti na tom, zda pozice s mnoha mezerami jsou zachovány, nebo ne, struktura shlukování zůstává zachována, což svědčí o nízké citlivosti

latentního prostoru na odstranění sloupců. To je cenné pozorování, protože v případě proteinového inženýrství nás zajímá zachování všech pozic v dotazované sekvenci pro generování slibných mutantů.



Obrázek 8.2: Latentní prostor pro *query* sekvenci PIP_BREBN/26-275 s ponechánými mezerami (**vlevo**) a bez mezer ponechaných mezer mimo pozice *query* sekvence (**vpravo**).

Analýza pozic *seed* sekvencí v latentním prostoru

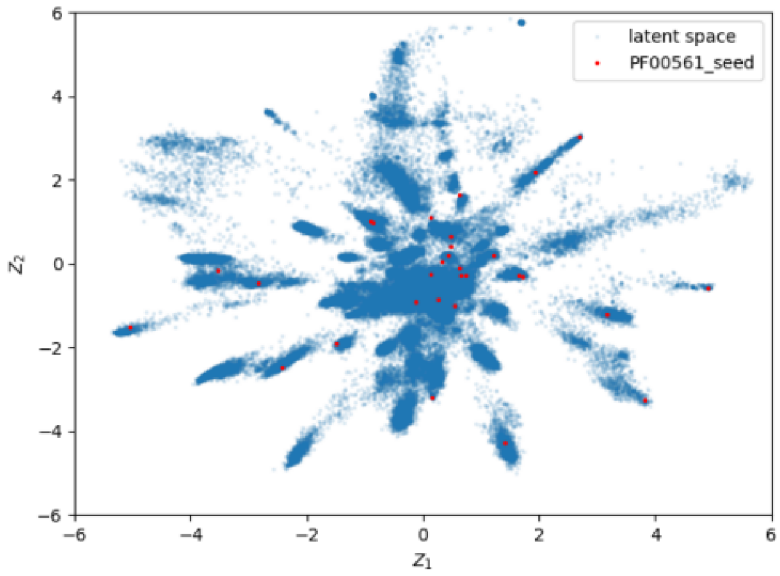
Účelem tohoto experimentu bylo namapovat *seed* sekvence *Pfam* rodiny PF00561 do latentního prostoru a zjistit, zda je jejich pokrytí rovnoměrné v celém prostoru, nebo je soustředěno v jeho určité části. Předpokladem bylo jejich rovnoměrné rozprostření do jednotlivých shluků v prostoru, jelikož tyto sekvence tvoří při sestavování *Pfam* rodiny šablony, proti kterým dochází k vyhledávání sekvencí této rodiny. Proto by každá *seed* sekvence měla mít v dané rodině určitý nemalý počet sekvencí sdílející vysokou podobnost. Předzpracování sekvencí bylo provedeno standardním protokolem dle sekce 7.1.1 bez kroku shlukování sekvencí.

Soubor *seed* sekvencí obsahoval 48 sekvencí. Překvapivě se mi podařilo identifikovat pouze 32 z nich v datové sadě PF00561 prostřednictvím shody názvu sekvencí.

Obrázek 8.3 ukazuje, že shluky v latentním prostoru jsou poměrně rovnoměrně pokryty *seed* sekvencemi, jak jsem předpokládal a variační autoenkodér je schopný zachytit sekvenční rozmanitost ve svém prostoru. Avšak některé části latentního prostoru nejsou pokryty, což může být způsobeno tím, že ne všechny *seed* sekvence byly nalezeny v zarovnání.

Vyhodnocení shody mezi reprezentativními podrodinami, evolučními větvemi a latentním prostorem

Cílem tohoto posledního experimentu bylo vyhodnotit, zda je zpřesnění zarovnání v souladu s naučeným latentním prostorem. Toto zpřesnění bylo provedeno pomocí podmnožin RPXX *Pfam*, které se skládají z reprezentativních proteomů (RP). (i) Každý člen RP musí být dobrým reprezentativním (v evolučním kontextu) proteomem; (ii) člen RP by měl být nejvíce funkčně charakterizovaným/anotovaným členem skupiny; a (iii) RP na různých prahových hodnotách by mělo být hierarchické [10]. XX označuje práh spolučlenství (15, 35, 55, 75). Tento experiment prokazuje pravidelné podvýběr RP75 do podmnožin RPXX. Zkoumanou rodinou byla opět *Pfam* rodina PF00561.



Obrázek 8.3: Pokrytí latentního prostoru pro datovou sadu PF00561 odpovídajícími *seed* sekvencemi (červeně).

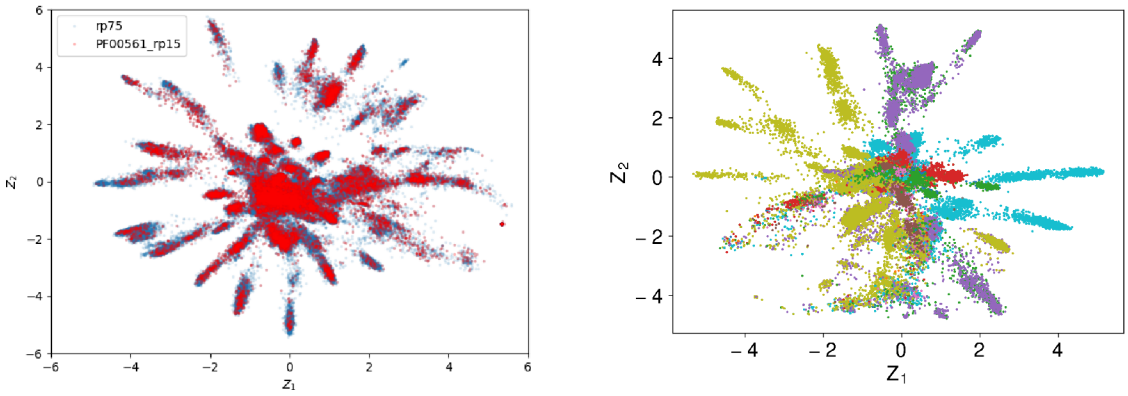
Obrázek ukazuje, že očekávané rovnoměrné podvzorkování RPXX podmnožin je reflektováno variačními autoenkodéry a i pro nejmenší množinu RP15 platí, že rovnoměrně pokrývá latentní prostor množiny RP75 90 (viz obrázek 8.4 vlevo).

Zda jednotlivé větve fylogenetického stromu lze úspěšné namapovat do latentního prostoru jako v původní studii [14] jsem vyzkoušel i na naší rodině proteinů. Výsledek mapování lze vidět na obrázku 8.4 vpravo. Fylogenetický strom byl vytvořen pomocí nástroje FastTree2 [52] a průchod společně se zpracováním vygenerovaného stromu byl realizován za využití knihovny ete3. Obrázek 8.4 vpravo odhaluje, že v případě reálné proteinové rodiny je výkon rovnoměrného mapování evoluce do prostoru slabší než v případě simulovaného zarovnání použitém v na obrázku 5.3 a jednotlivé evoluční větve jsou v latentním prostoru v některých případech promíchané. Původní studie ukazuje i výsledky na reálných proteinových rodinách, konkrétně na rodině domén *fibronektinu* typu III, které ukazují i pěkné hvězdicové uspořádání. V ní ale autoři vybrali 20 nejpočetnějších skupin sekvencí v daném fylogenetickém stromě, zatímco já ukazuji přesné výsledky pro určitý řez stromu.

8.2 Experimenty se sadou *HLDs*

Experiment, který bude popsán v následující sekci, byl zaměřen na úzkou optimalizaci modelu na úloze generovat relevantní sekvence pro *Dhaa* na datové sadě *HDLs*.

Jako *query* sekvence pro trénování byla vybrána gen DhaA s *uniprot id* P59336_S14 (Haloalkan dehalogenáza z Rhodococcus sp). Na konci fáze redukce délky sekvencí byla konečná délka zarovnání rovna 299 residuí. Následně bylo provedeno filtrování vstupního zarovnání pomocí shlukování, abych snížil identitu v souboru dat a zvýšil rozmanitost v latentním prostoru. Shlukováním sekvencí bylo provedeno podle 90% identity a výběrem jedné sekvence z každého shluku do finální datové sady. Výběr dotazovací sekvence P59336_S14 byl zajištěn. Výsledný počet sekvencí byl snížen na 12053.



Obrázek 8.4: Latentní prostor dobře reflektuje podvzorkovací proces v RPXX *Pfam* podrodinách (**vlevo**). Fylogenetický strom pro PF00561 namapován do latentního prostoru pro jednotlivé evoluční větve ve zkoumaný evoluční okamžik (**vpravo**).

8.2.1 Optimalizace parametrů modelu

Celkový model je složen z dílčích částí definovaných vlastní sadou parametrů, jejichž optimalizace má potenciál vylepšit jeho generativní vlastnosti. Proto jsem se rozhodl prozkoumávat následující parametry:

- Redukce vlivu regularizace ve ztrátové funkci v prospěch rekonstrukční chyby;
- Zvyšování dimensionality latentního prostoru
- Přidáváním více vrstev s proměnlivým počtem neuronů do enkodéru a dekodéru;
- Modifikování parametrů optimalizátoru.

V experimentech byla pozorována generativní schopnost modelu dle vztahu v 7.2 společně s průměrnou procentuální identitou pro každou sekvenci v trénovací sadě a identitou *query* sekvence. Pokud není specifikováno jinak, tak dimensionalita prostoru byla 2, C 2, jedna skrytá vrstva s 100 neurony a *weight decay factor* 0.01.

