Masarykova univerzita
Institut biostatistiky a analýz

VĚDECKÉ VÝPOČTY V BIOLOGII A BIOMEDICÍNĚ

Prof. RNDr. Jiří Hřebíček, CSc.
Doc. Ing. Jan Žižka, CSc.

Tento učební text vznikl za podpory MŠMT v rámci projektu FRVŠ č. 2588/2007
„Rozvoj výuky předmětu Vědecké výpočty v biologii a biomedicíně“

Prosinec 2007

Obsah

KAPITOLA 1 ÚVOD (JIŘÍ HŘEBÍČEK, JAN ŽIŽKA) 4

1.1 DATA, INFORMACE, ZNALOST 4
1.1.1 Data 4
1.1.2 Metadata 5
1.1.3 Strukturovaná data 5
1.1.4 Databáze 5
1.1.5 Informace 6
1.1.6 Metainformace 6
1.1.7 Znalost 6
1.1.8 Metaznalost 7
1.1.9 Poznatek 7
1.1.10 Učení 9
1.2 NEJISTOTA, NEURČITOST, PŘIBLIŽNOST, FUZZY MNOŽINY A FUZZY LOGIKA 9
1.2.1 Nejistota 10
1.2.2 Přibližnost 10
1.2.3 Nejednoznačnost a její odstraňování 10
1.2.4 Fuzzy množiny 12
1.2.5 Fuzzy logika 12
1.2.6 Fuzzy množinové operace 14
1.2.7 Fuzzy expertní systémy 15
1.2.8 Defuzzifikace 15
1.2.9 Fuzzy čísla 15
1.3 POČÍTAČ, SOFTWARE, HARDWARE, KOMUNIKACE, INTERNET 16
1.3.1 Počítač 16
1.3.2 Software 16
1.3.3 Počítačový software 16
1.3.4 Aplikační software 17
1.3.5 Open source software 17
1.3.6 Svobodný software 17
1.3.7 Freeware 18
1.3.8 Middleware 18
1.3.9 Hardware 18
1.3.10 Komunikace 18
1.3.11 Počítačová síť 19
1.3.12 Internet 19
1.3.13 Web 19
1.3.14 Distribuovaný výpočet 19
1.4 VĚDECKÉ VÝPOČTY A VÝPOČETNÍ VĚDA 19
1.4.1 Vědecké výpočty 20
1.4.2 Počítačová simulace 20
1.4.3 Klasifikace vědeckých výpočtů 22

KAPITOLA 2 STRUČNÁ HISTORIE VĚDECKÝCH VÝPOČTŮ/VÝPOČETNÍ VĚDY (JIŘÍ HŘEBÍČEK) 24

2.1 VÝVOJ V ZAHRANIČÍ 24
2.1.1 Virtuální superpočítač World Community Grid 25
2.1.2 BOINC - Berkeley Open Infrastructure for Network Computing 26
2.2 VÝVOJ V ČESKÉ REPUBLICE 27
2.2.1 Projekt EGEE - Enabling Grids for E-sciencE 27
2.2.2 Superpočítačové Centrum Brno 28

KAPITOLA 3 STRUČNÝ PŘEHLED TECHNOLOGIÍ PRO VĚDECKÉ VÝPOČTY V BIOLOGII A BIOMEDICÍNĚ (JIŘÍ HŘEBÍČEK) 29

3.1 VĚDECKÉ VÝPOČTY V BIOINFORMATICE 29
3.2 NEJPOUŽÍVANĚJŠÍ BIOLOGICKÉ DATABÁZE A FORMÁTY DAT V BIOINFORMATICE 29
3.2.1 EMBL 30
3.2.2 GenBank 30
3.2.3 DDBJ 30

3.2.4 Swiss-Prot ... 30
3.2.5 PIR ... 31
3.3 POČÍTAČOVÁ ANALÝZA VELKÝCH SOUBORŮ NUKLEOTIDOVÝCH SEKVENCÍ ... 31
3.3.1 Vyhledávání sekvencí ... 32
3.4 VĚDECKÉ VÝPOČTY V BIOMEDICÍNĚ ... 33
3.4.1 Vědecké výpočty v rámci projektu EGEE ... 35
3.4.2 Biomedicínské výpočty ve výzkumu ... 36

KAPITOLA 4 VĚDECKÉ VÝPOČTY S VYUŽITÍM MAPLE (JIŘÍ HŘEBÍČEK) ... 38

4.1 MAPLE ... 39
4.1.1 Uživatelské rozhraní v Maple ... 39
4.1.2 Maple aplikační centrum ... 41
4.1.3 Doplňující software spolupracující s Maple ... 43
4.1.3 Příklad použití Maple ... 44

KAPITOLA 5 UMĚLÁ INTELIGENCE A INTELIGENTNÍ AGENTI (JAN ŽIŽKA) ... 47

5.1 MODERNÍ POHLED NA POJEM UMĚLÁ INTELIGENCE ... 47
5.2 INTELIGENTNÍ AGENTI ... 50
5. 3 KOMUNIKACE AGENTŮ ... 52
5.4 KOMPONENTY KOMUNIKACE ... 53
5.5 DVA MODELY KOMUNIKACE ... 54

KAPITOLA 6 DOLOVÁNÍ Z DAT (JAN ŽIŽKA) ... 55

6.1 ÚČEL DOLOVÁNÍ Z DAT ... 55
6.2 ZÁKLADNÍ PROSTŘEDKY PRO DOLOVÁNÍ Z DAT ... 55
6.3 SOFTWAROVÝ SYSTÉM DOLOVÁNÍ Z DAT WEKA ... 56

KAPITOLA 7 STROJOVÉ UČENÍ (JAN ŽIŽKA) ... 64

7.1 INDUKTIVNÍ STROJOVÉ UČENÍ ... 64
7.2 ŘÍZENÉ A NEŘÍZENÉ UČENÍ ... 65
7.3 PODOBNOST ... 65
7.4 VYHODNOCENÍ VÝSLEDKŮ STROJOVÉHO UČENÍ ... 66
7.5 ZÁKLADNÍ ALGORITMY INDUKTIVNÍHO STROJOVÉHO UČENÍ ... 68
7.5.1 Rozhodovací stromy ... 68
7.5.2 Nejbližší soused ... 70
7.5.3. Umělé neuronové sítě ... 70
7.5.4 Genetické algoritmy ... 74
7.5.5 Bayesovské učení ... 79
7.5.6 Roje a mravenci ... 81

KAPITOLA 8 LITERATURA ... 84

Kapitola 1
Úvod
(Jiří Hřebíček, Jan Žižka)

Učební texty v tomto našem společném díle se zabývají v první kapitole přehledovým úvodem do terminologie oblastí souvisejících s vědeckými výpočty a postupně popisují studovanou problematiku, možnosti jejího řešení, snaží se vysvětlit vědecké výpočty a výpočetní věda v současném pojetí. Druhá kapitola je věnována historii vědeckých výpočtů a ve třetí kapitole jsou shrnuty technologie a jejich aplikace do oblasti biologie a biomedicíny. Ve čtvrté kapitole je stručně objasněn systém Maple, který je provozován Masarykově univerzitě v rámci multilicence. Jsou zde popsány některé jeho vlastnosti a na jednoduchém příkladu ukázáno jeho využití pro vědecké výpočty v biologii. V dalších třech kapitolách jsou uvedeny aplikace metod umělé inteligence, dolování dat i metody strojového učení pro oblasti biologie a biomedicíny.

Učební texty rovněž odkazují na současnou kvalitní výchozí literaturu, která je velmi rozsáhlá a popisuje problematiku až do detailů, což zde z důvodu rozsahu těchto učebních textů není realizovatelné. Umožňují však seznámit se s možnostmi současných vědeckých výpočtů dostupných pro oblasti biologie a biomedicíny na internetu a vyzkoušet si, jak adaptivní učení ve skutečnosti funguje, jak lze dolovat z dat znalost a jak ověřovat kvalitu znalosti; jsou uvedeny i odkazy na velmi kvalitní, uživatelsky pohodlný systém, který je k dispozici formou freeware zdarma, je rozvíjen na novozélandské univerzitě a používán k serióznímu výzkumu po celém světě.

K objasnění některých částí učebního textu jsou připojeny příklady, včetně ukázky vyhledávání z genetických databází a dolování znalosti z dat, která lze snadno získat z veřejně přístupného zdroje a na nichž lze vyzkoušet, jak ve skutečnosti funguje, jaká jsou pozitiva a negativa popisovaného přístupu a získat inspiraci pro jejich samostatné používání. Existuje již i velmi dobrý svobodný (či open source) software/freeware, který eliminuje ztrátu času věnovanou programování algoritmů (jak to umožňuje současné verze Maple), a namísto toho se zaměřit na vlastní řešení aktuálních problémů ve studiu matematické biologie či příbuzných oborů. Situace ovšem není ideální, jak dosvědčuje např. [Sonnenburg et al., 2007], v oblasti strojového učení, ale k postupnému zlepšování určitě dochází a nové přínosy a objevy systematicky posouvají oblast vědeckých výpočtů neustále dopředu v souladu s velmi naléhavými potřebami praxe v řadě mnoha oborů.

V následujících podkapitolách objasníme některé základní terminologické pojmy, se kterými budeme dále pracovat a které souvisejí s oblastí vědeckých výpočtů.

1.1 Data, informace, znalost

V tomto odstavci popíšeme pojmy, které souvisejí se základními pojmy data, informace, znalost a učení.

1.1.1 Data

Data[1] jsou zaznamenané údaje o nějakých skutečnostech světa (fakta[2]), které jsou schopná přenosu, uchování, interpretace či zpracování [Sklenák a kol**.,** 2001]. Data mohou mít různou

[1] Slovo data pochází z latinského slovesa *dato, -are, -avi, -atum*, tj. *dávat*, a podstatná jména z něj odvozená znamenají *dané, danost, údaj*. V češtině se výraz *data* používá pro množné číslo, jednotné číslo je *údaj*.
[2] skutečnost; událost, děj, skutek, který se opravdu stal a není vymyšlený

formu: celá nebo reálná čísla, binární hodnoty, nominální (nečíselné, vyjmenované hodnoty), texty, obrázky, grafy, mapy, zvuky, videa atd. a jejich reprezentaci v počítačovém systému, přičemž popis skutečností může obecně zahrnovat všechny výše zmíněné formy, včetně popisu dat prostřednictvím přirozeného jazyka. V počítači jsou data zaznamenávána v binární formě (tj. posloupnost nul a jedniček), přičemž každá forma dat má předepsaný způsob (formát) zápisu v počítači. Data mohou být získávána různým způsobem, např. senzory (kamery družic apod.) i lidmi (pozorováním a záznamem počtu druhů společenstva v nice). Primární data (např. prvotní naměřená data) mají tzv. hrubou formu, související s neurčitostmi nebo chybami měření, které je nutno v procesu předzpracování eliminovat a převést na formu vhodnější pro další zpracování. Tzn., že primární data mohou obsahovat např. různý druh *šumu*, zkreslující opravdové hodnoty. Odstraňování tohoto šumu je důležitou součástí předzpracování dat.

1.1.2 Metadata

Metadata[3] jsou strukturovaná data, která nesou informace o primárních datech. [Ressler a kol., 2006]. Pojem metadata je používán především v souvislosti s elektronickými zdroji a vztahuje se k datům v nejširším smyslu slova (datové soubory, texty, obrázky, grafy, hudba aj.). Funkce metadat je popisná, selekční a archivační. V souvislosti s těmito funkcemi se rozlišují metadata pro účely popisu, správy, právních nároků, technické funkčnosti, užití a archivace. Technická metadata jsou metadata vytvořená pro počítačový systém nebo vyrobená počítačovým systémem, která uvádějí, jak se systém nebo jeho obsah chová nebo co požaduje, aby mohl být provozován (protokol HTTP, parametry HW). Administrativní metadata jsou používaná pro řízení a správu informačních zdrojů, např. informace o umístění, údaje o době vzniku a poslední modifikaci, elektronický podpis aj.

1.1.3 Strukturovaná data

Strukturovaná data jsou údaje logicky uspořádané ve struktuře jednotlivých datových položek podle určitého systému, který určuje, jak má být souhrn datových objektů strukturován. Obecné typy struktury dat zahrnují např. pole, soubor, záznam, tabulku, strom atd. Každá struktura dat určuje organizaci (ukládání) dat tak, aby vyhovovala určitému účelu, umožňovala k uloženým datům potřebný přístup, popř. další zpracování. V počítačovém programování se může struktura dat zvolit nebo navrhnout tak, aby data mohla být dále zpracovávána různými algoritmy.

1.1.4 Databáze

Databázi (banku dat) lze obecně popsat jako systém sloužící k modelování objektů a vztahů reálného světa (včetně abstraktních nebo fiktivních) prostřednictvím digitálních dat uspořádaných tak, aby s nimi bylo možné efektivně manipulovat, tj. rychle vyhledat, načíst do paměti a provádět s nimi potřebné operace – zobrazení, přidání nových nebo aktualizace stávajících údajů, matematické výpočty, uspořádání do pohledů a sestav apod. Základními prvky databáze jsou data a program pro práci s nimi. Datový obsah tvoří množina jednotně strukturovaných dat uložených v paměti počítače nebo na záznamovém médiu, jež jsou navzájem v určitém vztahu a tvoří určitý celek z hlediska obsažených informací; data jsou přístupná výhradně pomocí speciálního programového vybavení - systému řízení báze dat. Podle typu obsažených dat se rozlišují databáze textové (mezi nimi lze dále vyčlenit databáze pouze s texty, bibliografické, referenční, faktografické), numerické, obrazové, multimediální. Podle způsobu práce uživatele s daty se rozlišují databáze umožňující zápis dat (např. firemní transakční systémy, modul katalogizace knihovnického systému) a databáze umožňující

[3] Metadata (z řeckého meta- = mezi, za + latinského data = to, co je dáno) jsou strukturovaná data o datech.

pouze vyhledávání a čtení dat (např. databáze v databázových centrech, OPAC, datové sklady). Přístup může být on-line i off-line. Pamětí počítače se rozumí jak paměť externí (např. disky), tak interní (RAM), v principu libovolná. Organizace dat může být různá, od jednoduchého sekvenčního uspořádání, které nemá např. vysoké režijní požadavky na paměť, ale vyhledávání konkrétního údaje je pro rozsáhlejší data pomalé, až po pokročilé systémy umožňující vyhledávání přímým přístupem, kde se za rychlost platí vysokou režií prostorovou (např. statické a dynamické indexové soubory, hashing, aj., což může běžně zabrat více místa, než data samotná).

1.1.5 Informace

Pojem **informace**[4] je definován v normě ISO/IEC 2382-1:1993: „Informační technologie – Slovník - Část 1: Základní pojmy“, jako “*Poznatek (znalost) týkající se jakýchkoliv objektů, např. faktů, událostí, věcí, procesů nebo myšlenek včetně pojmů, které mají v daném kontextu specifický význam*“. V nejobecnějším slova smyslu se informací chápe údaj o reálném prostředí, o jeho stavu a procesech v něm probíhajících. Informace snižuje nebo odstraňuje neurčitost systému (např. příjemce informace); množství informace je dáno rozdílem mezi stavem neurčitosti systému (entropie), kterou měl systém před přijetím informace a stavem neurčitosti, která se přijetím informace odstranila. Z našeho pohledu informace je taková část dat, která je z hlediska uživatele *relevantní* pro řešení specifického problému. V datech mohou být zjevné i zcela skryté nějaké spojitosti (resp. závislosti, pravidelnosti) mezi určitými hodnotami a objevením těchto spojitostí jsou nalezeny specifické vzory, které mohou dále vést k získání nové, v datech doposud ukryté *znalosti*. Obecně data ukrývají velké množství nejrůznějších, potenciálně užitečných informací. Objevení správné informace je převážně obtížný úkol, který nemá jednoznačné standardní řešení. Různé specifické problémy mohou používat jako zdroj tatáž data, ale relevantní části mohou být zcela (nebo částečně) odlišné – různé problémy mohou vyžadovat různou informaci pro řešení.

1.1.6 Metainformace

Metainformace je informace, která je v transformačním vztahu k jiné informaci. Metainformace je nástrojem popisu vztažné informace. Slouží jako prostředek vybavení obsahu popisované informace ve vyhledávacích systémech.

1.1.7 Znalost

Znalost je schopnost člověka nebo jakéhokoli jiného inteligentního systému uchovávat, komunikovat a zpracovávat informace do systematicky a hierarchicky uspořádaných znalostních struktur [Ressler a kol., 2006]. Znalost je charakterizována schopností abstrakce a generalizace dat a informací. Znalost je rovněž to, co jednotlivec vlastní (ví) po osvojení dat a informací a po jejich začlenění do souvislostí, tj. znalost je získána *zobecněním nalezené informace*. Znalost je výsledek poznávacího procesu, předpoklad uvědomělé činnosti. Kvalitní znalost umožňuje například přesné předpovědi týkající se vlastností dosud neznámých, v budoucnu teprve získaných či očekávaných dat (predikce), nebo zpětného doplnění chybějících hodnot (regrese). Automatizované získávání znalosti z rozsáhlých dat patří k velmi důležitému a potřebnému odvětví informatiky. Získaná znalost může mít velmi odlišnou formu, která se liší také ve srozumitelnosti. Stroj může z dat vydolovat správnou znalost, která pak velmi dobře funguje např. pro predikci, ale nemusí být zřejmé, proč stroj

[4] V latině, která dala světu termín informace, se sloveso *informo, -are, -avi, -atum* používalo k vyjádření následujících činností: *formovat, utvářet, vzdělávat, upravovat, podávat představu (pojem) něčeho*. Podstatné jméno *informatio, -onis, f.* pak označovalo *představu, obrys, výklad, poučení*. V dnešním jazyce však je význam slova posunut a my už nevystačíme s jeho interpretací v tom smyslu, jak mu rozuměli staří Římané

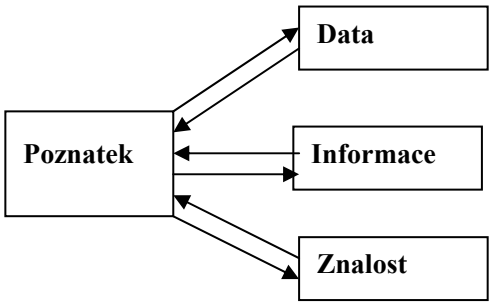
konkrétním způsobem odpovídá – typickým příkladem jsou umělé neuronové sítě. Naopak pravidla představují formu mnohem srozumitelnější.

1.1.8 Metaznalost

Metaznalost vyžaduje efektivní organizaci a správu různě reprezentované různé znalosti, aby bylo možno rozhodnout, který typ znalosti aplikovat na konkrétně řešený problém. Někdy lze na jeden problém použít několik znalostí, což odpovídá konzultaci několika expertů, kteří se ve všem nemusejí shodovat. Obdobně lze použít více znalostí na problém rozdělený na části, kde každý odborník se vyzná ve své specifické části, konečný výsledek lze získat např. hlasováním. Příkladem může být tzv. *boosting* metoda (*posilovaná* metoda), kdy lze generovat třeba více rozhodovacích stromů, kde se každý strom dopouští malé chyby pouze na "své" části dat, které rozumí, zatímco na ostatních částech poskytují lepší výsledky jiné stromy (a naopak). Pro výsledek klasifikace se pak uvažuje výsledek "hlasování", pro numerickou predikci průměr, a to jako mínění skupiny navzájem se doplňujících expertů.

1.1.9 Poznatek

Výsledek procesu poznávání skutečnosti je poznatek. Jde o souhrn pojmů a představ o světě.

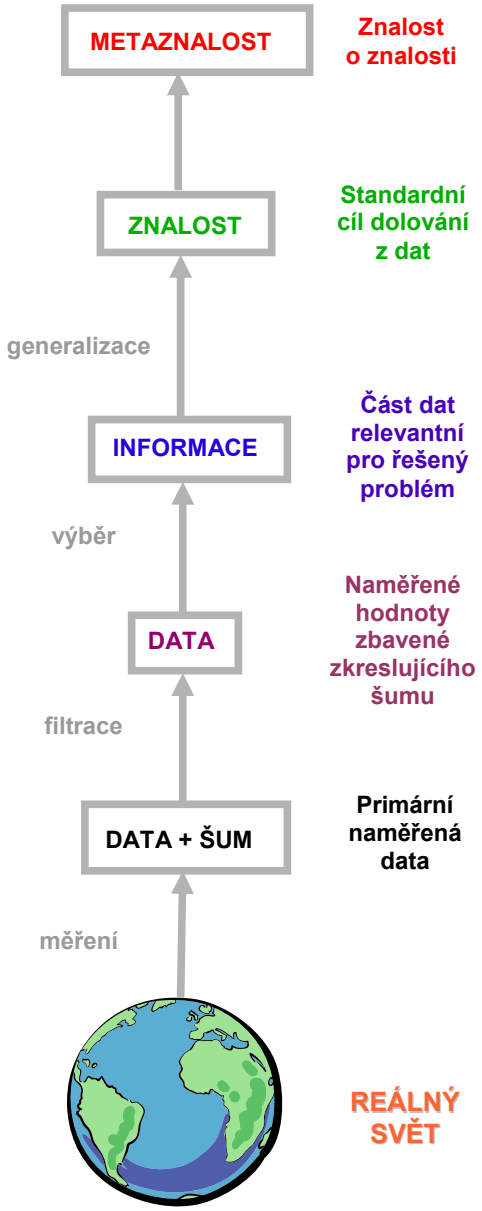


Obrázek č. 1.1: Vztahy mezi poznatkem, daty, informacemi a znalostí

Příklad: Uvažujme posloupnost 26 číslic

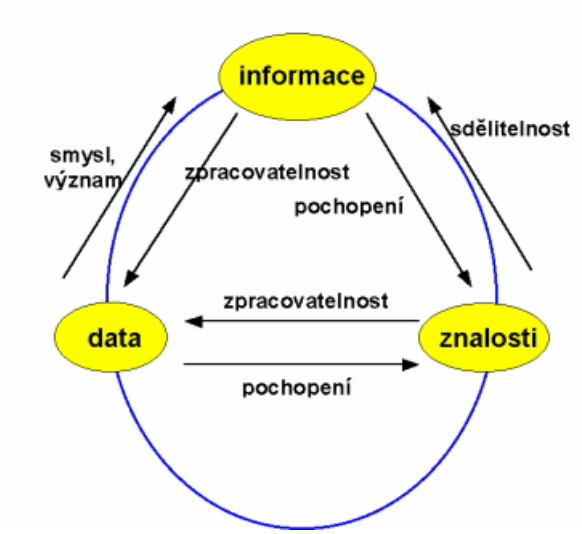
- 12345678901234567890 … *primární data*
- máme-li poznatek, že posloupnost obsahuje smysluplné údaje (nejen šum), pak jde o data … *data*
- předpokládejme, že máme znalost, jak tato data převést na informaci … *informace*
  Např.:
  1. rozdělíme posloupnost na dvojice: 13 90 76 … 15 (předpokládaná struktura)
  2. vynecháme každé číslo menší než 32: 90 76 65 … 43 (odstranění známého šumu)
  3. nahradíme zbývající čísla ASCII znaky (změna reprezentace dat):
  ZLATO 438 + cena stoupá (relevantní údaje – informace)
- Máme-li formalizovanou znalost reprezentovanou pravidlem:
  „IF zlato je levnější než 500 a cena stoupá, THEN nakupuj zlato",
  pak informaci můžeme využít.

Na obr. č. 1.2 jsou znázorněny hierarchické vztahy mezi reálným světem, daty, informací, znalostí a metaznalostí. Přechody mezi jednotlivými fázemi jsou uskutečňovány příslušnými procesy.



Obrázek č. 1.2: Vztahy mezi reálným světem, daty, informací, znalostí a metaznalostí

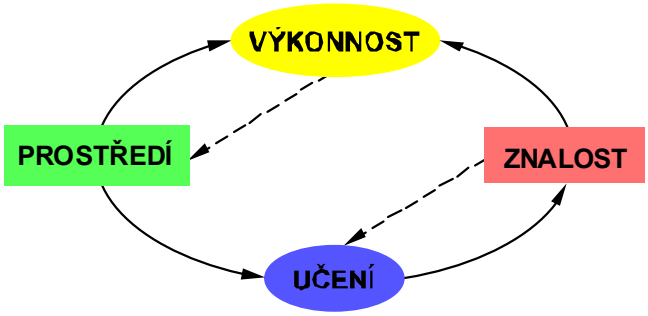
Výše uvedené termíny informace, data a znalosti lze v běžném hovoru považovat za synonymní (jsou natolik příbuzné, že je prakticky nelze definovat jinak než pomocí jich samých), pro pochopení podstaty informace stojí však zato, zamyslet se nad jejich odlišnostmi, které jsou uvedeny na obr. č. 1.3.



Obrázek č. 1.3: Vztahy mezi daty, informacemi a znalostmi

1.1.10 Učení

V odborné literatuře lze najít další definice pojmů *data, informace, znalost* a pojmu **učení**. Zejména pro *znalost* a *učení* neexistuje jasná, jednoznačná a obecně přijímaná (jediná) definice, protože vždy lze k příkladům odpovídajícím určité definici najít protipříklady. Znalost, získaná hierarchicky zobrazeným procesem dolování a učení se z dat, používá neformální a praktickou, byť ne zcela jednoznačnou definici [Langley, 1996], vztahující se k tématu probíranému v tomto textu: *Učení* je zlepšování *výkonnosti* v určitém *prostředí* pomocí získávání *znalostí* vycházejících z toho, co je známo a co je k dispozici v daném prostředí.



Obrázek č. 1.4: Interakční vztahy mezi prostředím, učením, znalostí a její efektivnosti

Použití získané znalosti záleží na potřebách uživatele a aplikace. Vztahy mezi získanou znalostí, učením, výkonností a prostředím ilustruje obr. 1.4. *Znalost* se získává *učením* a může zpětně učení ovlivňovat – např. s větší získanou znalostí lze upravit metodu dalšího učení. *Výkonnost* měří kvantitativně dosaženou znalost (např. přesnost predikce, srozumitelnost aj.). *Prostředí* představuje externí parametry ovlivňující učení a aplikovatelnost znalosti, avšak získaná znalost může prostředí zpětně ovlivňovat – to může být i cílem dobývání znalosti z dat poskytovaných prostředím. Zvýrazněné pojmy nelze tedy v procesu dolování z dat chápat izolovaně.

