VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

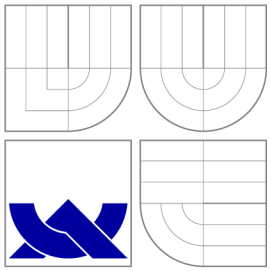
PREDIKCE SEKUNDÁRNÍ STRUKTURY PROTEINŮ S VYUŽITÍM NEURONOVÝCH SÍTÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

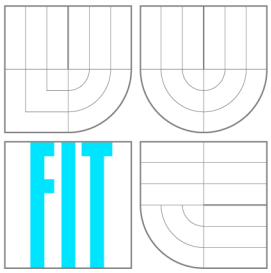
AUTOR PRÁCE
AUTHOR

RADEK ŠAMŠULA

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PREDIKCE SEKUNDÁRNÍ STRUKTURY PROTEINŮ S VYUŽITÍM NEURONOVÝCH SÍTÍ

NEURAL NETWORKS FOR PROTEIN STRUCTURE PREDICTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

RADEK ŠAMŠULA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADEK BURGET, Ph.D.

BRNO 2012

Abstrakt

Bakalářská práce se zabývá problematikou neuronových sítí a predikcí sekundární struktury proteinů. Blíže se zaměřuje na vícevrstvé neuronové sítě s učícím algoritmem zpětného šíření chyby a na jejich použití pro predikci. Je zkoumán vliv architektury sítě a nastavení jejich parametrů na výslednou predikci. Dále je diskutována vhodnost použití této sítě pro tento druh predikce.

Abstract

Bachelor's thesis studies the problems of neural networks and prediction of protein secondary structure. This thesis is focused on multilayer neural network with back-propagation learning algorithm and their use for prediction. It describes the influence of network architecture and their parameters settings on the prediction results. Furthermore suitability of the network for this kind of prediction is discused.

Klíčová slova

Neuron, neuronová síť, predikce, backpropagation, struktura proteinu.

Keywords

Neuron, neural network, prediction, backpropagation, protein structure.

Citace

Radek Šamšula: Predikce sekundární struktury proteinů s využitím neuronových sítí, bakalářská práce, Brno, FIT VUT v Brně, 2012

Predikce sekundární struktury proteinů s využitím neuronových sítí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Radka Burgeta. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal

.....
Radek Šamšula
16.5.2011

Poděkování

Děkuji Ing. Burgetovi za odborné rady a pomoc

Obsah

Kapitola 1

Úvod

V dnešní době, době vědeckého a technického pokroku, hrají nezastupitelnou roli výpočetní technologie. Důležitým odvětvím vědy, využívajícím tyto technologie, je předpovídání, neboli predikce, různých jevů. Právě touto problematikou se zabývá tato práce. Jedná se konkrétně o predikci využívající neuronové sítě.

Modely založené na neuronových sítích se dnes objevují v mnoha odvětvích lidské činnosti. Umělé neuronové sítě jsou statistické modelovací nástroje, které mají široké pole možných aplikací. Předpovídat je možné spotřebu energie, riziko lékařského zákroku, ekonomické a finanční závislosti, případně předpovědi chaotických časových řad. V mnoha případech tyto modely dosahovaly lepších výsledků než jiné přístupy[10]. Tato práce se však zaměřuje na předpovědi z oboru molekulární biologie.

Úvodní kapitola(2) nás seznámí s proteiny z hlediska molekulární biologie, s jejich stavbou a strukturou. Vysvětluje pojmy struktura proteinu a popisuje vztahy mezi jednotlivými typy struktur.

Další kapitoly(3,4) jsou věnovány teoretickým otázkám neuronových sítí, krátkému úvodu do historie vzniku a vývoje neuronových sítí. Poté se zaměří na konkrétní typy sítí.

V kapitole Návrh neuronové sítě pro predikci sekundární struktury proteinů (5) je rozebrán návrh samotné implementace neuronové sítě, které byla použita pro predikci sekundární struktury. Jsou zde podrobně rozebrány problémy, se kterými se můžeme setkat při návrhu aplikace pro predikci pomocí neuronových sítí.

Kapitola Experimenty(6) se na příkladech a konkrétních testech snaží objasnit určité zákonitosti fungování predikce pomocí neuronových sítí.

V poslední kapitole(7) jsou pak zhodnoceny výsledky práce.

Kapitola 2

Proteiny

Proteiny, česky také bílkoviny, tvoří většinu suché hmotnosti buňky. Nejsou však pouze stavebními kameny, z nichž je buňka postavena. Obstarávají také většinu buněčných funkcí. Některé proteiny přenášejí zprávy od jedněch buněk ke druhým, zatímco jiné slouží jako integrátory signálu, které předávají soubory signálů z plasmatické membrány do jádra jednotlivých buněk. Ještě další proteiny slouží jako drobné molekulové stroje s pohyblivými částmi: například *kinesin* posouvá organely cytoplasmou. Z chemického hlediska je jedná o vysokomolekulární látky složené z 20 různých aminokyselin, z nichž každá má jiné chemické vlastnosti.

Molekula proteinu je tvořena dlouhým řetězcem těchto aminokyselin. Tvar proteinu je dán pořadím těchto aminokyselin[4]. Mezi sousedními řetězci vzniká spojení *kovalentní peptidovou vazbou*, viz. obrázek 2.1.

Obrázek 2.1: Peptidová vazba, obrázek převzat z [19]

Kostru aminokyselin tvoří aminoskupina -NH2, karboxylová skupina -COOH a α uhlík C, na který je navázán postranní řetězec obvykle označovaný R. Z 20 aminokyselin, které se vždy vyskytují v lidském organismu, může v případě jednoduchého proteinu, složeného ze 100 aminokyselin, vzniknout 20^{100} (tj. asi $1,3 * 10^{130}$) rozdílných primárních proteinových struktur[4]. Z toho vyplývá, že existuje daleko větší množství různých proteinů, než je jich obsaženo ve všech živých organismech na Zemi. V tabulce 2.1 jsou vypsány známé aminokyseliny s jejich jednopísmennou a třípísmennou zkratkou.

2.1 Struktura proteinů

Proteiny tvoří trojrozměrné struktury. Tyto proteiny se ustalují v přirozeném uspořádání. Rozlišujeme několik typů struktur[4].

Aminokyselina	zkratka	
Asparagová kys.	Asp	D
Glutamová kys.	Glu	E
Arginin	Arg	R
Lysin	Lys	K
Histidin	His	H
Asparagin	Asn	N
Glutamin	Gln	Q
Serin	Ser	S
Threonin	Thr	T
Tyrosin	Tyr	Y
Alanin	Ala	A
Glycin	Gly	G
Valin	Val	V
Leucin	Leu	L
Isoleucin	Ile	I
Prolin	Pro	P
Fenylalanin	Phe	F
Methionin	Met	M
Tryptofan	Trp	W
Cystein	Cys	C

Tabulka 2.1: Aminokyseliny nacházející se v proteinech

2.1.1 Primární struktura

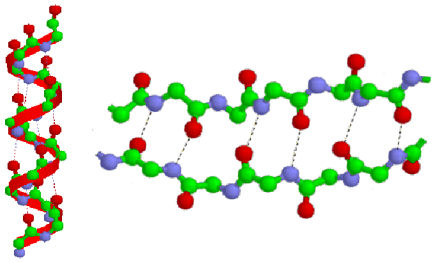
Primární struktura je dána pořadím aminokyselin v polypeptidovém řetězci. Standardně se zapisuje od N-konce k C-konci proteinu. Určuje chemické vlastnosti bílkoviny a také determinuje vyšší struktury. Primární struktura neříká, jak vypadá prostorové uspořádání proteinu[20].

2.1.2 Sekundární struktura

Sekundární struktura je přímo závislá na primární struktuře. Popisuje prostorové uspořádání polypeptidového řetězce „na krátkém úseku“, tj. mezi několika po sobě jdoucími aminokyselinami[20]. Existuje několik známých druhů sekundárních struktur. Nejpočetnějšími jsou *alfa šroubovice* (alfa-helix), struktura *skládaného listu* (beta-sheet) a obecně neuspořádaná struktura (random coil). Predikcí (na základě primární struktury) těchto struktur se blíže věnuje kapitola 5.

2.1.3 Terciární struktura

Za sekundární strukturu je označován celkový tvar jedné proteinové molekuly; prostorový vztah sekundární struktury k jiné. V bioinformatice se často používá spojení část řetězce(sekundární struktura) a celý řetězec(terciární struktura)[4].



Obrázek 2.2: Příklad sekundární struktury, α-helix(vlevo) a β-sheet, převzato z[1]

2.1.4 Kvartérní struktura

Struktura tvořená několika polypeptidovými řetězci(molekulami proteinů) označována jako proteinová podjednotka v kontextu proteinových komplexů. Kvartérní struktura též řeší prostorové uspořádání těchto podjednotek. Takovéto uspořádání vykazují jen složitější bílkoviny, např. fibrily kolagenu, nebo lidské DNA polymerázy[20].

Kapitola 3

Neuronové sítě a klasické modely

3.1 Vznik neurovýpočtů

Za počátek vzniku oboru neuronových sítí je považována práce Warrena McCullocha a Waltera Pittse z roku 1943[12], kteří vytvořili velmi jednoduchý matematický model *neuronu*, což je základní buňka nervové soustavy. Číselné hodnoty v tomto modelu byly převážně bipolární (tj. z množiny {-1,0,1}). Ukázali, že nejjednodušší typy neuronových sítí mohou z principu počítat libovolnou aritmetickou nebo logickou funkci. Ačkoliv nepočítali s možností bezprostředního praktického využití svého modelu, jejich článek měl velký vliv na ostatní badatele. Například zakladatel kybernetiky Norbert Weiner se jím inspiroval při studiu podobnosti činnosti nervové soustavy a systému výpočetní techniky. Nebo autor amerického projektu elektronických počítačů John von Neumann napsal práce[13, 14], ve kterých navrhoval výzkum počítačů, které by byly inspirovány činností mozku[23].