Z počátku jsem se zaměřil na **vliv regularizace**. Proces učení VAEs je založen na ztrátové funkci, které je složena z regularizační a rekonstrukční části (viz rovnice 5.4). Regularizační část je tvořena z Kullback–Leiblerovi divergence a nutí enkodér kódovat sekvence více do středu za vytváření více spojitého prostoru. Daní za to je snížení rekonstrukční schopnosti modelu. Podíl jednotlivých částí lze ovlivnit pomocí parametru C, kdy vliv regularizačního termu roste nepřímo úměrně k C. Z experimentů s architekturou mající jednu vrstvu se 100 neurony a s hodnotou $C = \{1, 2, 5, 25, 125, 625, 3125\}$ vyplynulo, že hodnota C **nezlepšuje generativní vlastnosti** modelu, avšak způsobuje rozptýlení prostoru, což by mohlo mít za následek vznik „slepých“ regionů v latentním prostoru. Proto byla hodnota C nastavena na počáteční hodnotu C (pokud nebude specifikováno jinak).

Se stejnou architekturou neuronových sítí jsem následně zkoumal **vliv dimensionality** latentního prostoru na rekonstrukční vlastnosti. Výsledky $D = \{2, 3, 4, 5, 6, 8, 16, 20, 33\}$ prokázaly, že zvyšující počet dimensí **zlepšuje generativní možnosti modelu**. Navzdory slibným výsledkům týkajícím se *Dhaa* párové identity (až 97% pro 20 dimensí), zvyšující se dimensionalita přináší nevýhodu v podobě horší interpretovatelnosti ve vícerozměrném prostoru a relativně malého zlepšení celkové párové podobnosti (74,5 na 79,5 %). Navíc při

33 dimenzích jsem pozoroval fenomén kolapsu dimenzionality, kdy přidávané komponenty do latentního prostoru nejsou schopné se naučit užitečnou informaci o datové distribuci. Hledáme tedy jinou možnost, jak zlepšit vlastnosti modelu při zachování nízké úrovně dimenzionality. Níže uvedené výsledky ukáží, že zlepšení modelu je možné i v případě dvourozměrného latentního prostoru.

Dalším zkoumaným parametrem byl **počet parametrů v neuronové síti**. První experiment byl zaměřen na ponechání pouze jedné skryté vrstvy s 200 neuronů. V dalších experimentech jsem zvětšil hloubku sítě na vrstvy s 100; 50 respektive 100; 50; 50 neuronů. Výsledky ukázaly, že zvětšování hloubky sítě neřeší problém generativního schopností modelu. V případě jedné vrstvy s 200 neurony došlo zvýšení generativních schopností. V případě 2 skrytých vrstev došlo k stlačení latentního prostoru směrem do středu, což bylo překvapující zjištění, které může mít 2 vysvětlení: s větší hloubkou roste vliv parametru C, nebo tu nebyl dostatečný počet trénovacích epoch, aby se model naučil korektní rozložení prostoru (10000 epoch). Pro 3 skryté vrstvy jsem pozoroval opět fenomén kolapsu dimenzionality, kdy síla parametrů modelu je dostatečná, aby mohla zakódovat všechny body do jedné dimense.

Poté jsem se snažil náš model více přeučit při trénování po 59000 epoch, které byly určeny na základě analýzy deviace validační a trénovací chyby. Testování bylo provedeno opět na vícero architekturách. Ani tento krok nebyl úspěšný mající identitu s původní *DhaA* sekvencí pouze 65 %.

V další kroku jsem přešel k testování vlivu hodnoty **weight decay** faktoru v nastavení optimalizátoru. Význam *weight decay* je v penalizování vysokých hodnot vah čímž je omezuje volnost modelu. *Weight decay* faktor je dobré použít v případě vyžadované regularizace neuronové sítě a v případě VAEs je běžnou praxí. V experimentech jsem otestoval hodnoty $w = \{0.01, 0.005, 0.001, 0.0005, 0.0001, 0.0\}$. Identita *DhaA* sekvence s klesající hodnotou *weight decay* rostla z původních 64 % až na 90 %. Nejlepší je model s faktorem nastaveným na 0. Odstranění gama členu ze ztrátové funkce odstraňuje regularizace vah, což umožňuje nastavit váhy na požadované hodnoty. Faktor rozpadu nastavený na 0 obecně zvyšuje párové skóre, ovšem musíme brát v potaz, že může způsobit horší výsledky v prázdných oblastech latentního prostoru.

Po vyhodnocení předešlých experimentů jsem se znovu vrátil k jejich optimalizaci tentokrát s nastavenou hodnotou *weight decay* na 0.0. Po provedení experimentů pro jednotlivé hodnoty parametru C, počtu dimenzí a velikosti/hloubky skrytých vrstev jsem rozhodl provést následující pokus generace proteinů pro laboratorní testy s **zvoleným modelem** majícím jednu skrytou vrstvu s počtem neuronů odpovídajícím šířce zarovnání. Parametr C byl ponechán na hodnotu 2, *weight decay* 0.0 a s učícím se krokem 0.001.

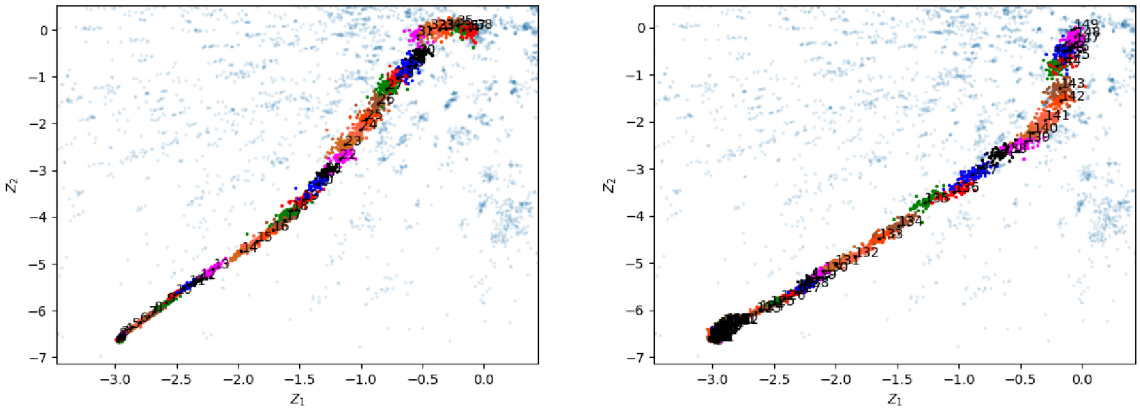
Výsledný model pro tuto datovou sadu byl složen z jedné skryté vrstvy mající 299 neuronů, *weight decay* faktor optimalizátoru byl nastaven na 0.0, byl zvolen 2 dimenzionální prostor a parametr C na 2.0. Model byl trénován po 16500 epoch.

8.2.2 Simulace řízené evoluce

Prvním experimentem, který byl vyzkoušen nad latentním prostorem byla simulace řízené evoluce (viz sekce 7.3). S vybraným modelem byly zaváděny mutace do *query* sekvence a sledovat vývoj v latentním prostoru, jak lze vidět na obrázku 8.5. Simulace byla provedena zaváděním 1 a 2 bodových mutací a v každém kole bylo vygenerováno 100 kandidátů. Dvoubodové mutace nejsou v obrázku 8.5 zachyceny. V levé části obrázku 8.5 byla aplikována pouze metrika blízkosti centra. Lze vidět, že v daném případě stačilo zavést 38 mutací.

Studie profilu generovaných sekvencí ovšem ukázalo zavádění velmi nepravděpodobných mutací. Proto byl proveden další experiment s funkcí biologicky stabilní sekvence (viz sekce 7.3) a jeho výsledek je vidět na obrázku 8.5 vpravo. Oproti funkci nejbližší sekvence byl počet vygenerovaných sekvencí větší. To bylo nejvíce způsobeno zaseknutím mutační evoluce v blízkém prostoru *query* sekvence, kdy navržené proteiny obsahovaly velmi netypické mutace, které měly nízkou přiřazenou pravděpodobnost od modelu a úspěšní mutanti se držely blízko původní sekvence. Evoluci se nakonec povedlo dostat z lokálního minima introdukcí mutace, která byla velice pravděpodobná a významná pro polohu v latentním prostoru. Studium těchto významných mutací bude předmětem dalšího pokračování projektu.

Nadále byl proveden statistický test na průměrný počet mutací nutných pro dosažení středové oblasti. Byl použit jedno-výběrový statistický t-test se statistickou významností $\alpha = 0.05$ vypočtený pomocí programovacího jazyka *R*. Bylo použito 500 běhů celkem ve třech scénářích. První scénář testoval střední hodnotu řízené evoluce pro funkci nejbližší sekvence s dvoubodovými mutacemi. Druhý a třetí scénář ověřoval střední hodnoty jedno bodových mutací pro funkci nejbližší respektive biologicky stabilní sekvence. Hodnota 95% konfidenčního intervalu byla v rozmezí 25.06 - 25.37, zatímco pro druhý respektive třetí scénář konfidenční intervaly nabývaly hodnot 35.50 - 35.89 respektive 92.77 - 96.17. Výsledky ukazují, že zavedením dvou bodových mutací došlo k snížení průměrného počtu vygenerovaných mutantů. To by mohlo naznačovat, že náš model zachycuje reálné evoluční události vyskytující se v přírodě, kdy mutace jedné pozice často vyžaduje komplementární mutaci residua na jiné i velmi vzdálené pozici, které jsou však prostorově blízké v ternární struktuře proteinu a navzájem interagují (vytváří sulfidové, potažmo vodíkové můstky). Naopak markantní rozdíl mezi středními hodnotami scénáře 2 a 3 naznačuje možné zavádění nepravděpodobných mutací pomocí funkce nejbližší sekvence, zatímco funkce biologicky stabilní sekvence tyto mutace potlačuje a po zavedení silné mutace se algoritmus dostane z lokálního minima a poté pokračuje ke středu prostoru (viz obrázek 8.5).