1.2 Nejistota, neurčitost, přibližnost, fuzzy množiny a fuzzy logika

Současné procesy vědeckého poznávání, snad v reakci na úzkou specializaci a dosaženou vysokou úroveň poznání řady vědeckých oborů, nyní hledají a také nacházejí svůj další

rozvojový potenciál v mezioborových vztazích, které přinášejí mnohdy nová a překvapující poznání, jež mají předpoklady se stát novým paradigmatem vědy 21. století. Růst lidského poznání tak člověku přináší stále větší možnost osvobodit se od závislosti na probíhajících procesech tím, že může s vysokou přesností předvídat jejich výsledky, a vyhnout se tak různým nepříznivým situacím, ale také může tyto procesy usměrňovat a měnit.

V okolním světě, a to jak v různých vědních oborech, tak i v každodenním životě, tedy existují procesy, u nichž i při současném vysokém stupni lidského poznání je často nemožné vzít v úvahu, resp. kontrolovat všechny podmínky, za nichž daný proces probíhá. Realizace takového souboru podmínek pak vede k *nejistému* výsledku, který nemůžeme předem jednoznačně určit, protože podléhá působení různě velkého množství často drobných, ne úplně nebo zcela zjistitelných vlivů, které jsou příčinou toho, že i při opakované realizaci tohoto souboru podmínek dostaneme různé, *náhodné* výsledky.

Proto další důležité pojmy, které zavedeme souvisejí již s vědeckými výpočty, metodami zpracování dat a informací v počítačových systémech budou nejistota, přibližnost, fuzzy množiny a operace s nimi, fuzzy čísla a fuzzy logika.

1.2.1 Nejistota

Libovolně naměřená data se mohou vyznačovat nejistotou, zda senzorem udávaná hodnota je jediná možná – tentýž senzor měřící tatáž data může poskytovat pokaždé více či méně odlišné hodnoty, i když by měl poskytovat tytéž hodnoty. Lze pak vycházet z nějaké pravděpodobné, průměrné hodnoty, kterou lze získat z více měření zpracovaných stochasticky. Touto oblastí se nebudeme zabývat, protože se jí běžně zabývá v oboru Matematické biologie řada jiných výukových předmětů (Bi5040 Biostatistika, Bi7490 Základy stochastického modelování).

1.2.2 Přibližnost

Dále mohou také existovat data, popisující entity velmi **přibližným** (neurčitým) způsobem, kde na rozdíl od nejistoty nelze stanovit očekávanou hodnotu tradičními metodami. Taková data se vyznačují tím, že nemají stanoveny ostré hranice, oddělující je např. jednoznačně od dat jiných, mohou se s jinými údaji poměrně libovolně (i subjektivně) překrývat a je velmi obtížné je zpracovávat. V odborné praxi je poměrně běžné používat vyjádření typu "*je to částečně pravda*", "*asi tak zhruba pět nebo šest, možná o něco víc*", nebo třeba "*obvykle sní dvě až čtyři vejce, někdy i pět*". Zde nelze aplikovat pravděpodobnostní počet, protože nejsou k dispozici klasická pozorování jevů, z nichž lze stanovit aposteriorní pravděpodobnosti. Jestliže je stoprocentně *možné*, že „*sní dvě nebo tři vejce*", pak nelze mluvit o pravděpodobnosti rovné 1.0 pro výskyt obou jevů, protože stoprocentní výskyt nějakého jevu vylučuje výskyt ostatních jevů.

1.2.3 Nejednoznačnost a její odstraňování

Nejednoznačnost (*ambiguita*) a její odstraňování (*desambiguace*) hraje výraznou roli v procesu komunikace (interpretační fáze). V ideálním případě má řečník *S* na mysli sdělení *P* a provede akci mluvení, která však může mít několik interpretací, a je otázkou, kterou z interpretací lze v dané situaci považovat za nejlepší vzhledem k předávání výroku *P*. Naslouchající *H* lze si být nejednoznačnosti vědom a *P* interpretuje – odstraňuje víceznačnost. Někdy může *H* být zmaten a požádat o vyjasnění, to ale při častém opakování vede k zdržování komunikace, zejména pokud dojde k dalšímu, vícenásobnému "vyjasňování vyjasňovaného". Existuje velmi mnoho narušení plynulé komunikace, ale ukazuje se, že *největším problémem je to, že většina toho, co je vyřčeno, je nejednoznačné*. Typickým příkladem jsou např. titulky zpráv.

- *Lexikální ambiguita* je nejjednodušším typem ambiguity, kde nějaké jedno slovo má více významů. Např. anglické přídavné jméno *hot* znamená *horký, kořeněný, nejnovější, nesnesitelný, dychtivý, rozčilený, skvělý, čerstvý, radioaktivní, sexy, populární, ukradený,* aj. Lexikální ambiguita zahrnuje také nejednoznačnost kategorie slova: *back* je příslovce v *go back*, přídavné jméno v *back door*, podstatné jméno v *the back of the room*, a sloveso v *back up your files*.
- *Syntaktická ambiguita (strukturální ambiguita)* se může vyskytnout jak s lexikální nejednoznačností, tak i bez ní. Např. *I smelled flowers in the garden* má dva výsledky rozboru: *"cítil jsem* květiny *v zahradě"* nebo *"cítil jsem květiny* v zahradě*"*. Syntaktická ambiguita vede k další ambiguitě, k sémantické.
- *Sémantická ambiguita* může vést k tomu, že se sémantická nejednoznačnost objeví i ve frázích, kde není lexikální nebo syntaktická ambiguita. Např. fráze *coast road* (*pobřežní silnice*) může znamenat *silnici vedoucí po pobřeží* nebo *silnici vedoucí k pobřeží*.
- *Referenční ambiguita* je všudypřítomná forma nejednoznačnosti. Anaforické (odkazové na předchozí kontext) výrazy typu *it* (to) se mohou odkazovat téměř na vše ("Řekl jsem mu *to*. A co on na *to* řekl? Že o *tom* neví.").
- *Pragmatická ambiguita* se objevuje tehdy, když promlouvající *S* a naslouchající *H* se neshodnou na tom, co je okamžitá situace. *S* říká *"Setkám se s tebou příští čtvrtek."* za předpokladu, že se mluví o 12. dni měsíce, ale *H* si myslí, že se jedná o 19. den, takže dojde k chybné komunikaci.
- *Lokální ambiguita* se někdy objevuje ve frázi či větě, kde lze udělat rozbor podřetězce několika způsoby, ale pouze jeden z nich se shoduje s širším kontextem celého řetězce. Např. v programovacím jazyce *C* je význam řetězce *"*c"* buď *"pointer na c"*, pokud se řetězec vyskytuje v deklaraci *char *c;* nebo *"vynásob hodnotou c"* pokud se vyskytuje ve výrazu *"2*c"*. Podobně v angličtině je podřetězec *"the radio broadcasts"* fráze podstatného jména v širším kontextu *"the radio broadcasts inform"*, zatímco v jiném kontextu *"the radio broadcasts information"* jde o frázi podstatného jména *"the radio"* následovanou další frází slovesnou *"broadcasts information"* (*broadcasts* je *vysílání* jako podstatné jméno v množném čísle nebo *vysílá* jako sloveso ve třetí osobě, tj. *rozhlasová vysílání informují* nebo *rozhlas vysílá informace*).
- *Vágnost* (ve smyslu nejasnost, neurčitost) je v přirozených jazycích obvyklá. Výrok *"Venku je horko"* se týká teploty, ale termín *"horko"* je vágní, subjektivně vnímaný.

Nejednoznačnost může vyplynout i z typu akce řeči: Na dotaz *"Víš, kolik je hodin?"* odpoví *H "Ano."* nebo může odpovědět hodnotou času, např. *"Je půl sedmé a tři minuty"*. Obojí může být pro tentýž dotaz správné, ovšem v závislosti na typu otázky a např. kontextu. Desambiguace je v principu problém diagnózy, tj. rozboru a rozpoznání. Naslouchající *H* si udržuje model světa a na základě vyslechnutí nové akce řeči přidá možné interpretace toho, co slyšel, k modelu jako hypotézy. K interpretacím lze využít různé techniky zpracování nejistoty a přibližnosti (logika monotonická a nemonotonická, pravděpodobnosti, shlukování, stochastická simulace, Bayesovy metody, tvorba a zpracování pravidel, Dempster-Shaferovu metodu, znalostní inženýrství, fuzzy usuzování, aj.). K úspěšnému výsledku je zapotřebí, aby svět obsahoval co nejvíce informací o světě. Např. ke korektnímu vyřešení syntaktické ambiguity v anglické větě *"Joe saw the Grand Canyon flying to New York." (Pepa viděl Grand Canyon při letu do New Yorku.)* je nutno vědět, že je mnohem pravděpodobnější, že Pepa – a nikoliv Grand Canyon – letěl letadlem (a tedy že Pepa neviděl Grand Canyon letící do New Yorku).

Je také zapotřebí mít dobrý model o názorech *S* a *H*, aby se dalo určit, co měl *S* v úmyslu říci. Např. obvyklá interpretace anglického sdělení *"I am not a crook"* je, že *"nejsem zloděj"*, i když lze sdělení také přeložit jako *"nejsem pastýřská berla"*. Podobně slovo *"bank"* znamená mj. *"břeh, lavice, banka"*, ale je značně pravděpodobné, o co jde ve výroku *"Peter keeps his money in the bank"*, tj. Petr asi nezahrabává své peníze do břehu (i když i to je možné).

Zcela jistým a posledním důvodem, proč správná interpretace je tak velmi obtížným problémem, je skutečnost, že může existovat *několik* správných interpretací. Např. vtipy jsou často založeny na tom, že *H* se pobaví pomocí dvou současně platných interpretací. Většina programů, schopných porozumění, zatím ignoruje tyto skutečnosti (možnost současného výskytu několika případů).

Existují tedy dva různé pojmy: *pravděpodobnost* a *možnost*. Jedním z odvětví matematiky, které se zabývá *možností*, je tzv. *teorie fuzzy množin* [Kruse, Gebhardt, Klawonn, 1994].

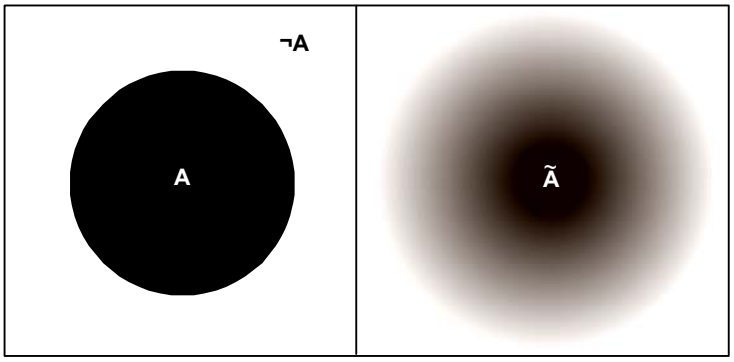
1.2.4 Fuzzy množiny

Fuzzy množina modeluje *přibližné* (nikoliv tedy *pravděpodobné*) neurčité hodnoty, které mohou být čísla (mluvíme pak o *fuzzy číslech* a existují i fuzzy aritmetiky) nebo i nominální hodnoty ("*skoro zelený, mírně do modra*" je příklad jedné z možných vyjmenovaných barev nějakého objektu, třeba automobilu). Fuzzy množiny představují rozšíření klasických množin a vyznačují se tím, že do nich mohou prvky patřit jen částečně, neboli určitý prvek může náležet s různou mírou příslušnosti do více množin zároveň. Pak lze modelovat tvrzení typu "*částečně je to pravda, částečně nepravda, a částečně neznámo*", s čímž si samozřejmě klasická dvouhodnotová (nebo vícehodnotová) logika neporadí.

1.2.5 Fuzzy logika

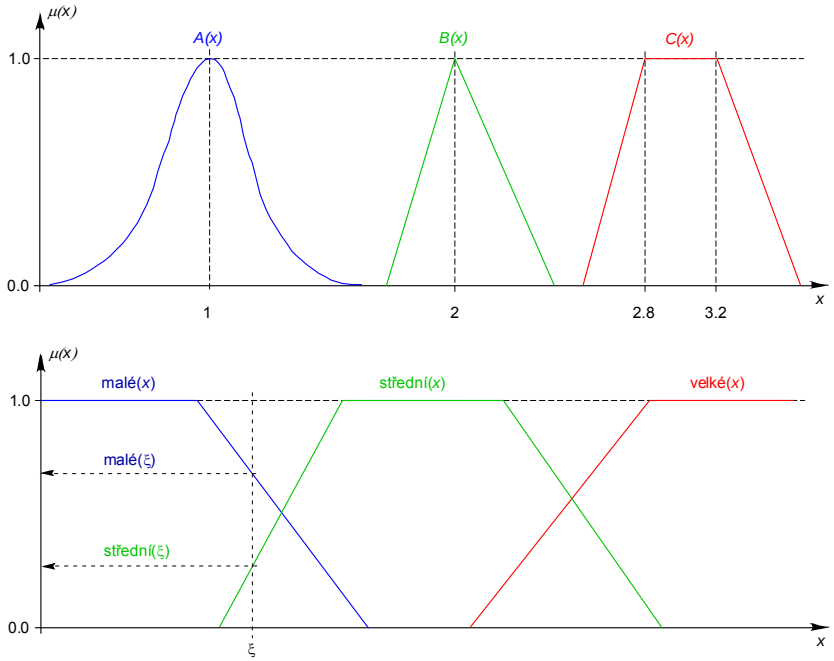
Fuzzy logika, která vznikla odvozením z fuzzy množin představuje jednu z možností, jak definovat vícehodnotovou logiku (zde vlastně logiku s nekonečným počtem hodnot mezi 0 a 1). Např. pravdivostní hodnotu rovnu 0.75 lze interpretovat jako "*ze tří čtvrtin pravda*", což může v řadě situací dávat smysl. Zakladatel teorie fuzzy množin je profesor Lotfi A. Zadeh (nar. 1921), původem z Ázerbajdžánu v bývalém SSSR. Vystudoval a pracoval v USA, kde je od r. 1990 ředitelem v Berkeley Initiative of Soft Computing (BISC) na University of California. Jeho idea byla modelovat přibližné, neostré (*fuzzy*) hodnoty vyskytující se typicky v běžně mluvené řeči, kde se používají hodnoty jako "*…za necelou hodinu budeme vědět o dost víc; zdali je to dostatečně blízko nebo spíše daleko…*". Aniž by uvedené sdělení obsahovalo nějaké přesné, ostré (*crisp*) číselné hodnoty, v určitém *daném kontextu* jsou lidé schopni si takto předávat smysluplné informace. Idea vychází ze skutečnosti, že v běžném životě nejsou známy přesné hodnoty, avšak přibližné výsledky plynoucí z přibližných hodnot v praxi fungují docela dobře.

Pojem *necelá hodina* může znamenat lecjaký časový údaj, daný i subjektivním postojem: jde o 50 až 59 minut? Nebo 40 až 50? Nebo 40 až 55? Kterékoliv uvedené časové rozpětí je možné; jedná se o intervaly bez ostrých hranic, kde pojmy jako *nejméně půl hodiny* a *méně než hodina* se mohou různě vzájemně překrývat, takže třeba hodnota ¾ hodiny může patřit do obou pojmů. Zadeh navrhl fuzzy množiny [Zadeh, 1965], nemající ostré hranice, pro reprezentaci neostrých hodnot. Fuzzy množina $\tilde{A}$ obecně nemá ostré oddělení od svého doplňku (pravá část následujícího obrázku), na rozdíl od klasické množiny A (levá část obr. č. 1.5):



Obrázek č. 1.5: Rozdíl mezi množinou A a fuzzy množinou Ã

Přechod od úplného náležení prvku do množiny až do jeho nenáležení je u fuzzy množin plynulý, takže lze říci, že fuzzy množina sdílí prvky se svým doplňkem (existuje neprázdný průnik), přičemž míra náležení prvků x z univerza[5] X (univerzum je *vždy* ostrá množina) do množiny $\tilde{A}$ je obecně definována nějakou funkcí $\mu(x)$ (funkce příslušnosti, *membership function*). U klasických množin je $\mu(x)$ funkce se skokovým přechodem mezi množinou A a jejím doplňkem $\neg A$, zatímco u fuzzy množin může přechod být libovolný pozvolný.



Obrázek č. 1.6: Přibližné vyjádření čísel

[5] Nechť X je neprázdná množina zvaná **universum**

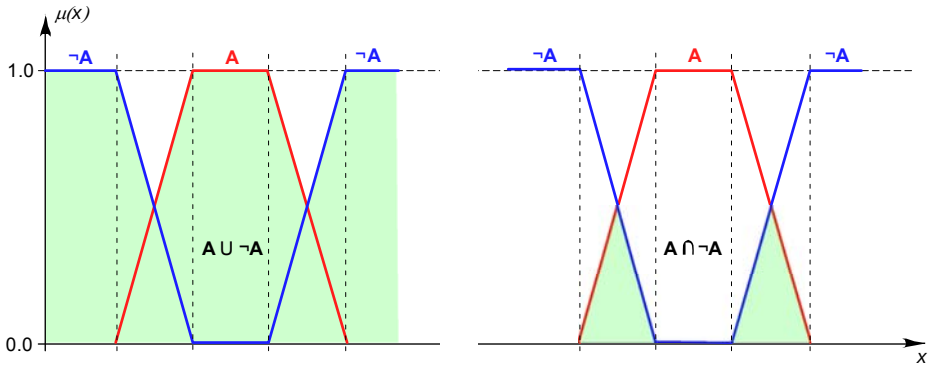
Lze tak modelovat i přibližná čísla, jak ukazuje obr. č. 1.6, např. $A(x)$ znamená *asi1*, $B(x)$ *přibližně 2*, $C(x)$ *okolo 3*. V praxi se často používá pro $\mu(x)$ lomená čára ($B(x)$ jako jedna neostrá hodnota a $C(x)$ jako neostrý interval), kterou lze definovat jednoduše na rozdíl od komplikovanějších funkcí typu $A(x)$.
Obdobně lze definovat *lingvistické* hodnoty typu *malý, střední, velký* apod. Hodnota ξ v obr. č. 6 patří s určitou mírou náležení, danou funkcí příslušnosti $\mu(x)$, jak do hodnot malých, tak do hodnot středních, i když do obou jen částečně. Spíše je ξ malé než střední, určitě není velké: *malé*(ξ) > *střední*(ξ)*; velké*$(\xi) = 0$.
Zároveň je vidět, že některé hodnoty nemusí patřit do žádné na 100 % (např. přechod mezi *malé* a *střední*). Naopak mohou existovat i takové hodnoty univerza, které patří do více fuzzy množin na 100 %, definice příslušností je vždy závislá na konkrétní aplikaci, a to určit bývá obtížný úkol.

1.2.6 Fuzzy množinové operace

Fuzzy množiny jsou rozšířením klasických množin: mají-li prvky fuzzy množiny své funkce příslušnosti definovány skokově (hodnoty pouze 0 nebo 1), pak **fuzzy množinové operace** dají stejné výsledky jako klasické množinové operace. Obvyklé definice základních množinových operací pro fuzzy množiny (označené zde jako $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$) a prvek x z univerza X jsou následující [$A(x)$, $B(x)$, resp. $C(x)$ znamenají hodnoty příslušnosti x do $\boldsymbol{A}$, $\boldsymbol{B}$, resp. $\boldsymbol{C}$]:

- sjednocení: $C(x) = \max[A(x), B(x)]$ pro $\boldsymbol{C} = \boldsymbol{A} \cup \boldsymbol{B}$,
- průnik: $C(x) = \min[A(x), B(x)]$ pro $\boldsymbol{C} = \boldsymbol{A} \cap \boldsymbol{B}$,
- doplněk: $\neg A(x) = 1 - A(x)$.

Avšak snadno se lze přesvědčit, že vzhledem k definici doplňku není u klasických a fuzzy množin shodný výsledek pro *pravidlo vyloučeného třetího* (sjednocení množiny a doplňku dává univerzum) a *pravidlo kontradikce* (průnik množiny a doplňku dává prázdnou množinu). Fuzzy množinový doplněk může být tedy problematický, avšak ve velké většině praktických aplikací (např. řízení procesů) tento nedostatek není na závadu, nemusí-li se s doplňky pracovat.



Obrázek č. 1.7: Přibližné vyjádření čísel

Fuzzy množiny představují užitečný nástroj pro rozšíření databázových systémů, kde umožňují výběry na základě větší či menší podobnosti hodnot atributů vůči zadanému dotazu. Fuzzy množiny a odpovídající fuzzy logiku lze použít také pro řadu oborů, kde je zapotřebí inference – rozšířené jsou aplikace s fuzzy pravidly, která obsahují fuzzy množiny modelující přibližné hodnoty v antecedentech i konsekventech, což umožňuje činnost fuzzy expertních systémů. Pro inferenci se nejčastěji používá zobecněný *modus ponens*, někdy *modus tolens*.

1.2.7 Fuzzy expertní systémy

Fuzzy expertní systémy jsou založeny na znalostní bázi obsahující fuzzy pravidla, což vede k rozdílům oproti klasickým expertním systémům používajícím pravidla, která vycházejí z tradiční logiky a vyhodnocování. Jedním z hlavních rozdílů je možnost platnosti více pravidel naráz, přičemž aktivovaná fuzzy pravidla, která přinášejí výsledek vzhledem k zadanému dotazu, mohou mít různý stupeň platnosti, jejich konsekventy je nutno sloučit dohromady a tento výsledek interpretovat. Konsekventy se obvykle slučují operací *max*, tj. množinovým sjednocením, protože pravé strany pravidel jsou obecně rovněž fuzzy množiny.

1.2.8 Defuzzifikace

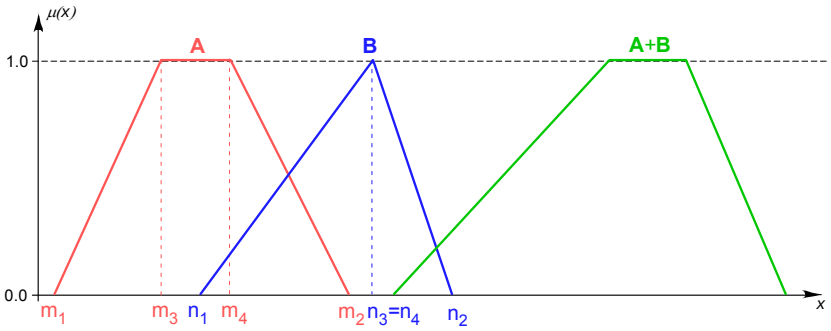
Ne vždy bývá snadná interpretace přibližného výsledku, zejména v aplikacích vyžadujících jako odpověď jednu ostrou hodnotu (třeba velikost elektrického proudu pro servomotor ovládající rameno robota nemůže být "asi tak 57 Ampér"). Existují operace tzv. **defuzzifikace** (převod přibližných hodnot na ostré), např. často používané je stanovení ostré hodnoty jako souřadnice těžiště plochy, představované komplikovanou výslednou fuzzy množinou, na ose univerza výstupních hodnot. Každá z metod defuzzifikace má kromě výhod i nevýhody, které často zabraňují jejímu použití: má-li robot před sebou překážku a může-li ji objet zprava (regulátor pohybu po vyhodnocení údajů ze senzorů pomocí fuzzy pravidel vydá fuzzy pokyn "zatoč mírně doprava …") nebo stejně tak zleva ("… nebo zatoč mírně doleva"), kde operace *nebo* se realizuje množinovým sjednocením *max.* Výsledek defuzzifikovaný metodou těžiště (nebo metodou obdobnou) poskytne však ostrý pokyn "jeď rovně" jako jakousi střední hodnotu z možných akcí a robot do překážky narazí, pokud nemá inteligentnější způsob interpretace neostrého výsledku inference. K obtížnosti interpretace přispívá i skutečnost, že je definována řada různých operací fuzzy implikace (a nejčastěji používaná tzv. *Mamdaniho implikace* navíc implikací není, i když pro řízení procesů ve velké většině vyhovuje). Každá z inferenčních i defuzzifikačních metod má své výhody a nedostatky. Alternativní přístup, odstraňující nutnost defuzzifikace, představuje metoda *Takagi-Sugeno*, která do konsekventů pravidel nedává fuzzy množiny, ale ostré funkce vstupních proměnných x_i, které nejčastěji mapují kartézský součin příslušných univerz X_i do výsledného univerza Y; někdy dokonce stačí do konsekventů pravidel vložit pouze ostré konstanty.

1.2.9 Fuzzy čísla

Kromě logických a množinových výpočtů lze počítat i s hodnotami vyjádřenými **fuzzy čísly**. Fuzzy číslo je rovněž fuzzy množina, představující nějakou neostrou hodnotu. Existuje několik typů fuzzy aritmetiky, například fuzzy obdoba *intervalové aritmetiky*, kdy se uvažují souřadnice fuzzy čísel **A** a **B** na úrovni tzv. α-řezů $\mu(x) = 0$ a $\mu(x) = 1$, což jsou intervaly, na něž se aplikuje aritmetika, která dá souřadnice výsledných intervalů na obou použitých úrovních (viz následující obrázek):

- sečítání: $[m_1 + n_1, m_2 + n_2]$ pro $\mu(x) = 0$ a $[m_3 + n_3, m_4 + n_4]$ pro $\mu(x) = 1$;
- odečítání: $[m_1 - n_2, m_2 - n_1]$ pro $\mu(x) = 0$ a $[m_3 - n_4, m_4 - n_3]$ pro $\mu(x) = 1$;
- násobení: $[m_1 \cdot n_1, m_2 \cdot n_2]$ pro $\mu(x) = 0$ a $[m_3 \cdot n_3, m_4 \cdot n_4]$ pro $\mu(x) = 1$;
- dělení: $[m_1 / n_2, m_2 / n_1]$ pro $\mu(x) = 0$ a $[m_3 / n_4, m_4 / n_3]$ pro $\mu(x) = 1$.