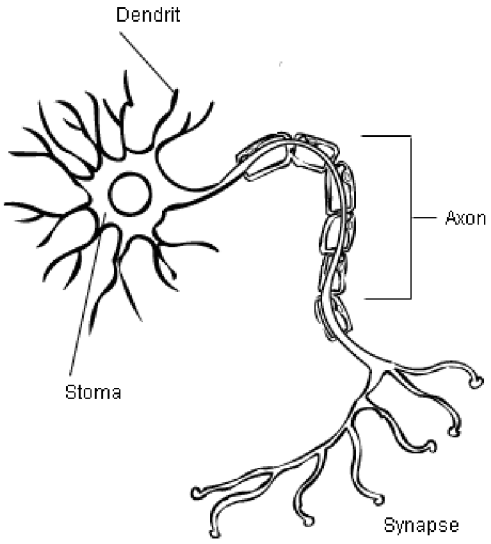
3.2 Modely neuronových sítí

3.2.1 Inspirace biologickým neuronem

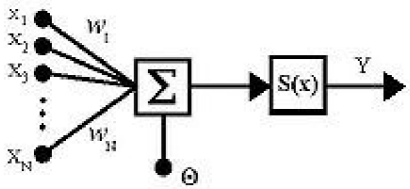
Formální neuron tvořící matematické neuronové sítě prakticky vychází z biologického neuronu. Jeho funkce v síti mají co nejlépe odrážet funkce svého biologického vzoru. Proto je dobré vědět něco a funkcích biologického neuronu v živém mozku. Neuron je základní prvek nervové soustavy. Je uzpůsoben k přenosu signálu, takže jej kromě těla, tzv. *stoma*, tvoří i vstupní a výstupní kanály: *dendrity* a *axon*[23]. Vstupy do neuronu, tedy dendrity jsou krátké(2-3 mm) výběžky z těla neuronu(v jednom neuronu 10^2 až 10^5). Zatímco výstup, axon, je jeden o délce 1 cm až několik desítek cm[21]. Z axonu obvykle odbočuje řada větví, tzv. *terminálů*, zakončených blánou, která se převážně stýká s výběžky, tzv. *trny*, dendritů jiných neuronů. K přenosu informace pak slouží unikátní přenosové mezineuronové rozhraní, tzv. *synapse*. *Synaptická vazba* může být inhibiční nebo excitační[23]. Biologický neuron je znázorněn na obrázku 3.1.

3.2.2 Formální neuron

Základem matematické sítě je formální neuron (dále jen neuron), což je přeformulování funkce biologického neuronu do matematické řeči. Schéma umělého neuronu je na obrázku 3.2



Obrázek 3.1: Biologický neuron



Obrázek 3.2: Model umělého neuronu

Neuron má n obecně reálných vstupů $x_1,\ldots,x_n$, které modelují dendrity. Vstupy jsou ohodnoceny obecně reálnými synaptickými *váhami* $w_1,\ldots,w_n$, které určují jejich propustnost. Suma vstupních hodnot představuje *vnitřní potenciál* neuronu[23]:

$$\xi = \sum_{i=1}^{n} w_i x_i \tag{3.1}$$

Hodnota vnitřního potenciálu ξ po dosažení tzv. *prahové* hodnoty h indukuje *výstup*(stav) neuronu y, který modeluje elektrický impuls axonu. Nelineární nárůst výstupní hodnoty $y = \sigma(\xi)$ při dosažení prahové hodnoty potenciálu h je dán tzv. *aktivační*(*přenosovou*)*funkcí* σ. Nejjednodušším typem aktivační funkce je tzv. *ostrá nelinearita*, která má tvar:

$$\sigma(\xi) = \begin{cases} 1 & jestlie \quad \xi \geq h \\ 0 & jestlie \quad \xi < h \end{cases} \tag{3.2}$$

3.2.3 Neuronová síť

Základním prvkem umělé neuronové sítě je formální neuron. Tyto neurony jsou vzájemně propojeny tak, že výstup neuronu je vstupem obecně více neuronů podobně, jako terminály

axonu biologického neuronu jsou spojeny s dendrity jiných neuronů. Počet neuronů a jejich vzájemné propojení určuje *topologii* sítě. V takové síti můžeme rozdělit několik druhů neuronů, *vstupní*, *výstupní* a *skryté*[23].

Neuronová síť se v čase vyvíjí, mění se jejich *konfigurace*. V souvislosti se změnou různých charakteristik sítě v čase lze hovořit o *dynamice sítě*. Dynamiku sítě lze rozdělit do tří kategorií: *organizační* (změna topologie), *aktivní* (změna stavu) a *adaptivní* (změna konfigurace) dynamika. V biologické neuronové síti tyto dynamiky probíhají současně[23].

Organizační dynamika specifikuje architekturu sítě a její případnou změnu(u některých typ sítí, viz. dále). Změna topologie se většinou uplatňuje v rámci adaptivního režimu tak, že je síť rozšířena o případné další prvky a odpovídající spoje. Ve většině modelů se však topologie sítě nemění. V zásadě existují dva typy organizační stavby sítě: *cyklická* a *acyklická*(resp. *dopředná*). V cyklické síti existuje alespoň jedna skupina neuronů(dva a více), které svými spoji vytváří uzavřený okruh (*cyklus*). Naopak u acyklické sítě neexistuje ani jeden uzavřený cyklus. Acyklickou architekturu lze rozdělit do tzv. *vrstev*, kde spoje jdou jedním směrem od vstupů do nižších vrstev, dále do vyšších, až k výstupním a obecně můžou přeskočit jednu či více vrstev. Nikde však nespojují prvky jedné vrstvy, ani se nevrací na nižší vrstvu.

Aktivní dynamika specifikuje *počáteční stav* sítě a jeho změny v čase při pevné topologii a konfiguraci. V počátečním(nulovém) čase se na prvky vstupní vrstvy nastaví vstupy a zbylé neurony jsou ve své počáteční konfiguraci. Po inicializaci stavu sítě probíhá průchod signálu sítí (vlastní výpočet), od vstupů až k výstupům. Obecně se uvažuje spojitý vývoj průběh této operace, kdy stav sítě je spojitou funkcí v čase, která je obvykle zadaná diferenciální rovnicí. Většinou však aktivační dynamika probíhá v diskrétním čase. Na začátku je čas nastaven na 0 a stav sítě se mění krokově s narůstajícím diskrétním časem.

Aktivní dynamika také určuje funkci jednoho každého neuronu. Ve většině sítí je tato funkce pro každý neuron stejné, hovoříme pak o *homogenní* neuronové síti. Součástí aktivační dynamiky je i diskrétní přenosová funkce neuronu, často aproximována spojitou aktivační funkcí, popř. nahrazena úplně jinou. V této praxi se setkáváme např. s následujícími *sigmoidními aktivačními funkcemi*: *ostrá nelinearita* (hard limiter)(rovnice 3.3), *saturovaná lineární funkce*(satured-linear function)(rovnice 3.4), *standardní (logická) sigmoida*(standard sigmoid)(rovnice 3.5), *hyperbolický tangens*(3.6) apod. Obrázky přenosových funkcí převzaty z [18]

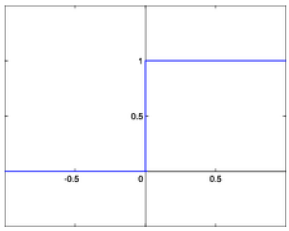
$$\sigma(\xi) = \begin{cases} 1 & \xi \geq 1 \\ 0 & \xi < 0 \end{cases} \quad \text{ostrá nelinearita} \tag{3.3}$$

$$\sigma(\xi) = \begin{cases} 1 & \xi < 1 \\ \xi & 0 \leq \xi \leq 1 \\ 0 & \xi < 0 \end{cases} \quad \text{saturovaná lineární funkce} \tag{3.4}$$

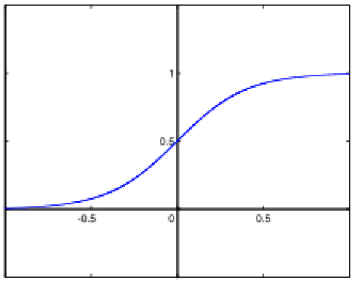
$$\sigma(\xi) = \frac{1}{1 + e^{-\xi}} \quad \text{standardní (logická) sigmoida} \tag{3.5}$$

$$\sigma\xi = \frac{1 - e^{-\xi}}{1 + e^{-\xi}} \quad \text{hyperbolický tangens} \tag{3.6}$$

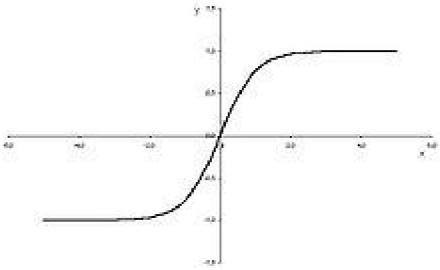
V režimu **adaptivní dynamiky** probíhá učení sítě, tzn. nastavování vah a biasů mezi jednotlivými vrstvami sítě. Adaptivní dynamika specifikuje počáteční konfiguraci a jakým způsobem se v čase mění. Cílem adaptivní dynamiky je najít takovou konfiguraci sítě, která by v aktivním režimu realizovala předepsanou funkci[23]. Příkladem učícího algoritmu je algoritmu backpropagation, který podrobněji proberu v 5.4.