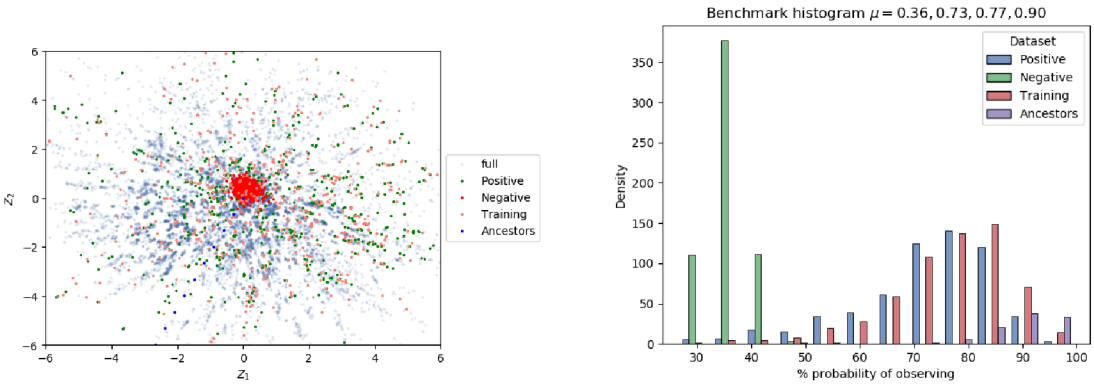
Obrázek 8.5: Simulace řízené evoluce v latentním prostoru pro jednobodové mutace. **Vlevo**: funkce nejbližší sekvence (viz sekce 7.3) má plynulou trajektorii do středu, **vpravo**: funkce biologicky stabilní sekvence prokázala problematiku zavádění zcela náhodných velice nestandardních mutací (lokální minimum kolem *query* sekvence).

8.2.3 Aplikace přímé rekonstrukční strategie

Po analýze náhodných mutací jsem přešel k dalšímu experimentu, který již využíval generativní schopnosti našeho modelu. Experiment si kladl za cíl navrhnout proteiny pro následné testy jejich aktivity a stability v laboratořích za využití přímé evoluční strategie.

Konstrukce statistických kontrol

Odhadnutí generativní kvality VAEs a stanovení úrovně statistických hodnot průměrné rekonstrukční shody bylo provedeno počítáním pravděpodobnostních distribucí na čtyřech podmnožinách trénovací sady tzv. pozitivní, negativní, trénovací a ancestrální (viz 7.2.1 a obrázek 8.6). Ancestrální sada byla přidána pro kontrolu distribuce v případě generovaných sekvencí. Každá podmnožina, kromě ancestrální, obsahovala množství sekvencí odpovídající 5 % trénovací sady. Jak jde vidět na obrázku 8.6 negativní kontrolní množina je dobře separována a positivní a trénovací kontrola sdílejí rozložení.



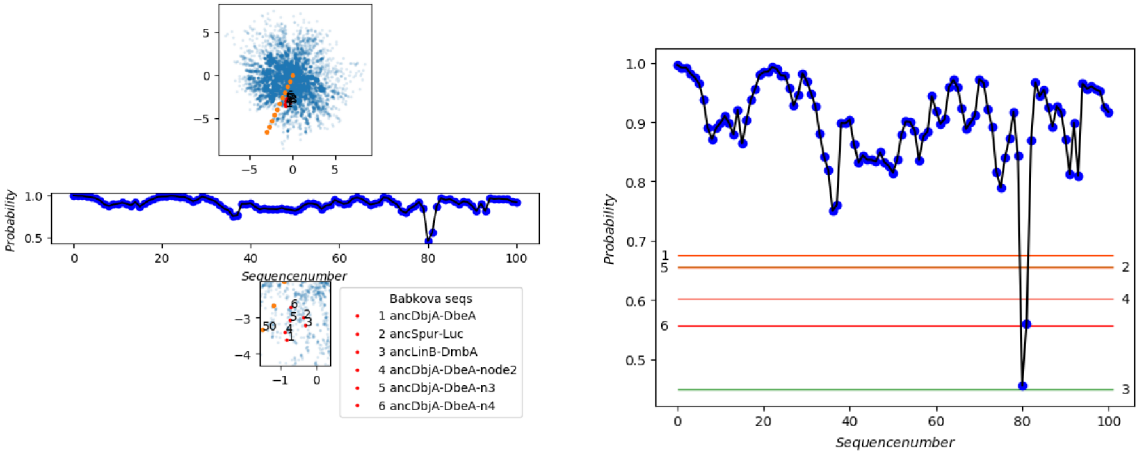
Obrázek 8.6: Latentní (**vlevo**) a distribuční (**vpravo**) rozložení statistických kontrol průměrné rekonstrukční shody.

Společně s jednotlivými podmnožinami byly analyzovány i laboratorně ověřené varianty ancestrální sekvencí vytvořené prostřednictvím klasický technik. Podle autorky jsem tuto skupinu proteinů označil jako sekvence Babkové [3]. Díky těmto sekvencí a analýze jejich statistických a latentních vlastností byl získán vhled do interpretace modelu.

Selekce kandidátů

Obrázek 8.7 ukazuje výsledky protokolu přímé rekonstrukce aplikované na datovou sadu *HLDs*. Jako první statistiku jsem sestavil profil pozorované sekvenční pravděpodobnosti. Profil v obrázku 8.7 ukazuje, jak si model byl jistý jednotlivými sekvencemi generovaných touto strategií. Profil odhaluje regiony, kde jsou sekvence pravděpodobnějšími členy vstupního zarovnání oproti jiným oblastem. Současně jsou v profilu zobrazeny i sekvence Babkové včetně jejich umístění v latentním prostoru (obrázek 8.7 vlevo), které je dobře strukturované do středu evoluční trajektorie. K získání přehledu o hodnotách profilu byly sekvence Babkové zarovnány do vstupní množiny a byl provedena stejná statistická analýza. Jak ukazuje obrázek 8.7 vpravo, pravděpodobnosti jsou v rozmezí 45 až 70 procent.

Abych lépe pochopil rozdělení pravděpodobností sekvencí a možný práh pro náhodné sekvence, vypočítal jsem rozdělení pravděpodobností pro 4 podskupiny dat (viz 7.2.1).



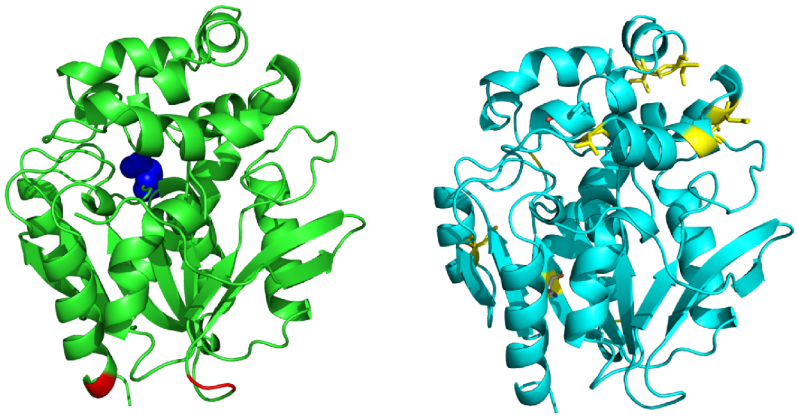
Obrázek 8.7: Statistický profil pozorovaných sekvencí strategie přímé evoluce. **Vlevo**: latentní prostor experimentu včetně pozic vygenerovaných sekvencí. Sekvence Babkové jsou zarovnány a zobrazeny v latentním prostoru, kde jsou umístěny blízko středu evoluční trajektorie. **Vpravo**: detail profilu sekvencí společně s pravděpodobnostními hodnotami pro sekvence Babkové.

Tréninkové a pozitivní podmnožiny měly podobná rozdělení, což svědčí o vysoké kvalitě natrénovaného VAEs, přičemž rozdělení se pohybovaly kolem hodnoty pravděpodobnosti 76 %. Tato rozdělení se navíc dobře odlišují od negativní trénovací množiny. Všechny negativní sekvence měly pravděpodobnost nižší než 50 %. Naproti tomu rekonstruovaní předci vykazovali posun pravděpodobností k vyšším hodnotám. Vzhledem k tomu, že pravděpodobnost sekvence koreluje s fitness proteinu, vysoké hodnoty u vygenerovaných přímých předků je potvrzují jako slibné cíle pro hlubší analýzu a charakterizaci v laboratoři.

Vzhledem k novosti metody jsem se však rozhodl doplnit analýzu předků o výpočet dalších vlastností generovaných sekvencí. Identita sekvence k původní může být důležitý ukazatel, protože v experimentu je cíleno na optimalizaci určitého proteinu a příliš velký rozdíl může odpovídat výrazně odlišnému proteinu. Další vlastností, kterou jsem se rozhodl prozkoumat, je identita sekvence vůči nejbližší sekvenci v zarovnání, abychom pochopili rozmanitost generovaných předků a také jejich blízkost ke známým proteinům. Kromě výše zmíněných sekvenčních identit jsme také vypočítali počet sekvenčních zbytků, které mají vyšší než 90% pravděpodobnost, že se nacházejí na výstupu dekodéru, což naznačuje jistotu VAEs u vybraných aminokyselin. Poslední sledovanou statistikou jsou počty inzercí/delecí a substitucí v dotazované sekvenci, které byly zavedeny do vygenerovaných předků. Výsledek vynesený do grafu lze vidět v příloze A.1.

Kromě statistického vyhodnocení došlo i k detailní studii vybraných variant z pohledu biologie (obrázek 8.8). Navrhované delece případně inzerce ve variantě 16 byly umístěny na biologicky relevantních místech převážně v koncových částech či mimo α-helix a β-list. Graf z přílohy A.1 byl předmětem debaty konečném výběru kandidátů do kola laboratorních testů. Po diskuzi s biologickými experty z týmu Loschmidtových laboratoří bylo pro první kolo laboratorních testů vybráno celkem 9 proteinů pokrývající celé spektrum statistických vlastností. Konkrétně se jednalo o varianty značené jako *AncDhaA* 1-9, které v příloze A.1 lze najít pod čísly 38, 44, 59, 68, 75, 108, 110, 113 a 125. Varianta *AncDhaA1* byla jediná nemající ani jednu inzerci či deleci avšak obsahovala 45 substitucí. Počet substitucí se u variant pohyboval od 45 do 138. Počet inzercí a delecí byl pro prvních pět variant

AncDhaA1-5 v rozmezí 0 až 21. Pro zbylé varianty *AncDhaA6-9* nacházející se blíže středu jejich počet dosahoval rozmezí 91 až 138. Pouze varianta *AncDhaA1* vykázala rozpustnost a funkčnost sníženou o 50 %.