Dělení nelze obecně realizovat, je-li **A** tzv. *nulové* fuzzy číslo, což je fuzzy množina obsahující jako prvek nulu s nenulovou hodnotou příslušnosti ($\mathbf{A} / \mathbf{B} = \mathbf{A} \cdot \mathbf{B}^{-1}$, tj. násobení inverzní hodnotou).
Odečtení je definováno jako přičtení $-\mathbf{A}$, kde $-\mathbf{A}$ je tzv. *obraz*: $\mu_{-\mathbf{A}}(x) = \mu_{\mathbf{A}}(-x)$.



Obrázek č. 1.8: Přibližné vyjádření čísel

Přibližnosti obsažené v jednotlivých operandech se ve výsledcích operací kumulují, takže například sečtení více fuzzy čísel dá jako výsledek velice přibližnou hodnotu.

1.3 Počítač, software, hardware, komunikace, internet

V tomto odstavci zavedeme pojmy, které souvisejí s počítačovými systémy a komunikací mezi nimi, jako je software, hardware, počítačové sítě.

1.3.1 Počítač

Počítač je specializované zařízení, které podle předem připravených instrukcí zpracovává data. V dalším budeme uvažovat pouze číslicové počítače, které sestávají ze dvou základních druhů komponent: **technického vybavení** (hardware), tj. fyzických komponent, skládající se z různých (převážně elektronických) dílů; **programového vybavení** (software), tj. informací složených z řady instrukcí, které jsou počítačem postupně provedeny. Obvykle není software nic jiného než zvláštní druh dat uložený v paměti počítače podobně jako ostatní data. Počítače mají nespočetně mnoho podob. Název počítač je svým způsobem anachronismus. Dnešní počítače jsou ve skutečnosti „stroje na zpracování informací“, a to ve všech svých podobách:: obrazové, textové, zvukové a filmové/video. Umožňují tyto informace pořizovat, ukládat, upravovat, poskytovat v požadovanou dobu, předávat na požadované místo v požadovaném tvaru.

1.3.2 Software

Software nebo též **programové vybavení** počítače je množina všech počítačových programů (tj. zaznamenaného postupu počítačových operací, který speciálním způsobem popisuje praktickou realizaci zadané úlohy – algoritmus výpočtu) uložených v počítači. Obvykle je počítačový program zapsán v nějakém programovacím jazyku nebo strojovém kódu počítače a je tvořen posloupností příkazů. Software vzniká jeho programováním, tj. činností programátorů, která zahrnuje tvorbu *algoritmu*[6] a *programu* [Gonnet, Baeza-Yates, 2006]. **Algoritmem** rozumíme obecný postup řešení dané úlohy. **Program** je zápis algoritmu ve zvoleném programovacím jazyce. Počítačový program obsahuje sekvence instrukcí, které jsou vykonávány procesorem počítače. Počítačový program je neoddělitelný doplněk hardware počítače.

1.3.3 Počítačový software

Počítačový software se dělí na dvě základní skupiny: *systémový software* a *aplikační software* (aplikace). Systémový software je programové vybavení počítače, které umožňuje spouštění nebo zpracování aplikačního software. Typický představitel systémového software

[6] http://cs.wikipedia.org/wiki/Algoritmus

je *operační systém*. Aplikační software (programy) je programové vybavení, které je navrženo a vytvořeno pro řešení nějakého konkrétního problému.

1.3.4 Aplikační software

Pojem **aplikační software** nebo aplikační programové vybavení znamená takové programové vybavení počítače, které je určeno pro přímou či částečně nepřímou interakci s uživatelem, na rozdíl od obecného pojmu programové vybavení (software), které nemusí být v interakci s uživatelem (např. programové vybavení moderní telefonní ústředny). Často se termín zkracuje na slovo aplikace. Účelem aplikačního software je zpracování a řešení konkrétního problému (uživatele).

1.3.5 Open source software

Původním záměrem **open source software** bylo zbavit se problémů dvojznačnosti termínu *free software* a pomoci k lepšímu prosazení sdílení software v komerční sféře. Open source nebo také open-source software (OSS) je počítačový software s otevřeným zdrojovým kódem[7]. Otevřenost zde znamená jak technickou dostupnost kódu, tak legální dostupnost – licenci software, která umožňuje, při dodržení jistých podmínek, uživatelům zdrojový kód využívat, například prohlížet a upravovat.
V užším smyslu se OSS míní software s licencí vyhovující definici prosazované *Open Source Initiative*[8] (OSI). Pro odlišení se někdy open source software vyhovující požadavkům OSI označuje Open Source (s velkými písmeny).

1.3.6 Svobodný software

Za **svobodný software** (*free software*) pokládáme takový software, ke kterému je k dispozici také zdrojový kód, spolu s právem tento software používat, modifikovat a distribuovat se zachováním určitých práv a svobod pro jejich koncového uživatele. Jde o právo:

- Spouštět program za jakýmkoliv účelem.
- Studovat, jak program pracuje a přizpůsobit ho svým potřebám. Předpokladem k tomu je přístup ke zdrojovému kódu.
- Redistribuovat kopie dle svobodné vůle.
- Vylepšovat program a zveřejňovat zlepšení, aby z nich mohla mít prospěch celá komunita. Předpokladem je opět přístup ke zdrojovému kódu.

Za získání kopií svobodného software můžete platit nebo je obdržet zdarma, ovšem bez ohledu na způsob, jak jste je získali, máte vždy svobodu kopírovat a měnit software, dokonce prodávat nebo darovat jeho kopie nebo pozměněné verze. Uživateli je rovněž umožněno modifikovat program a používat tuto modifikovanou verzi soukromě ve svém zaměstnání nebo volném čase bez toho, aniž by se musel zmiňovat o existenci programu.

Důležitou roli v oblasti svobodného software hraje nadace *Free Software Foundation* (FSF)[9], česky *Nadace pro svobodný software*, která byla založena již v roce 1985 s cílem podporovat práva uživatelů počítačů používat, studovat, kopírovat, modifikovat a redistribuovat počítačové programy. FSF podporuje vývoj svobodného softwaru, jmenovitě pak operačního systému GNU[10]. Dnes jsou hojně používány různé varianty operačního systému GNU s

[7] Jako zdrojový kód či zdrojový text se označuje text počítačového programu zapsaný v některém (obvykle vyšším) programovacím jazyce.
[8] http://en.wikipedia.org/wiki/Open_Source_Initiative
http://www.oreilly.com/catalog/opensources/book/perens.html
[9] http://www.fsf.org/
[10] GNU je projekt zaměřený na svobodný software, inspirovaný operačními systémy unixového typu. Původní cíl byl vyvinout operační systém se svobodnou licencí, který však neobsahuje žádný kód původního UNIXu.

jádrem Linux; ačkoliv se těmto systémům často říká Linux, přesnější pojmenování pro ně je GNU/Linux. Nadace FSF také se stará o právní a organizační stránky projektu GNU a o rozšiřování povědomí o svobodném software. Myšlenky svobodného software byly formulovány prostřednictvím copyleft[11] GNU *General Public License*[12] (česky „všeobecná veřejná licence GNU“) a GNU *Lesser General Public License* (původně nazývaná GNU „*Library General Public License“*), které se časem staly nejrozšířenějšími licencemi svobodného software[13].

Svobodný software není úplně totéž jako Open source software; záměna termínů není v některých případech možná. Požadavky (*The Open Source Definition*) na Open source software jsou nepatrně méně striktní. Rozdíl je spíše v ideologii. GNU uznává většinu Open source licencí jako licence svobodného software, ale vzhledem k jejich nekompatibilitě s GPL je nedoporučuje používat.

1.3.7 Freeware

Samotné slovo *free software* má v angličtině však také druhý význam, který znamená software zadarmo, tedy něco zcela odlišného. Tomu se však obvykle říká **freeware**. Tento termín *freeware* nemá zatím jasnou a přijatelnou definici, ale je běžně používán pro balíky programů, u kterých je dovolena distribuce, ale ne modifikace (zdrojové kódy nejsou dostupné) jako u *svobodného software*.

1.3.8 Middleware

Middleware je software, který slouží jako konverzní nebo překladatelská vrstva. Slouží také jako integrátor. Během desetiletí bylo vytvářeno mnoho middleware programů, aby umožnily komunikaci mezi různými částmi softwaru, které jsou od různých výrobců, beží na jiných platformách a nebo obojí. Nyní jsou zde již hotové balíčky s middlewarem, které umožňují komunikovat mezi jednotlivými programy. Middleware samozřejmě není jen jeden, ale jsou různé jeho typy, dělící se podle jeho funkcí, tedy podle toho, co s čím komunikuje a co má middleware vlastně na starost.

1.3.9 Hardware

Hardware označuje veškeré fyzicky existující technické vybavení počítače na rozdíl od dat a programů (označovaných jako software). Samotná hranice mezi softwarem a hardwarem však není nijak ostrá – existuje tzv. firmware, což je název pro programy napevno vestavěné v hardware. U moderních počítačů zpracovávaná data i prováděné instrukce jsou umístěny v paměti počítače, řídící jednotka zajišťuje načítání instrukcí a dat z paměti (a jejich zápis zpět do paměti), aritmeticko-logická jednotka provádí operace s načtenými daty, přičemž data je také možné zapisovat na vstupně/výstupní porty i je z nich načítat. U současných osobních počítačů se hardware se rozděluje podle komponent.

1.3.10 Komunikace

Komunikace je sdělování informací, myšlenek, názorů, a pocitů mezi živými bytostmi, lidmi i živočichy obvykle prostřednictvím společné soustavy symbolů. Pojmem *počítačová komunikace* se rozumí vzájemné předávaní relevantních datových zpráv mezi počítači v

[11] Copyleft používá copyrightového práva, ale „převrací" jej tak, aby sloužilo k opačným účelům než obvykle: namísto prostředku ke spoutání software, se stává prostředkem k udržení software svobodným.
[12] http://gnu.cz/article/30/pdf/gpl-cz.pdf
[13] http://www.gnu.org/licenses/license-list.cs.html

elektronické podobě pomocí hardware a software pro elektronickou komunikaci. Elektronická komunikace představuje výměnu dat mezi různými softwary.

1.3.11 Počítačová síť

Počítačová síť je souhrnné označení pro technické prostředky, které realizují spojení a výměnu informací mezi počítači. Umožňují tedy uživatelům komunikaci podle určitých pravidel, za účelem sdílení využívání společných zdrojů nebo výměny zpráv.

1.3.12 Internet

Internet je celosvětová počítačová síť, která spojuje jednotlivé menší sítě pomocí sady *protokolů IP*[14]. Název pochází z anglického slova network (síť), podle něhož tradičně názvy amerických počítačových sítí končily „*-net*“, a mezinárodní (původně latinské) předpony *inter-* (mezi), vyjadřující, že internet propojil a vstřebal různé starší, dílčí, specializované, proprietární nebo lokální sítě. Internet slouží k přenášení informací a poskytování mnoha služeb, jako jsou elektronická pošta, chat, www stránky, sdílení souborů, on-line hraní her, vyhledávání, katalog a další.

1.3.13 Web

World Wide Web (**WWW**, také pouze zkráceně web), ve volném překladu „Celosvětová pavučina“, je označení pro aplikace *internetového protokolu HTTP*[15]. Je tím myšlena soustava propojených hypertextových dokumentů. Dokumenty umístěné na počítačových serverech jsou adresovány pomocí *URL*[16], jehož součástí je i doména a jméno počítače. Název naprosté většiny těchto serverů začíná zkratkou www, i když je možné používat libovolné jméno vyhovující pravidlům URL.

1.3.14 Distribuovaný výpočet

Distribuovaný výpočet je výpočet rozdělený na více menších méně náročných úloh za účelem rychlejšího vyřízení požadavku předaného programu. Lze ho využít jen u výpočtů, jejichž algoritmus lze paralelizovat – převést na paralelní verzi, kdy vzájemně nezávislé části výpočtu běží současně. Výpočet lze distribuovat buď na úrovni operačního systému s přesměrováním komunikace na jiné členy počítačové sítě (klastru), nebo přímo v režii programu, který se nainstaluje v podobě mnoha klientů na každý z počítačů tvořících klastr. Mezi typické výpočty, které je vhodné řešit distribuovaně, patří analýza velkého množství statistických dat, „zpětné inženýrství“ DNA, modelování struktury proteinů i výpočty jako generování fraktálů nebo zkoumání vesmírného vlnění, zda je v něm přítomen rádiový signál mimozemšťanů.

1.4 Vědecké výpočty a výpočetní věda

V tomto odstavci již můžeme zavést terminologii, která souvisí s vědeckými výpočty s využitím pojmů zavedených v předchozích odstavcích.

---

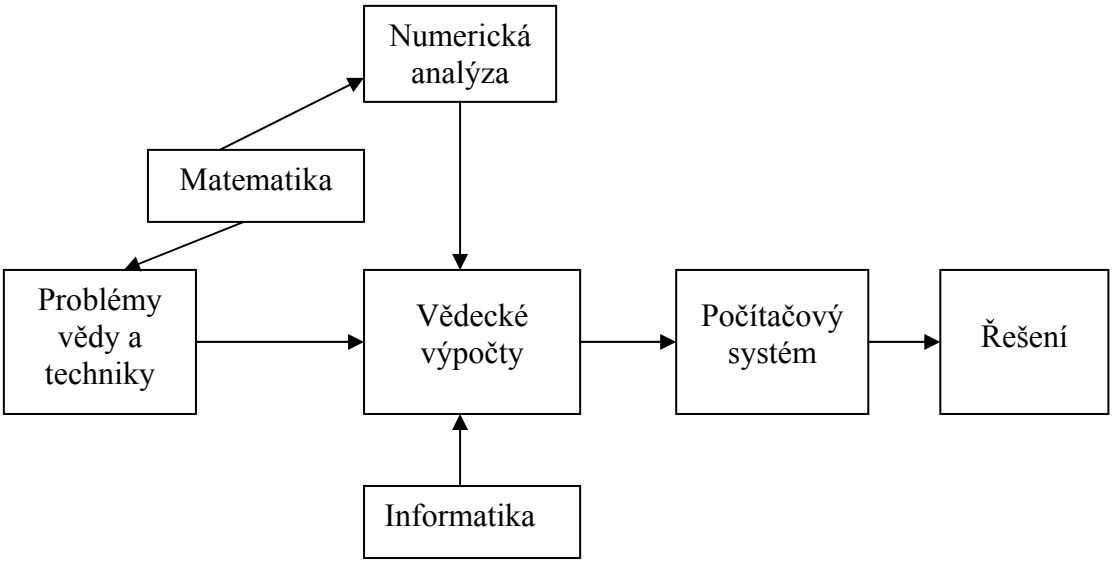
[14] IP (anglicky Internet Protocol) je datový protokol používaný pro přenos dat přes paketové sítě. Tvoří základní protokol Internetu, viz např. http://cs.wikipedia.org/wiki/Internet_Protocol.

[15] HTTP (Hyper Text Transfer Protocol) je internetový protokol určený původně pro výměnu hypertextových dokumentů ve formátu HTML. K protokolu HTTP existuje také jeho bezpečnější verze HTTPS, která umožňuje přenášená data šifrovat a tím chránit před odposlechem či jiným narušením.

[16] URL, celým názvem Uniform Resource Locator („jednotný lokátor zdrojů“) je řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací (ve smyslu dokument nebo služba) na Internetu. URL definuje doménovou adresu serveru, umístění zdroje na serveru a protokol, kterým je možné zdroj zpřístupnit.

1.4.1 Vědecké výpočty

Výpočetní věda[17] (*Computational Science*), resp. její synonymum **vědecké výpočty** (*Scientific Computing*), je v současné době jedna z velmi rychle se rozvíjejících oblastí vědy, která využívá informační a komunikační technologie k analýze vědeckých, sociálně vědeckých a technických problémů, aby dosáhla jejich pochopení pomocí náročných výpočtů (*HPC - High Performace Computing*), které vyžaduje analýza jejich matematických modelů.



Obrázek č. 1.9: Vědecké výpočty a související oblasti [zdroj: Golub/Ortega 1993]

Vědecké výpočty/výpočetní věda nemají dosud ustálenou definici a jsou spíše uměním, kde se prolínají různé vědecké disciplíny s matematikou a informatikou. Je to de facto počítačová revoluce ve způsobu vědecké práce. Toto nové paradigma má dvě významná použití: *řešení dosud nevyřešených sporných problémů a výzkum nových, přičemž oba typy problémů by nebylo možno řešit bez vědeckých výpočtů/výpočetní vědy*. Tato vědní disciplína umožňuje pochopit řešený problém především pomocí analýzy jeho matematického modelu implementovaného a řešeného pomocí počítačového systému, tím se liší např. od informatiky (*tj. oboru základního i aplikovaného výzkumu zabývající se shromažďováním, klasifikací, ukládáním, výběrem a šířením zaznamenané znalosti*) a tradičních forem vědy a techniky postavené na teorii a experimentech.

V praxi je typická aplikace vědeckých výpočtů/výpočetní vědy při počítačové simulaci[18] nebo v dalších formách výpočtů řešení problémů v různých vědních disciplínách. Ve vědeckých výpočtech/výpočetní vědě se využívají jak symbolické, tak i numerické výpočty a metody.

1.4.2 Počítačová simulace

Počítačová (numerická) simulace[19] je další vědeckou metodou, která rozšiřuje tradiční teorie a fyzické experimenty a přináší nové postupy řešení problémů. Numerická analýza a simulace je důležitou podporu pro postupy a techniky využívané ve vědeckých výpočtech. Počítačovou simulaci ve vědeckých výpočtech rozumíme běh aplikačního programu, pomocí

[17] http://en.wikipedia.org/wiki/Scientific_computing
[18] Simulace je v matematice a kybernetice vědecká metoda, při které se zkoumají vlastnosti nějakého systému pomocí experimentů s jeho matematickým modelem.
[19] Počítačová simulace je experiment s počítačovým modelem.

kterého simulujeme chování abstraktního modelu konkrétního systému. Tradičně, formální modelování systému se děje prostřednictvím matematického modelu, který se pokouší nalézt analytické řešení problému, které umožňuje predikovat chování systému na základě jeho parametrů, počátečních a okrajových podmínek. Počítačová simulace je často užívána jako doplnění nebo nahrazení modelovaného systému, u něhož nalezení uzavřené analytické formy řešení není možné. Existuje mnoho typů počítačové simulace; společným rysem, který všechny simulace sdílejí je pokus generovat vzorek reprezentativních scénářů pro model, ve kterém úplný výpočet všech možných stavů modelu je nemožný nebo nepřípustný.

Počítačovou simulaci dělíme vzhledem různým kritériím na:

- *Stochastickou* (užívající generátorů náhodných čísel a metodu Monte Carlo) nebo *deterministickou* (jako speciální případ deterministické je chaotická). Stochastická simulace je typicky využívána pro diskrétní systémy, kde se události vyskytují s určitou pravděpodobností a nemohou být popsány přímo např. pomocí diferenciálních rovnic. Jevy v této kategorii zahrnují např. genetický drift, biochemické nebo genové sítě s malým počtem molekul.
- *Statickou* a *dynamickou* (závislou na čase). Ve statické simulaci se používají rovnice definující rovnovážné vztahy mezi prvky modelovaného systému a hledá se stav ve které je systém v rovnováze. Tento způsob je často využíván při simulaci fyzikálních systémů, které jsou dynamické, aby se zjednodušila jejich simulace než se přistoupí k dynamické simulaci.
- *Spojitou a diskrétní*. Spojitá dynamická simulace je řešena většinou pomocí aplikačních programů pro numerické řešení diferenciálně algebraických rovnic nebo diferenciálních rovnic (buď parciálních nebo obyčejných). Speciálním typem diskrétní simulace, která nezávisí na modelu s příslušnou rovnicí, ale může být víceméně representována formálně, je *„agent based“ (zprostředkovatelská) simulace*. V této simulaci nejsou individuální entity (např. molekuly, buňky, stromy nebo zákazníci) modelu reprezentovány přímo, nýbrž zprostředkovaně (např. jejich hustotou nebo koncentrací) a je řízen vnitřní stav modelu pomocí množiny chování nebo pravidel, která určují jak agentův (zprostředkovatelův) stav, který je aktualizován od jednoho časového kroku k následujícímu.
- *Lokální a distribuovanou* (je řešena na počítačových sítích a prostřednictvím Internetu).

Vědci a technici vyvíjejí informační a komunikační technologie (krátce ICT) pro výpočetní vědu/vědecké výpočty jako jsou např. počítačové programy, aplikační programové vybavení (zkráceně SW), které modelují studované systémy v nejrůznějších oblastech vědy a spouští tyto programy/SW s různými množinami vstupních parametrů, aby nalezli jejich řešení. Tyto modely vyžadují ohromné množství náročných výpočtů (obvykle v pohyblivé řádové čárce) a jsou často zpracovávány na superpočítačích nebo distribuovaných výpočetních systémech a platformách. Pomocí těchto ICT se často modelují proměnlivé podmínky reálného světa jako je počasí, proudění vzduchu kolem letícího letadla nebo v plících, deformace kloubů kostry při dlouhodobém zatížení nebo karosérie automobilu při nárazu, pohyb hvězd ve vesmíru, šíření karcinomů nebo výbuch zařízení apod.

Oblast výzkumu, ve které je potřeba zpracovat ohromné množství dat a ve které je největší potenciál pro využití superpočítačů, je výzkum lidského genomu. V této oblasti se provádějí

hlavně operace nad velkými databázemi, proto se využívají ty největší databázové systémy využívající paralelní architektury postavené na procesorech typu RISC[20].

Na závěr tohoto odstavce shrneme*, že výpočetní věda/vědecké výpočty se nyní uvažují jako třetí způsob/technika ve vědeckém bádán doplňující a rozšiřující experimentování/pozorování a teorii.* Tento názor lze najít ve známé monografii prof. S. Wolframa „A New Kind of Science" [Wolfram, 2002] i v knize prof. Gandera a Hřebíčka „Solving Scientific Problems Using Maple and MATLAB" [Gander, Hřebíček, 2005], která poprvé vyšla v roce 1992. V dalším se omezíme jen na termín vědecké výpočty, pod kterými budeme rozumět i výpočetní vědu. Využití vědeckých výpočtů, matematických nástrojů a teorie vede k vývoji a ke zlepšení současného stavu poznání světa.

Programovacími jazyky, které se společně používají v programování především matematických aspektů vědeckých výpočtů jsou hlavně Fortran a C, nově C#, Matlab (numerické výpočty), Maple, Mathematica aj. (symbolické výpočty), kterými se nebudeme zabývat.

1.4.3 Klasifikace vědeckých výpočtů

Vědecké výpočty můžeme rozdělit na tři oblasti:

1. *Počítačovou (numerickou) simulaci*, která má různé cíle závisející na podstatě simulované úlohy, např.:
   - Rekonstrukce a pochopení známých událostí (např. zemětřesení, tsunami a další přírodní katastrofy).
   - Predikce budoucí nebo nezpozorované situace (např. šíření karcinomu, počasí, chování atomových částic).

   V poslední době se provádí mnoho simulací, ať už různých výrobních technologií nebo účinků nových léků na organizmy, chování těles při namáhání až po reagenci a energii molekul.
2. *Přizpůsobení matematického modelu a datovou analýzu*, např.:
   - Vhodným laděním modelů nebo řešení rovnic, aby odrážely pozorování, jevy a objekty, aby modelovaly omezení (např. šíření nafty v geofyzice, počítačová lingvistika).
   - Využití teorie grafů v modelování sítí, zvláště těch spojení jedinců, organizací a webových míst.
3. *Metody a algoritmy*. Algoritmy a matematické metody užívané v vědeckých výpočtech jsou různé a jsou na přírodovědecké fakultě zavedeny ve výuce v předmětu M5180 Numerické metody II [Horová, Zelinka, 2004].

V odborné literatuře jsou vědecké výpočty/výpočetní věda nejčastěji děleny do následujících subkategorií [Wikipedia[21], 2007], které ukazují šíři jejich bádání:

1. Bioinformatika
2. Chemoinformatika
3. Chemometrika
4. Výpočetní chemie
5. Výpočetní biologie
6. Výpočetní matematika
7. Výpočetní mechanika

[20] RISC - Reduced Instruction Set Computer je architektura, kdy procesor umí relativně málo druhů instrukcí, ty jsou však velmi dobře propracované, vykonávají se rychle a jejich kód má vždy stejnou délku.
[21] http://en.wikipedia.org/wiki/Scientific_computing

8. Výpočetní fyzika
9. Výpočetní inženýrství
10. Náročné výpočty
11. Výpočetní elektromagnetismus
12. Výpočetní dynamika tekutin
13. Výpočetní ekonomie
14. Environmentální simulace
15. Finanční modelování
16. Geografické informační systémy (GIS)
17. Strojové učení
18. Analýza sítí
19. Numerické předpovědi počasí
20. Rozeznávání obrazu

Další možný způsob rozdělení vědeckých výpočtů podle oborů lze rovněž nalézt v odborné literatuře:

- Počítačová dynamika tekutin
- Věda o atmosféře
- Seismologie
- Strukturální analýza
- Chemie
- Magnetická hydrodynamika
- Globální modelování oceánu/klimatu
- Environmentální studie
- Nukleární technika

Některé netradiční a nové vznikající oblasti vědeckých výpočtů

- Biologie
- Ekonomika
- Materiálový výzkum
- Zpracování obrazů v medicíně
- Zoologie

Z toho je vidět, že vědecké výpočty se dotýkají takřka věch oblastí vědy. V našem učebním textu se zaměříme na oblasti vědeckých výpočtů, které jsou spjaty s oblastí biologie a medicíny.

Kapitola 2
Stručná historie vědeckých výpočtů/výpočetní vědy
(Jiří Hřebíček)

2.1 Vývoj v zahraničí

Počátek vědeckých výpočtů/výpočetní vědy lze datovat od čtyřicátých let minulého století v souvislosti s vývojem balistických raket a atomových zbraní, kdy počítače ukázaly své možnosti a potenciál využitím jak starých, tak nově vyvinutých matematických metod, které vyžadovaly extrémní výpočty. Již v roce 1943 američtí vědci J. P. Eckert a J. V. Mauchly vyvinuli první programovatelný elektronkový počítač ENIAC, který byl využíván k výpočtům balistických drah raket, vodíkové bomby, ve výzkumu návrhu draků letadel ve větrném tunelu a v předpovědích počasí.