Obrázek 3.3: Ostrá nelinearita



Obrázek 3.4: Standardní sigmoida



Obrázek 3.5: Hyperbolické tangens

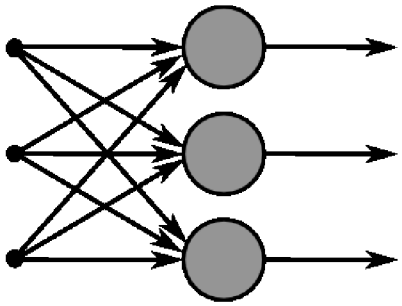
3.3 Síť perceptronů

Síť perceptronů je historicky prvním úspěšným modelem neuronové sítě. Organizační dynamika je velmi jednoduchá, jedná se o architekturu jednovrstvé sítě $n-m$. To znamená, že síť se skládá z n vstupních neuronů a m výstupních, tak, že výstup každého neuronu $x_1,\ldots,x_n$ ze vstupní vrstvy je vstupem do každého neuronu $y_1,\ldots,y_m$ výstupní vrstvy. Dále w_{ji} představuje reálnou synaptickou váhu od i-tého vstupního k j-témů výstupnímu neuronu a w_{j0} je bias j-tého výstupního neuronu[23, 21]. Jednoduchá síť perceptronů je znázorněna na obrázku 3.3

Aktivní dynamika sítě perceptronů popisuje způsob výpočtu výstupu sítě na základě vstupů. Stavy vstupních neuronů sítě se nastaví na vstup sítě a výstupní neurony počítají svůj binární stav, který určuje výstup sítě, podobně jako formální neuron(3.1):

$$\xi_j = \sum_{i=0}^{n} w_{ji}x_i \quad j = 1,\ldots,m, \tag{3.7}$$

kde koeficienty $\mathbf{w} = (w_{10},\ldots,w_{1n},\ldots,w_{m0},\ldots,w_{mn})$ tvoří koeficienty sítě[23].



Obrázek 3.6: Síť perceptronů

Cílem adaptivní dynamiky je, aby síť pro každý vstup $\mathbf{x}_k (k = 1, \ldots, p)$ z tréninkové množiny odpovídalo v aktivním režimu požadovaným výstupům $\mathbf{d}_k$, tj. aby platilo:

$$\mathbf{y}(\mathbf{w}, \mathbf{x}_k) = \mathbf{d}_k \quad k = 1, \ldots, p. \tag{3.8}$$

3.4 Vícevrstvá síť a backpropagation

Vícevrstvá neuronová síť s učícím algoritmem zpětného šíření chyby(*backpropagation*) je asi neznámějším a nejpoužívanějším typem sítě. Tento algoritmus používá asi 80 procent všech aplikací neuronových sítí[23]. Model vícevrstvé sítě je takovým zobecněním sítě perceptronů. Mezi vstupní a výstupní vrstvu sítě vstupují tzv. skryté vrstvy. Tento model se někdy nazývá *vícevrstvý perceptron*. Blíže se s tímto modelem seznámíme v kapitole Návrh neuronové sítě(5). V téže kapitole bude implementován pro predikci sekundární struktury proteinů

3.5 MADALINE

Základním prvkem v tomto modelu je neuron ADALINE(**ADA**ptive **LIN**ear **E**lement), který je velmi podobný perceptronu. MADALINE(Multiple ADALINE) byl navržen Windrowem a Hoffem[16, 17] a formálně téměř identický se sítí perceptronů, viz. 3.3, i když původně vychází z jiných principů.

Organizační dynamika MADALINE(tj. architektura) a značení parametrů je stejné jako u sítě perceptronů, avšak místo perceptronu uvažujeme ADALINE.

Aktivní dynamika se u tohoto modelu liší tím, že výstupy sítě mohou být obecně reálné a jednotlivé ADALINE realizují lineární funkci, tj. chybí nelineární aktivační funkce. Funkce MADALINE $\mathbf{y}(\mathbf{w}) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, která závisí na konfiguraci $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_m)$, kde $\mathbf{w}_1 = (w_{10}, \ldots, w_{1n}), \ldots, \mathbf{w}_m = (w_{m0}, \ldots, w_{mn})$, je tedy dána afinní kombinací vstupů:

$$y_j = \sum_{i=0}^{n} w_{ji} x_i j = 1, \ldots, m. \tag{3.9}$$

Také geometrický význam funkce j-tého ADALINE se liší od perceptronu. Uvažujeme vstup $\mathbf{x} = (x_1, \ldots, x_n)$, tj. bod $[x_1, \ldots, x_n]$ v n-rozměrném vstupním prostoru. Nadrovina s koeficienty $\mathbf{w}_j$ pro daný j-tý ADALINE daná rovnicí

$$w_{j0} + \sum_{i=1}^{n} w_{ji} x_i = 0 \tag{3.10}$$

rozděluje tento prostor na dva podprostory, ve kterých má hodnota výstupu y_j (3.9) odlišné znaménko.

V **adaptivním** režimu je požadovaná funkce MADALINE zadána tréninkovou posloupností, kde reálné vstupy tréninkových vzorů $\mathbf{x}_l$ jsou generovány náhodně s daným rozdělením pravděpodobnosti a u každého je dán požadovaný reálný výstup $\mathbf{d}_k$. Model MADALINE má diskrétní adaptivní dynamiku, tj. v každém časovém kroku učení $t = 1, 2, \ldots$ je síti předložen k-tý vzor $(k = t)$ z tréninkové posloupnosti, podle kterého jsou adaptovány váhy. Je zajímavé, že adaptivní dynamika MADALINE je formálně totožná s adaptací perceptronu. Widrow a Hoff dokázali, že adaptivní proces konverguje z libovolné počáteční konfigurace $\mathbf{w}^{(0)}$ ke konfiguraci $\mathbf{w}^*$, která minimalizuje chybové funkce[23]

3.6 Sítě s kaskádovou architekturou

Tento model nepatří mezi klasické modely neuronových sítí, avšak je jistým zobecněním perceptronových sítí. Používá učící algoritmus *kaskádové korelace* a vznikl na přelomu 80. a 90. let. Autory jsou Fahlman a Labiere[7].

Organizační dynamika(architektura) je dopředná síť s jednou vrstvou n vstupních jednotek, skrytou vrstvou obsahující h perceptronů a lineární výstupní vrstvou o m jednotkách. Perceptronové jednotky skryté vrstvy jsou navíc pospojovány laterálními spoji, které jdou jedním směrem, řekněme zleva doprava ve smyslu očíslování skrytých jednotek. Každý perceptron dostává jako vstupní signály i výstupy ze všech předchozích jednotek ve skryté vrstvě. Tzn., že i-tá jednotka ve skryté vrstvě má jako obvykle n vstupů spojených se vstupní vrstvou a dále $i-1$ vstupů napojených na výstupy jednotek skryté vrstvy. Výstupní jednotky jsou spojeny se všemi jednotkami ze skryté i vstupní vrstvy.

Funkce $\mathbf{f} : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ realizovaná kaskádovou sítí v **aktivním** režimu se díky rekurentním vztahům mezi skrytými jednotkami zapisuje jedním vzorcem poměrně nesnadno, ale samotný způsob, jakým síť počítá odezvu na předložený vstup $\mathbf{x} \in \mathbb{R}^n$, je jednoduchý. Jednotky ve skryté vrstvě postupně dle svého pořadí i spočtou výstupní hodnotu z_i podle obvyklé přechodové funkce známé z perceptronu:

$$z_i = \sigma\left(\sum_{j=0}^{n} w_{ij}x_j + \sum_{j=1}^{i-1} w_{i(n+j)}z_j\right) \quad pro \quad i = 1, \ldots, h, \tag{3.11}$$

kde w_{ij} jsou váhy, σ je aktivační funkce.

Pro každý perceptron i jsou tedy váhy příslušející výstupům od předcházejících jednotek zařazeny na konec váhového vektoru, odpovídají vždy pozicím $n+1, \ldots, n+i-1$. V další fázi výpočtu potom jednotky ve výstupní vrstvě spočtou lineární kombinaci svých vstupů a aplikují přechodovou funkci[23]:

$$y_i = \sigma\left(\sum_{j=0}^{n+h} w_{ij}z_j\right) \quad pro \quad i = 1, \ldots, m. \tag{3.12}$$

Adaptivní dynamika je realizována algoritmem kaskádové korelace. Ta je příkladem konstruktivního či inkrementálního algoritmu. Učení začíná s minimální konfigurací sítě, neboli bez skrytých jednotek. Učení probíhá ve dvou časových škálách: v každém velkém cyklu učíme aktuální konfiguraci sítě pomocí lineárního gradientního sestupu. Fahlman v [7] používá algoritmus *quickprop*, který je podrobně popsán v [6]. Jde o jednu ze sofistikovaných variant algoritmu zpětného šíření, která vychází z Newtonovy gradientní metody. V

momentě, kdy už nedochází k výraznému zlepšení chyby sítě, přidáme do sítě další skrytou jednotku a opakujeme gradientní učení. Konfiguraci vah nově přidané jednotky se snažíme nastavit tak, aby co nejvíce snížila chybu sítě. Toho docílíme maximální korelací této nové jednotky s chybou sítě[23].

Hlavní rysy algoritmu kaskádové korelace jsou následující:

- Díky inkrementálnímu (čí konstruktivnímu) učení není potřeba dopředu určovat rozměry sítě(počet skrytých jednotek).
- Inkrementální mechanismus učení umožňuje snadné pozdější učení nových vzorů.
- Kaskádová korelace je typicky rychlejší než zpětné šíření chyby, protože se v každém okamžiku adaptuje pouze jedna vrstva vah.
- Časově nejnáročnější část,maximalizace korelace,lze snadno paralelizovat.

Kapitola 4

Asociativní neuronové sítě

4.1 Lineární asociativní síť

Lineární asociativní síť, navržená Andersonem[5] je příkladem modelu neuronové sítě, který se využívá jako asociativní paměť. Na rozdíl od klasických počítačů, kdy klíčem k vyhledání položky v paměti je adresa, u asociativní paměti probíhá vybavení příslušné informace na základě její částečné znalosti (asociace). Například v databázových aplikacích je znalost některých položek záznamu postačující k vyhledání celého záznamu. Podobně u člověka např. černobílá fotografie přítele pomůže vybavit barvu jeho vlasů či očí. V zásadě rozlišujeme dva typy asociativní paměti, a to paměť *autoasociativní* a paměť *heteroasociativní*. U autoasociativní paměti se jedná o upřesnění či zúplnění vstupní informace na základě již naučeného, což v našem případě s černobílou fotografií znamená vybavení odpovídajícího barevného obrazu. Naproti tomu u heteroasociativní paměti dochází k vybavení určité sdružené informace na základě vstupní asociace, což v uvedeném případě může odpovídat určení jména osoby na fotografii[23].