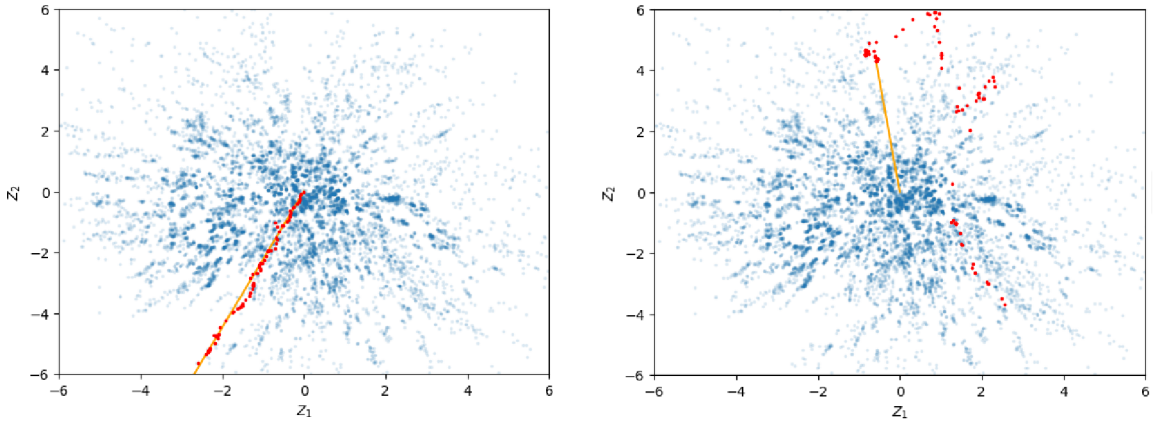


Obrázek 8.8: Příklady přímých předků vygenerovaných na základě latentního prostoru pro *DhaA*. **Vlevo**: Varianta *AncDhaA16* s identitou 69,23 % pouze zvýrazněné pozice inzerce/-delece, červeně - delece (dvě ve smyčce, pozice 29 a 30, a jedna ve šroubovici, pozice 291), modře - inzerce S (jedna ve smyčce, pozice 133). Vpravo: Varianta *AncDhaA7* s 93,89% identitou WT. Všech 10 navržených záměn je zvýrazněno.

8.2.4 Testování robustnosti metody

Při práci s datovou sadou *HLDs* jsem se zaměřil i na zkoumání robustnosti metody použití VAEs pro ancestrální sekvence pomocí vícenásobné náhodné inicializace. Účelem bylo zjistit citlivost trénovacího procesu a vliv na rozložení latentního prostoru při náhodném nastavení počátečních vah neuronové sítě.

Pro testování robustnosti byl nejdříve natrénován referenční model, který vygeneroval 100 ancestrálních sekvencí pomocí strategie přímé evoluce. Následně se stejnou datovou sadou bylo natrénováno dalších 16 modelů se shodnou architekturou a nastavením trénovacích i vnitřních parametrů jako referenční model. Na počátku trénovací fáze byly náhodně transformovány váhy pomocí normálního rozdělení kolem střední hodnoty 0 s variací 0.15. Takto inicializované modely byly natrénovány a v jejich latentním prostoru byly vyneseny pozice ancestrálních sekvencí navržených referenčním modelem. V ideálním případě by ve všech případech zakódované sekvence kopírovaly trajektorii do středu prostoru, což není náš případ, jak ukazuje obrázek 8.9. Náhodně inicializované modely VAEs nejsou schopny se naučit shodnou distribuci mutací skrze celý latentní prostor. Tento fakt klade určité limity univerzálnímu použití VAEs pro ancestrální rekonstrukci, kdy při stejné vstupní sadě můžeme dostat odlišné ancestrální sekvence. Stabilizace latentního prostoru je oblast, které bych chtěl v budoucnosti věnovat pozornost.



Obrázek 8.9: Rozložení ancestrální sekvencí vygenerovaných strategií přímé evoluce referenčním modelem pro náhodně inicializované modely. **Vlevo**: referenční model, **vpravo**: náhodně inicializovaný model mapující ancestrální sekvence referenčního modelu.

8.3 Experimenty se sadou *HLD I-II*

Výsledky předešlého experimentu 8.2 v profilu uvedeném v příloze A.1 ukázaly tendenci generovat vzdálené sekvence od původního proteinu *DhaA*. Navíc laboratorní testy ukázaly ztrátu funkce a nízkou rozpustnost některých z navržených proteinů, která je častým problémem při navrhování designů pomocí generativních modelů [23].

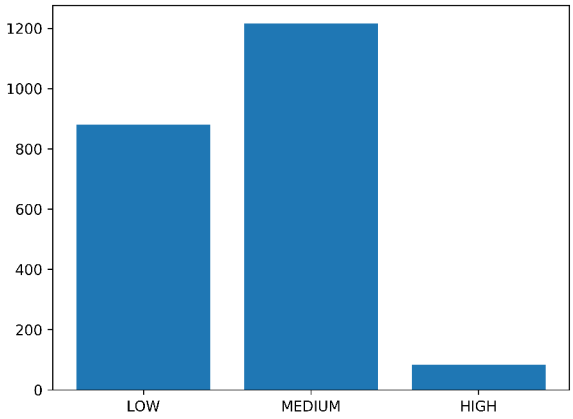
Pro adresaci těchto problémů jsem se rozhodl navrhnout ještě více specifické zarovnání sekvencí *HLD I-II* sugerující esenciální katalytická residua a použít podmíněný variační autoenkodér (CVAE) pro zlepšení rozpustnosti proteinů. Dle aktuálních publikací v oblasti generativních modelů byla rozšířena metodika evaluace kvality modelu o sledování statistik prvního a druhého řádu. Pro kontrolu zachování struktury latentního prostoru bylo provedeno testování na zachycení evolučních souvislostí.

Vstupní vícenásobné zarovnání sekvencí po předzpracování obsahovalo 2181 sekvencí se šířkou 291 residuí.

8.3.1 Optimalizace modelu

Před generováním kandidátních proteinových sekvencí byla provedena analýza výkonosti modelu na této menší datové sadě, které bere v potaz i predikovanou rozpustnost sekvencí diskretizovanou do 3 úrovní: *LOW*, *MEDIUM* a *HIGH*. U všech trénovací sekvencí bylo zajištěno zachování příslušných úrovní rozpustnosti pro všechny fáze evaluace, generování a trénování. V případě náhodných kontrol byly značky rozpustnosti vygenerovány náhodně z uniformního rozdělení. Jak ukazuje distribuce hodnot predikovaných rozpustností na obrázku 8.10, datová sada obsahuje pouze 84 sekvencí klasifikovaných jako vysoce rozpustných. Referenční sekvence *DhaA_S19* patří do skupiny sekvencí s vysokou rozpustností. Proto se předpokládá, že generování sekvencí s podmíněním vysoké rozpustnosti by mělo zapracovávat více residuí blízkých cílovému proteinu.

Pro selekci konečného modelu s vysokou biologickou podobností k cílovým vstupním sekvencím jsem vyhodnocoval hodnoty statistik prvního a druhého řádu (viz sekce 7.2.2). Testoval jsem modely mající 1-3 skryté vrstvy s počtem neuronů od 450 až 50 a dimensionalitou 2-7. Výsledky modelů ukázaly velmi dobrou schopnost VAEs zachytávat frekvenční



Obrázek 8.10: Rozložení predikovaných hodnot rozpustností pro všechny sekvence použité k trénování.

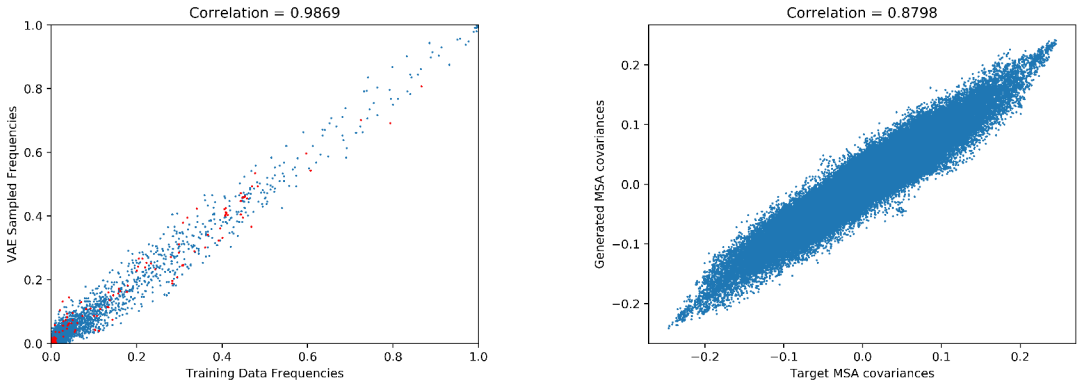
statistiky samostatných aminokyselinových pozic (obrázek 8.11 vlevo). Korelace u všech modelů dosahovala hodnoty 96 až 99 procent. Pro kontrolu distribuce mezer v generovaných i vstupních sekvencích jsou v obrázku 8.11 vlevo vyznačeny červené body značící frekvenci výskytu mezer v dané pozici u generované nebo vstupní sekvence. Pro tento konkrétní případ lze vidět existenci několika pozic bohatých na mezery. Důležitou informací pro nás je, že model nemá tendenci vkládat do generovaných sekvencí mezery jako nahrazující symboly v případě nejistoty.

Před provedením experimentů bylo očekáváno, že přidání dalších skrytých vrstev do modelu výrazně zlepší hodnoty statistik druhého řádu (obrázek 8.11 vpravo). Tento předpoklad se však nepotvrdil. Model s jednou vrstvou byl schopný zachytit vysokou korelaci téměř totožnou s hlubší variantou (87,98 % oproti 88,16 %) mající 450 a 100 neuronů ve skrytých vrstvách. V případě ostatních modelů byla hodnota korelace kovariance podobně vysoká nepřekračující hodnotu 88 % a nejnižší hodnota byla naměřena 86 % pro model s dvěma vrstvami každý mající 100 neuronů.