Vědecké výpočty se nejvíce rozvíjely ve Spojených státech, kde byly podporovány americkými vládymi. *Národní vědecká nadace USA* (NSF - National Science Foundation) začala v roce 1950 financovat výzkum, který umožnil zásadní vývoj v této oblasti. V té době firma IBM vyvinula model IBM 7030 prvních paralelních počítačů, které překonávaly problémy s operacemi v operační paměti pomocí tzv. pipeliningu[22]. V roce 1964 vytvořil geniální konstruktér Seymour Cray vektorový počítač Cray-1 s výkonem 133 MegaFlops[23], který jako první použil funkční víceprocesorový paralelismus. Tyto počítače byly postaveny speciálně pro náročné vědecké výpočty při použití tehdy nejmodernějších, avšak velmi drahých technologií. Jejich složitá architektura byla náročná na chlazení a byly velmi drahé, například Cray-1 stál nejméně 8,8 miliónu dolarů. Existovaly však i jiné americké vektorové počítače, například firmy CDC, dále takzvané minisuperpočítače firem jako CONVEX, KSR a Alliant.

V letech 1970 až 1980 se výpočetní věda zasloužila o superpočítačovou revoluci, vznik nových vysoce výkonných, víceprocesorových a paralelních počítačů a zasáhla celou řadu dalších odvětví vědy a kosmického a leteckého průmyslu, např. v komercionalizaci návrhů supersonických letadel. Byla to opět americká nadace NSF, která prostřednictvím svého úřadu pro náročné vědecké výpočty financovala od roku 1984 vytvoření pěti superpočítačových center (San Diego, Illinois, Pittsburgh, Cornell a Princeton), které sehrály důležitou roli na poli náročných výpočtů, počítačové vizualizace, grafiky a vývoje sítě NSFNET. Tato síť urychlila tempo dalšího technického zdokonalování při propojování novějších rychlejších superpočítačů prostřednictvím novějších rychlejších linek, znovu a znovu rozšiřovaných a zdokonalovaných v letech 1986, 1988, 1990, ...

V devadesátých letech nastal rozkvět výpočetní vědy, prudký rozvoj paralelních výpočtů, včetně rozšíření počítačů a jejich výpočetní mohutnosti, vývoje v jejich nasazení a důležitosti. Výpočetní věda se tak stala intelektuální disciplínou, výkonnou a nepostradatelnou analytickou metodou. Superpočítače japonských firem NEC, Hitachi, Fijitsu nebo německé Siemens-Nixdorf přinesly konkurenci a jisté zlevnění superpočítačů a v druhé polovině devadesátých let vytlačily z trhu vektorové modely firmy Cray, v té době už vlastněné firmou SGI. Vědci na univerzitě v Tokiu sestavili v roce 1996 tehdy nejrychlejší počítač pro vědecké

[22] Proudové zpracování - technika práce procesoru koncipovaná tak, aby v době, kdy jedna část procesoru provádí určitou fázi jedné instrukce, mohla jiná část procesoru pracovat na jiné fázi jiné instrukce.

[23] Výkon počítačů pro náročné výpočty se měří ve **Flops** - floting point operations per second, tedy v počtu operací v pohyblivé řádové čárce za sekundu. Používají se násobné předpony soustavy SI: Mega=10^6, Giga=10^9, Tera=10^{12}, Peta=10^{15}.

výpočty. Počítač dosahuje 1,08 TeraFlops (trilión operací v pohyblivé řádové čárce za sekundu). Japonští vědci Junichiro Makino a Makoto Taiji na tomto počítači GRAPE-4 provedli simulaci komplexních interakcí mezi astronomickými objekty, jako jsou hvězdy a galaxie. Tento typ simulace se označuje jako problém *N* těles, protože chování každého z *N* objektů je ovlivněno všemi ostatními objekty. Tento problém nelze řešit explicitně diferenciálními rovnicemi, ale pouze numericky. Takový výpočet byl velmi náročný na strojový čas. Počítač GRAPE-4 je 100krát rychlejší, než nejrychlejší počítače za posledních deset let. Dosud se pro problém *N* těles používaly rafinované efektivní algoritmy. GRAPE-4 používal 1692 procesorů, z nichž každý dosahuje rychlosti 640 MegaFlops. Tokijští vědci chtějí sestavit počítač s rychlostí 10^{15} operací za sekundu, kdy bude použito 20 tisíc procesorů s rychlostí 50 GigaFlops.

USA zavedly celní opatření, která prakticky zabránila dovozu superpočítačů z Japonska, což způsobilo v USA postupné zastarávání superpočítačů a mnoho výpočtů je dodnes prováděno mimo USA, hlavně vzhledem k vysokorychlostním sítím. Dále v USA začal vývoj alternativních superpočítačů, například heterogenní paralelně-vektorový počítač SV1 (a hlavně budoucí SV2) firmy SGI. Obecně se masivně paralelní[24] počítače s více než 64 procesory označují jako superpočítače. Mezi superpočítače se také někdy zařazují stovky osobních počítačů či pracovních stanic propojených sítí, které tvoří tzv. počítačovou farmu.
V roce 1997 NSF změnila podporu financování superpočítačových center v USA na financování *Partnerství pro rozvinutou výpočetní infrastrukturu* (PACI) pro aliance více než 50 akademických pracovišť v USA. Cílem bylo vybudovat prototyp další generace informační a výpočetní infrastruktury pro vysoce náročné výpočty. Za vládní podpory byl v USA od roku 1998 do 2004 realizován projekt masivně paralelních superpočítačů na bázi standardních procesorů firem Intel, IBM a SGI. Jsou to např. superpočítače IBM ASCI White, Intel ASCI Red, IBM ASCI Blue-Pacific, SGI ASCI Blue Mountain. V rámci tohoto projektu byla snaha vyvinout efektivní výpočetní modely pro určité konkrétní úlohy a získat z masivně paralelního superpočítače víc než obvyklých 10 % teoretického instalovaného výkonu. Takže již v roce 2000 v rámci grantu NSF pro tera výpočty bylo dosaženo v Pittsburghském superpočítačovém centrum výkonu, který přesáhl 6 TeraFlops.

2.1.1 Virtuální superpočítač World Community Grid

V listopadu 2004 byl spuštěn virtuální superpočítač World Community Grid[25], což je největší humanitární výpočetní sít světa, která využívá nevyužitý výpočetní výkon soukromých a firemních počítačů při řešení nejobtížnějších společenských problémů světa. Po celém světě se používá více než miliarda počítačů, z nichž každý se může potenciálně připojit ke světovému veřejnému gridu. Výpočetní sítě jsou rychle se rozvíjející technologií, která spojuje výkon tisíců a miliónů jednotlivých počítačů do obřího virtuálního systému s obrovskou výpočetní kapacitou. Technologie gridu umožňuje dosahovat takového výpočetního výkonu, který zdaleka přesahuje možnosti největších superpočítačů světa.

Společnost IBM poskytla své informační technologie a prostředky k provozu projektu **Help Defeat Cance**r[26] (*Pomozte porazit rakovinu*) v síti World Community Grid a spolupracuje od roku 2006 s vědci Univerzity lékařství a stomatologie New Jersey – Lékařské školy Roberta Wood Johnsona a Institutu rakoviny v New Jersey na tomto projektu, který uplatní možnosti superpočítačů v boji s rakovinou. Help Defeat Cancer je třetím projektem, který využije

[24] Masivně paralelní je více než 64 procesorový počítač s distribuovanou pamětí, kdy každý procesor má přímý přístup jen ke své paměti a o data z paměti jiného procesoru musí požádat, nějčastěji formou zprávy.
[25] http://www.worldcommunitygrid.org/
[26] http://pleiad.umdnj.edu/%7Ewill/IBM/index.html

výpočetní výkon virtuálního superpočítače World Community Grid. Projekt dá vědeckým pracovníkům příležitost simultánně analyzovat velké počty mikroskopických vzorků rakovinných tkání (TMA), a tak umožní za kratší dobu uskutečnit větší množství experimentů. Pro lepší představu „World Community Grid" umožňuje projektu Help Defeat Cancer analyzovat za jediný den takový počet vzorků, na jejichž analýzu by běžný počítač potřeboval zhruba 130 let.

Poznámka: Nevyužitý čas svého počítače může prostřednictvím projektu World Community Grid darovat každý, kdo si z jeho webu stáhne bezplatný software a zaregistruje se. Více než 300 000 jednotlivců nyní prostřednictvím sítě World Community Grid poskytuje výkon zhruba 400 000 počítačů pro rozvoj výzkumu rakoviny. K síti World Community Grid se mohou připojit počítače s operačními systémy Windows, Linux i Mac.

2.1.2 BOINC - Berkeley Open Infrastructure for Network Computing

Berkeley Open Infrastructure for Network Computing[27] (BOINC) je open source software pro distribuované výpočty, která umožňuje provozovat projekty jako je např. SETI@Home[28], který využívá internetového připojení počítačů pro hledání mimozemské inteligence (SETI). Tohoto projektu se může zúčastnit každý uživatel, bude-li používat open source software, který stahuje a analyzuje data z rádiového teleskopu. Existuje malá, ale významná pravděpodobnost, že právě počítač připojeného uživatele zachytí slabý šum mimozemské civilizace. SETI@Home je jedním nejznámějších projektů využívající BOINC.

BOINC je vyvíjen skupinou, která pochází z University of California v USA. BOINC je navržen jako otevřená struktura pro každého, kdo se chce zapojit do projektu distribuovaných výpočtů. Záměrem BOINCu je umožnit badatelům různých oborů, například molekulární biologie, klimatologie nebo astrofyziky, jednoduchý přístup do výpočetní sítě osobních počítačů na celém světě s ohromným výkonem. BOINC je založen na myšlence, že po světě běží naprostá většina počítačů nevyužita. Moderní operační systémy dokáží tento nevyužitý výkon spotřebovat, aniž by došlo k výraznému zpomalení aplikací, které uživatel používá. Většina projektů využívajících BOINC je neziskových a závisejí, pokud ne zcela, tak převážně na dobrovolnících[29]. To ale neznamená, že BOINC nemůže být použit pro zisk. BOINC sestává ze serveru a klientů, kteří spolu komunikují při distribuci pracovních jednotek. Každý klient pak jednu jednotku zpracuje a vrátí ji serveru, aby si posléze vyžádal další.

Z různých statistických důvodů BOINC zahrnuje také systém ohodnocení uživatelů. Tento systém také zabraňuje podvádění, a tím znehodnocení vědeckých výpočtů. Pro ohodnocování uživatelů se používá jednotka *cobblestone* pojmenovaném po Jeffu Cobbovi z projektu SETI@Home. Uživatel získá 100 cobblestonů, pokud jeho počítač s následujícími parametry pracoval 1 den: 1 miliarda operací za sekundu s čísly s plovoucí řádovou čárkou (založeno na benchmarku Whetstone[30]); 1 miliarda operací za sekundu s celými čísly (založeno na benchmarku Dhrystone[31]). V případě, že má uživatel rychlejší počítač, cobblestony mu přibývají rychleji, a naopak.

[27] http://boinc.berkeley.edu/
[28] http://setiathome.berkeley.edu/
[29] http://petrdrhlik.webzdarma.cz/veda_a_technika/distribuovane-vypocty/strucne-popisy-jednotlivych-projektu-boinc.htm
[30] http://freespace.virgin.net/roy.longbottom/whetstone.htm
[31] http://en.wikipedia.org/wiki/DMIPS

2.2 Vývoj v České republice

V České republice jsou nejvýkonnější počítače v akademické sféře rozmístěny na centrálních pracovištích velkých vysokých škol: Univerzity Karlovy (UK) a Českého vysokého učení technického (ČVUT) v Praze; Západočeské univerzity (ZČU) v Plzni; Vysokého učení technického (VUT) a Masarykovy univerzity (MU) v Brně, Vysoké škole báňské (Technická univerzita (VŠB-TU)) v Ostravě a v Českém hydrometeorologickém ústavu (ČHMÚ) v Praze. V roce 1996 vznikl projekt *META Centrum*, který sdružoval výše uvedené školy a od roku 1999 je součástí výzkumného záměru sdružení CESNET[32], kdy se na jeho realizaci podílí zejména ZČU, UK a MU. Hlavním cílem tohoto projektu bylo vytvoření virtuálního distribuovaného *META Počítače* a odpovídajícího podpůrného prostředí. To by mělo přispět jak k podstatně efektivnějšímu využití instalované výpočetní techniky, tak i k možnosti využít výpočetní zdroje na těchto školách pro řešení i velmi náročných vědeckých výpočtů, které kteréhokoliv samostatné pracoviště v České republice (ČR) nebylo schopno řešit.
Projekt *METACentrum* zastřešuje většinu aktivit souvisejících v ČR s gridy[33], superpočítači, klastrovým nebo gridovým počítáním a/nebo výkonnými výpočty počítáním obecně. Vlastním cílem projektu *METACentrum* je provoz a správa současných výpočetních zdrojů a v budoucnosti další rozšiřování výpočetní kapacity největších akademických center ČR. Systémy projektem spravované vytváří *virtuální distribuovaný počítač*, v poslední době označovaný jako *Grid*. V současné době jsou hlavními výpočetními středisky *METACentra*: Superpočítačové centrum Brno, Superpočítačové centrum UK a Západočeské superpočítačové centrum.

2.2.1 Projekt EGEE - Enabling Grids for E-sciencE

V současnosti se snahy ČR reprezentované sdružením CESNET soustřeďují na řešení projektu EGEE[34] (Enabling Grids for E-sciencE) sdružující experty z 45 zemí a 240 organizací [Matyska, 2006]. Výsledkem projektu EGEE má být funkční celoevropská gridová infrastruktura umožňující široké evropské vědecké komunitě využívání výpočetních zdrojů nemajících prozatím v evropském měřítku konkurence. Projekt EGEE/EGEE-II sdružuje experty z 32 zemí a 90 organizací s cílem využít současného rozvoje gridových technologií k vybudování servisní gridové infrastruktury dostupné evropské vědecké komunitě 24 hodin denně. Jeho cílem je poskytnout vědcům z akademické sféry i průmyslu přístup k výpočetním zdrojům nezávislým na jejich zeměpisné poloze. Projekt se také zaměřuje na oslovení širokého spektra nových potenciálních uživatelů gridu. Projekt se primárně soustřeďuje na čtyři klíčové oblasti:

- vybudovat konzistentní, robustní a bezpečný grid (síť výpočetních kapacit),
- umožnit jednoduchou integraci nových dostupných výpočetních zdrojů,
- kontinuálně zlepšovat a udržovat middleware s cílem poskytovat spolehlivý servis uživatelům,
- získat nové uživatele z průmyslu a akademické sféry a zajistit jim vysoký standard uživatelské podpory, který potřebují.

Biomedicína je hlavní aplikační oblastí projektu EGEE. V této oblasti již bylo nasazeno nebo je sem právě převáděno 23 aplikací. Oblast se dělí na tři podoblasti: *zpracování lékařských*

[32] Sdružení CESNET založily vysoké školy a Akademie věd České republiky v roce 1996. Jeho hlavním cílem je provozovat a rozvíjet páteřní akademickou počítačovou síť České republiky. Současná generace této sítě se nazývá CESNET2 a nabízí na páteřních trasách přenosové rychlosti v řádu gigabitů za sekundu.
[33] Gridem rozumíme dynamický virtuální výpočetní, informační či znalostní systém tvořený soustavou především výkonných počítačů propojených vysokorychlostní sítí a poskytujících datové a další služby. Grid je tedy infrastruktura, určená pro řešení nejnáročnějších výpočetních a datových problémů v kterékoliv odborné oblasti
[34] http://www.eu-egee.org/

obrazových materiálů, biomedicína a výzkum léčiv. V každé podoblasti již bylo na bázi infrastruktury EGEE realizováno mnoho jednotlivých aplikací. Tyto aplikace zatěžují middleware specifickými požadavky, které se týkají především bezpečnosti (citlivost dat), správy dat (komplexní datové struktury a distribuce) a realizace velkého množství náročných malých úloh. Biomedicínské aplikace se již staly pravidelnými uživateli infrastruktury (za měsíc je realizováno přibližně 15 000 úloh) a výpočetně náročné analýzy příprav molekul („molecular docking“) sloužící k výzkumu nových léků, které by vyžadovaly 80 let výpočtů na jednom procesoru, byly vyřešeny za jediný měsíc.

2.2.2 Superpočítačové Centrum Brno

Superpočítačové Centrum Brno (SCB) vzniklo v roce 1994, v rámci Ústavu výpočetní techniky MU a bylo pověřeno koordinací národních superpočítačově orientovaných aktivit v rámci projektu Fondu rozvoje VŠ. Hlavním cílem SCB je poskytovat uživatelům z MU přístup k výkonné výpočetní technice umožňující účinně řešit vědecké výpočty z různých oblastí. Kromě běžného provozu výpočetní infrastruktury SCB se podílí na rozvoji výpočetních a datových kapacit, a to jak v rámci ČR, tak i v mezinárodním měřítku. Národní spolupráce je realizována v rámci projektu (či aktivity) METACentrum výzkumného záměru sdružení CESNMET, do níž je SCB intenzivně zapojeno a hraje v něm vedoucí roli. Právě aktivita METACentrum vytváří zmíněný národní Grid, v této fázi virtuální počítač, který umožňuje efektivní sdílení výpočetní techniky instalované v rámci předcházejících superpočítačových projektů na různých vysokých školách ČR, a současně umožňuje řešit úlohy ve vědeckých výpočtech, které svými požadavky (na paměť, výkon centrálního procesoru aj....) přesahují možnosti jednotlivých dílčích superpočítačových center (uzlů gridu).

Kapitola 3
Stručný přehled technologií pro vědecké výpočty v biologii a biomedicíně

(Jiří Hřebíček)

V první kapitole jsme uvedli dva důležité pojmy svobodný software a open source software, které se týkají vědeckých výpočtů na internetu a ve druhé kapitole jsme shrnuli historii vědeckých výpočtů v zahraničí, v České republice i na MU, které se týkají možností využití dostupného hardware pro náročné výpočty. V této kapitole stručně shrneme možnosti řešení problémů v oblasti biologie a biomedicíny pomocí počítačových systémů.

3.1 Vědecké výpočty v bioinformatice

Bioinformatika je vědní disciplína, která se zabývá metodami pro shromažďování, organizování, archivování, analýzu a vizualizaci rozsáhlých souborů biologických, biomedicínských a zdravotnických dat, zejména molekulárně-biologických dat [Cvrčková 2006].
Výpočetní biologie, (*Computational Biology*) je vědní disciplína, která se zabývá vývojem a aplikací metod analýzy dat, teoretických metod, matematickým modelováním, technikami počítačové simulace ve studiu chování biologických a společenských systémů. Její vývoj je pozitivně ovlivňován společností *International Society for Computational Biology*[35].

Vědecké výpočty v bioinformatice se prolínají s výpočty v dalších příbuzných biologických oborech, jako např. molekulární biologií, genomikou, proteomikou, genetikou, výpočetní biologií, matematickou biologií, systémovou biologií, teoretickou biologií, dále pak biomedicínskou informatikou, biomedicínským inženýrstvím, výpočetní chemií, informatikou a počítačovou lingvistikou.

Souvislost vědeckých výpočtů v bioinformatice a výpočetní biologii spočívá v užití společných matematických metod, aby byly získány užitečné informace z dat, která jsou získávána prostřednictvím výkonných biologických technik, např. v analýze **genomu**[36]. Oblast technik analýzy genomu sestává z vývoje nových technologií a také zlepšování těch současných, široce používaných technologií napomáhajících analýze komplexní genové informace, jako jsou např. Microarrays, SKY, PCR, CGH atd. Tyto zahrnují především takové techniky, které se zabývají analýzou DNA a její struktury, ale i metody zabývající se analýzou genové exprese (Differential Display, SAGE, RDA …) a proteinů, které by mohly být užitečné při objasňování funkce genomu.

3.2 Nejpoužívanější biologické databáze a formáty dat v bioinformatice

Mnoho databází, programů a analýz týkající se geonomu je přístupných veřejně na Internetu, v České republice je např. v ÚMG AV ČR organizována databáze užitečných bioinformatických linek[37]. Velká bioinformatická centra v Evropě jako European

[35] http://www.iscb.org/

[36] Genom je veškerá genetická informace uložená v DNA (u některých virů v RNA) konkrétního organismu. Zahrnuje všechny geny a nekódující sekvence. Přesněji řečeno, genom organismu je kompletní sekvence DNA (popř. RNA) jedné sady chromozómů. Termín genom může být chápán úžeji jako kompletní výčet veškeré jaderné DNA, avšak také může být do něj zahrnuta navíc i veškerá DNA těch organel (semiautonomní organely - mitochondrie, chloroplasty,.), které dědí svou vlastní DNA.

[37] http://bio.img.cas.cz/links/

Bioinformatic Institute[38] a Expert Protein Analysis System[39] nabízejí celou řadu zajímavých služeb a databází. V USA je jedním z nejvíce využívaných zdrojů informací National Center for Biotechnology Information[40]. Na Internetu lze nalézt další databáze a analýzy lidského, šimpanzího a dalších genomů[41],[42]. Na Přírodovědecké fakultě UK lze nalézt podrobnější informace v elektronických učebních textech předmětu Bioinformatika[43].

3.2.1 EMBL

Databáze EMBL[44] je organizována Evropskou molekulárně biologickou laboratoří (EMBL - European Molecular Biology Laboratory). Je to veřejná evropská primární nukleotidová databáze, která je vytvářena v součinnosti s ostatními nukleotidovými databázemi GENBANK (USA) a DDBJ (Japonsko). Je velmi dobře přístupná spolu s mnoha odvozenými a dalšími databázemi přes SRS[45] (Sequence Retrieval System) vyhledávací systém. Databáze obsahuje všechna data zaslaná vědeckou komunitou, a to bez kontroly. Z tohoto důvodu může obsahovat určité procento chyb.

3.2.2 GenBank

Distribuční formát nukleotidové **databáze GenBank**[46], který je podobný formátu databáze EMBL, je však lépe čitelný – namísto dvou písmenného identifikátoru se používá celé slovo. Databáze GenBank je nukleotidová databáze, kterou organizuje od roku 1988 Národní centrum pro biologické informace[47] (NCBI – National Center for Biological Information) v USA. V důsledku výměnné spolupráce s ostatními nukleotidovými databázemi obsahuje v zásadě táž data jako EMBL. GenBank je výborně propojena s mnoha dalšími databázemi a výše uvedená adresa je výchozím bodem hledání pro velkou část vědecké komunity. Bohužel, při práci je třeba mít na paměti, že (stejně jako EMBL) neobsahuje všechny dostupné sekvence, hlavně z velkých genomových projektů.

3.2.3 DDBJ

Databáze **DDBJ**[48] - DNA datová databanka Japonska (DNA Data Bank of Japan) zahájila svou činnost v roce 1986 v Národním ústavu genetiky (National Institute of Genetics - NIG). V současné době je DDBJ samostatná DNA databanka v Japonsku, která shromažďuje DNA sekvence hlavně od japonských vědců i vědců z jiných zemí v distribučních formátech EMBL/EBI a GenBank/NCBI. Tyto tři databanky sdílí virtuálně táž data v daném čase.

3.2.4 Swiss-Prot

Swiss-Prot[49] databáze má formát proteinových sekvencí podobný formátu EMBL. Databáze Swiss-Prot je anotovaná proteinová databáze organizovaná hlavně Švýcarským

[38] http://www.ebi.ac.uk/
[39] http://www.expasy.org/
[40] http://www.ncbi.nlm.nih.gov/
[41] http://genome.ucsc.edu/
[42] http://www.ensembl.org/index.html
[43] http://bio.img.cas.cz/PrfUK2003/
[44] http://www.ebi.ac.uk/embl/
[45] http://www.biowisdom.com/navigation/srs/srs
[46] http://www.ncbi.nlm.nih.gov/Genbank/GenbankOverview.html
[47] http://www.ncbi.nlm.nih.gov/
[48] http://www.ddbj.nig.ac.jp/Welcome-e.html
[49] http://www.expasy.org/sprot/

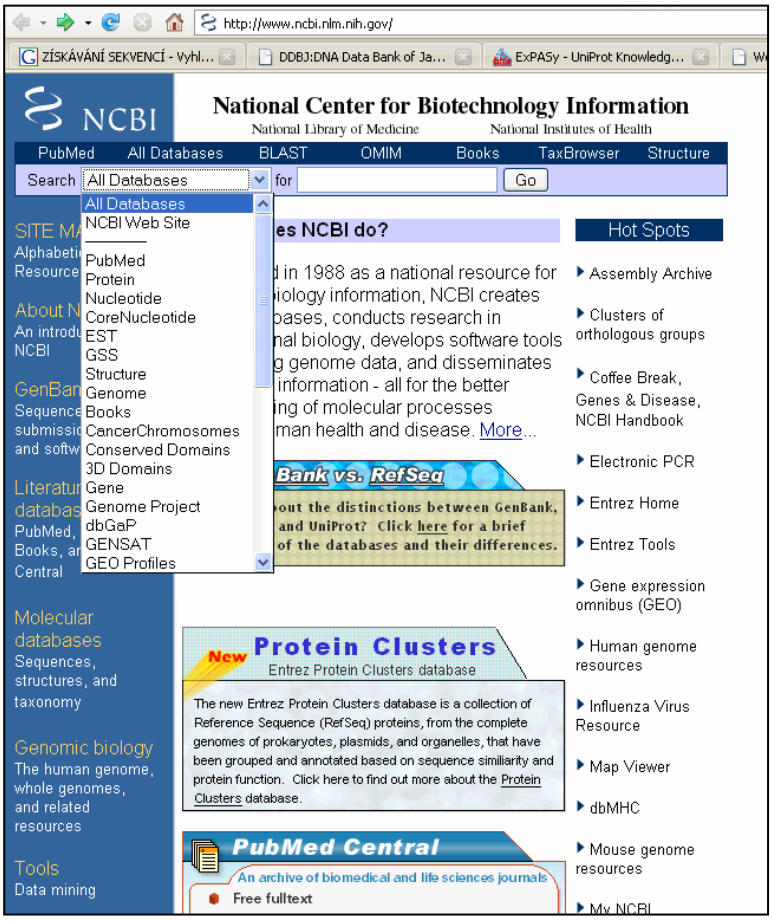
bioinformatickým institutem (Swiss Institute of Bioinformatics - SIB)[50]. Úzce spolupracuje s EMBL a společně vytvářejí EMBL, také proteinovou databázi. Tyto dvě databáze dohromady pokrývají všechny "existující" či "smysluplné" proteinové sekvence. Autoři z literatury průběžně doplňují nové informace a v současné době se pravděpodobně jedná o nejkvalitnější molekulárně-biologickou databázi. Pro vědecké použití je Swiss-Prot volně k dispozici.