Organizační a aktivační dynamika lineární asociativní neuronové sítě je téměř identická jako u modelu MADALINE, popsaného v 3.5. Rozdíl spočívá v tom, že lineární asociativní síť v aktivním režimu místo afinních kombinací počítá jen lineární kombinace vstupů, tj. chybí formální jednotkový vstup a odpovídající biasy jsou nulové. Formálně lze funkci lineárně asociativní sítě $\mathbf{y}(\mathbf{w}) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ zapsat[23]:

$$y_i = \sum_{i=1}^{n} w_{ij} x_i \quad j = 1, \ldots, m. \tag{4.1}$$

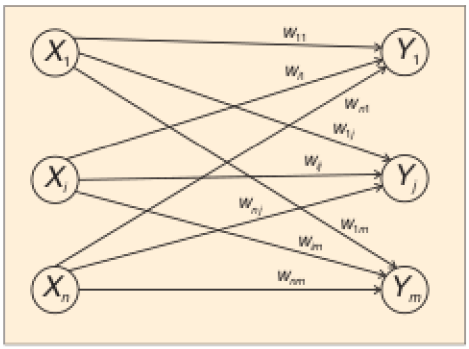
Asociativní paměť neuronové sítě odpovídá síti, ve které jsou váhové hodnoty determinovány takovým způsobem, aby si byla schopna zapamatovat množinu P asociovaných vzorů. Každou asociaci tvoří pár vektorů $\mathbf{s}(p)$, $\mathbf{t}(p)$, kde $p = 1, 2, \ldots, P$, kde každý vektor $\mathbf{s}(p)$ obsahuje n komponent a každý vektor $\mathbf{t}(p)$ obsahuje m komponent. Jeden z možných způsobů adaptace této sítě je motivován neurofyziologickým *Hebbovým zákonem*, který uvádí, že změna synaptické váhy spoje mezi dvěma neurony je úměrná jejich souhlasné aktivitě (tj. součinu jejich stavů). Opačná aktivita neuronů tuto vazbu zeslabuje. Donald Hebb tímto způsobem vysvětloval vznik podmíněných reflexů, kdy současná aktivita (popř. pasivita) prvního neuronu odpovídající podmínce (příčině) a druhého neuronu vyvolávající reflex posiluje synaptickou vazbu spoje směrem od prvního neuronu k druhému. *Hebbovo adaptační pravidlo pro lineární asociativní neuronové sítě* je nejběžnější metodou pro stanovení váho-

vých hodnot na spojích mezi jednotlivými neurony. Pracuje s vektory, které jsou zapsány v binární i bipolární reprezentaci. Jeho algoritmus probíhá v následujících krocích[15]:

1. Inicializujeme všechny váhové hodnoty číslem 0, tj. $w_{ij} = 0, (i = 1, \ldots, n; j = 1, \ldots, m)$.
2. Vstupní vektor $\mathbf{s} = (s_1, \ldots, s_i, \ldots, s_n)$ tvoří sloupcovou matici $\mathbf{S}$ typu $nX1$, tj. $\mathbf{S} = \mathbf{s^T}$.
3. Asociovaný vektor $\mathbf{t} = (t_1, \ldots, t_j, \ldots, t_m)$ tvoří řádkovou matici $\mathbf{T}$ typu $1Xm$, tj. $\mathbf{T} = \mathbf{t}$.
4. Součin obou matic $\mathbf{S}$ a $\mathbf{T}$ pak tvoří váhovou matici $\mathbf{W}$, ve které jsou uloženy informace o asociovaném páru vektorů $\mathbf{s} : \mathbf{t}$.

4.1.1 Heteroasociativní paměť neuronové sítě

Architektura heteroasociativní paměti neuronové sítě je zobrazena na obr. 1 a její adaptace probíhá podle Hebbova adaptačního pravidla pro lineární asociativní neuronové sítě. Učení heteroasociativní paměť není iterační.



Obrázek 4.1: Architektura heteroasociativní paměti neuronové sítě

Algoritmus heteroasociativní paměti nalezne pro každý vektor $\mathbf{x}$ ve vrstvě X vhodný vektor $\mathbf{y}$ ve vrstvě Y. Algoritmus probíhá v následujících krocích:

0. Adaptace sítě, tj. nastavení všech váhových hodnot $w_{ij} (i = 1, \ldots, n; j = 1, \ldots, m)$ podle Hebbova adaptačního pravidla pro lineární asociativní neuronové sítě.
1. Pro vektor $\mathbf{x} = (x_1, ..., x_i, ..., x_n)$ opakovat kroky (2 až 4).
2. Inicializovat vrstvu X daným vstupním vektorem.
3. Vypočítat potenciál neuronů ve vrstvě Y

$$y\_in_j = \sum_{i=1}^{n} x_i w_{ij} \quad pro (j = 1, \ldots, m). \tag{4.2}$$

4. Vypočítat aktivaci neuronů ve vrstvě Y:

$$y = \begin{cases} 1 & pokud \quad y\_in_j > 0 \\ 0 & pokud \quad y\_in_j = 0 \\ -1 & pokud \quad y\_in_j < 0, \end{cases} \tag{4.3}$$

pro *bipolární* reprezentaci (práh $\theta = 0$).

5. Vypočítat aktivaci neuronů ve vrstvě Y:

$$y_j = \begin{cases} 1 & pokud \quad y\_in_j > 0 \\ 0 & pokud \quad y\_in_j \leq 0 \end{cases} \tag{4.4}$$

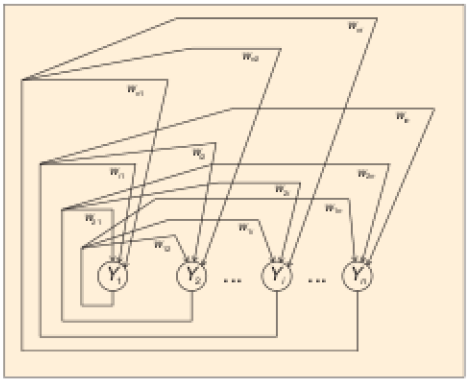
pro *binární* reprezentaci (práh $\theta = 0$).

4.1.2 Autoasociativní paměť neuronové sítě

Autoasociativní neuronové sítě s dopředným sířením signálu jsou speciálním případem heteroasociativních sítí, kdy jsou oba vektory tvořící množinu P asociovaných vzorů $(\mathbf{s}(p), \mathbf{t}(p)), kdep = 1, 2, \ldots, P)$ identické. Každou asociaci proto tvoří pár vektorů $\mathbf{s}(p) : \mathbf{s}(p), kdep = 1, 2, \ldots, P$. Každý vektor $\mathbf{s}(p)$ obsahuje n komponent. Váhové hodnoty na příslušných spojích jsou nastaveny Hebbovým adaptačním pravidlem pro lineární asociativní neuronové sítě[15].

4.2 Hopfieldova síť

Tento model byl navržen McCullochem a Pittsem[12] a dodnes je předmětem bádání mnoha badatelů. Avšak až Hopfield[9], který při analýze stability této sítě využil průhlednou analogii s fyzikální teorií magnetických materiálů, tento model proslavil[23].



Obrázek 4.2: Model diskrétní Hopfieldovy sítě

Diskrétní Hopfieldova síť se používá jako iterativní autoasociativní paměť. Model Hopfieldovy neuronové sítě je založen na využití energetické funkce svázané s neuronovou sítí tak, jak je to běžné u fyzikálních systémů. Architektura Hopfieldovy sítě s n neurony je znázorněna na obr. 4.2. Každý neuron je zde spojen se všemi ostatními neurony sítě. Všechny neurony v síti jsou tedy zároveň vstupní i výstupní. Každý spoj v síti mezi neuronem $i(i = 1, \ldots, n)$ a neuronem $j(j = 1, \ldots, n)$ je ohodnocen celočíselnými synaptickými vahami w_{ij} a w_{ji}, které jsou symetrické, tj. $w_{ij} = w_{ji}$. V základním modelu neuvažujeme biasy neuronů (všechny prahy neuronů jsou nulové) a žádný neuron není spojen sám se sebou, tj. odpovídající váhy $w_{jj} = 0(j = 1, \ldots, n)$[15].

4.3 Spojitá Hopfieldova síť

Na rozdíl od analogových modelů, které byli v této práci doposud popsaný, kde reálný stav neuronu je spojitou funkcí vnitřního potenciálu, je tato síť příkladem modelu, u kterého

je vývoj reálného stavu v aktivním režimu navíc spojitou funkcí času. Aktivní (případně adaptivní) dynamika je v takovém případě obvykle zadána diferenciální rovnicí. Tyto modely, pokud nepracujeme s jejich diskrétní verzí, nejsou vhodné pro simulaci na klasických počítačích. Jsou výhodné pro analogovou hardwarovou implementaci pomocí elektrických obvodů[23].

4.4 Boltzmannův stroj

Boltzmannův stroj je model je stochastickou variantou Hopfieldovy sítě se skrytými neurony. Boltzmannův stroj lze použít např. jako heteroasociativní paměť(viz. 4.1.1)

Kapitola 5

Návrh neuronové sítě pro predikci sekundární struktury proteinů

V předchozích kapitolách jsme se zběžně seznámili s různými typy neuronových sítí a jejími rozdíly. V následující kapitole se blíže zaměříme na dopřednou neuronovou síť a učící algoritmus zpětného šíření chyby (algoritmus back-propagation) a na použíti těchto technik na predikce sekundární struktury proteinů. Cílem návrhu je vytvořit funkční model sítě pro predikci sekundární struktury proteinů. Nejedná se o triviální úkol, protože predikce sekundární struktury proteinů sama o sobě je velmi složitý proces. Predikce založená na generalizaci vstupních dat sítě, kdy trénovací množina je nesrovnatelně menší než počet možných kombinací vstupů, přináší mnohá úskalí. Správnou analýzou problematiky se pokusím co možná nejlépe postihnout nastavení všech možných parametrů, které mohou ovlivnit proces učení.