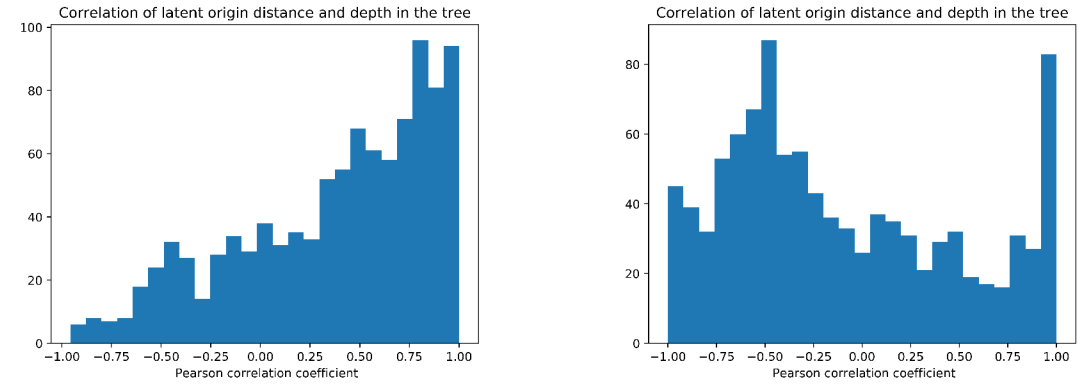
Kromě schopnosti generovat velmi blízké sekvence původnímu zarovnání byla důležitým ukazatel pro výběr modelu i jeho schopnost zachytávat evoluční závislosti v prostoru (viz sekce 7.2.3). Proto byly sledovány statistiky korelace vzdáleností zakódování sekvencí v latentním prostoru s hloubkou odpovídajícího uzlu ve fylogenetického stromu. Zejména důležitá byla vlastnost lokace ancestrálních sekvencí k centru prostoru jelikož i evoluční strategie ve své *fitness* funkci počítá s tímto předpokladem.

Statistiky prvního a druhé řádu vykazovaly podobné výsledky pro většinu testovaných modelů, proto hlavním činitel pro výběr modelu byla zvolena právě analýza struktury latentního prostoru. Obrázek 8.12 ukazuje výsledky korelací hloubky uzlu fylogenetického stromu se vzdáleností od středu prostoru. V levé části obrázku 8.12 lze pozorovat relativně dobré výsledky pro model s jednou vrstvou. Zajímavým zjištěním byla neschopnost hlubších modelů mapovat evoluční vztahy s ohledem na vzdálenost od středu prostoru. Vícevrstvé modely vykazují pro většinu mapovaných větví do latentního prostoru žádnou nebo mírně opačnou korelaci.

S předchozím zjištěním byla zavedena další metrika pro detailnější analýzu dějů v latentním prostoru. Bylo předpokládáno, že evoluční vztahy jsou v hlubších modelech stále

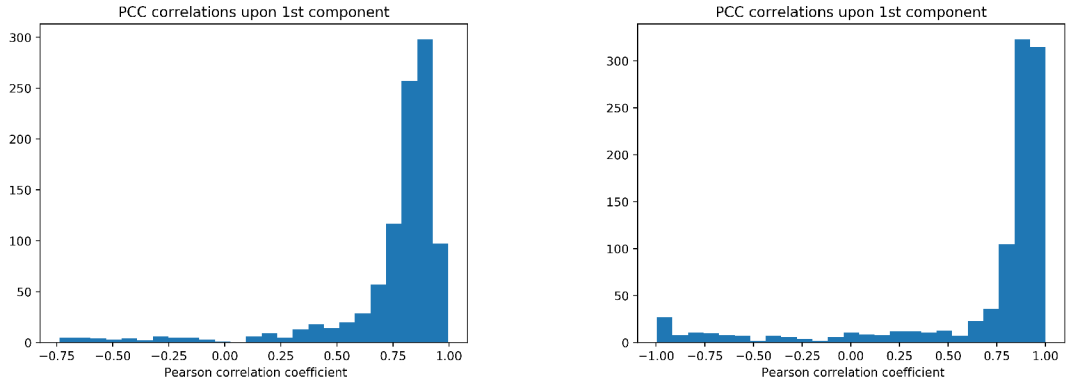


Obrázek 8.11: Statistiky prvního a druhého řádu pro konečný model s jednou skrytou vrstvou s 450 neurony vybraného pro experiment. **Vlevo**: statistika prvního řádu s vysokou korelací značí schopnost VAEs reflektovat frekvence aminokyselin na daných pozicích. **Vpravo**: korelace kovariance statistik druhého řádu.



Obrázek 8.12: Histogram hodnot korelací vzdálenosti zakódované pozice sekvence od středu latentním prostoru s hloubkou odpovídajícího uzlu ve fylogenetickém stromu. **Vlevo**: model s jednou vrstvou s 450 neurony vybraného pro experiment, **vpravo** hlubší model s dvěma vrstvami mající 450 a 100 neuronů.

zachyceny však jiným způsobem než umístění blíže ke středu, protože silnější enkodér je schopný zavádět složitější souvislosti do latentního prostoru. Proto byla měřena korelace pozice kolem osy první komponenty PCA s pozicí ve fylogenetickém stromu (viz sekce 7.2.3). Hypotézu potvrdily výsledky na obrázku 8.13, které ukazují silnou korelaci pro hluboký a jednovrstvý model s první komponentou PCA.



Obrázek 8.13: Histogram hodnot korelací pozic v latentním prostoru kolem první komponenty PCA s hloubkou v fylogenetickém stromu. **Vlevo**: model s jednou vrstvou s 450 neurony vybraného pro experiment, **vpravo** hlubší model s dvěma vrstvami mající 450 a 100 neuronů.

V této práci jsem pracoval s předpokladem mapování ancestrální sekvencí blíže ke středu prostoru, proto relevantními modely pro mě byly ty s jednou vrstvou. Navíc jednovrstvé modely vykázaly podobnou schopnost zachytávat jednoduché i párové frekvenční statistiky s hlubšími modely pro danou datovou sadu. Dalším zajímavým zjištěním podporujícím aplikaci jednovrstvého modelu oproti hlubšímu byla schopnost rekonstruovat *query* sekvenci s identitou nad 98 % pouze v případech jednodušších modelů (obrázek 8.14 vlevo). Hlubší modely dosahovaly rekonstrukce kolem 88 % (kolem 33 mutací), čímž byly nevhodné pro naše účely generace blízkých sekvencí.

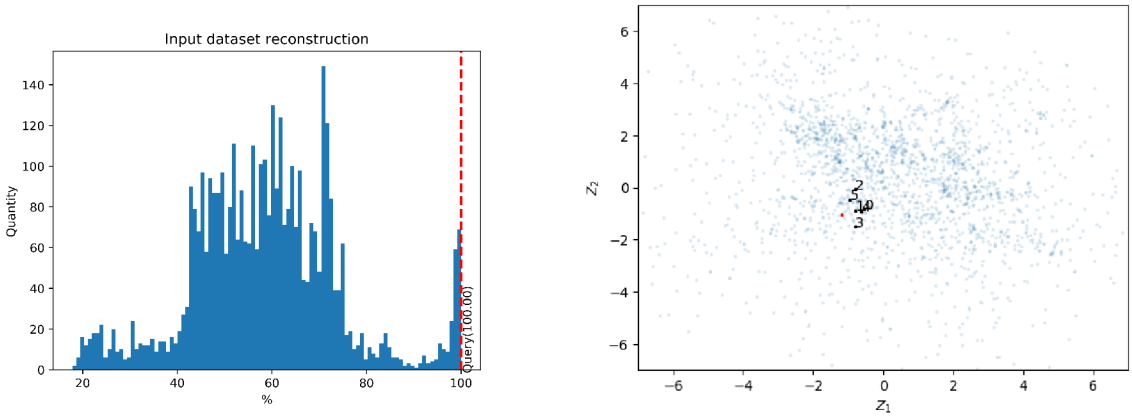
Proto **zvoleným modelem** pro experimenty se stal jednovrstvý model s 450 neurony a dvou dimenzionálním latentním prostorem učeným po 13500 epoch. *Weight decay* faktor byl nastaven na 0.0 a parametr C na 2.0.

Výsledný latentní prostor pro datovou sadu vykazuje méně shlukovanou strukturu. Umístění *query* sekvence je ve vnitřní části. Experimentálně ověřené ancestrální sekvence Babkové [3] jsou téměř všechny umístěné blíže ke středu naznačující.

Aplikace evoluční strategie adaptace kovarianční matice

V předchozím experimentu byla použita přímá evoluční strategie. Vygenerovaný profil a testy robustnosti ukázaly, že použití statické evoluční strategie není vhodné pro sofistikované prohledávání za účelem maximalizace vybraných sledovaných hodnot.

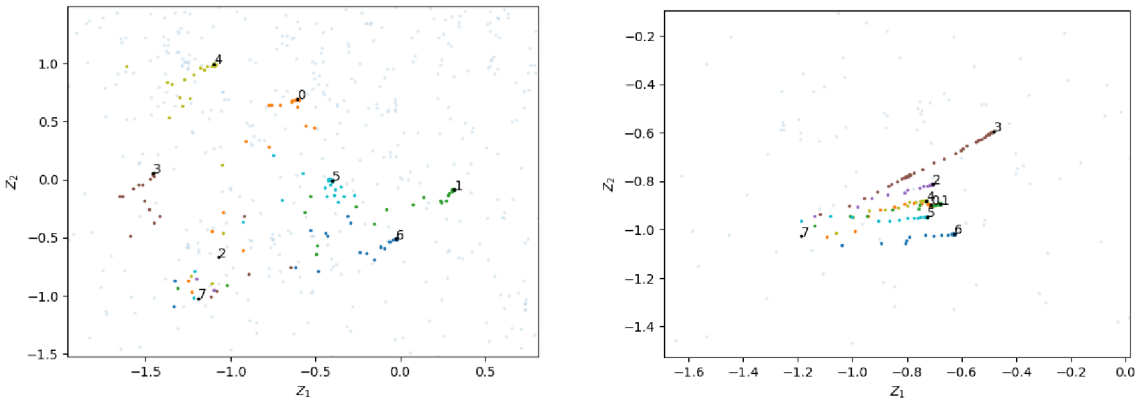
Proto byla sestavena nová metoda využívající evoluční strategii adaptace kovarianční matice (viz sekce 7.3.3). Jejím cílem bylo nalezení sekvence mající 92,5% podobnost, která je lokalizována blíže k centru prostoru a maximalizuje svoji marginální pravděpodobnost. V případě prohledávání oblasti s dostatečnou důvěrou v interpolaci známých hodnot stabilit mutantů pomocí Gaussových procesů je využita i tato hodnota.