3.2.5 PIR

Stejně jako v případě EMBL/Swiss-Prot je i formát PIR formát používán pro proteiny. Je organizován je podobně jako GenBank v NCBI. Zde s dalšími organizacemi vytváří PIR-International[51] anotovanou databázi analogickou Swiss-Prot.

3.3 Počítačová analýza velkých souborů nukleotidových sekvencí

Vědecké výpočty v bioinformatice se mimo jiné věnují zpracování a počítačové analýze velkých souborů nukleotidových sekvencí, jaké jsou generovány například právě v genomových projektech a na ní navazující analýze aminokyselinových sekvencí proteinů.



Obrázek č. 3.1: Vyhledávání podle klíčových slov

Pro získání úplné nukleotidové sekvence genomu se skládají do kontinuálních lineárních řetězců nukleotidové sekvence získané sekvenováním jednotlivých klonů. V typickém případě lze stanovit jedním sekvenováním pořadí několika set až jednoho tisíce nukleotidů. Z

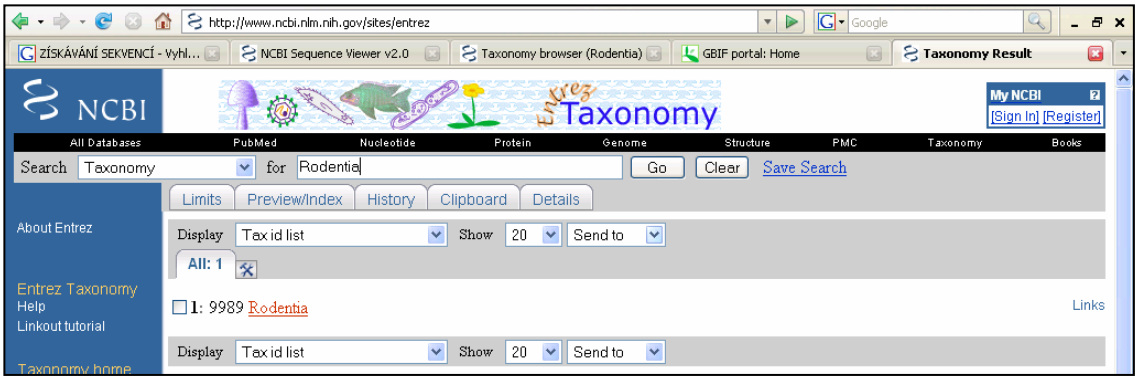
[50] http://www.isb-sib.ch/
[51] http://www-nbrf.georgetown.edu/

takových parciálních sekvencí se pak postupně skládají delší a delší řetězce (tzv. kontigy) až v ideálním případě získáme celou sekvenci, například sekvenci genomu nebo velkého úseku DNA který studujeme. To jsou první počítačové operace v sekvenačních projektech.

3.3.1 Vyhledávání sekvencí

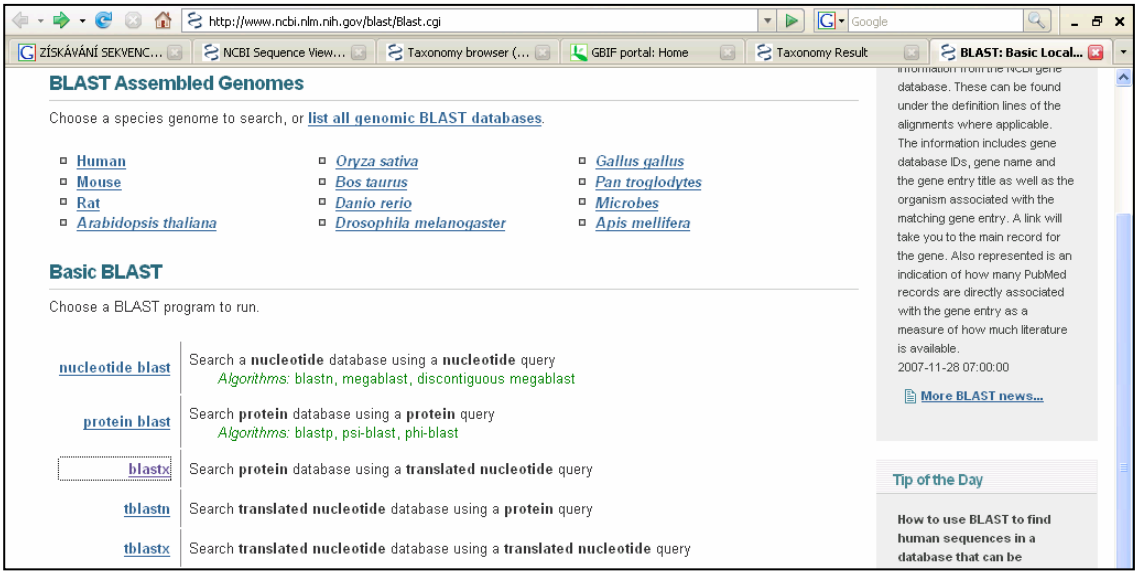
Na internetu je celá řada zdrojů informací, kde je možno najít příslušné sekvence, které jsme uvedli v předchozím odstavci: EMBL, GenBank a DDBJ. Tyto databáze data vzájemně sdílejí, a proto stačí vyhledávat podle klíčových slov jen v jedné z nich, např. GenBank.

Příklad 3.1:
Rodentia[Organism] AND 12S rRNA[Gene Name]
Názvy taxonů musí odpovídat názvům použitým v taxonomické databázi GenBank[52] nebo je možno používat identifikační čísla z této taxonomické databáze (v případě Rodentia je to Tax id 9989).



Obrázek č. 3.2: Vyhledávání v taxonomické databázi

Nevýhodou tohoto způsobu vyhledávání sekvencí je, že se nenaleznou geny, jejichž název v anotaci se liší od zadaného názvu (synonymum nebo chybně anotovaný gen).



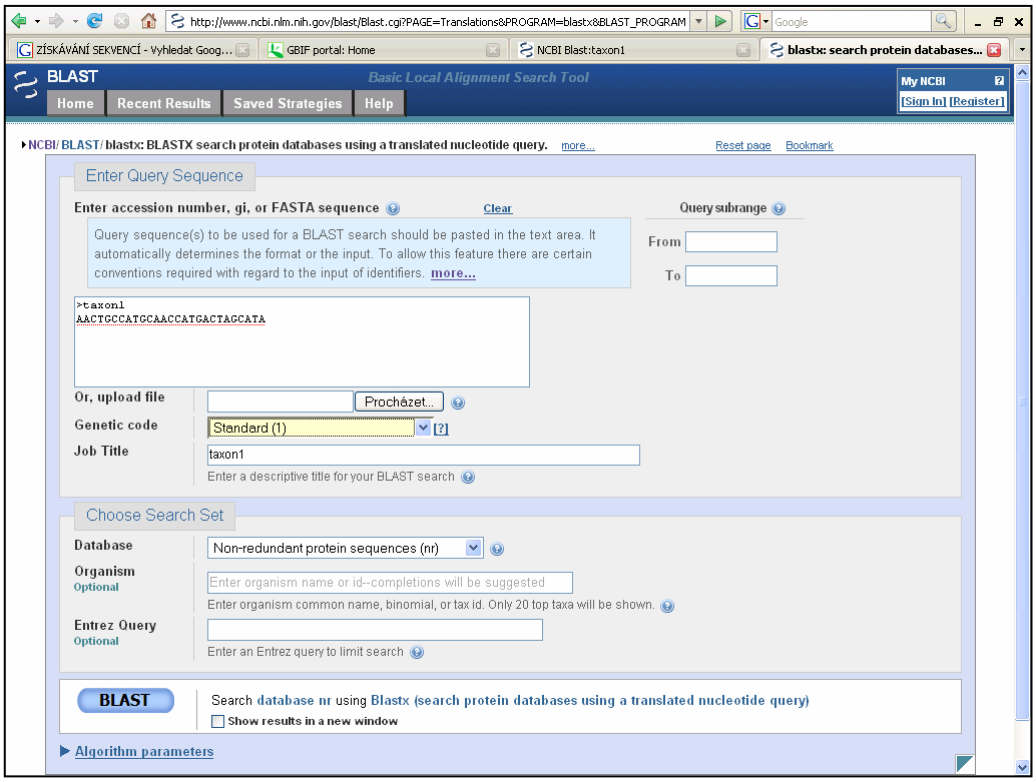
Obrázek č. 3.3: Vyhledávání podle sekvence (BLAST)

[52] http://www.ncbi.nlm.nih.gov/sites/entrez?db=taxonomy

Další možnost vyhledávaní v databázi GenBank je vyhledávání podle sekvence (BLAST)[53]. Základní druhy vyhledávání v BLAST, viz obr. č. 3.3, jsou:

- Nukleotide blast – dotazem na sekvenci nukleotidů prohledává nukleotidové databáze;
- Protein blast – dotazem na sekvenci proteinů prohledává proteinové databáze;
- blastx – dotazem na sekvence nukleotidů prohledává proteinové databáze;
- tblastn – dotazem na sekvence proteinů prohledává nukleotidové databáze;
- tblastx – dotazem na sekvence nukleotidů prohledává nukleotidové databáze.
-

Volbou blastx se objeví zadávací formulář, viz obr. 3.4, kam je nutno zadat v poli Enter Query Sequence údaje ve formátu FASTA (základní, možný vstupní formát u mnoha programů na tvorbu alignmentu), např.
>taxon1
AACTGCCATGCAACCATGACTAGCATA



Obrázek č. 3.4: Zadávací formulář BLASTX

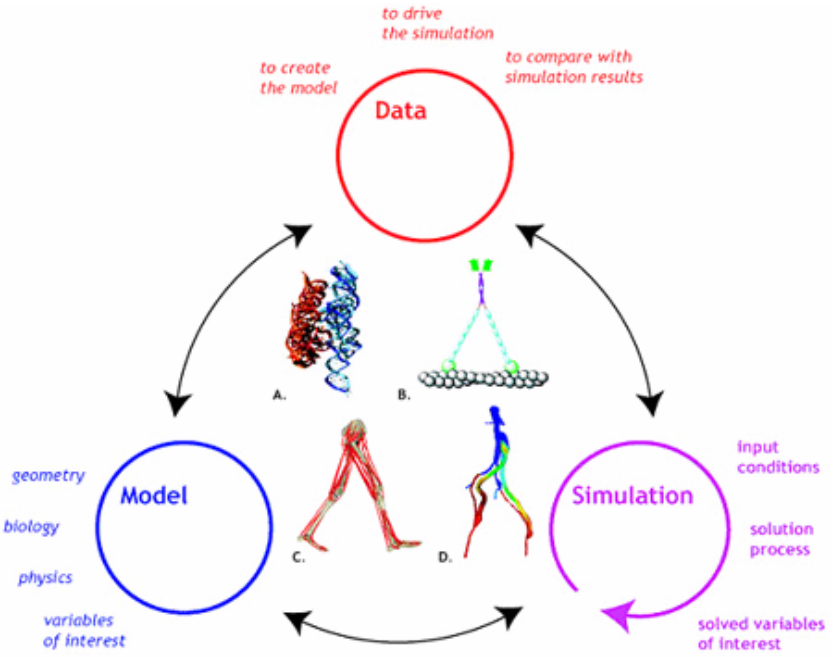
Nevýhodou tohoto způsobu vyhledávání sekvencí je, že se mezi nalezenými sekvencemi se mohou (a většinou budou) vyskytovat i jiné geny, které jsou vašemu genu podobné (např. jiní členové genové rodiny). Odkaz *Taxonomy reports* umožňuje snadnou orientaci v taxonomickém zařazení nalezených položek.

3.4 Vědecké výpočty v biomedicíně

Biomedicínské výpočty kombinují diagnostiku a výzkum v biologii a zdravotnických vědách s možnosti a schopnostmi moderních vědeckých výpočtů a spadají do oblasti

[53] http://www.ncbi.nlm.nih.gov/blast/Blast.cgi

biomedicínckého inženýrství, viz Biomedical Computational Review[54]. V této nově se rychle rozvíjející disciplíně jsou počítače využívány, aby urychlily výzkum a studium pacientova chování pomocí simulačních metod a vizualizovaly složité biologické modely. Tím se zkracují cykly ve zdravotnickém výzkumu, právě tak, jako rozšiřují jeho hranice poznání. Biomedicínckě výpočty, to není jen pouhé použití počítačů v laboratořích k řízení diagnostických přístrojů a zařízení, ale představují integraci nového způsobu myšlení a logického uvažování do tradičních prakticky orientovaných lékařských oborů a věd o živé přírodě. Centrem výzkumu biomedicínckých výpočtů je National Institutes of Health (NIH), NIH Center for Biomedical Computing[55] na universitě ve Standfordu v USA, se kterým spolupracují další centra biomedicínských výpočtů v USA[56].



Obrázek č. 3.5: Výzkum biomedicínckých výpočtů v NIH Center for Biomedical Computing

Moderní biomedicínské výpočty jsou zaměřeny do širokého pole aplikačních oblastí, jako jsou[57]:

- aplikace inženýrské systémové analýzy (fyziologické modelování, simulace, řízení) na biologické problémy;
- detekce, měření a monitorování fyziologických signálů (tj. biosenzory a biomedicínská instrumentace);
- diagnostická interpretace pomocí metod zpracování signálů aplikovaných na bioelektrická data;
- terapeutické a rehabilitační procedury a přístroje (rehabilitační inženýrství);
- řízení přístrojů pro náhradu či posílení funkcí těla (umělé orgány);
- počítačová analýza dat pacientů a klinické rozhodování (tj. lékařská informatika a umělá inteligence);

[54] http://biomedicalcomputationreview.org/
[55] http://simbios.stanford.edu/index.html
[56] http://www.bisti.nih.gov/ncbc/
[57] http://biomedicalcomputationreview.org/archive.html

- lékařské zobrazování (tj. grafické znázornění anatomických detailů nebo fyziologických funkcí);
- vytvoření nových biologických produktů (tj. biotechnologie a tkáňové inženýrství).

3.4.1 Vědecké výpočty v rámci projektu EGEE

Superpočítačové centrum Brno na Masarykově univerzitě se podílí na řešení projektu EGEEII financovaného EU, přičemž níže naleznete přehled aplikací z oblasti biomedicíny, které umožňují provádět vědecké výpočty v infrastruktuře projektu EGEE[58].

Sektor **pořizování a zpracování lékařských obrazových materiálů** zahrnuje počítačovou analýzu digitálních lékařských obrazových materiálů. Patří sem sdílení lékařských dat, výpočetně náročné lékařské procedury, zpracování velkých datových sad a statistické studie u velkých populačních vzorků. Jedná se o následující aplikační software:

- **GATE**[59] je simulátor fungující na principu Monte Carlo, který umožňuje plánovat radioterapii na základě snímků pacienta. Gridovou infrastrukturu EGEE využívá ke snížení času potřebného k realizaci Monte Carlo simulací na úroveň, která by měla z klinického hlediska smysl.
- **CDSS**[60] (Clinical Decision Support System) využívá obrazovou klasifikaci založenou na odborné znalosti k tomu, aby pomohl při klinických rozhodnutích. Grid je využíván ke shromažďování velkých objemů dat i k efektivnímu učení klasifikačního softwaru na základě těchto rozsáhlých datových vzorků.
- Aplikace **Pharmacokinetics**[61] studuje difúzi kontrastního činidla v játrech pomocí sekvence snímků získaných magnetickou rezonancí. Artefakty způsobené pohybem pacienta znemožňují přímé srovnání snímků. Paralelizované výpočty návaznosti snímků řešené pomocí gridu však dovolují sekvenci analyzovat v rozumném čase.
- **SiMRI3D**[62] je simulace obrazových materiálů získaných magnetickou rezonancí. Slouží k tvorbě umělých, ale realistických trojrozměrných snímků, které mohou být použity při analýze snímků z dokonale známých zdrojů, ke studiu artefaktů a k dalšímu rozvoji a optimalizaci sekvencí magnetické rezonance.
- Aplikace **gPTM3D**[63] umožňuje interaktivní rekonstrukci trojrozměrných lékařských snímků, např. objemovou rekonstrukci velkých nebo složitých orgánů. Kvalita služeb potřebná pro interaktivitu znamená, že v některých místech gridu musí být pro tuto třídu úloh definována vysoká priorita.
- **Bronze Standard**[64] je aplikace, která vyhodnocuje algoritmy řešící registraci (přenos mezi různými souřadnými soustavami) lékařských obrazových materiálů. Objem zpracovávaných dat a cena výpočtů jsou mimo možnosti standardních počítačů, ale aplikaci lze snadno distribuovat v prostředí gridu.

Sektor **bioinformatiky** se zabývá analýzou genových sekvencí. Zahrnuje genomiku, proteomiku a fylogenezi. Jde o následující aplikační software:

- **Grid Protein Sequence Analysis**[65], **GPS@**, je webový portál, který poskytuje uživatelsky přívětivé rozhraní bioinformatickým prostředkům v gridové infrastruktuře

[58] http://public.eu-egee.org/applications/biomed.html
[59] http://opengatecollaboration.healthgrid.org/
[60] http://egee-na4.ct.infn.it/biomed/CDSS.html
[61] http://egee-na4.ct.infn.it/biomed/pharmacokinetics.html
[62] http://egee-na4.ct.infn.it/biomed/SiMRI3D.html
[63] http://egee-na4.ct.infn.it/biomed/gPTM3D.html
[64] http://egee-na4.ct.infn.it/biomed/BronzeStandards.html
[65] http://gpsa.ibcp.fr/

EGEE. Prototyp portálu GPS@ je již on-line a poskytuje rozhraní pro 13 programů s podporou gridové infrastruktury z celkových 46 programů původního portálu.

- **xmipp_MLrefine**[66] slouží k trojrozměrné strukturální analýze velkých makromolekulárních komplexů. V rekonstrukčním procesu je zkombinováno mnoho obrazových materiálů získaných elektronovou mikroskopií, které odpovídají různým pohledům na vzorek. Snímky však obvykle trpí vysokou hladinou šumu, a proto je k nalezení nejpravděpodobnějšího modelu, jenž popisuje experimentální data, potřeba mnoho iterací. Snímky získané elektronovým mikroskopem podléhají vlivu mnoha forem aberace. Matematicky je rozdíl mezi teoretickou projekcí a skutečnou experimentální projekcí modelován pomocí přenosové funkce kontrastu (Contrast Transfer Function – CTF). Chceme-li zjistit skutečný tvar funkce CTF, která ovlivňuje experimentální snímky, je třeba použít simulaci – metodu Xmipp_assign_multiple_CTFs.
- **SPLATCHE**[67] (SPatiaL And Temporal Coalescences in Heterogeneous Environment) je buněčný nástroj pro modelování evoluce genomu. Umožňuje rekonstruovat globální šíření prapůvodního člověka v genograficky realistické krajině a generovat molekulární diverzitu různých lidských populací.

Sektor **výzkumu léků** se snaží přispět ke zrychlení procesu hledání nových léčiv pomocí počítačových (*in silico*) simulací proteinových struktur a dynamiky. Jde o následující aplikační software:

- Aplikace **WISDOM**[68] realizuje náročné výpočty v rámci počítačového (in silico) výzkumu léků proti nově vznikajícím a opomíjeným chorobám. Tyto výpočty přichytávání molekul určují, jak dobře určité léky přilnou na specifická místa cílových virů – u těch, které se přichytí dobře, je vyšší pravděpodobnost, že budou vůči virům aktivní. Výpočty již byly úspěšně realizovány v případě malárie a ptačí chřipky, a do budoucna jsou plánovány další cílové choroby.
- **GridGRAMM**[69] je jednoduché rozhraní pro realizaci molekulárního přichytávání na webu. Výsledky zahrnují kvalitativní skóre a různé metody přístupu k trojrozměrné struktuře komplexu. Molekulární přichytávání lze využít ke studiu molekulárních interakcí, a tedy k analýze interakcí mezi enzymem a substrátem. To je užitečné pro návrh léčiv a snahu porozumět patologickým mutacím.
- Cílem projektu **GROCK**[70] (Grid Dock) je poskytnout jednoduchý způsob, jak provádět masový rozbor molekulárních interakcí pomocí webu. Uživatelé mají možnost analyzovat jednu molekulu vůči celé databázi známých struktur.

Infrastrukturu EGEE lze využít dalším možným aplikacím ve vědeckých výpočtech. Více informací o tom, jak se zúčastnit projektu EGEE II, a další informace o aplikacích, které běží na infrastruktuře EGEE, naleznete na portálu EGEE pro uživatele a aplikace[71].

3.4.2 Biomedicínské výpočty ve výzkumu

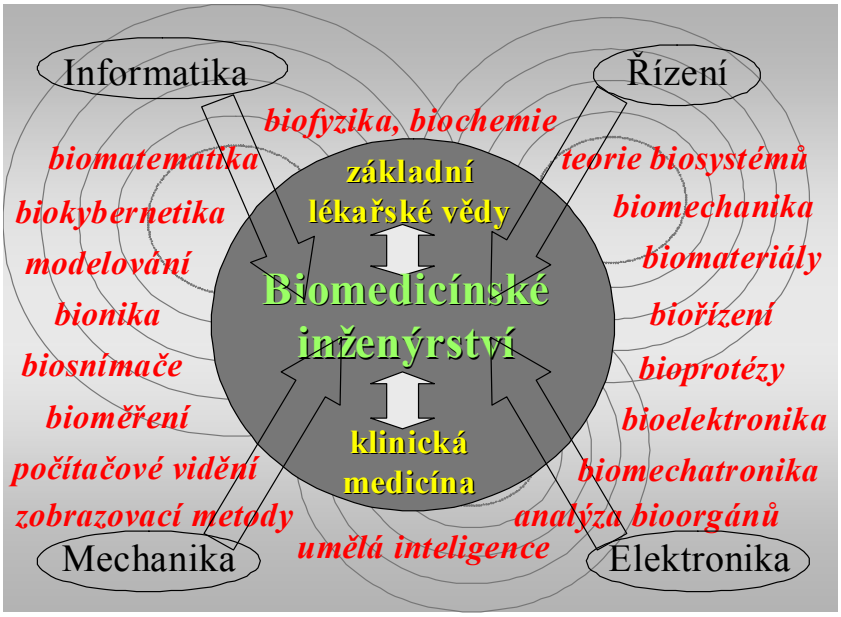
V současné době biomedicínské výpočty se využívají v následujících oblastech výzkumu:

- výzkum nových materiálů a konstrukcí pro implantované umělé orgány a skeletární náhrady;

[66] http://egee-na4.ct.infn.it/biomed/xmipp_MLrefine.html
[67] http://cmpg.unibe.ch/software/splatche/
[68] http://wisdom.eu-egee.fr/
[69] http://egee-na4.ct.infn.it/biomed/GridGRAMM.html
[70] http://egee-na4.ct.infn.it/biomed/GROCK.html
[71] http://egeena4.lal.in2p3.fr/

- návrh instrumentace pro sportovní lékařství;
- vývoj nových dentálních materiálů a konstrukcí;
- návrh komunikačních pomůcek pro postižené;
- studium dynamiky plicní tekutiny;
- studium biomechaniky lidského těla;
- vývoj materiálů použitelných jako náhrada lidské kůže;
- vývoj nových diagnostických nástrojů pro analýzu krve;
- počítačové modelování funkce lidského srdce a mozku;
- tvorba nových metod a aplikačního softwaru pro analýzu lékařských dat;
- analýza bezpečnosti lékařských přístrojů;
- vývoj nových diagnostických zobrazovacích systémů;
- návrh telemetrických systémů pro monitorování pacientů;
- návrh biomedicínských senzorů pro měření proměnných veličin v lidském fyziologickém systému;
- vývoj znalostních systémů pro diagnostiku;
- aplikace metod strojového učení na podporu diagnostiky;
- návrh zpětnovazebních řídicích systémů pro dávkování léků;
- modelování fyziologických systémů lidského těla.

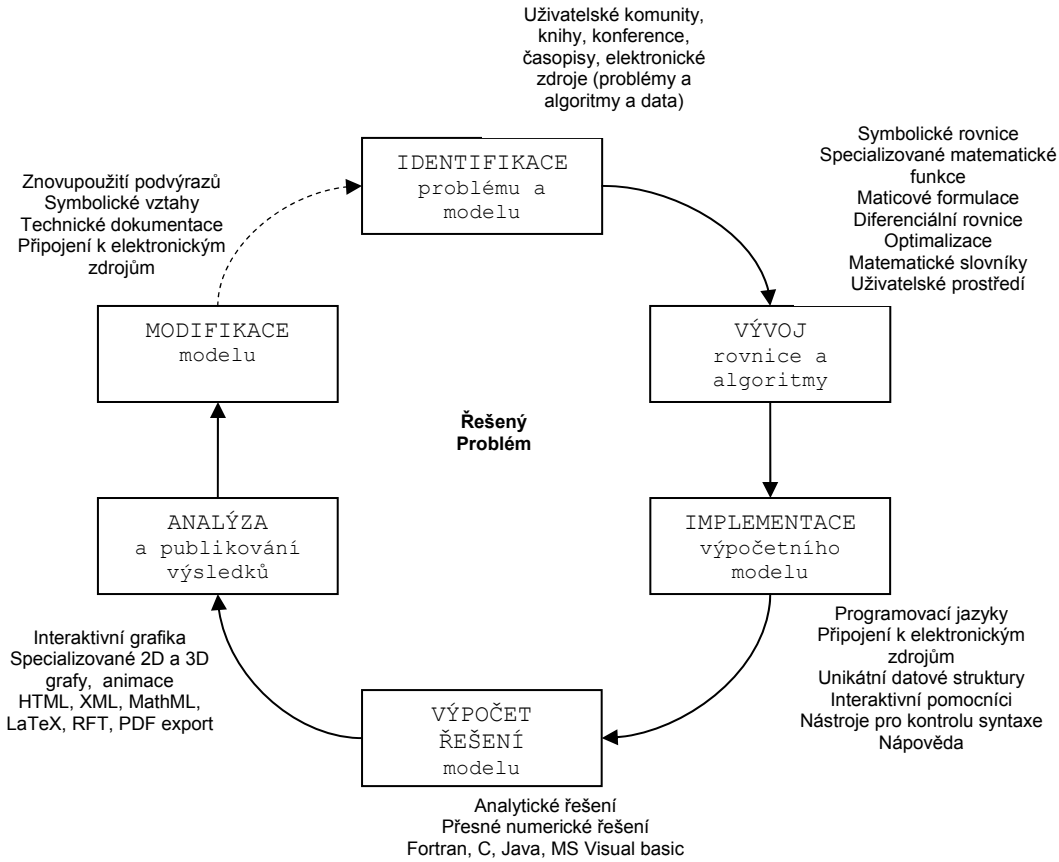
Biomedicínské výpočty jsou součástí biomedicínské inženýrství, což je interdisciplinární vědní obor, který souvisí s matematickou biologií (biomatematika), obr. 3.6.