5.1 Analýza vstupů a výstupů sítě

Před samotným návrhem dynamiky sítě je nutné si ujasnit některá fakt netýkající se přímo návrhu sítě. Síť musí umět zpracovávat sekvence primární struktury proteinů v určitém formátu. Pro predikci sekundární struktury jedné aminokyseliny v proteinu nestačí znát prvek samotný. Je nutné znát i jeho okolí v řetězci. Nejčastěji používanou délkou řetězce pro predikci proteinu je 7 prvků, kdy predikovaný je prostřední z nich. Je nutné si uvědomit že každý prvek může nabývat 20 různých hodnot, viz. tabulka 2.1. Dvacet různých aminokyselin tvořících řetězec o sedmi prvcích dává $7^{20} = 7.979 * 10^6$ možných řetězců na vstupu do sítě. Jen malé procento těchto řetězců, resp. aminokyselina na prostředním místě, má známou sekundární strukturu. Aby mohl být řetězec použit k učení sítě, musí být známá jeho sekundární struktura.

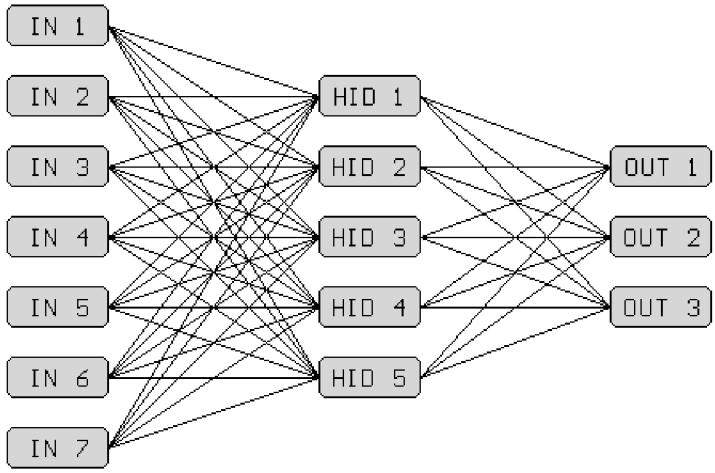
Existují různé databáze, shromažďující informace o proteinech a jejich strukturách. Například databáze UniProt[3]. V této databázi lze získat data potřebná k učení sítě. Mimo jiné i známe sekundární struktury různých proteinů. Tři nejčastější struktury jsou zmíněny v kapitole 2 v části věnované sekundární struktuře. Jsou to *alfa-helix*, *beta-sheet* a *random coil*, což jsou tří možné výstupy predikce neuronovou sítí. Jak se tyto informace projeví na architektuře sítě popíše organizační dynamika v této kapitole (část 5.2).

Pro převod informací u proteinech z textového tvaru, jak jej nabízí databáze Uniprot[3], do souboru řetězců jsou použil mnou vytvořený skript v pythonu.

5.2 Organizační dynamika

Budeme uvažovat acyklickou síť s jednou skrytou vrstvou neuronů, tzn. síť bude mít celkem 3 vrstvy neuronů. Tato topologie se v průběhu aktivní ani adaptivní dynamiky nemění.

Počet neuronů vstupní vrstvy musí odpovídat vstupním datům jdoucím do neuronové sítě. V předchozí části(5.1) byly řečeny důležité informace. Vstupem neuronové sítě je sedmice znaků odpovídajících aminokyselinám v řetězci(znaky odpovídají znakům v tabulce 2.1). Pro každý znak z řetězce je tedy ve vstupní vrstvě sítě umístěno 20 neuronů. Vstupnímu neuronu, který odpovídá znaku z řetězce je jeho vstupní hodnota nastavena na 1, ostatním 19 na 0. Ve vstupní vrstvě je takto umístěno $7 * 20 = 140$ neuronů.



Obrázek 5.1: Modelu neuronové sítě, převzato z [11]

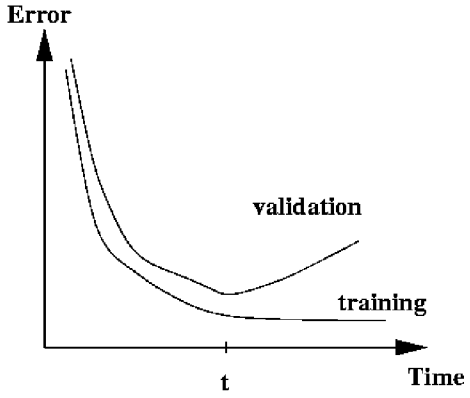
Výstupní vrstva má tři neurony, jeden pro každou možnou predikovanou sekundární strukturu. V textu zavedeme následující označení, množinu n neuronů vstupní vrstvy označíme $\mathbf{X}$ a množinu m neuronů výstupní vrstvy označíme $\mathbf{Y}$. Jednotlivé neurony značím indexy i, j apod. a ξ_j je reálný vnitřní potenciál a y_j reálný stav. Dále $j_{\leftarrow}$ je množina všech neuronů, které jsou vstupem do neuronu j. Spoj mezi neuronem i jedné vrstvy a neuronem j vrstvy vyšší je označen w_{ij}. Uvažujeme síť, kde každý neuron vrstvy i je propojen se všemi neurony vyšší vrstvy j. Na obrázku 5.2 je znázorněn příklad modelu neuronové sítě, kde každý neuron nižší vrstvy je propojen se všemi neurony vrstvy vyšší. Každý neuron v síti obsahuje svůj práh, neboli bias h. Při implementaci algoritmu backpropagation zjistíme, že pro každý neuron je dále potřeba uchovat jeho chybu err určenou během trénování sítě(viz. 5.4) a také všechny váhy w_{ij} k vyšší vrstvě vypočtené v předchozím cyklu učení, označené jako wo. Architektura jednoho neuronu pak může v pseudokódu vypadat následovně jako v 5.1

Největším problémem při návrhu architektury sítě je volba počtu neuronů skryté vrstvy. Neexistuje žádné pravidlo pro výpočet tohoto počtu. U sítí s jednou, či dvěma skrytými vrstvami se očekává, že algoritmus backpropagation zobecní příslušné vztahy z trénovací množiny a zohlední je ve vahách jednotlivých neuronů. Počet neuronů skryté vrstvy by měl

struct Node {		
	double ξ	vnitřní potenciál neuronu
	double y	reálný stav neuronu, resp. výstup
	double h	práh(bias)
	double err	chyba neuronu
	double $w[j]$	množina vah k neuronům vyšší vrstvy
	double $wo[j]$	váhy z předchozího trénovacího cyklu
}		

Tabulka 5.1: Jeden uzel sítě

odpovídat složitosti řešeného problému. Příliš malá síť nemůže řešit složité úlohy. Učení s malou sítí se obvykle zastaví v nějakém lokálním minimu a je třeba topologii doplnit o další neurony skryté vrstvy. Pokud je ale počet skrytých neuronů příliš velký, nejen, že roste časová náročnost učení, ale nemusí také dojít ke správné generalizaci vzorů, které nebyli součástí tréningové množiny. Tomuto jevu se říká *přeučení*(overfitting) a průběh učení, při kterém tento jen nastane je znázorněn na obrázku 5.2, kde *Error* je chyba predikce, *Time* je čas běhu učení, *training* pak znázorňuje pokles chyby sítě v čase pro vzory z trénovací množiny. *Validation* pak znázorňuje změnu chyby predikce pro vzory z tzv. *testovací*, neboli *validační* množiny vzorů. čas t je pak ideální čas učení. Vzory z tréningové množiny jsou sice rozpoznávány s velkou přesností, ostatní jsou naopak rozpoznány velmi málo.



Obrázek 5.2: Přeučení sítě, převzato z [2]

5.3 Aktivní dynamika

V aktivním režimu je daný vstup sítě zpracován a převeden na výstup. Výpočet probíhá v diskrétním čase. Na začátku, v čase 0 je síť inicializována. Stavy neuronů $y_i (i \in \mathbf{X})$ vstupní vrstvy jsou nastaveny na odpovídající vstupy(viz. část 5.2), ostatní neurony nemají určen stav. Všechny váhy a biasy v sítí jsou nastaveny na velmi malé hodnoty, obecně náhodné v intervalu $(-1, 1)$. V diskrétním čase $t > 0$ jsou vypočteny hodnoty vnitřních potenciálů

(podobně jako u formálního neuronu, viz. rov. 3.1)

$$\xi_j = \sum_{i \in j_{\leftarrow}} w_{ij} y_i \tag{5.1}$$

všech neuronů j, jejichž vstupy(z $j_{\leftarrow}$) již mají určen svůj stav. Z takto vypočteného potenciálu je stanoven reálný stav $y_j = \sigma(\xi_j)$ neuronu j pomocí aktivační funkce, např. standardní sigmoidy(rov. 3.5):

$$y_j = \sigma_j(\xi_j), \qquad kde \quad \sigma_i(\xi_i) = \frac{1}{1 + e^{-\xi}} \tag{5.2}$$

Takto jsou postupně vypočteny všechny vnitřní potenciály, potažmo reálné stavy neuronů v síti, od vstupní, přes skrytou, po výstupní vrstvu. Výstupem aktivního režimu je pak hodnota reálných stavu je třech výstupních neuronech, pomocí níž lze predikovat sekundární struktury vstupního řetězce. Každé možné struktuře je přiřazen jeden výstupní neuron(více v části 5.4). V implementaci jsou uvažovány tři nejčastější struktury(Helix, Turn, Coil). Struktura, jejíž neuron má nejvyšší hodnotu, je výslednou predikovanou strukturou.

5.4 Adaptivní dynamika

Požadovaná funkce je zadána tréninkovou množinou, na které probíhá učení sítě. Jeden tréningový vzor z tréninkové množiny je podobný vzoru pro predikci. Je to řetězec sedmi po sobě jdoucích aminokyselin, kde prostřední z nich má známou sekundární strukturu. Znak odpovídající této struktuře je taktéž obsažen v tréninkovém vzoru. Tento vzor je načten na vstup sítě stejně jako v aktivním režimu. Podle znaku známé sekundární struktury jsou nastaveny neurony výstupní vrstvy.