Obrázek 8.14: Rekonstrukce sekvencí a latentní prostor generovaný vybraným modelem pro experimenty s jednou vrstvou s 450 neurony. Zobrazené sekvence Babkové společně s *query* sekvencí znázorněnou červeně.

Úroveň cílové podobnosti byla zvolena na základě známé podobnosti funkčních sekvencí Babkové navržených pomocí metody ancestrální rekonstrukce. V dané studii [3] nejbližší navržená sekvence k enzymům *DbjA* a *DbeA* měla sekvenční podobnost kolem 88 %. Navržená sekvence sdílející vývoj s *DhaA* s ní měla sekvenční podobnost pouze 70 %. Mz jsme se rozhodli hledat pro bližší varianty proteinu s 92,5% podobností s *DhaA*.

Evoluční prohledávací algoritmus běžel v 10 iteracích a každá populace měla 128 členů. Výběr kandidátů probíhal za pomoci *pareto* fronty a váhované *fitness* funkce. Výsledky vyhledávání lze vidět na obrázku 8.15. V případě váhované funkce byl vliv jednotlivých složek *fitness* funkce nastaven dle experimentů s algoritmem a jeho chováním v latentním prostoru. Výsledné nastavení prohledává širší oblast v případě váhované *fitness* funkce. Pro *pareto* frontu výsledkem byly výsledkem přímější trajektorie. Současně se algoritmu častěji podařilo najít optimální řešení pro podobnost generovaných sekvenci a ukončit tak prohledávání předčasně. Takto navržené sekvence mají vysokou podobnost s *query* sekvencí a umístěním v latentním prostoru by měly sdílet některé rysy ancestrálních sekvencí.



Obrázek 8.15: Výsledky prohledávání evoluční strategie pro váhovanou *fitness* funkci **vlevo**, a *pareto* frontu **vpravo**. Bod označený číslem 7 značí pozici *query* sekvence. Čísla 0-6 jsou výsledky prvních 7 běhů evoluce.

V současné podobě navržená *fitness* funkce uvažuje pouze sekvenční a prostorové ohodnocení. V budoucím vývoji bude upřena pozornost na vylepšení relevantnosti *fitness* funkce pro proteinové sekvence. Do *fitness* funkce bude přidána predikce sekundárních struktur a penalizováno jejich porušení. Tím bude maximalizována pravděpodobnost správné funkčnosti vygenerovaných proteinů. Nadále bude uvažováno přidání nástroje *foldX*, který dokáže predikovat škodlivost či užitečnost použitých mutací při výpočtu Gibbsovy volné energie.

8.4 Porovnání výsledků s literaturou

Využití generativních modelů pro návrh proteinů je nová oblast aktivního výzkumu. Mnoho studií je věnováno pochopení vlastností prostorů generovaných těmito modely [14][13] bez využití jejich generativních vlastností. Generativní modely jsou také využívány jako indikátory prospěšnosti mutací dle zkoumání pravděpodobnostních hodnot poskytovaných modelem. Poměrem pravděpodobnostní hodnoty modifikované a původní varianty proteinu byl vytvořen tzv. evoluční index. Ten byl použit pro následné trénování klasifikátoru, který dosahoval *state-of-the-art* výsledků pro klasifikaci patogenních mutací [20].

V literatuře byly popsány studie zabývající se generací nových enzymů. Studie [54] vykázala schopnost sítí typu GANs vytvářet funkční enzymy se shodnou úrovní aktivity. Po trénovací fázi, při které kontrolovali různorodost generovaných proteinů, vybrali 60 vzorků k následné laboratorní analýze. Testy ukázaly, že 24 % kandidátů bylo rozpustných a 13 variant vykázalo katalytickou aktivitu. Sekvence sdílely identitu s původním proteinem malátdehydrogenásy v rozmezí 48 až 98 %. Nejvzdálenější varianta, která vykázala katalytickou aktivitu, měla 106 aminokyselinových substitucí.

Další studie [21] generovala vzorky ze zakódování původní sekvence postupném rozšiřování vzorkovací oblasti. Takto vygenerovali celkem 87 variant, které následně ověřili v *E.coli* a nejlépe projevujících se 12 variant vyzkoušeli v savčích buňkách. Výsledné měření ukázaly zlepšenu stabilitu o 12°C.

V [23] byly použity podmíněné variační autoenkodéry, které byly schopné vygenerovat varianty proteinů s modifikovanou funkcí. Takto dokázali kontrolovat rozpustnost studovaného protein (*luxA*), kterou výrazně zvýšili oproti její přírodní variantě.

V naší studii jsme se zaměřili na využití vlastností generovaného latentního prostoru pro generování nových variant za účelem vylepšení jejich stability. Tím se odlišujeme od předchozích studií aplikující převážně náhodné vzorkování prostoru. Navíc jsme nezůstali pouze u teoretické studie, ale vybrané kandidáty s experimentu s přímou evolucí jsme podrobili laboratorním testům. Naše prostředky nám dovolily ověřit menší množinu generovaných sekvencí než v případě předchozích studií. Celkem jsem ověřili vlastnosti 9 variant pokrývající různé oblasti statických metrik. 1 z 9 variant prokázala schopnost rozpustnosti a současně i shodné katalytické aktivity *in vitro*. Tato varianta obsahovala 45 substitucí. Hodnoty stability nebyly v době psání této diplomové práce k dispozici, proto nemůžeme vyvodit závěr vliv vzorkovací strategie na optimalizaci sledované vlastnosti proteinu.

Kapitola 9

Závěr

Oblast proteinové inženýrství zažívá v posledních letech nárůst jak vědecké, tak i průmyslové pozornosti, jelikož zde proteiny představují možnou náhradu složitých chemických postupů. Formování proteinových struktur je dlouhodobě evolučně optimalizovaný přírodní proces, který se lidé snaží pochopit od minulého století. Jedním z přístupu pro studium evolučních souvislostí je metoda ancestrální rekonstrukce sekvencí, která byla stěžejním tématem této diplomové práce. Ancestrální rekonstrukce proteinových sekvencí je významnou a úspěšnou metodou používanou v oblasti proteinového inženýrství pro vylepšení stability a aktivity proteinů. Cílem této diplomové práce bylo seznámení se s její problematikou a prozkoumaní alternativního přístupu, využívající soudobou obrovskou expanzi datových a výpočetních zdrojů společně s pokroky v oblasti strojového učení bez učitele, k odvození ancestrálních sekvencí pomocí latentních modelů.

V práci nejprve bylo provedeno seznámení se základními pojmy domény molekulární biologie, která se zabývá stavbou a syntézou proteinů. Dále byla vysvětlena tvorba vícenásobného zarovnání proteinových sekvencí společně s jejich významem pro tvorbu fylogenetických stromů, které jsou základním stavebním kamenem pro ancestrální rekonstrukci sekvencí. Byla také uvedena teorie generativních modelů s detailním popisem variačních autoenkodérů.

V článku *Deciphering protein evolution and fitness landscapes with latent space models* byla představena schopnost latentního prostoru generovaného pomocí variačních autoenkodérů zachycovat evoluční vztahy v proteinové *pfam* rodině domény *fibronektinu* typu III. Tato skutečnost byla inspirací mého výzkumu v oblasti využití modelu pro generaci ancestrálních sekvencí. Původní protokol využíval převážně latentní reprezentaci proteinů a oblasti využití generativního procesu nebyla věnována dostatečná pozornost. V mé práci jsem si proto kladl za cíl optimalizaci modelu a vstupního zarovnání společně s definicí strategií vhodných pro tvorbu ancestrálních sekvencí.

V prvním fázi projektu došlo k úspěšnému replikování výsledků popisovaných v referenční publikaci na *pfam* proteinové rodině $\alpha\beta$hydroláz. Byla provede studie různých protokolů předzpracování vstupního zarovnání s ohledem na vlastnosti latentního prostoru. Latentní prostor projevil nízkou citlivost v případě odebrání sloupců ze vstupního zarovnání, což bylo cenné pozorování, protože v případě optimalizace jednoho proteinu nás zajímají pouze jeho pozice.

Druhý experiment se již věnoval detailní studii optimalizace zarovnání a modelu společně s návrhem strategií za účelem rekonstrukce ancestrálních sekvencí proteinu *DhaA*. Pro protein *DhaA* bylo vytvořeno optimalizované zarovnání pomocí nástroje EnzymeMiner. Po řadě experimentů, byl vybrán model maximalizující podobnost rekonstruovaných

sekvencí včetně *DhaA* a generativní schopnost modelu. V prvním kole výběru kandidátů byla aplikována strategie simulace řízené evoluce. Za účelem využití skutečného potenciálu generativního procesu byla po diskuzi navrhnuta strategie přímé evoluce. K experimentům byl generován velmi obsáhlý soubor statistických informací, který v případě druhé strategie vykazoval zajímavé trendy. Po diskuzi a studii navržených mutací s biologickými odborníky bylo vybráno k laboratorním testům celkem 12 kandidátů. Výsledky ukázaly problémy s rozpustností většiny navržených proteinů. Avšak jeden z blízkých mutantů byl úspěšně podroben laboratorním testům a vykázal shodnou funkčnost.