Obrázek č. 3.6: Biomedicínské inženýrství

Kapitola 4
Vědecké výpočty s využitím Maple
(Jiří Hřebíček)

Současné vědecké výpočty v biologii a biomedicíně souvisí s řešením reálných problémů pomocí informačních a komunikačních technologií (ICT). Požadavky na vědecké výpočty, na jejich vysokou přesnost, maximální vizualizaci a interaktivní komunikaci s řešitelem atd. – vedly k vytvoření univerzálních komplexních programových systémů, jako jsou např. systémy Maple[72] od Maplesoft Inc. (Kanada), MathCAD[73] od PTC Corporate Headquarters, (USA) Mathematica[74] od Wolfram Reasearch, Inc. (USA), MuPAD[75] od SciFace Software GmbH & Co. KG (Německo) atd. Tyto systémy lze využít ve vědeckých výpočtech během celého procesu řešení problému, tj.: jeho identifikace, analýzy, vývoje, implementace, výpočtu, vizualizace, simulace, příp. optimalizace řešení, zpětné vazby, odezvy a ověřování, případně možné modifikace matematického modelu znázorněného na obr. č. 4.1.



Obrázek č. 4.1: Interaktivní proces řešení problému ve vědeckých výpočtech

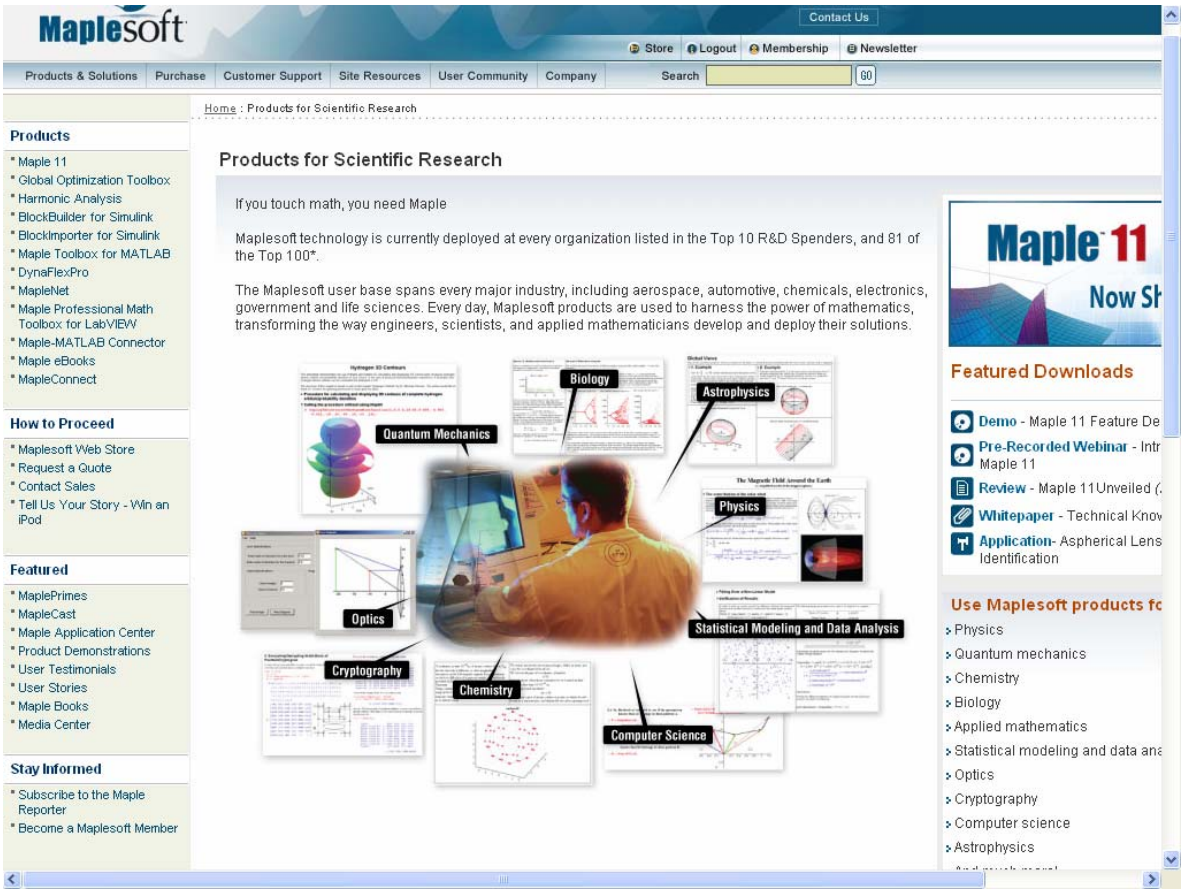
Na obr. č. 4.1 lze vidět, že vědecké výpočty tvoří interaktivní proces vycházející z principů matematického modelování s četnými zpětnými vazbami, který se několikrát opakuje

[72] http://www.maplesoft.com/
[73] http://www.ptc.com/appserver/mkt/products/home.jsp?k=3901
[74] http://www.wolfram.com/
[75] http://www.mupad.de/products/

(Hřebíček, Škrdla, 2006). V tomto modelu se neomezujeme pouze na výpočty s využitím ICT, ale chápeme jej obecněji, jako proces přípravy výpočtu, jeho implementace a efektivní realizace , analýzy výsledků a případné modifikace a opětovného provedení výpočtu s cílem vyšší účinnosti či přesnosti.

4.1 Maple

Počítačový systém Maple je jednou z možných informačních a komunikačních technologií, kterou lze velmi efektivně zapojit do výukového procesu v oblasti vědeckých výpočtů k modelování i simulací v biologii a biomedicíně, neboť Masarykova univerzita má zakoupenu jeho multilicenci. Maple poskytuje nepřeberné množství funkcí pro použití základních (diferenciální a integrální počet) i náročnějších matematických struktur (obyčejné, parciální, algebro-diferenciální rovnice, optimalizace aj.). Umožňuje velmi přirozenou a intuitivní formou při jeho obsluze provádění jednoduchých i složitých vědeckých výpočtů. Na obr. č. 4.2 jsou vidět oblasti vědeckých výpočtů, kde lze Maple použít ve výzkumu.

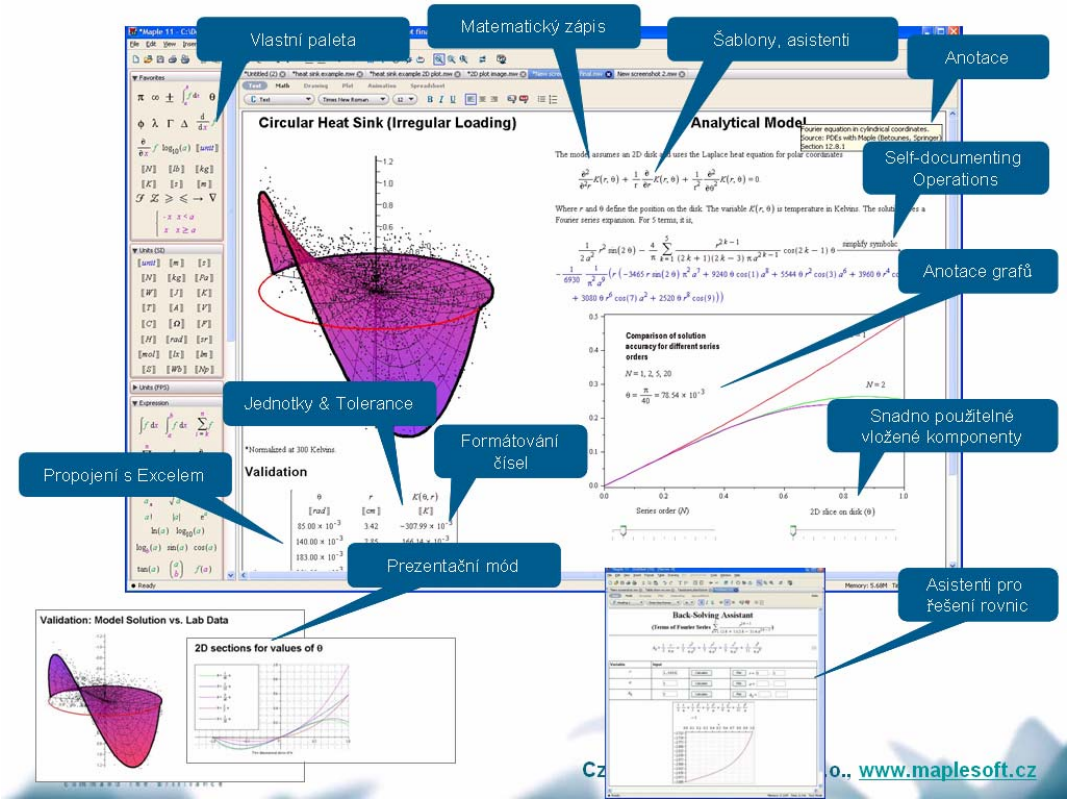


Obrázek č. 4.2: Využití Maple 11 ve výzkumu

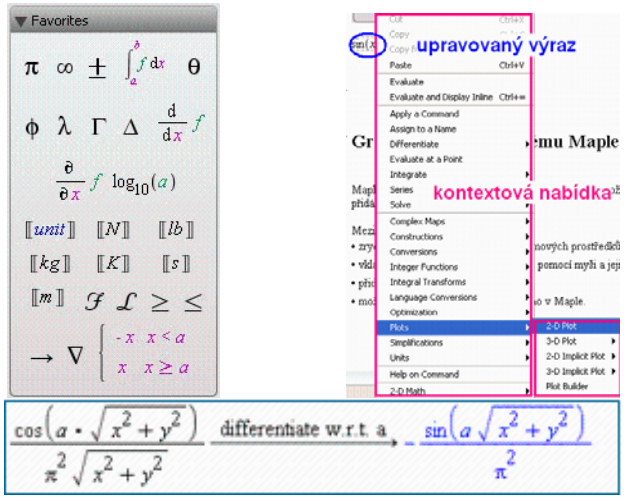
4.1.1 Uživatelské rozhraní v Maple

Nové verze systému Maple přinesly zásadní změnu filozofie používání systému Maple jako celku (Hřebíček, Žák, 2007). Jeho předchozí použití bylo založeno na znalosti programovacího jazyka systému Maple, který sloužil k provádění matematických výpočtů. Uživatel se musel naučit základní datové struktury Maple, práci s nimi a dále byl nucen učit se množství příkazů pro jednotlivé konkrétní operace s objekty Maple. To však bylo od verze systému Maple 10 změněno pomocí nového uživatelského rozhraní Maple (GUI – Graphical User Interface) ve prospěch „pohodlí“ uživatele, kterému se zjednodušil zápis textu i

matematických výrazů do Maple zápisníku (Worksheet) a mohl začít používat kontextové nabídky kliknutím na pravé tlačítko myši. V GUI Maple byl vytvořen zcela nový typ dokumentu, (*Rich Technical Dokument - RTD)*, který umožňuje pracovat se systémem Maple cíleně, velmi intuitivně a interaktivně pomocí kontextové nabídky. Uživatel je schopen jen využitím vlastních základních počítačových dovedností vytvářet plně interaktivní a komplexní dokumenty, které mohou sloužit dokonce jako výstup technických aplikací, popř. jako dokumentace k řešenému matematickému problému. Tento typ uživatelského rozhraní, na obr. č. 4.3, budeme v dalším označovat dokument.

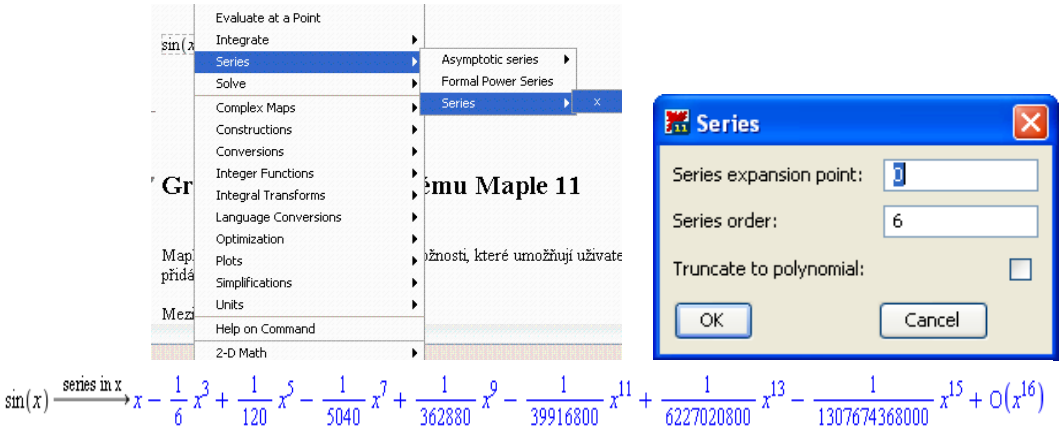


Obrázek č. 4.3: Uživatelské rozhraní systému Maple 11 – dokument



Obrázek č. 4.4: Paleta oblíbených vstupů, kontextová nabídka, popiska provedené operace

V novém GUI Maplu může uživatel pracovat jen pomocí kontextové nabídky a výstup zvolené operace je popsán formou editovatelného popisku, který dokumentuje právě provedenou operaci, do nabídky Maplu byla přidána nově paleta oblíbených nástrojů, která je osobně editovatelná, viz obr. č. 4.4. Kontextová nabídka umožňuje přistupovat i k výpočetnímu jádru Maple, a tato činnost je opět nově v Maple 11 plně dokumentována. Na obr. č. 4.5 je ukázána interaktivita prostředí systému Maple, kde se znázorňuje jednoduchost rozvoje funkce sin(x) do řady v dokumentu.



Obrázek č. 4.5: Znázornění rozvoj funkce sin(x) do mocninné řady v dokumentu - kontextová nabídka, upřesnění parametrů, výsledek

Další předností tohoto typu dokumentu je možnost použití systému Maple bez znalosti jeho příkazů s využitím:

- palety nástrojů;
- šablon běžných problémů (task templates), asistentů (assistants) a tutorů (tutors) v sekci tools;
- rozšířené kontextové nabídky;
- nástrojů pro rozpoznávání znaků;
- interaktivních průvodců umožňujících např. import dat a jejich analýzu, atd.

Je nutné též poznamenat, že tradiční zápisník v Maple je uživatelům stále k dispozici, ale neumožňuje jednoduše vytvářet plně interaktivní dokumenty.

4.1.2 Maple aplikační centrum

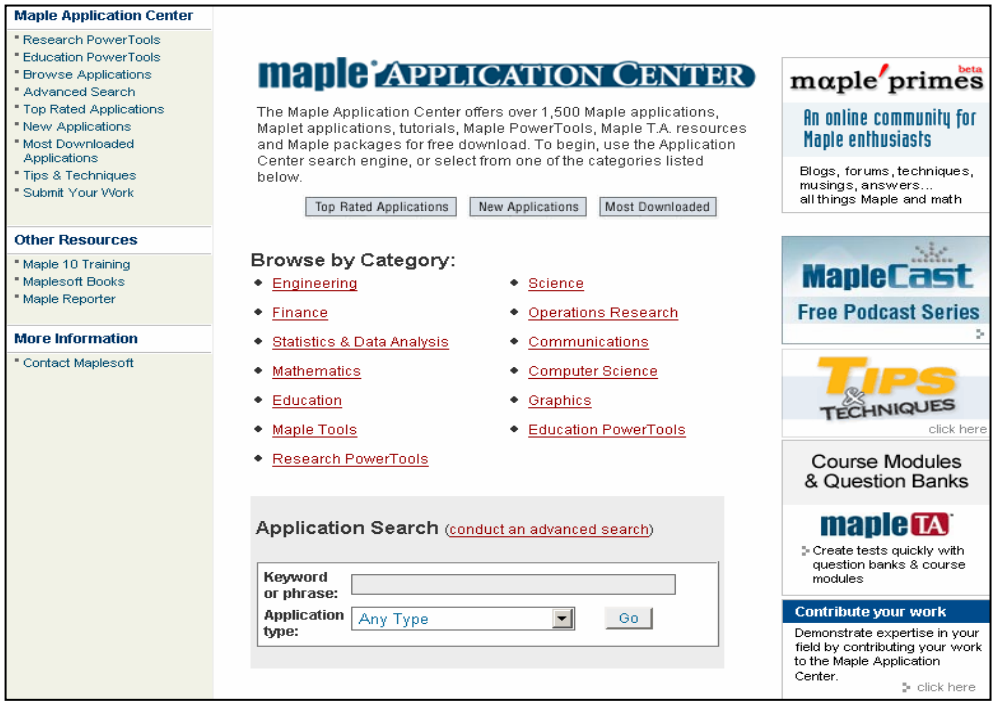
Pokud jde o uživatelskou podporu, tak z webového portálu Maplesoftu lze načíst a využít řadu velmi rozsáhlých knihoven řešených příkladů užitím Maple, a to v různých vědních oborech. Uživateli se stačí zaregistrovat a pak může z tohoto portálu stahovat tisíce řešených příkladů ve formátu HTML nebo přímo Maple zápisníků.

Maple aplikační centrum[76] , obr. č. 4.6, nabízí uživatelům ke volnému stažení více než 1500 Maple aplikací, tj. Maplet aplikací, tutoriálů, zápisníků a balíků konkrétních řešených problematik. V oblasti Biologie je ke stažení v současné době 25 aplikačních programů ve fomě Maple zápisníků.

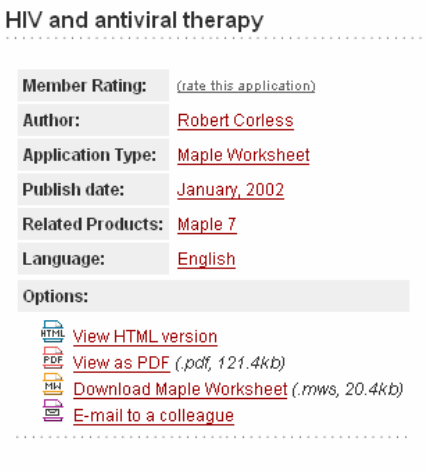
U každé aplikace, obr. č. 4.7, je uveden její rating, autor aplikace, typ aplikace, datum její publikace v Aplikačním centru, typ verze produktu, v níž je aplikace vytvořena a abstrakt.

[76] http://www.maplesoft.com/applications/

Aplikaci je možno si prohlížet buď v HTML formátu nebo v pdf formátu. Dále je zde ke stažení ve formátu mw (mws) Maple zápisníku vlastní aplikace a možnost zaslat kolegům e-mail o této aplikaci.

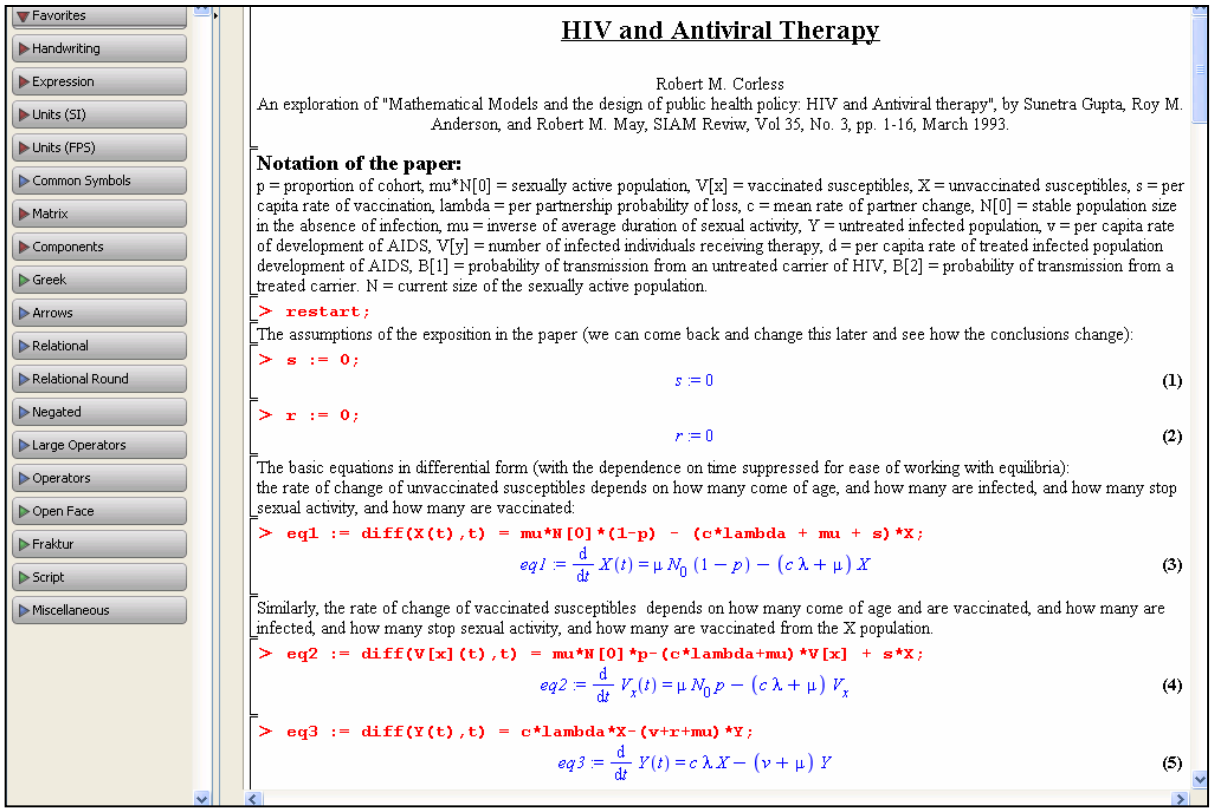


Obrázek č. 4.6: Maple Aplikační centrum



Obrázek č. 4.7: Informace o aplikaci

Postup při stažení aplikačního programu z Aplikačního centra je velmi jednoduchý. V sekci věda si vyhledáme oblast Biologie a v této oblasti příslušný Maple zápisník, který se týká antivirové terapie HIV. Na obr. č. 4.7 je informace o této aplikaci, která vychází z článku [Gupta, Anderson, May 1993] a byla vytvořena Robertem Corlessem v lednu 2002 ve Maple 7. Na obr. č. 4.8 je uveden stažený Maple zápisník této aplikace, který je možno jako svobodný software dále upravovat.



Obrázek č. 4.8: Část staženého zápisníku aplikace z obr. č. 4.7

4.1.3 Doplňující software spolupracující s Maple

Společnost Maplesoft Inc. již delší dobu nabízí komplexní řešení pro nejrůznější oblasti vědeckých výpočtů, které využívá nejen systémy Maple, MapleNet, popř. Maple T.A., ale zejména nové vlastní produkty, a dále produkty vyvinuté tzv. *třetími stranami*. K dispozici jsou nové vlastní produkty společnosti Maplesoft, např.:

- *Maple Toolbox for Matlab*, který umožňuje využít v Maple výpočetních možností systému Matlab, a naopak. Je určen pro přímou komunikaci mezi oběma matematickými systémy, a to nejen při výpočtech a předávání si výsledků z jednoho prostředí do druhého, ale poskytuje také možnost přístupu do celého výpočetního jádra systému Maple přímo z Matlabu. Tento toolbox je velmi užitečný zejména při simulacích výrobních procesů a složitých fyzikálních a chemických dějů.
- *Global Optimization Toolbox*, který umožňuje formulovat globální optimalizační problémy jednodušeji a užitím systému Maple získat rychle nejlepší možný extrém. K dispozici je již i elektronická publikace zabývající se využitím tohoto toolboxu.
- *Database Integration Toolbox*, který umožňuje rychle vytvářet aplikace, které kombinují velké datové soubory s možnostmi analýzy a vizualizací pomocí Maple.
- *Maple Professional Math Toolbox for LabVIEW*, který rozšiřuje možnosti systému LabVIEW užitím sofistikovaných symbolických a numerických možností Maple.
- *BlockBuilder for Simulink*, který exportuje dynamické modely systému a analytické algoritmy z systému BlockBuilder do systému Simulink.

Maplesoft prostřednictvím programu *MapleConnect*[77] nabízí produkty třetích stran, které jsou určeny pro uživatele systému Maple. Tyto doplňky zejména usnadňují práci v systému Maple. Jsou to např.:

- *HPC-Grid Toolbox for Maple* je balík nástrojů pro distribuované výpočty v počítačových sítích typu Grid užívající Maple.
- *DynaFlexPro*, který modeluje a simuluje dynamiku mechanických systémů. Je postaven na rychlém vytváření modelů užitím blokových diagramů a menu.
- *nVizx for Maple,* který je určený pro velmi kvalitní vizualizace.
- *ICP for Maple* tvoří balík vývojových nástrojů, které umožňují rychle a jednoduše identifikaci v inženýrských systémech. Je ideálním řešením pro návrh nových systému a jejich začlenění např. do systému Simulink apod.
- *Mathematics for Chemistry with Symbolic Computation* je elektronickou publikací aplikace matematiky v chemických výpočtech, která je ve formě zápisníků. Její příklady jsou zvoleny z různých oblastí chemie.
- *PSC Functions* je balík funkcí určený k modelování křivek a ploch a řešení problémů v 3D geometrii.