Výstupní vrstva neuronů navíc, kromě záznamů uvedených v tab. 5.1 obsahuje záznam T, udávající očekávanou hodnotu na výstupu z tohoto neuronu na konci predikce. Tyto hodnoty jsou nastaveny na 1 pro odpovídající znak sekundární struktury, resp. na 0 u ostatních dvou neuronů.

Celý algoritmus backpropagation je pak přehledně zapsán v 5.2

Nyní se podrobněji zaměříme na problematiku tohoto algoritmu a na zkoumání jevů, které mohou nastat při učení.

První část adaptivní dynamiky probíhá stejně jako v aktivním režimu. Váhy v síti jsou nastaveny na malé náhodné hodnoty, např. v intervalu $(-1, 1)$ nebo menším. Žádná hodnota by však neměla překročit 1. Biasy jsou inicializovány podobně. V kapitole Experimenty (6) se pokusím zjistit závislost nastavení vah a biasů na učící schopnosti sítě.

Nastává aktivní režim učení. Postupně jsou vypočteny všechny vnitřní potenciály a reálné stavy všech neuronů v síti od nejnižší po nejvyšší vrstvu. Hodnota vnitřního potenciálu ξ_j je vypočtena jako součet reálných stavů y_i neuronů předchozí vrstvy násobených příslušnými váhami w_ij mezi nimi. Tento vnitřní potenciál je upraven o bias h_j neuronu. Takto vypočtený vnitřní potenciál

$$\xi_j = \sum_i w_{ij} y_i + h_j \tag{5.3}$$

je nutné převést na výstup neuronu. K tomu slouží aktivační funkce neuronu $y_j = \sigma(\xi_j)$. V našem případě je použita standardní sigmoida představená v části 3.2.3

$$y_j = \frac{1}{1 + e^{-\xi_j}} \tag{5.4}$$

1. Inicializuj síť, nastav všechny váhy a biasy.
2. dokud nejsou splněny ukončovací podmínky
   - pro každý vzorek z trénovací množiny proveď
     (a) //Aktivní režim
     (b) pro každý neuron j ve skryté nebo výstupní vrstvě vypočítej vzhledem v nižší vrstvě i
        i. $\xi_j = \sum_i w_{ij} y_i + h_j$
        ii. $y_j = \sigma(\xi_j)$
     (c) //Adaptivní režim
     (d) pro každý neuron j ve výstupní vrstvě
        i. $err_j = y_j(1 - y_j)(T_j - y_j)$
     (e) pro každý neuron j ve skryté vrstvě, vzhledem v neuronům k na vyšší vrstvě
        i. $err_j = y_j(1 - y_j) \sum_k err_k w_{jk}$
     (f) pro každou w_{ij} váhu v síti
        i. $\Delta w_{ij} = (l) err_j y_i$
        ii. $M = (m) * (w_{ij} * wo_{ij})$
        iii. $w_{ij} = w_{ij} + \Delta w_{ij} + M$
     (g) pro každý práh h_i v síti
        i. $\Delta h_j = (l) err_j$
        ii. $h_j += \Delta h_j$

Tabulka 5.2: Backpropagation algoritmus, inspirace v [8]

Tato funkce je diferencovatelná a umožňuje modelovat klasifikační problémy lineárně neoddělitelné. Tím dostáváme reálný stav y_j neuronu, který je delegován na další vrstvu neuronové sítě. Tímto způsobem jsou vypočteny všechny stavy v síti.

Nyní se dostáváme k fázi učení (adaptaci). Reálné stavy výstupních neuronů y_j jsou porovnány s očekávanými výstupy T_j a je spočítána chyba err_j každého neuronu výstupní vrstvy.

$$err_j = y_j(1 - y_j)(T_j - y_j) \tag{5.5}$$

Všimněte si, že $y_j(1 - y_j)$ je derivací aktivační funkce. Tato chyba je z neuronů k výstupní vrstvy přenesena na neurony j skryté vrstvy

$$err_j = y_j(1 - y_j) \sum_k err_k w_{jk} \tag{5.6}$$

Neurony vstupní vrstvy jsou obrazy reálných vstupů, tudíž neuvažujeme jejich chybu a není potřeba ji počítat. Podle vypočtených chyb je potřeba rekonfigurovat síť a to následovně. Pro všechny váhy w_{ij} v sítí

$$\begin{aligned} \Delta w_{ij} &= (l) err_j y_i \\ w_{ij} &= w_{ij} + \Delta w_{ij} + M, \end{aligned} \tag{5.7}$$

kde l je koeficient učení(learning rate), obvykle z $(0, 1)$. Použití koeficientu učení je jedním z nástrojů, jak se vyhnout uváznutí konfigurace sítě v lokálním minimu. Zvolení příliš malého

koeficientu může vést k neúměrnému prodloužení doby učení, naopak příliš velký koeficient může způsobit kolísání výsledků mezi špatnými výsledky a k ideálnímu stavu se síť nikdy nemusí dostat. M je *Momentum*. Momentum lze volně přeložit jako setrvačnost učení. Lze jej spočítat jako

$$M = (m) * (w_{ij} * wo_{ij}), \tag{5.8}$$

kde m je tzv. *momentum term*, obecně v intervalu (0,1) a je obdobou koeficientu učení. Pro oba tyto koeficienty l a m neexistuji ideální hodnoty, je nutno je volit experimentálně vzhledem ke druhu řešeného problému. Pokud je při učení použito momentum, koeficient učení nabývá zpravidla vyšších hodnot, což ve výsledku může urychlit učení. Zároveň je potřeba přenastavit všechny prahy h_i v síti

$$\begin{aligned} \Delta h_j &= (l) err_j \\ h_j &+= \Delta h_j \end{aligned} \tag{5.9}$$

Literatura[8] uvádí, že koeficient lze v přibývajícími cykly t učení snižovat způsobem $1/t$. Takové snižování je v případě našeho trénovaní, kdy dochází k desítkám a stovkám tisíc cyklů, nemožné. V pozdějších fázích učení by došlo k tak velkému snížení, že by se učení stalo neefektivní a téměř nekonečné. Možným řešením je upravit vzorec $1/t$ a dosáhnout menší strmosti klesání koeficientu učení.

Celý proces učení pomocí zpětného šíření chyby(backpropagation) probíhá v cyklech. V každém cyklu je načteno D vzorku vstupních dat pro učení. V případě, že by byla načítána v každém cyklu ve stejném pořadí, mohlo by dojít k ovlivnění učení sítě, kdy by se síť dokázala naučit predikovat jen určité procento vzorků z trénovací množiny. Proto je nutné v každém cyklu učení posílat tyto trénovací vzorky v jiném, nejlépe náhodném pořadí.

Velikost D trénovací množiny nás přivádí k další problematice, a to k časová náročnosti učení sítě. Časová složitost O učení metodou zpětného šíření chyby je $O(D \times \mathbf{w})$, kde $\mathbf{w}$ je množina všech vah v síti[8]. V další kapitole(6) se toto pokusím uvěřit.

Kapitola 6

Experimenty

V předchozí kapitole byly rozebrána problematika návrhu a implementace neuronové sítě a učícího algoritmu zpětného šíření chyby. Tato kapitola se zaměřuje na zkoumání tohoto algoritmu na konkrétních datových vzorech a sítích.

Za účelem experimentování bylo vytvořeno několik různě velkých množin trénovacích vzorů, jejich přehled je v tabulce 6.1. Tyto soubory byly vytvořeny skriptem napsaným v Pythonu. Soubory *teacher*, *teacher2*, *teacher3* a *teacher4* byly použity k učení sítě. Soubor s větším číslem (soubor *teacher* uvažujme jako *teacher1*) je vždy podmnožinou souboru s číslem větším. Tedy, například soubor *teacher2* obsahuje stejné trénovací vzory jako soubor *teacher*, jen neobsahuje všechny. Soubor *sourcefile* slouží jako množina testovacích vzorů. Tyto vzory mají známou sekundární struktury, nebyly však použity k trénování.

Soubor	Počet vzorů
teacher	3733
teacher2	1059
teacher3	531
teacher4	78
sourcefile	395

Tabulka 6.1: Soubory s trénovacími vzory

6.1 Časová složitost algoritmu

V tabulce 6.2 je ukázána na příkladu několika různě velkých trénovacích množin časová náročnost učení v absolutních číslech. Testy byly prováděny na počítači s dvou-jádrovým procesorem o výkonu 2x1.6GHz, pamětí 1.5GiB a systémem Linux. Je patrná lineární závislost jak na velikosti sítě, tak na počtu trénovacích vzorů.

Soubor	Počet uzlů skryté vrstvy			
	15	30	50	140
teacher	302.6	588.6	973.6	2704.2
teacher2	86.2	167.6	275.4	766.4
teacher3	44.2	84.8	138.4	383.6
teacher4	6.8	13.0	21.0	56.6

Tabulka 6.2: Časová náročnost učení 1 cyklu učení, v ms

6.2 Nastavení konfigurace

Důležitých faktorem při volbě konfigurace sítě je nastavení počátečních hodnot vah a biasů. Špatná volba může výrazně ovlivnit dobu učení a dokonce i míru naučení sítě. Tabulka 6.3 znázorňuje tyto rozdíly. Test proběhl při učení sítě na souboru teacher3 s koeficientem učení $l = 0.05$, defaultním $m = 0.5$ po $c = 100$ cyklech.

Interval	správně predikovaných vzorů z trénovací množiny (v %)									
(-0.2,0.2)	60.83	70.25	68.73	61.39	66.48	68.17	58.38	63.28	68.55	70.06
(-0.5, 0.5)	69.49	67.23	61.96	68.93	69.12	69.12	65.16	70.06	68.74	71.94
(-0.8, 0.8)	64.03	64.78	61.02	63.47	63.84	62.15	62.15	67.42	64.41	51.22
(-1,1)	63.28	57.06	53.68	59.52	51.22	53.86	64.97	57.25	56.50	57.44

Interval	Aritmetický průměr
(-0.2,0.2)	65.611
(-0.5, 0.5)	68.173
(-0.8, 0.8)	62.448
(-1,1)	57.478

Tabulka 6.3: Vliv volby náhodných hodnot při inicializaci sítě

Z tabulek je patrné, že menší rozsah volby náhodných hodnot vah a biasů dává po několika prvních cyklech učení větší přesnost. Nejlepších hodnot síť dosahuje při volbě intervalu $(-0.5, 0.5)$. Tento interval je nastaven jako defaultní v všech sítích, na kterých byly prováděny experimenty.