Po analýze laboratorních výsledků a studiu velmi aktivního souběžného výzkumu v naší oblasti jsme se rozhodli použít sofistikovanější metriky měření kvality modelu pomocí statistik prvního a druhého řádu. Dále došlo k modifikaci modelu na podmíněný variační autoenkodér využívající predikovanou rozpustnost od nástroje SolutProt. Také bylo navrženo ještě užší vstupní zarovnání vyřazující podrodiny HLD-III a HLD-IV z trénovací sady, protože jejich členové vykazovali nízké hodnoty rozpustnosti. V souvislosti s přímou evoluční strategií byly provedeny experimenty robustnosti trénovacího procesu variačních autoenkodérů. Ty ukázaly náchylnost trénovacího modelu na náhodnou inicializaci počátečních vah. Z tohoto důvodu a důvodu sledování trendů vlastností generovaných sekvencí z latentního prostoru a jejich maximalizace byla implementována evoluční strategie založena na metodě adaptace kovarianční matice. Výsledky posledního experimentu ještě nebyly laboratorně ověřeny.

Současná vysoká frekvence publikovaných článků na tématiku generativních modelů v proteinovém inženýrství nasvědčuje o aktuálnosti naší metody. Originálnost naší práce spočívá v hlubokém studiu vlastností zachycených latentním prostorem se současným návrhem sofistikovaných generativních strategií. Proto budoucí práce na projektu bude směřovat optimalizací těchto strategií a zvyšování robustnosti celého procesu inference ancestrálních sekvencí.

V rámci práce bylo provedeno studium současné literatury k problematice latentních modelů a jejich aplikací v proteinové doméně. Současně byly navrženy nová, optimálnější zarovnání a strategie pro generování relevantních proteinů s velkým počtem experimentů. Vybraní kandidáti byli podrobeni laboratorním testů, kdy na základě jejich výsledků a studia literatury byl proveden další krok optimalizace všech částí našeho programu.

Literatura

[1] Alberts, B. *Základy buněčné biologie : Úvod do molekulární biologie buňky.* Ústí nad Labem: Espero Publishing, 1998. ISBN 80-902906-0-4.

[2] Altschul, S. F., Gish, W., Miller, W., Myers, E. W. a Lipman, D. J. Basic local alignment search tool. *Journal of Molecular Biology.* 1990, sv. 215, č. 3, s. 403–410. DOI: https://doi.org/10.1016/S0022-2836(05)80360-2. ISSN 0022-2836. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0022283605803602.

[3] Babkova, P., Sebestova, E., Brezovsky, J., Chaloupkova, R. a Damborsky, J. Ancestral haloalkane dehalogenases show robustness and unique substrate specificity. *ChemBioChem.* 2017, sv. 18, č. 14, s. 1448–1456.

[4] Bateman, A., Coin, L., Durbin, R., Finn, R. D., Hollich, V. et al. The Pfam protein families database. *Nucleic acids research.* Oxford University Press. 2004, sv. 32, suppl_1, s. D138–D141.

[5] Betz, S. F. Disulfide bonds and the stability of globular proteins. *Protein Science.* Wiley Online Library. 1993, sv. 2, č. 10, s. 1551–1558.

[6] Blackshields, G., Sievers, F., Shi, W., Wilm, A. a Higgins, D. G. Sequence embedding for fast construction of guide trees for multiple sequence alignment. *Algorithms for Molecular Biology.* Springer. 2010, sv. 5, č. 1, s. 21.

[7] Blei, D. M., Kucukelbir, A. a McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association.* Taylor & Francis. 2017, sv. 112, č. 518, s. 859–877.

[8] Bontrager, P., Roy, A., Togelius, J., Memon, N. a Ross, A. Deepmasterprints: Generating masterprints for dictionary attacks via latent variable evolution. In: IEEE. *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS).* 2018, s. 1–9.

[9] Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R. et al. Generating sentences from a continuous space. *ArXiv preprint arXiv:1511.06349.* 2015.

[10] Chen, C., Natale, D. A., Finn, R. D., Huang, H., Zhang, J. et al. Representative proteomes: a stable, scalable and unbiased proteome set for sequence analysis and functional annotation. *PloS one.* Public Library of Science San Francisco, USA. 2011, sv. 6, č. 4, s. e18910.

[11] Ciampi, M. S. Rho-dependent terminators and transcription termination. *Microbiology*. Microbiology Society. 2006, sv. 152, č. 9, s. 2515–2528.

[12] Consortium, U. UniProt: a hub for protein information. *Nucleic acids research*. Oxford University Press. 2015, sv. 43, D1, s. D204–D212.

[13] Detlefsen, N. S., Hauberg, S. a Boomsma, W. Learning meaningful representations of protein sequences. *Nature Communications*. Nature Publishing Group. 2022, sv. 13, č. 1, s. 1–12.

[14] Ding, X., Zou, Z. a Brooks Iii, C. L. Deciphering protein evolution and fitness landscapes with latent space models. *Nature communications*. Nature Publishing Group. 2019, sv. 10, č. 1, s. 1–13.

[15] Do, C. B. a Batzoglou, S. What is the expectation maximization algorithm? *Nature biotechnology*. Nature Publishing Group. 2008, sv. 26, č. 8, s. 897–899.

[16] Edgar, R. C. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*. Březen 2004, sv. 32, č. 5, s. 1792–1797. DOI: 10.1093/nar/gkh340. ISSN 0305-1048. Dostupné z: https://doi.org/10.1093/nar/gkh340.

[17] Felsenstein, J. Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of molecular evolution*. Springer. 1981, sv. 17, č. 6, s. 368–376.

[18] Fiser, A. Template-based protein structure modeling. In: *Computational biology*. Springer, 2010, s. 73–94.

[19] Fitch, W. M. a Margoliash, E. Construction of phylogenetic trees: a method based on mutation distances as estimated from cytochrome c sequences is of general applicability. *Science*. American Association for the Advancement of Science. 1967, sv. 155, č. 3760, s. 279–284.

[20] Frazer, J., Notin, P., Dias, M., Gomez, A., Min, J. K. et al. Disease variant prediction with deep generative models of evolutionary data. *Nature*. Nature Publishing Group. 2021, sv. 599, č. 7883, s. 91–95.

[21] Giessel, A., Dousis, A., Ravichandran, K., Smith, K., Sur, S. et al. Therapeutic enzyme engineering using a generative neural network. *Scientific Reports*. Nature Publishing Group. 2022, sv. 12, č. 1, s. 1–17.

[22] Hansen, N. a Ostermeier, A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*. MIT Press. 2001, sv. 9, č. 2, s. 159–195.

[23] Hawkins Hooker, A., Depardieu, F., Baur, S., Couairon, G., Chen, A. et al. Generating functional protein variants with variational autoencoders. *PLoS computational biology*. Public Library of Science San Francisco, CA USA. 2021, sv. 17, č. 2, s. e1008736.

[24] Heim, M., Römer, L. a Scheibel, T. Hierarchical structures made of proteins. The complex architecture of spider webs and their constituent silk proteins. *Chemical Society Reviews*. Royal Society of Chemistry. 2010, sv. 39, č. 1, s. 156–164.

[25] Henikoff, S. a Henikoff, J. G. Position-based sequence weights. *Journal of molecular biology*. Elsevier. 1994, sv. 243, č. 4, s. 574–578.

[26] Higgins, D. G. a Sharp, P. M. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*. Elsevier. 1988, sv. 73, č. 1, s. 237–244.

[27] Hinton, G. E. Learning translation invariant recognition in a massively parallel networks. In: Springer. *International Conference on Parallel Architectures and Languages Europe*. 1987, s. 1–13.

[28] Hochberg, G. K. a Thornton, J. W. Reconstructing ancient proteins to understand the causes of structure and function. *Annual Review of Biophysics*. Annual Reviews. 2017, sv. 46, s. 247–269.

[29] Hon, J., Borko, S., Stourac, J., Prokop, Z., Zendulka, J. et al. EnzymeMiner: automated mining of soluble enzymes with diverse structures, catalytic properties and stabilities. *Nucleic acids research*. Oxford University Press. 2020, sv. 48, W1, s. W104–W109.

[30] Huelsenbeck, J. P., Ronquist, F., Nielsen, R. a Bollback, J. P. Bayesian inference of phylogeny and its impact on evolutionary biology. *Science*. American Association for the Advancement of Science. 2001, sv. 294, č. 5550, s. 2310–2314.

[31] Janssen, D. B. Evolving haloalkane dehalogenases. *Current Opinion in Chemical Biology*. 2004, sv. 8, č. 2, s. 150 – 159. DOI: https://doi.org/10.1016/j.cbpa.2004.02.012. ISSN 1367-5931. Dostupné z: http://www.sciencedirect.com/science/article/pii/S1367593104000274.

[32] Jermann, T. M., Opitz, J. G., Stackhouse, J. a Benner, S. A. Reconstructing the evolutionary history of the artiodactyl ribonuclease superfamily. *Nature*. Nature Publishing Group. 1995, sv. 374, č. 6517, s. 57–59.

[33] Johnson, S. R., Monaco, S., Massie, K. a Syed, Z. Generating novel protein sequences using Gibbs sampling of masked language models. *BioRxiv*. Cold Spring Harbor Laboratory. 2021.

[34] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S. a Saul, L. K. An introduction to variational methods for graphical models. *Machine learning*. Springer. 1999, sv. 37, č. 2, s. 183–233.

[35] Kapoor, S., Rafiq, A. a Sharma, S. Protein engineering and its applications in food industry. *Critical reviews in food science and nutrition*. Taylor & Francis. 2017, sv. 57, č. 11, s. 2321–2329.

[36] Kejnovský, E. Genetický kód, Náhodné zamrznutí, nebo postupný vývoj? *Vesmír*. Dostupné z: https://vesmir.cz/cz/casopis/archiv-casopisu/2007/cislo-9/geneticky-kod.html#&gid=1&pid=1.