Kombinace systému Maple a některých systémů matematického (Matlab) a statistického software (Statgraphics) je možná nejen pomocí doplňkových toolboxů či programů Maple, ale např. i pomocí technologie OpenMaple, která otevírá uživatelům všech jiných systémů možnosti, jak využít nástrojů Maple.

4.1.3 Příklad použití Maple

V následujícím příkladu ukážeme jednoduché použití Maple pro vědecké výpočty. Iniciálním krokem při vytváření matematického modelu (nebo jeho modifikaci) je jeho identifikace, kdy se rozhodujeme o typu modelu a vlastnostech systému zahrnutých do tohoto modelu. Jako příklad uvedeme modelování růstu populace nějakého společenstva (např. myší). Z praxe je známo, že populace při ideálních podmínkách roste exponenciálně. Přirozený úbytek populace je nepatrný, pouze snižuje přirozený přírůstek. Populaci omezíme pouze dalším druhem – populací dravce (např. káně myšilov), který se bude živit populací kořisti (myší). V literatuře [Hřebíček, Škrdla, 2006] lze nalézt Lotka-Volterrův model ovlivňujících se populací, popsaný následujícími rovnicemi:

$$\begin{aligned} \frac{diff\ X(t)}{t} &= \alpha_1 * \mathrm{X(t)} - \beta_1 * \mathrm{X(t)} * \mathrm{Y(t)} \\ \frac{diff\ Y(t)}{t} &= -\alpha_2 * Y(t) + \beta_2 * X(t) * Y(t) \end{aligned}, \qquad (4.1)$$

kde α_1 je konstanta přirozeného růstu populace kořisti $X(t)$ a β_1 konstanta úmrtnosti populace v závislosti na počtu dravců $Y(t)$. Konstanta α_2 určuje přirozený úbytek populace dravce $Y(t)$ a konstanta β_2 přírůstek populace dravce v závislosti na počtu ulovené kořisti. Tento model je dynamický a spojitý, neboť simuluje vývoj populace v čase t.

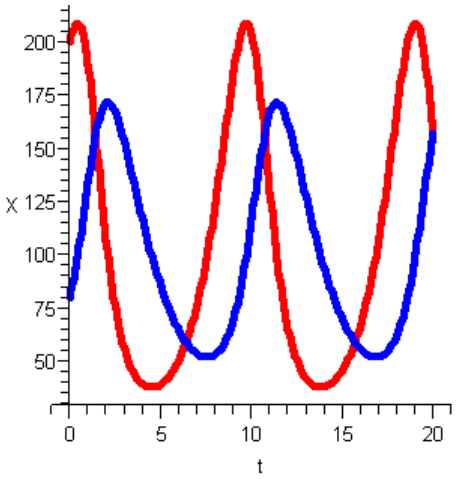
Implementace v programovacím jazyku Maple může být následující:

```
> restart;
with(plots): with(DEtools):
alpha[1] := 1.0:
alpha[2] := 0.5:
beta[1] := 0.01:
```

[77] http://www.maplesoft.com/products/thirdparty/index.aspx

```
beta[2] := 0.005:
de1 := diff(X(t), t) = alpha[1]*X(t) - beta[1]*X(t)*Y(t);
de2 := diff(Y(t), t) = -alpha[2]*Y(t) + beta[2]*X(t)*Y(t);
inits := [X(0) = 200, Y(0) = 80]:
myopts := stepsize=0.1, arrows=none:
plot1 := DEplot([de1, de2], [X, Y], t=0..20, [inits],
scene=[t, X], linecolor=red, myopts):
plot2 := DEplot([de1, de2], [X, Y], t=0..20, [inits],
scene=[t, Y], linecolor=blue, myopts):
> display(plot1, plot2
```

Výsledek je zobrazen na obr. č. 4.9 a zobrazuje vývoj počtu obou populací v čase. Na grafu lze pozorovat periodické střídání období velkého rozmnožení populace kořisti, po kterém přichází rozšíření populace dravce. To následně vede k značnému úbytku populace kořisti, a tedy i potravy pro dravce a jeho postupné vymírání. Poté se opět dostáváme do stavu, kdy populace kořisti nemá přirozeného soupeře a začíná se přemnožovat.

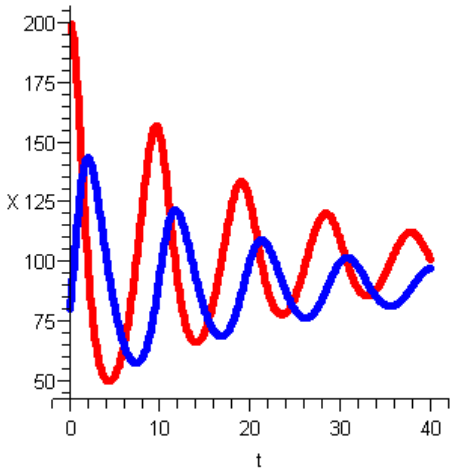


Obrázek č. 4.9: Řešení základního Lotka-Volterrova modelu

Nyní se dostáváme k fázi modifikace, kdy zjistíme, že model nevyhovuje přesně našim požadavkům a modelovanému systému. Rozhodneme se tedy pro úpravu modelu – tentokrát půjde o omezení růstu populace kořisti, například kvůli omezeným zdrojům potravy nebo prostoru, kde se může vyskytovat. V tomto případě už populace kořisti bez populace dravce neroste neomezeně, ale po dosažení určitého počtu jedinců se její počet ustálí. I pro tuto modifikaci lze nalézt upravený Lotka-Volterrův model popsaný rovnicemi (4.2):

$$\begin{aligned} \frac{diff\ X(t)}{t} &= \alpha_1 * \mathrm{X(t)} * (1 - \frac{\mathrm{X(t)}}{\mathrm{L}}) - \beta_1 * \mathrm{X(t)} * \mathrm{Y(t)} \\ \frac{diff\ Y(t)}{t} &= -\alpha_2 * Y(t) + \beta_2 * X(t) * Y(t) \end{aligned} \quad , \qquad (4.2)$$

kde L je zavedený limit růstu populace kořisti, přičemž ostatní předpoklady z modelu (4.1) zůstávají v platnosti. Úprava implementace modelu (4..2) v Maple je snadná, a proto zde uvádíme pouze výsledný graf, který je zobrazen na obr. č. 4.10.



Obr. 4.10: Řešení upraveného Lotka-Volterrova modelu

Z tohoto řešení lze opět vypozorovat periodicky se opakující stavy, kdy populace kořisti nebo dravce dosahuje extrému. Tyto extrémy se postupně snižují až dochází k postupnému ustálení počtu obou populací.

Kapitola 5
Umělá inteligence a inteligentní agenti
(Jan Žižka)

V této kapitole je velmi stručně shrnut současný moderní směr umělé inteligence spočívající v orientaci na tzv. *agenty*, přesněji řečeno na inteligentní umělé agenty. Mohou to být jak fyzičtí roboti, tak i softwarové systémy. Jejich cílem je provádět *racionální činnosti* a *být adaptivní* vzhledem k prostředí, pro něž jsou určeni. Racionalitou se nemíní dokonalost, i když dokonalost lze chápat jako ideál.

Agenti by měli být schopni navzájem *spolupracovat*, proto je zdůrazněna role *komunikace* a velmi stručně nastíněny možnosti a problémy. Agenti by měli být schopni se *přizpůsobit* situacím, o kterých nemohli být předem spolehlivě (nebo vůbec) informováni, ale které se mohou vyskytnout. Proto je důležitý i požadavek na adaptabilitu agentů. *Inteligence* nespočívá ve schopnosti všechno vědět, ale v tom, že agent disponuje určitou znalostí o světě a je schopen tuto znalost použít i v situacích, které jsou pouze podobné tomu, co zná, třebaže někdy podobnost nemusí být ani výrazná.

Agenti potřebují mít *schopnost se učit* neboli získávat znalost. Jsou aktivní ve světě, o němž dostávají prostřednictvím senzorů množství údajů, často v nedokonalé formě, někdy i s narušenou srozumitelností či s výpadky hodnot. Takto získaná data agent musí zpracovat tak, že z nich vybere jen to, co potřebuje k řešení aktuálního úkolu, a je-li schopen tento výběr zobecnit, a tím ho rozšířit, může s úspěchem zvládnout i situace, které se objeví v budoucnu. Nebudou identické se znalostí vloženou do agenta při jeho implementaci, ale na základě podobnosti bude agent schopen vyřešit i je.

K učení může agent použít různé metody. Současný moderní obor, zvaný *dolování z dat*, umožňuje odhalit v datech *informaci* a *znalost*, i když je to v datech někde zcela skryto. Vydolování znalosti umožňuje rovněž moderní, dnes již rozsáhlý obor zvaný *strojové učení*, jehož algoritmy jsou agentovi k dispozici a tvoří základní nástroj dolování.

5.1 Moderní pohled na pojem umělá inteligence

Umělá inteligence (AI, *Artificial Intelligence,* [Shapiro a kol., 1992]) se obecně a z různých hledisek zabývá možnostmi, jak dosáhnout cíle, aby stroje (např. počítače, roboti) byly schopny dělat činnosti, které dělá mysl živých bytostí. Umělá inteligence se začala rozvíjet jako obor, který se – spojen se vznikem prvních počítačů těsně po konci druhé světové války – začal cíleně pokoušet nejen porozumět tomu, jak biologické inteligentní bytosti myslí, ale i vytvářet obdobné, umělé či strojové entity. Pojem umělá inteligence vznikl, částečně i jako vtip, přibližně v polovině 50. let 20. století. Obdobně tomu, jak probíhal vývoj i jiných oborů, umělá inteligence prošla obdobími nadneseného očekávání i zklamání, periodami úspěchů i selhání. Navzdory nemožnosti vždy splnit naděje do ní vkládané však dosáhla a dosahuje řady úspěchů spojených s rozvojem výzkumu počítačové vědy a příslušných oborů, v nichž se počítače postupně uplatňují. Umělá inteligence jako odvětví nemá jednoznačnou definici a ke své existenci ji ani nepotřebuje – inteligence biologických systémů je rovněž spíše intuitivně chápaný než exaktně definovaný pojem.
Jeden z předních zakladatelů umělé inteligence, britský matematik Alan Mathison Turing (1912-1954), se nepokoušel stanovit obsah pojmu výčtem ne vždy jasných požadavků určujících, co je a co není inteligentní. Místo toho v roce 1950 navrhl známý Turingův test [Turing, 1950], který by za inteligentní entitu považoval cokoliv, co není možno z hlediska

reakcí odlišit od lidské bytosti, která je všeobecně považována za inteligentní. V jeho testu vystupují dva skrytí dotazovaní (muž A a žena B) a tazatel C, z nichž B se snaží tazateli C pomáhat a A ho mást při zjišťování skutečnosti, kdo z obou skrytých a dálnopisem odpovídajících (tj. elektronicky komunikujících) je muž a kdo žena. K čemu dojde, když místo A bude stroj? Bude tazatel C stejně často neúspěšný? Těmito pragmatickými otázkami Turing nahradil tu původní: *Mohou stroje myslet?* Zatím nejsou známy umělé objekty, vyznačující se schopnostmi zcela srovnatelnými s lidskými, protože ke splnění Turingova testu by musely být schopny pro komunikaci používat přirozený jazyk, umět ukládat fakta a vjemy do paměti ve formě informace a znalostí, usuzovat pomocí uložené informace tak, aby mohly odpovídat na otázky a vytvářet závěry a formou učení se přizpůsobovat novým, odlišným okolnostem nebo rozlišovat dosud neznámé vjemy a vzory.

Turing také ve svém článku v poslední kapitole nazvané *Learning Machines* (*Učící se stroje*) navrhuje, abychom se nesnažili jen o přímé sestavení "dospělého" stroje se znalostmi uloženými zvnějšku norimberským trychtýřem, ale spíše aby stroj měl schopnosti "novorozence" se učit sám a tak získávat adaptivně znalosti. Argumentoval proti původní myšlence Lady Lovelace (1842, *Babbage's analytical engine*), že "…analytický stroj nemá žádné ambice cokoliv *vytvořit.* Může dělat *cokoliv, o čem víme, jak mu přikázat*, aby to udělal". Turing rovněž zastával názor, že "učitel" by se neměl starat o to, k čemu detailně uvnitř stroje při učení dochází – výukový proces lze prostě považovat za hledání takové formy chování "žáka", která uspokojí "učitele" (nebo nějaké kritérium). Takový postup se samozřejmě velice liší od tradičního programování počítačů, ale z dnešního hlediska byl předpovězen správně.

V podstatě dnes existují čtyři hlavní pohledy na umělou inteligenci [Russel a Norvig, 2003]:

- systémy *myslící jako lidé* (kognitivní vědy);
- systémy *chovající se jako lidé*;
- systémy *myslící racionálně* (např. usuzující pomocí nějakého logického systému);
- systémy *chovající se racionálně* (dosahující co nejlepších očekávaných výsledků).

Rozlišením mezi lidskými a racionálními systémy se rozumí skutečnost, že lidé nejsou bezchybní, zatímco po umělé inteligenci bezchybnost požadujeme jako jeden z ideálních cílů. K rozvoji umělé inteligence přispívají všechna čtyři uvedená hlediska, a to jak nezbytností využívat empirické přístupy využívající k prokazování správnosti výsledky experimentů v příslušných konkrétních oborech, tak i aplikací matematických a inženýrských metod. K racionálním systémům patří například roboti – stroje, které by měly bezchybně provádět činnosti přinejmenším tak dobře jako lidé, v prostředích, kde inteligence je nezbytná pro schopnost adaptace. Namátkou lze zmínit např. roboty zkoumající velmi vzdálené planety, kde není možno bez přílišného zpoždění ovládat stroj lidmi na dálku, nebo inteligentní stroje schopné pomáhat v záchraně lidí, počínaje třeba vyhledáváním postižených při katastrofách (najít živé oběti zasypané v troskách) a konče např. konzultací při stanovení jejich správného způsobu vyproštění a záchrany. Stroje totiž nepodléhají únavě jako lidé, nejsou natolik omezovány fyzickými podmínkami, v nichž pracují, a metody umělé inteligence do nich lze přenášet kopírováním, takže mohou být vyráběny sériově.

Činností v různých podmínkách pak mohou ze strojů vlivem adaptačních schopností, daných určitým druhem inteligence, vznikat individuální entity, které si však mohou prostřednictvím komunikace znalost vyměňovat a doplňovat. K tomu může docházet třeba tak, že stroj zpracovávající inteligentně environmentální informaci v prostředí, kde došlo k náhlému znečistění, předá získanou znalost obdobným strojům v jiných oblastech, takže experti mohou

disponovat ekvivalentní podporou své práce (nalezení zdroje znečistění, návrh možností dekontaminace apod.). Kromě toho s případným zánikem inteligentního stroje nezaniknou jeho znalosti a zkušenosti.

Jak lidé, tak i systémy moderní umělé inteligence se typicky vyznačují tím, že jejich činnost má probíhat v reálných, vysoce komplikovaných prostředích, přičemž obvykle není k dispozici veškerá potřebná informace a znalost. Často nejsou ani disponibilní data, z nichž by se dala potřebná informace nebo znalost získat, nebo je naopak dat velmi mnoho, takže není zřejmé, která jejich část je pro řešení daného problému důležitá, nebo jsou data velmi složitá, mnohorozměrná, popisující objekty různým způsobem (číselně, binárně, výčtem vlastností, volným přirozeným jazykem) a jejich zpracování klade příliš vysoké, v potřebném čase nezvládnutelné výpočetní požadavky, anebo se data vyznačují kombinací nejrůznějších možných nedostatků včetně šumu a chybějících hodnot. Za takových okolností nelze očekávat dokonalé chování libovolného systému, a je nutno počítat pouze s omezenou racionalitou.

K typickým modelovým úlohám, které se pro výzkum možností aplikací metod umělé inteligence používají, patří např. hry, na nichž lze srozumitelně demonstrovat potenciální obtíže. Klasickým příkladem jsou šachy, které ani při dokonalé znalosti konkrétní situace a existenci jednoduchých pravidel hry nikdy nebudou *dokonale* hrány ani lidmi, ani inteligentními stroji, protože složitost vyhledávání správného koncového řešení zdaleka přesahuje jakékoliv praktické možnosti, včetně toho, že by hypoteticky mohl být každý atom ve známém vesmíru použit jako prvek paměti – i těch atomů je nedostatek, když počet možných pozic podle pravidel lze vyjádřit odhadem jako jedničku následovanou desítkami nul. Přesto mohou někteří lidé hrát složité hry velice dobře, a rozvoj počítačové vědy s možností využívat paralelní vysoce výkonný hardware IBM Deep Blue (specializovaný na konkrétního lidského hráče) už také dokázal v r. 1997 porazit i lidského mistra světa v šachu, fenomenálního Garry Kasparova – samozřejmě, že ani člověk, ani počítač nehráli bezchybně, jak ukázaly "post mortem" analýzy, ale to není podstatné; podstatné je, že stroj dokázal na nejvyšší úrovni úspěšně inteligentně fungovat. Ovšem jiné hry, např. japonská *go*, však představují vlivem své obrovské složitosti úkol zatím neřešitelný.

Problémy, jejichž řešení lidé i stroje hledají pomocí své inteligence, se vyznačují nemožností nalézt uspokojivý výsledek tradičními metodami, např. matematickou analýzou, modelováním, statistikou apod., a to z libovolných důvodů. V principu může třeba být řešení nalezeno, např. tak, že se zjistí všechny možnosti a z nich se vybere hledaný optimální výsledek – tento jednoduchý postup však v reálném světě příliš často není možný, protože všech možností může být tolik, že řešení je zcela nezvládnutelné z časových nebo i paměťových důvodů. Na náhodné nalezení výsledku spoléhat nelze, takže je zapotřebí hledat způsoby omezení extrémní výpočetní složitosti. To velmi často lze, avšak je nutno za to zaplatit cenu v tom, že již není *zaručeno* nalezení optima. Nahrazení exaktního způsobu hledání optima např. heuristickým přístupem v praxi obvykle dá i docela dobrý výsledek, ale vzhledem k tomu, že nemusí být známo, jak vlastně optimum vypadá, nelze říci, zda optima bylo dosaženo či ne.

Problém hledání optima si lze představit tak, že reálná úloha se převede z reálného světa do světa umělého, který je abstraktnější, popisy objektů bývají zcela odlišné, počet dimenzí může být velmi výrazně odlišný apod. Abstraktní svět si lze představit jako mnohorozměrnou, libovolně členitou (nelineární mnohorozměrná plocha nemusí být např. všude spojitá ani hladká) a zamlženou krajinu, v níž není vidět, kde je nejvyšší vrchol (globální extrém), který se má nalézt a kde typicky hrozí uváznutí v lokálním extrému (nižší vrcholek nebo údolí),

nebo není vůbec zřejmé, kterým směrem se vydat směrem nahoru k alespoň lokálnímu extrému (rozsáhlé roviny, sedlové body). Žádná známá metoda hledání není imunní vůči některým nedostatkům zmíněné krajiny a systematické prohledání není možné kvůli výpočetní složitosti. Inteligence, ať přirozená nebo umělá, může velmi dobře v neznámé situaci odhadnout vhodný postup, i když bez záruky. Inspirací umělé inteligence je schopnost biologických systémů řešit problémy v obdobných situacích, kde navíc je k dispozici jen nejistá nebo i velmi vágní, přibližná informace, kdy jsou známy jen pravděpodobné, nejasné a neostré hodnoty. Lze říci, že umělá inteligence se zabývá prohledáváním komplexních prostorů s ideálním cílem najít optimum.

5.2 Inteligentní agenti

Agentem rozumíme cokoliv, co vnímá své prostředí pomocí senzorů a na prostředí působí pomocí efektorů [Russel a Norvig, 2003]. *Člověk-agent* z tohoto hlediska má oči, uši a další orgány jako senzory; dále má ruce, nohy, ústa a další části těla jako efektory. *Robot-agent* nahrazuje kamerami a infračervenými detektory senzory. Efektory jsou vlastně tvořeny různými motory. *Softwarový agent* disponuje bitovými řetězci pro vnímání i působení na prostředí. Cílem je návrh (umělých) agentů, kteří inteligentně a výkonně jsou schopni působit na své prostředí.

Za *racionálního agenta* budeme považovat takového, který provádí správné věci. V prvním přiblížení je *správná věc* taková věc, která umožňuje agentovi nejlepší úspěch. *Jak* a *kdy* se měří *výkonnost agenta?* Pro různé agenty neexistuje stejný způsob měření úspěšnosti. Měření (a vyhodnocení) míry úspěšnosti není vhodné provádět subjektivně (tedy přímo agentem), neboť např. agent nemá schopnost to udělat apod. Proto se používá *objektivní* míra úspěšnosti, kdy je agent pozorován např. námi z vnějšku v jeho prostředí.

Uveďme příklad: agent, jehož úkolem je vysávat nečistotu z podlahy, může být jednoduše posuzován z hlediska množství odstraněné špíny během osmi hodin. Komplexnější posouzení může vzít do úvahy faktor spotřeby elektrického proudu, množství produkovaného hluku apod. Další otázkou je *kdy* agentovu výkonnost vyhodnocovat. Na začátku vysávání může některý agent vysávat usilovně a za hodinu polevit, jiný může pracovat rovnoměrně celou pracovní dobu, další např. celou dobu své existence. Tomu je nutno přizpůsobit měření. Je nutno také rozlišovat mezi racionalitou a vševědoucností (kde agent zná *skutečný* výsledek svých akcí a tomu může přizpůsobit svou činnost; to je však v realitě nemožné docílit). Racionalita je zaměřena na *očekávaný* úspěch na základě toho, co je agentem vnímáno (mnoho věcí nelze předvídat a vnímat). Důležitou součástí *racionality* je získávání užitečné informace. Např. před přechodem křižovatky by se měl robot rozhlédnout doleva a doprava; bez toho jeho (*post mortem* zkoumaná) sekvence vnímání neodhalí, že vkročil do silnice bez rozhlédnutí a srazilo jej auto – takový robot ovšem není racionální a jeho akce *přejít křižovatku* má nízkou míru výkonnosti. Pokud tedy závisí chování agenta na jeho sekvenci vnímání do určitého okamžiku, lze každého agenta popsat pomocí tabulky akcí, které vykonává jako odezvu na každou (doposud) možnou sekvenci vnímání (pro mnoho agentů by byl seznam velice dlouhý). Tento seznam budeme nazývat *mapováním ze sekvencí vnímání do akcí*. Mapování popisuje agenta a *ideální mapování* ideálního agenta. Není ovšem nutno vytvářet explicitní tabulku s buňkou pro každou možnou sekvenci vnímání – často lze definovat specifikaci bez vyčerpávajícího výčtu. (Např. výpočet druhé mocniny na kalkulačce nepotřebuje znát hodnotu pro každou možnou kombinaci stlačení tlačítek – mapování lze vytvořit třeba metodou Newtona.)

Definice ideálního racionálního agenta: Pro jakoukoliv sekvenci vnímání, ideální racionální agent učiní vše, co se od něj očekává, pro maximalizaci míry jeho výkonnosti, a to na základě evidence poskytnuté sekvencí vnímání a znalosti, kterou agent disponuje. ■

Ideální racionální agent by měl obsahovat nějakou "vestavěnou" znalost. Pokud takovou má a je založen výhradně na ní, pak ovšem postrádá *autonomii* (jeho akce pak nezávisí na vnímání). Autonomie systému je dána vlastní zkušeností a znalostí agenta. Nelze ovšem např. požadovat kompletní autonomii agenta na slovo "jdi" a je nutno se vyvarovat jeho nahodilé činnosti. Agent by měl mít schopnost se učit (získávat znalost). Skutečně autonomní inteligentní agent musí být schopen úspěšné činnosti v širokém rozsahu různých prostředí, za předpokladu, že má dost času k adaptaci. Některá reálná prostředí mohou být ve skutečnosti velmi jednoduchá (robot pro třídění součástek). Oproti tomu někteří *softboti* (*softbot* = *software robot*) existují ve složitých, neomezených doménách (např. softbot navržený pro létání na leteckém simulátoru pro Boening 747, softbot pro prohlížení on-line zpráv a výběr zajímavých pro zákazníky a mnoho dalších). Prostředí, v nichž se agent pohybuje, mají několik charakteristik. Základní odlišnosti:

- *Přístupné a nepřístupné:* Zda agentovy senzory umožňují poskytnout kompletní informaci o stavu prostředí. Pokud ano, je prostředí *přístupné*, jinak je *nepřístupné*. Senzory by měly umožnit získat údaje relevantní pro výběr akce. Přístupné prostředí má tu výhodu, že agent nemusí udržovat vnitřní stavy sledovaného světa.
- *Deterministické a nedeterministické:* Je-li následující stav prostředí zcela určen současným stavem a akcemi zvolenými agentem, pak jde o *deterministické* prostředí. V principu nemá agent starosti s nejistotou v přístupném a deterministickém prostředí. Zda je prostředí deterministické se obvykle musí určit přímo z *hlediska agenta*.
- *Epizodické a neepizodické:* V epizodickém prostředí je agentova znalost či zkušenost rozdělena mezi "epizody". Každá epizoda se skládá z agentových vnímání následovaných akcemi. Kvalita agentovy akce závisí pouze na dané epizodě, protože následné epizody na předcházejících nezávisejí, co do jejich vlastní kvality. Epizodická prostředí mají výhodu v tom, že agent nemusí myslet dopředu.
- *Statické a dynamické:* Pokud se prostředí mění během agentova uvažování, pak je prostředí z jeho hlediska *dynamické*, jinak je *statické*. Statická prostředí mají výhodu v tom, že během svého uvažování nemusí agent sledovat svět, a také se nemusí zabývat časem.
- *Semidynamické:* Pokud se prostředí nemění v čase, ale agentova výkonnost na čase záleží, mluvíme o *semidynamickém* prostředí.
- *Diskrétní a kontinuální:* Existuje-li omezené množství odlišných a jasně určených vnímání a akcí, pak jde o *diskrétní* prostředí (např. šachy jsou diskrétní prostředí – existuje pevné množství možných tahů v každé pozici).