6.3 Vliv počtu neuronů skryté vrstvy

Následující experiment zkoumá vliv velikosti skryté vrstvy na učící schopnosti neuronové sítě. Jak už bylo řečeno, příliš malá síť nedokáže dostatečně generalizovat trénovací vzory, naopak příliš velkou síť může být obtížné naučit úspěšně predikovat trénovací vzory. V tabulce 6.4 jsou vidět rozdíly v úspěšnosti predikce u jednotlivých testovacích souborů. Je patrná jistá vzrůstající tendence míry naučení sítě při rostoucím počtu neuronů skryté vrstvy.

V tabulce 6.5 je patrné, že 140 neuronů skryté vrstvy je pro tak malou trénovací množinu

Skrytých neuronů	teacher	teacher2	teacher3	teacher4
30	49.611	45.987	48.776	38.462
140	54.567	45.609	47.269	48.718
200	53.657	44.947	47.646	50.000

Tabulka 6.4: Soubor *teacher*, naučení sítě po 10 000 cyklech

Skrytých neuronů	teacher	teacher2	teacher3	teacher4
30	40.155	55.996	77.401	87.190
140	39.754	50.047	64.783	69.231

Tabulka 6.5: Soubor *teacher2*, naučení sítě po 10 000 cyklech

mnoho a síť se učí hůř. V tomto případě je se jedná o *teacher2* To samé lze vypozorovat i pro *teacher3* v tabulkách 6.6 a 6.7. Predikce trénovacích vzorů je úspěšná, ale síť není schopná generalizovat vzory.

6.4 Trénování a testování

Pro predikci sekundární struktury proteinů byla v minulosti úspěšně použita vícevrstvá dopředná neuronová síť s jednou skrytou vrstvou. V literatuře [22] je popsána architektura sítě, kdy na vstup byl přiveden řetězec délky 13 aminokyselin, z čehož plyne, že vstupní síť obsahovala $13 * 20 = 260$ neuronů vstupní vrstvy. Ve skryté vrstvě pak bylo 40 neuronů a na výstupu neurony 3. Topologie sítě tedy byla $260 - 40 - 3$.

Mnou implementovaná síť uvažuje řetězec 7 aminokyselin na vstupu. Pro základní učení a testování jsem zvolil skrytou vrstvu o 30 neuronech. Topologie této sítě se dá tedy zapsat jako $140 - 30 - 3$. Takto nastavené topologie sítě obsahuje $140 * 30 + 30 * 3 = 4290$ vah mezi neurony, které je nutno v každém cyklu pro každý trénovací vzor zvlášť nastavovat. Každý jeden neuron přidaný do skryté vrstvy zvyšuje počet vah o $140 * 1 + 1 * 3 = 143$. Na konfiguraci $140 - 30 - 3$ byla prováděna série trénování pomocí různých trénovacích souborů.

V tabulce 6.8 jsou vypsány výsledky několika experimentů se sítí učenou na souboru *teacher*. Při narůstajícím počtu trénovacích cyklů narůstá míra naučení sítě. Síť má při inicializaci úspěšnost predikce asi 33, což odpovídá náhodnému výběru. Při učení na souboru *teacher* dojde už během prvních několika stovkách cyklů téměř k ustálení naučenosti sítě zhruba na 52 procentech.

Skrytých neuronů	teacher	teacher2	teacher3	teacher4
30	40.155	55.996	77.401	87.190
140	39.754	50.047	64.783	69.231

Tabulka 6.6: Soubor *teacher3*, naučení sítě po 10 000 cyklech

Skrytých neuronů	teacher	teacher2	teacher3	teacher4
30	39.620	54.958	74.953	83.333
200	39.593	49.009	63.842	58.974

Tabulka 6.7: Soubor *teacher3*, naučení sítě po 1 000 cyklech

	Počet	Predikovaný soubor				
	cyklů učení	teacher1	teacher2	teacher3	teacher4	sourcefile
1	50	46.933	43.437	47.081	53.846	32.658
2	100	46.451	39.094	41.243	47.436	33.924
3	500	51.005	46.648	48.399	50.000	33.418
4	1 000	52.933	46.742	47.458	48.718	32.658
5	10 000	49.611	45.987	48.776	38.462	33.924
6	100 000	52.585	46.081	49.906	47.436	34.430

Tabulka 6.8: Síť naučená vzory ze souboru *teacher*

V tabulce 6.9 jsou pak výsledky při učení sítě na souboru teacher2. Úspěšnost predikce vzorů, které byly použity pro učení dosahuje zhruba 63 až 64 procent. Při více než 100000 míra naučení sítě stagnuje nenarůstá procentuální úspěšnost predikce ani na jedné trénovací množině.

Každý řádek výsledků v tabulce vznikal při unikátním spuštění učení, nejedná se o výsledky jednoho procesu učení. Na přesnější určení střední hodnoty míry naučení při určitém počtu cyklů by bylo potřeba několik desítek jednotlivých procesů učení. To je při takovéto časové náročnosti učení obtížné. V tabulce jsou zaznamenány střední hodnoty ze tří procesů učení.

Přesto, že soubor *teacher* obsahuje největší množinu pro trénování, úspěšnost predikce na trénovací množině *sourcefile* nikdy nepřesáhne 35 procent. Menší trénovací množina vzorů, *teacher2* dosahuje úspěšnosti až 41 procent pro testovací množinu *sourcefile*. Jedním z možných vysvětlení tohoto jevu je to, množina trénovacích vzorů *teacher* může obsahovat více stejných řetězců primární struktury, které mají odlišnou očekávanou sekundární strukturu. Tím dochází kde "zmatení"sítě při učení.

	Počet	Predikovaný soubor				
	cyklů učení	teacher1	teacher2	teacher3	teacher4	sourcefile
1	10	35.387	42.682	45.009	53.846	36.456
2	50	36.619	46.930	51.036	55.128	37.215
3	100	38.762	52.502	57.439	60,256	37.975
4	500	41.361	62.040	68.738	79.487	38.734
5	1 000	40.611	62.323	68.550	80.770	40.760
6	10 000	53.335	43.060	45.951	50.000	31.130
7	100 000	41.656	63.740	70.810	82.051	35.190

Tabulka 6.9: Síť naučená vzory ze souboru *teacher2*

V tabulce 6.10 jsou výsledky při učení sítě na souboru teacher3. Tato tabulka ukazuje

to, co už bylo řečeno výše. Menší trénovací množina se snaze naučí predikovat sama sebe. Vzory z testovací množiny *sourcefile* ovšem predikuje hůř.

	Počet	Predikovaný soubor				
	cyklů učení	teacher1	teacher2	teacher3	teacher4	sourcefile
1	10	36.298	41.454	48.023	51.282	34.430
2	50	39.137	50.142	67.043	74.360	37.975
3	100	38.227	49.292	64.595	73.077	35.949
4	500	39.968	54.769	76.083	89.744	36.709
5	1 000	39.620	54.958	74.953	83.333	35.696
6	10 000	40.155	55.996	77.401	87.190	35.696
7	100 000	39.459	55.524	76.648	89.744	40.253

Tabulka 6.10: Síť naučená vzory ze souboru *teacher3*

V tabulce 6.11 jsou výsledky při učení sítě na souboru teacher4. Na této sítí lze velice snadno dosáhnout přeučení sítě. Pro takto malou trénovací množinu lze síť naučit na 100 procent, úspěšnost predikce jiných, než trénovacích dat pak bude velmi malá.

Už při 500 cyklech je úspěšnost predikce trénovací množiny 100 procent, jenže jak je patrné na řádcích 3 a 4, s rostoucím počtem cyklů neroste přesnost predikce pro testovací množinu, naopak dokonce klesá, což je právě způsobeno přeučením sítě pro trénovací množinu.

	Počet	Predikovaný soubor				
	cyklů učení	teacher1	teacher2	teacher3	teacher4	sourcefile
1	50	34.476	37.583	40.866	82.051	35.949
2	100	34.717	38.904	42.185	84.615	37.468
3	500	34.610	39.282	44.821	100.00	35.949
4	1 000	35.709	37.205	42.373	100.00	30.378

Tabulka 6.11: Síť naučená vzory ze souboru *teacher4*

Při bližším prozkoumání souborů s výsledky je vidět, že mnohdy, hlavně u větší trénovací množiny s větším počtem neuronů skryté vrstvy, vychází stejná hodnota pro 2 a více výstupů ze sítě, nejčastěji hodnota 1, což mimo jiné může vzniknout zaokrouhlením výsledku aktivační funkce. Tento fakt značně snižuje celkovou přesnost predikce sítě, protožw program neví, kterou ze stejných hodnot má interpretovat jako správnou. Hodnoty vyhodnocuje zleva od neuronu 0 po neuron 2 výstupní vrstvy, což odpovídá pořadí H, S, T. Mají li tedy neurony 0 a 1 výstupní vrstvy stejné reálný stav na konci predikce, program vybere neuron H jako správný, což ovšem nemusí být pravda.

6.5 Možnosti dalšího rozvoje

Existují metody, kterými lze snížit časovou náročnost učení neuronové sítě. Jednou z těchto metod je např. užití Momenta, blíže popsaného v kapitole 5 a použitého při implementaci aplikace. Mezi pokročilejší metody patří např. tzv. *škálování* (*scaling*)[23], jehož použití by mohlo zefektivnit učení.