[37] Kingma, D. P. a Welling, M. Auto-encoding variational bayes. *ArXiv preprint arXiv:1312.6114*. 2013.

[38] Kingma, D. P. a Welling, M. An introduction to variational autoencoders. *ArXiv preprint arXiv:1906.02691*. 2019.

[39] Kokkonen, P., Koudeláková, T., Chaloupkova, R., Daniel, L., Prokop, Z. et al. Structure-function relationships and engineering of haloalkane dehalogenases. *Aerobic utilization of hydrocarbons, oils and lipids. Cham: Springer International Publishing*. 2017, s. 1–21.

[40] Koudelakova, T., Bidmanova, S., Dvorak, P., Pavelka, A., Chaloupkova, R. et al. Haloalkane dehalogenases: biotechnological applications. *Biotechnology Journal*. Wiley Online Library. 2013, sv. 8, č. 1, s. 32–45.

[41] Le, S. Q. a Gascuel, O. An improved general amino acid replacement matrix. *Molecular biology and evolution*. Oxford University Press. 2008, sv. 25, č. 7, s. 1307–1320.

[42] Maddison, W. P., Donoghue, M. J. a Maddison, D. R. Outgroup analysis and parsimony. *Systematic biology*. Society of Systematic Zoology. 1984, sv. 33, č. 1, s. 83–103.

[43] Martínek, T. *Vícenásobné zarovnání*. FIT VUT Brno. Cit. 3-1-2021.

[44] Martínek, T. *Zarovnání sekvencí*. FIT VUT Brno. Cit. 3-1-2021.

[45] McGee, F., Hauri, S., Novinger, Q., Vucetic, S., Levy, R. M. et al. The generative capacity of probabilistic protein sequence models. *Nature communications*. Nature Publishing Group. 2021, sv. 12, č. 1, s. 1–14.

[46] Merkl, R. a Sterner, R. Ancestral protein reconstruction: techniques and applications. *Biological chemistry*. De Gruyter. 2016, sv. 397, č. 1, s. 1–21.

[47] Musil, M., Stourac, J., Bendl, J., Brezovsky, J., Prokop, Z. et al. FireProt: web server for automated design of thermostable proteins. *Nucleic Acids Research*. Duben 2017, sv. 45, W1, s. W393–W399. DOI: 10.1093/nar/gkx285. ISSN 0305-1048. Dostupné z: https://doi.org/10.1093/nar/gkx285.

[48] Needleman, S. B. a Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*. 1970, sv. 48, č. 3, s. 443 – 453. DOI: https://doi.org/10.1016/0022-2836(70)90057-4. ISSN 0022-2836. Dostupné z: http://www.sciencedirect.com/science/article/pii/0022283670900574.

[49] Newman, J., Peat, T. S., Richard, R., Kan, L., Swanson, P. E. et al. Haloalkane dehalogenases: structure of a Rhodococcus enzyme. *Biochemistry*. ACS Publications. 1999, sv. 38, č. 49, s. 16105–16114.

[50] Nguyen, V., Wilson, C., Hoemberger, M., Stiller, J. B., Agafonov, R. V. et al. Evolutionary drivers of thermoadaptation in enzyme catalysis. *Science*. American Association for the Advancement of Science. 2017, sv. 355, č. 6322, s. 289–294.

[51] Pauling, L., Zuckerkandl, E., Henriksen, T. a Lövstad, R. Chemical paleogenetics. *Acta chem. scand.* 1963, sv. 17, s. S9–S16.

[52] Price, M. N., Dehal, P. S. a Arkin, A. P. FastTree 2–approximately maximum-likelihood trees for large alignments. *PloS one*. Public Library of Science San Francisco, USA. 2010, sv. 5, č. 3, s. e9490.

[53] Ravanbakhsh, S., Lanusse, F., Mandelbaum, R., Schneider, J. a Poczos, B. Enabling dark energy science with deep generative models of galaxy images. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2017, sv. 31, č. 1.

[54] Repecka, D., Jauniskis, V., Karpus, L., Rembeza, E., Rokaitis, I. et al. Expanding functional protein sequence spaces using generative adversarial networks. *Nature Machine Intelligence*. Nature Publishing Group. 2021, sv. 3, č. 4, s. 324–333.

[55] Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z. et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*. National Acad Sciences. 2021, sv. 118, č. 15.

[56] Saitou, N. a Nei, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*. Červenec 1987, sv. 4, č. 4, s. 406–425. DOI: 10.1093/oxfordjournals.molbev.a040454. ISSN 0737-4038. Dostupné z: https://doi.org/10.1093/oxfordjournals.molbev.a040454.

[57] Sarma, R. *Protein engineering: applications in science, medicine, and industry*. Elsevier, 2012. ISBN 978-0124312319.

[58] Sievers, F. a Higgins, D. G. Clustal omega. *Current protocols in bioinformatics*. Wiley Online Library. 2014, sv. 48, č. 1, s. 3–13.

[59] Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K. et al. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular systems biology*. John Wiley & Sons, Ltd Chichester, UK. 2011, sv. 7, č. 1, s. 539.

[60] Söding, J. Protein homology detection by HMM–HMM comparison. *Bioinformatics*. Oxford University Press. 2005, sv. 21, č. 7, s. 951–960.

[61] Sohn, K., Lee, H. a Yan, X. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*. 2015, sv. 28.

[62] Stourac, J., Dubrava, J., Musil, M., Horackova, J., Damborsky, J. et al. FireProtDB: database of manually curated protein stability data. *Nucleic acids research*. Oxford University Press. 2021, sv. 49, D1, s. D319–D324.

[63] Teng, S., Srivastava, A. K. a Wang, L. Sequence feature-based prediction of protein stability changes upon amino acid substitutions. *BMC genomics*. Springer. 2010, sv. 11, S2, s. S5.

[64] Thompson, J. D., Higgins, D. G. a Gibson, T. J. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research*. Oxford University Press. 1994, sv. 22, č. 22, s. 4673–4680.

[65] Tria, F. D. K., Landan, G. a Dagan, T. Phylogenetic rooting using minimal ancestor deviation. *Nature Ecology & Evolution*. Nature Publishing Group. 2017, sv. 1, č. 1, s. 1–7.

[66] Vasina, M., Vanacek, P., Hon, J., Kovar, D., Faldynova, H. et al. Functional and Mechanistic Characterization of an Enzyme Family Combining Bioinformatics and High-Throughput Microfluidics. 2021.

[67] Vialle, R. A., Tamuri, A. U. a Goldman, N. Alignment modulates ancestral sequence reconstruction accuracy. *Molecular biology and evolution*. Oxford University Press. 2018, sv. 35, č. 7, s. 1783–1797.

[68] Weng, L. From gan to wgan. *ArXiv preprint arXiv:1904.08994*. 2019.

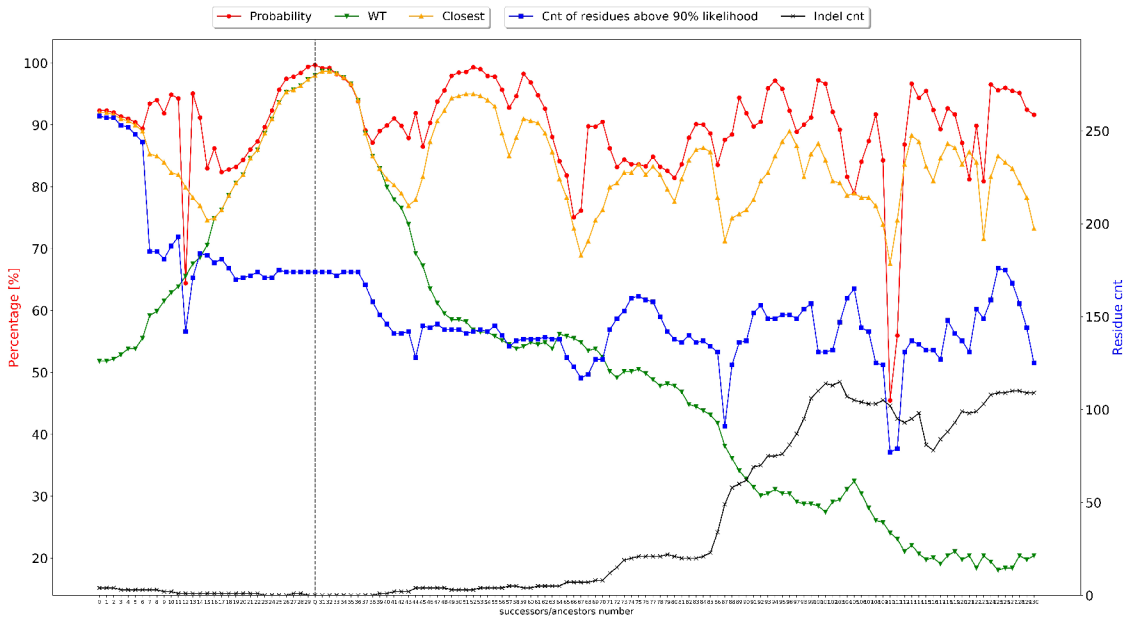
[69] Whitford, D. *Proteins: structure and function*. John Wiley & Sons, 2013. ISBN 978-1-118-68572-3.

[70] Wikipedia, the free encyclopedia. *Splicing*. 2006. [Online; navštíveno 30. prosince 2020]. Dostupné z: https://upload.wikimedia.org/wikipedia/commons/1/17/Pre-mRNA_to_mRNA.png.

[71] Yang, Z. et al. PAML: a program package for phylogenetic analysis by maximum likelihood. *Computer applications in the biosciences*. Citeseer. 1997, sv. 13, č. 5, s. 555–556.

Příloha A

Graf výstupu experimentu



Obrázek A.1: Grafické znázornění statistické analýzy navržených proteinových sekvencí. Je porovnávána průměrná rekonstrukční chyba (červeně), sekvenční podobnost s *query* sekvencí (zeleně), sekvenční podobnost k nejbližší sekvenci z trénovací sady v latentním prostoru (žlutě), počet residuí majících alespoň 90% pravděpodobnost (modře) a počet vložení/delecí v sekvenci.