U predikce sekundární struktury proteinů, kde existuje obrovské množství všech možných vstupů a nesrovnatelně malé množství známých vstupů je velkým problémem možnost přeučení nebo nedoučení sítě(také popsáno v kapitole 5). Někdy je proto výhodné pro trénování používat dvě disjunktní podmnožiny, množinu trénovacích vzorů i množinu testovacích vzorů. Na trénovacích vzorech síť trénovat a na testovacích vzorech počítat chybu sítě (momentálně je testovací množina používána pouze pro predikce, až po skončení fáze učení). Jako kritérium ukončení učení pak volíme situaci, kdy součet chyb sítě dosáhne minima.

Další technika, která ovlivňuje učení je způsob úpravy sítě. Literatura[22] uvádí, úprava vah pro každý tréninkový vzor bývá v případě predikce sekundární struktury proteinu efektivnější, než tzv. *dávkové učení*(*batch-learning*), kde jsou váhy nastaveny až po celém tréninkovém cyklu, tj. po jednom průchodu všech tréninkových vzorů sítí. Užitím dávkového učení by se teoreticky zrychlilo učení, díky tomu, že by se váhy nenastavovali pro každý vzor. Počítat a pamatovat v nějaké proměnné by se ale museli stejně.

Samotná implementace programu není zcela efektivní. Uložení každého neuronu do zvláštní struktury a následný a přístup a modifikace může být časově náročná. Určitě existují efektivnější konstrukce. Po dosažení kratšího času potřebného k učení by bylo dobré některé výše zmíněné experimenty provést víckrát přesněji tak zjistit střední hodnotu naměřených hodnot, především u experimentů s trénováním souboru *teacher* a také experimenty s větším množstvím neuronů skryté vrstvy. V současném stavu, kdy trénování středně velké sítě souborem vzorů *teacher* při dostatečným počtu cyklů trvá řádově dny, je veškeré experimentování časově velmi náročné.

Kapitola 7

Závěr

Tato práce se pokouší přiblížit problematiku predikce sekundární struktury proteinů. Vysvětluje pojmy jako protein a jeho struktury. Ale především se zabývá predikci sekundární struktury, konkrétně pomocí neuronových sítí. V první části práce jsou vyjmenovány a porovnány různé principy z oblasti neuronových sítí. Od nejjednoduššího perceptronu inspirovaného lidským neuronem, přes složitější sítě perceptronů po asociativní neuronové sítě.

Obsáhlá část práce se pak z teoretického hlediska věnuje algoritmu zpětného šíření chyby v dopředné neuronové síti.

Cílem této práce bylo navrhnout neuronovou síť, která by byla schopná s určitou pravděpodobností predikovat sekundární strukturu proteinu, na základě jeho struktury primární a otestovat její vlastnosti. Velká část této práce je proto zaměřena právě na analýzu a rozbor této problematiky. Neuronová síť navržená pro predikci sekundární struktury proteinů využívá učícího algoritmu zpětného šíření chyby (backpropagation algoritmus), který používá asi 80 procent všech aplikací, zabývajících se predikcí. Existují různé variace algoritmu backpropagation, které s menším či větším úspěchem ovlivňují učící schopnosti neuronové sítě. Některé z nich jsou v této práci popsány a využity. Jako součást této práce vznikla webová aplikace, která má za úkol demonstrovat predikci sekundární struktury proteinu. Na čtyřech vybraných, předem naučených sítích lze predikovat uživatelem zadaný zdrojový soubor a prohlédnout si výsledky. Neuronová síť implementovaná v rámci této práce je velmi jednoduchá, neřeší některé aspekty učení, které by mohli zásadně, ovlivnit schopnost učení sítě, viz. 6.5. Proto její schopnost učení se nedosahuje takových výsledků. Ale jak bylo nastíněno, prostor k rozvoji zde je, stejně jako v celém oboru neuronových sítí a umělé inteligence.

Literatura

[1] Molecular Biology Web Book. [online], [cit. 8.4.2012].
URL http://www.web-books.com/MoBio/Free/Chap2.htm

[2] Overfitting. [online], [cit. 25.4.2012].
URL http://www.willamette.edu/~gorr/classes/cs449/overfitting.html

[3] Uniprot. [online], [cit. 7.4.2012].
URL http://www.uniprot.org

[4] Alberts, B.; Bray, D.; Johnson, A.; aj.: *Základy buněčné biologie*. Espero Publishing, 2005, iSBN-10: 80-902906-2-0.

[5] Anderson, J. A.: A memory storage model utilizing spatial correlation functions. *Kybernetik*, , č. 5, 1968: s. 113–119.

[6] Fahlman, S. E.: Faster-learning variations on back-propagation: An empirical study. In *Proceedings of the 1988 Connectionist Models Summer School*, editace D. S. T. et al., Morgan Kaufman, 1988.

[7] Fahlman, S. E.; Lebiere, C.: The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems II*, editace D. S. Touretzky, 1990, s. 524–532.

[8] Han, J.; Kember, M.: *Data Mining: Concepts and Techniques. Second Edition.* Elsevier Inc., 2006, iSBN 1-55860-901-3.

[9] Hopfield, J. J.: Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Science*, ročník 79, 1982, s. 2554–2558.

[10] Hu, Y. H.: *Handbook of Neural Network Signal Processing.* CRC Press, Inc., 2000, iSBN 0849323592.

[11] Ladislav Metelka, Č.: Emapirická Objektivní Analýza Pole Slunečního Svitu Pomocí Neuronových Sítí. [online], [cit. 14.4.2012].
URL http://old.chmi.cz/HK/OK/PUBLIKACE/SSV/SSV.htm

[12] McCulloch, W. S.; Pitts, W.: A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, , č. 5, 1943: s. 115–133.

[13] von Neumann, J.: The general and logical theory of automata. In *Cerebral Machanisms in Behavior*, editace L. A. Jeffress, Willey, New York, 1951, s. 1–41.

[14] von Neumann, J.: Probabilistic logics and the synthesis of reliable organisms from unreliable components. In *Automata Studies*, ročník 34, editace C. E. Shannon; J. McCarthy, Princeton University Press, Princeton NJ, 1956, s. 43–89.

[15] Volná, E.: Asociativní neuronové sítě. *Automatizace*, , č. 11, 2008.

[16] Widrow, B.: Generelization and information storage in networks of ADALINE 'neurons'. In *Self-organazing systems*, editace M. C. Yovits; G. T. Jacobi; G. D. Goldstein, Spartan Books, Washington DC, 1962, s. 435–361.

[17] Widrow, B.; Hoff, M. E.: Adaptive switching circuits. In *IRE WESCON Convention Records*, ročník 4, 1960, s. 96–104.

[18] Wikipedia: Neuronová síť. [online], [cit. 9.4.2012].
URL http://cs.wikipedia.org/wiki/Neuronov%C3%A1_s%C3%AD%C5%A5

[19] Wikipedia: Peptidová vazba. [online], [cit. 4.4.2012].
URL http://cs.wikipedia.org/wiki/Peptidová_vazba

[20] Wikipedia: Protein. [online], [cit. 8.4.2012].
URL http://en.wikipedia.org/wiki/Protein

[21] Zbořil, F. V.: Soft Computing. [online], [cit. 20.4.2012].
URL https://www.fit.vutbr.cz/study/courses/SFC/private/10sfc_2.pdf

[22] Zvelebil, M.; Baum, J.: *Understanding Bioinformatics*. Garland Science, London, 2007, iSBN 978-0815340249.

[23] Šíma, J.; Neruda, R.: *Teoretické otázky neuronových sítí*. Matfyzpress, 1996, iSBN 80-85863-18-9.

Příloha A

Obsah CD

- zdrojové soubory programu pro predikci v C++
- datové soubory potřebné pro práci s programem
- Pythonovský skript pro tvorbu trénovacích souborů
- webová aplikace pro demostraci programu
- technická zpráva ve formátu PDF
- zdrojové soubory k technické zprávě

Příloha B

Manuál

B.1 Programu pro predikci sekundární struktury proteinů

Program pro predikci sekundární struktury proteinů (dále jen program) je napsán v jazyce C++ v prostředí Linux. Jedná se o konzolovou aplikaci, tedy aplikaci spouštěnou z příkazová řádky. Pro přeložení zdrojových kódů slouží soubor *Makefile* umístěný v kořenovém adresáři programu. Pro překlad je nutné mít nainstalován překladač jazyka C++, *g++* a program *make*. Makefile obsahuje rovněž skript pro vytvoření datových souborů vyžadovaných k učení a predikci. K tomu je potřeba mít nainstalovaný program *Python*.

Použití *make* v kořenovém adresáři programu:

- *make* - přeloží spustitelný program, jeho název je *prog*
- *make clean* - vymaže soubory vzniklé při překladu
- *make transform* - provede pythonovský skript, který prohledá složku *src* a upraví textové soubory stažené z databáze Uniprot([3]) do podoby řetězců, které příjme program, tyto řetězce uloží do souboru *sourcefile*. Totéž provede se složkou *teach* a data uloží do souboru *teacher*.
- *make transclean* - smaže soubory *sourcefile* a *teacher*
- *make stats* - vypíše statistiky predikce pro soubor *results*

Program *prog* lze spustit s několika parametry:

- -h: tiskne nápovědu
- -t: soubor s daty k učení sítě, default: teacher
- -s: zdrojový soubor pro predikci, default: sourcefile
- -d: výstupní soubor s predikovanými daty, default: results
- -l: koeficient učení, default 0.05
- -c: maximální počet cyklu predikce, default 100
- -n: počet neuronu skryte vrstvy, POZOR při používání zároveň se –save, při –load nutno opět zadat -n, default 30

- -e: požadovaná efektivita sítě při predikci v intervalu (0,1], default: 0.8
- -m: momentum pro trénování v intervalu (0,1), default: 0.5
- –save: uložení konfigurace sítě do souboru, default: network
- –load: načtení konfigurace sítě ze souboru, default: network

B.2 Webová aplikace

Pro spuštění webové aplikace na localhostu je nutný běžící webserver s nastavenou cestou do adresáře s aplikací.

V aplikaci je na výběr několik naučených sítí, na kterých lze predikovat. Pro predikci lze bud použít textový souboru ve stejném formátu, jaký přijímá program pro predikci, případně lze zadat jeden řetězec ručně.

Výstupem je pak textový soubor přístupný přes odkaz na webu.