Zjevně nejobtížnější případ nastane, když je prostředí nepřístupné, neepizodické a dynamické; obdobně je to i v případě prostředí dynamického. Většina reálných situací je tak složitá, že to, zda je prostředí skutečně deterministické, je záležitost sporná, a pro praktické účely se zachází se situací jako s nedeterministickou. Tab. č. 6 ukazuje *typy prostředí* pro některá známější prostředí:

Tabulka č. 6: Typy prostředí

Prostředí	Přístupné	Deterministické	Epizodické	Statické	Diskrétní
Šachy s hodinami	ano	ano	ne	semi	ano
Šachy bez hodin	ano	ano	ne	ano	ano
Poker	ne	ne	ne	ano	ano
Backgammon	ano	ne	ne	ano	ano
Řízení auta	ne	ne	ne	ne	ne
Medicínská diagnostika	ne	ne	ne	ne	ne
Analýza obrazů	ano	ano	ano	semi	ne
Robot třídící součástky	ne	ne	ano	ne	ne
Řízení čističky	ne	ne	ne	ne	ne
Interaktivní učitel angličtiny	ne	ne	ne	ne	ano

Hodnoty v tabulce mohou ovšem záviset na tom, jak je *vytvořen pojem* (tzv. konceptualizace) prostředí a agentů. *Poker* je deterministický, pokud si agent může udržovat údaje o sledování pořadí karet v balíčku, jinak bude poker nedeterministický. Obdobně může dojít k tomu, že prostředí je epizodické na vyšší úrovni, než se odehrávají agentovy *jednotlivé* akce. Např. *šachový turnaj* se skládá z posloupnosti her (partií). Každá partie je epizoda, protože přínos tahů, provedených agentem v jedné partii pro celkovou úroveň agentovy výkonnosti, není ovlivněn tahy *příští* partie. Na druhé straně je nutno uvažovat fakt, že během jedné partie jsou jednotlivé tahy spolu určitým způsobem provázány, takže agent je musí předjímat v určitém kvantu předem.

5. 3 Komunikace agentů

Podílí-li se na řešení úlohy více agentů, je zapotřebí, aby byli mezi sebou navzájem schopni komunikovat. Situaci může komplikovat skutečnost, že agenti mohou pocházet od různých výrobců (zdrojů), mohou být založeni na různých technologiích apod. Odpovídá to situaci, kdy mezi sebou mají spolupracovat „lidští“ experti z různých oborů, mluvící různými jazyky, mající různé vzdělání, různý původ a sledující různé cíle (např. vědci, výrobci, uživatelé aj.).

Agenti mohou mít jako jednu ze svých dostupných akcí také schopnost produkovat jazyk [Russel a Norvig, 2003]. Tato činnost se nazývá *akce řeči* (*komunikace*), přičemž pojem “řeč” má význam “volná řeč”, nikoliv jen “mluvení”, takže k akci „řeči“ se počítá také psaní, znaková řeč apod. Z tohoto hlediska i pojem *slovo* zde má širší význam. Skupina agentů může kolektivně nebo individuálně získávat nějaké výhody, pokud může dělat něco z následujících činností:

- *Informovat* jeden druhého o části prozkoumaného světa, takže mají k dalšímu zkoumání menší část světa.
- *Dotazovat se* na konkrétní vlastnosti světa.
- *Odpovídat* na dotazy.
- *Žádat* (úkolovat) jiné agenty, aby vykonali nějakou akci.
- *Slibovat* provedení nějaké akce nebo *nabízet* podmínky.
- *Potvrzovat* žádosti a nabídky.
- *Sdílet* dojmy a zkušenosti (získané znalosti) jiných.

Obtížné pro agenta je určit, *kdy* provést akci komunikace, tj. aby komunikace byla opravdu zapotřebí, a *jaká* komunikační akce se má udělat. Z určitého hlediska se jedná o *plánovací* problém – agent si může vybrat se souboru možných akcí a nějakým způsobem musí

otestovat, kterou akci vybrat k dosažení cíle, tj. jak patřičně předat informaci jinému agentovi. Všechny plánovací potíže se projevují i při výběru komunikační akce.

Dalším problémem je nedeterminismus: příkaz/žádost jednoho agenta může nebo nemusí být proveden/provedena, takže je zapotřebí mít k dispozici podmíněné plány. Konverzace se nedá naplánovat od začátku do konce, pouze je možné zkonstruovat obecný plán pro konverzaci, vygenerovat první větu, následně počkat na vjem odpovědi a reagovat na odpověď. Problém porozumění akce řeči je podobný jiným problémům porozumění (porozumění obrazovému vjemu, medicínské diagnóze apod.). K dispozici je soubor nejednoznačných vstupů, z nichž je nutno zpětně vypracovat rozlišení stavu světa, který vstup vyprodukoval. Část problému porozumění je závislá na jazyku – je nutno znát syntaxi a sémantiku jazyka k pochopení důvodu, proč jiný agent vykonal konkrétní komunikační akci. Tuto úlohu příjemce zprávy musí rozhodnout, které stavy jsou pravděpodobnější. Uvedené záležitosti jsou spojeny s vědou zabývající se přirozenými záležitostmi – např. porozuměním přirozenému jazyku a jeho zpracováním. V zásadě lze odlišovat *formální jazyky* (např. C++ a logika prvního řádu jsou rigidně definovány) a *přirozené jazyky*, např. čínština, angličtina aj. používané lidmi ke vzájemné komunikaci. Umělá inteligence využívá pro formální jazyky notaci v tzv. Backus-Naurově formě (BNF).

Formální jazyk je soubor řetězců složených ze sekvencí symbolů, které jsou převzaty z konečné množiny *terminálních symbolů*. Určitou potíží při práci s formálními a přirozenými jazyky je velké množství různých formalizmů a notací pro zápis gramatik – většina je však založena na myšlence *frázové struktury*, tj. řetězce jsou složeny z podřetězců zvaných *fráze* z různých kategorií (slovní druhy: podstatná jména, slovesa, atd.). Fráze jsou vhodným prostředkem, jemuž lze přiřazovat jisté významy (sémantiku). Kategorizace frází napomáhá popsat přípustné řetězce jazyka. Lze říci, že jakákoliv fráze podstatného jména může být kombinována (při dodržení určitých podmínek) se slovesnou frází pro vytvoření fráze z kategorie *věta*. V různých jazycích existují ovšem různé gramatiky také vlivem odlišné podstaty (v angličtině je slovosled vázán mnohem přísnějšími pravidly než v češtině, takže přehození slov v češtině nemusí zrušit správnost kategorie věty, v angličtině však ano). Gramatické kategorie a pravidla jsou součástí jazykovědy.

5.4 Komponenty komunikace

Typická komunikační epizoda, kdy řečník (*speaker*) *S* chce sdělit propozici *P* naslouchajícímu (*hearer*) *H* pomocí slov (*words*) *W*, se skládá ze sedmi procesů. Tři procesy souvisejí s řečníkem:

Intence:	*S* chce, aby *H* uvěřil *P* (typicky *S* sám věří *P*).
Generace:	*S* vybere slova *W* (protože vyjadřují význam *P*).
Syntéza:	*S* vysloví slova *W* (a obvykle je adresuje na *H*).

Čtyři procesy souvisejí s naslouchajícím:

Percepce:	*H* vnímá *W'* (ideálně $W' = W$, ale je možné nějaké zkreslení vjemu apod.).
Analýza:	*H* inferuje, že *W'* má možné významy $P_1, \ldots, P_n$ (slova a fráze mohou mít několik významů).
Desambiguace:	*H* inferuje, že *S* měl v úmyslu sdělit P_i (ideálně $P_i = P$, ale je možná chybná interpretace vjemu).
Inkorporace:	*H* se rozhodne uvěřit (přijmout) P_i (nebo P_i odmítne, pokud je mimo to, čemu již věří).

Předpokladem komunikace je, že: a) agenti používají stejný jazyk; b) sdílejí stejný kontext a c) jsou alespoň částečně racionální (aby správně reagovali na vjem řeči).

5.5 Dva modely komunikace

K současným nejvýznamnějším směrům studia komunikace agentů patří způsoby změny reprezentace "vnitřních názorů" agentů na slova, a naopak změny přijímaných slov na „vnitřní názory" agentů, kde *vnitřní názor* znamená znalost uloženou v nějaké vhodné formě v agentově znalostní bázi (*znalostní bází* se zde rozumí soubor reprezentací faktů o světě – znalostní báze poskytuje odpovědi na dotazy). Dva z existujících modelů jsou následující:

- *Kódované zprávy:* řečník *S* má na mysli určitou propozici *P* a zakóduje ji do slov (nebo nějakých jiných příznaků) *W*. Naslouchající *H* se pokusí dekódovat zprávu *W* k získání původní propozice *P* (např. Morseův kód – z teček a čárek, používaný kdysi pro výměnu zpráv). V tomto modelu se v ideálním případě přenáší původní význam z hlavy prostřednictvím přenášené zprávy a interpretace *H*, aniž by došlo k jakékoliv změně obsahu. Změnu obsahu však může způsobit např. šum komunikačního kanálu nebo chyba při kódování či dekódování.

Limitace modelu kódovaných a dekódovaných zpráv vedla k jinému modelu:

- *Situační jazyk:* význam zprávy závisí jak na slovech, tak i na situaci, v níž jsou slova vyslovena. Funkce kódování/dekódování zde mají navíc argument, který reprezentuje konkrétní, okamžitou situaci. To souvisí se skutečností, že tatáž slova mohou mít v různých situacích zcela různé významy: "Já jsem teď zde" znamená pro Pepu v Brně v pondělí něco zcela jiného než pro Frantu v úterý v Ostravě apod. Situační jazykový model má rovněž možný zdroj selhání komunikace: mají-li *S* i *H* odlišné představy o aktuální situaci, pak nemusí být zpráva předána na základě původního úmyslu.

Agenti mohou také komunikovat dvěma různými způsoby: oba sdílejí společný vnitřní reprezentační jazyk, nepotřebují tedy žádný externí jazyk. Jiní agenti nemusejí splňovat žádný předpoklad o vzájemném vnitřním jazyku, ale sdílejí komunikační jazyk (např. podmnožinu angličtiny).

Kapitola 6
Dolování z dat
(Jan Žižka)

6.1 Účel dolování z dat

Současná společnost se stále více zaměřuje na shromažďování a následné zpracování dat, jejichž objem narůstá se zrychlením vývoje ICT. Proto pro extrémní množství dat je nutné jejich strojové zpracování. Zpracování dat však nezahrnuje pouze běžné databázové aplikace, zaměřené na prosté vyhledávání a předávání nalezených údajů uložených v databázi. Data jsou zaznamenaná fakta, avšak jen v základní, hrubé formě, která sama o sobě nemusejí explicitně poskytovat znalost potřebnou k řešení reálných problémů reálného světa. Je zřejmé, že cíleně shromažďované údaje v sobě nějakou znalost obsahují, tato znalost je v datech ukryta. Potenciální informace v datech představuje určitou strukturu nebo vzory či nějaké pravidelnosti, jejichž zobecněním lze skrytou znalost odhalit, a pak výhodně využít jak v komerční, tak ve výzkumné oblasti. Objevováním informace a znalosti v datech [Witten a Frank, 2005] se zabývá moderní, stále se rozvíjející odvětví zvané **dolování z dat** (*Data Mining*) s cílem automatické extrakce implicitní, dosud neznámé a potenciálně užitečné informace, resp. znalosti, z dat. Odhalení výrazných vzorů v datech umožňuje jejich následné zobecnění a aplikaci pro predikci budoucích dat. Dolování v datech však představuje většinou obtížný a výpočetně náročný úkol. Mnoho nalezených vzorů bývá banálních a nezajímavých, i když informace v datech existují. Řada dalších dat je nepodložená nebo čistě náhodná koincidence. Odhalené vzory bývají nepřesné, nalezená pravidla mívají však mnoho výjimek, nezanedbatelná část případů není pokryta žádnými pravidly. Důvodem bývá nedokonalost dat, chybějící a zašuměné hodnoty, opominuté důležité atributy, a naopak zbytečně naměřené irelevantní neužitečné vlastnosti zkoumaných entit. Někdy není dat nebo jejich určitých částí dostatek;doplnit je není možné ať již třeba z důvodů fyzikálních vztahů nebo z finančních důvodů. Přesto i nedokonalá znalost bývá lepší, než není znalost žádná. Dolování z dat zahrnuje přípravu dat, hledání vhodného algoritmu pro zobecnění, experimentální výpočty a důkladnou interpretaci výsledků. Celý postup se může několikrát opakovat na základě zjištění z experimentů.

6.2 Základní prostředky pro dolování z dat

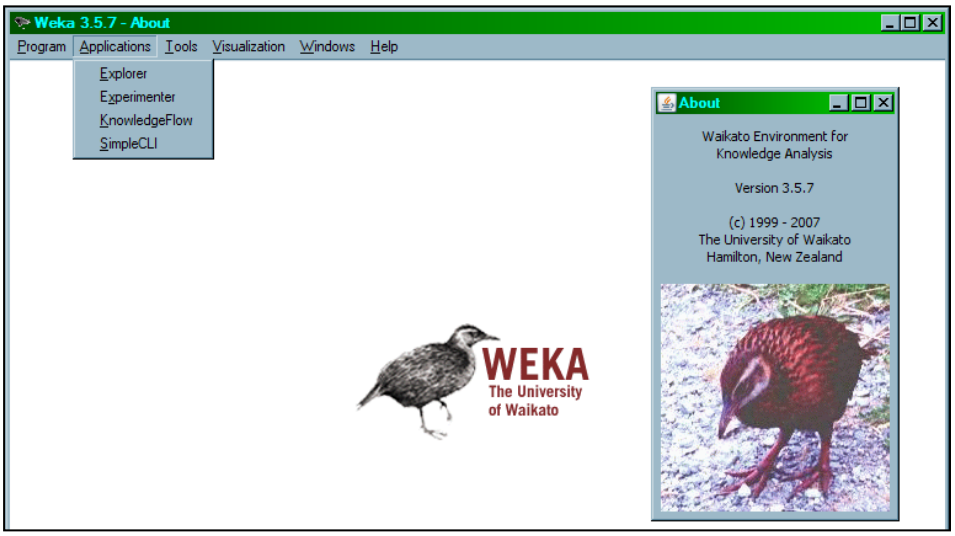
Technickou základnu dolování z dat poskytuje obor *strojové učení* [Mitchell, 1997], nebo též *algoritmické učení* [Hutchinson, 1994], zejména jeho vysoce pragmaticky zaměřená větev *induktivní*, daty řízené učení. Známé datové příklady poslouží, podobně jako při učení lidí, k získání obecné znalosti, která je pak použitelná pro předpověď, vysvětlení, porozumění, klasifikaci/kategorizaci, doplnění hodnot údajů nezískaných v minulosti apod. Pro uvedené činnosti lze samozřejmě použít i jiné postupy, jsou-li aplikovatelné, např. *dedukce, abdukce,* aj. Deduktivní znalost však v praxi nebývá často dostupná, například dokázaný matematický vzorec nebo model, takže induktivním učením lze objevit i znalost později aplikovatelnou pro deduktivní účely – může to třeba být neznámá, velmi složitá funkční závislost, aproximovaná s dostatečnou přesností nějakým algoritmem, např. umělou neuronovou sítí.

6.3 Softwarový systém dolování z dat WEKA

Systém WEKA[78] (*Waikato Environment for Knowledge Analysis, Waikato prostředí pro analýzu znalosti*) lze volně získat, viz [WEKA, 2007]. Implementován je na University of Waikato, Hamilton, New Zealand v jazyce Java, a lze ho snadno nainstalovat a používat [Witten a Frank, 2005] pod operačními systémy Linux i MS Windows. WEKA je stále průběžně vyvíjen a rozšiřován o aktuální algoritmy strojového učení a metody dolování z dat včetně některých podpůrných nástrojů pro přípravu dat a zpracování výsledků. Jedním z cílů je i dosažení pokud možno obecné použitelnosti, takže není (na rozdíl od řady různých komerčních systémů) orientován na speciální formáty dat a předpoklady použití. Součástí instalace jsou i některé známé datové soubory v textovém formátu (přejaté např. z [ICS-UCI, 2007]), na nichž si lze snadno vyzkoušet mnohé možnosti a nástroje systému WEKA, který obsahuje rovněž vlastní editor textových dat a jejich konverzní funkce pro některé běžně užívané textové (*.csv, *.data a *.names, *.xrff a vlastní *.arff) a binární formáty (*.bsi).

Pro velmi stručnou ilustraci použití a možností softwarového nástroje WEKA jsou zde použita data *soybean*, která se zabývají devatenácti možnými chorobami rostliny pěstované pro získávání sojových bobů. Data jsou součástí instalace WEKA, ale lze je volně získat i přes Internet, viz [ICS-UCI] v seznamu referencí. Rostlina je popsána 35 atributy (nominálními; některé lze seřadit podle hodnot) a klasifikačními třídami. Soubor *soybean* obsahuje šum i chybějící hodnoty (hodnota "dna" v datech znamená pouze "does not apply" neboli N/A, tj. "nelze aplikovat na daný případ"), takže představuje dobrý příklad dat z reálného světa a slouží jako jeden ze standardních problémů, na kterém se dá studovat např. chování různých algoritmů a metod. V souboru je k dispozici 683 instancí. Cílem je na základě kombinace hodnot atributů stanovit příslušnou chorobu rostliny, neboli klasifikace. Atributy udávají např. datum (date), rozsah poškozené oblasti (area-damaged), míru klíčení (germination), stav listů (leaves) a kořenů (roots); další – detailnější popis je obsažen přímo v datech.

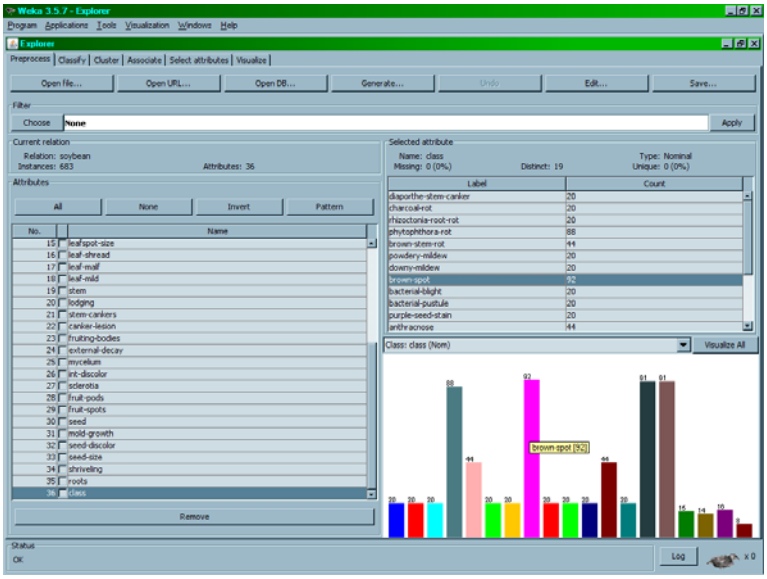
Po vyvolání systému WEKA se objeví úvodní okno, kde v položce menu *Applications* lze vybrat jeden z nástrojů – zde bude použit nástroj zvaný *Explorer*, viz obr. č. 6.1.



Obrázek č. 6.1: Úvodní okno systému WEKA

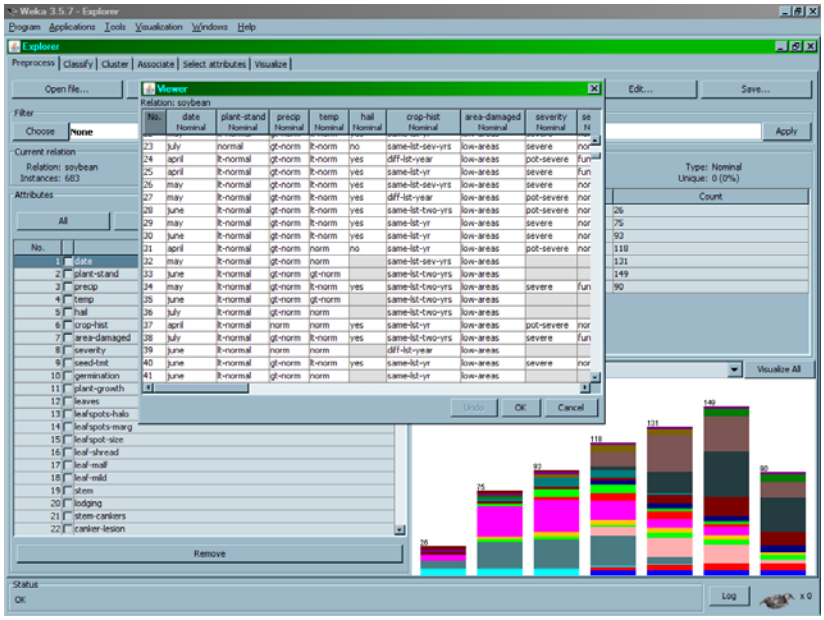
78 Weka (*Gallirallus australis,* zvaný též slepice Maori podle původních obyvatel Nového Zélandu) je unikátní nelétající pták, cca 50 cm a 1 kg, který žije divoce na třech hlavních ostrovech Nového Zélandu a je údajně zvídavý.

Dále pomocí *Open File* (viz obr. č. 6.2) se stanoví jméno a formát vstupního datového souboru, zde *soybean.arff* (textový, jednoduchý a srozumitelný vlastní formát systému WEKA). Okno *Explorer* zobrazuje pro zvolený atribut rozložení jeho hodnot (zde formou histogramu). Nabízené položky v menu umožňují např. výběr nástroje pro předzpracování dat, filtraci, editaci, konverzi, apod. Dále si lze zvolit požadovanou činnost (klasifikaci *Classify*, shlukování *Cluster*, aj.).



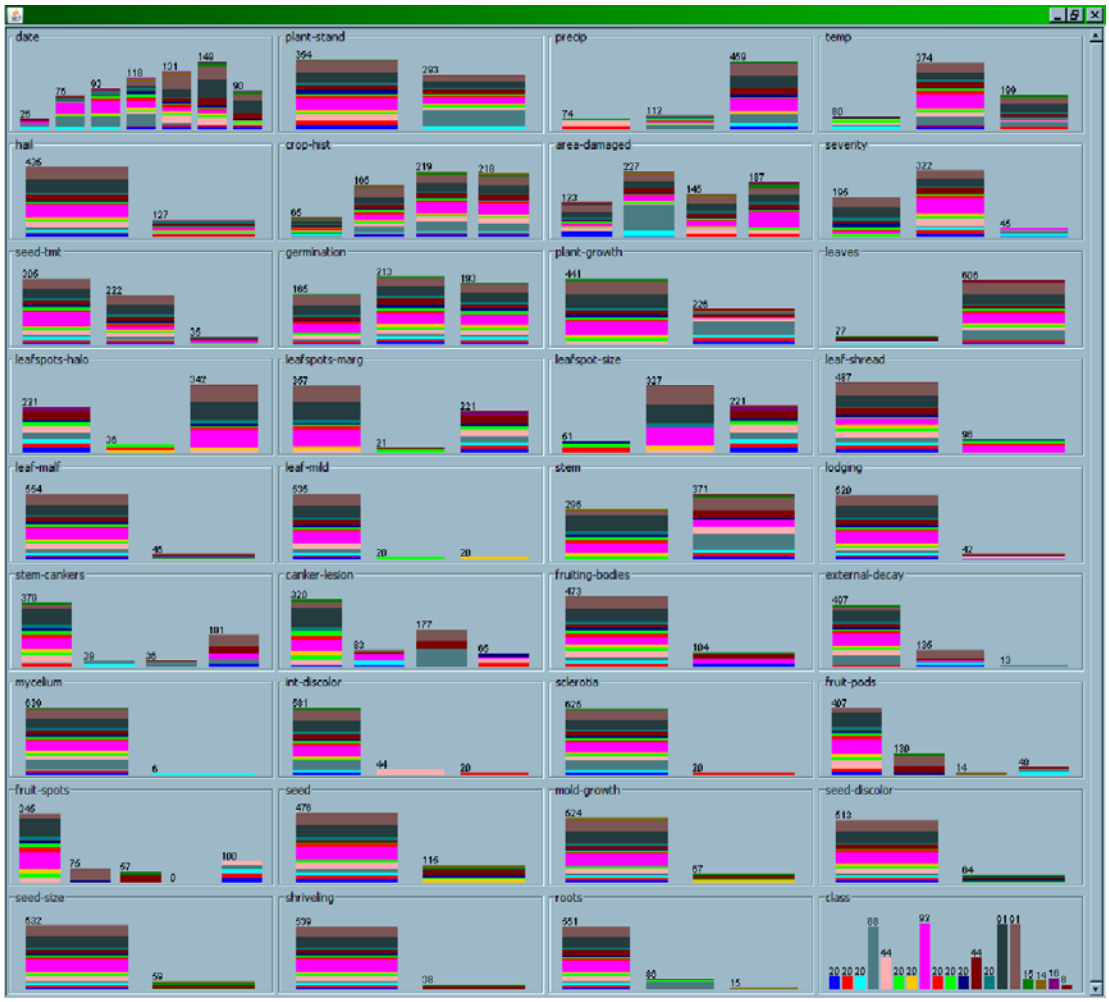
Obrázek č. 6.2: Zadání vstupního datového souboru

Například položka menu *Edit* zobrazí okno *Viewer,* obr. č. 6.3, kde lze prohlížet a různě upravovat aktuální data, a to i bez nepříjemných omezení na počet sloupců a řádků, což je typicky na závadu u různých komerčních i nekomerčních tabulkových procesorů – je nutno ovšem počítat s tím, že editace rozsáhlých dat i zde vyžaduje velkou paměť a odpovídající čas.



Obrázek č. 6.3: Okno *Viewer*

Položka menu *Visualize All* umožňuje zobrazit rozložení hodnot všech atributů (včetně tříd), obr. č. 6.4:

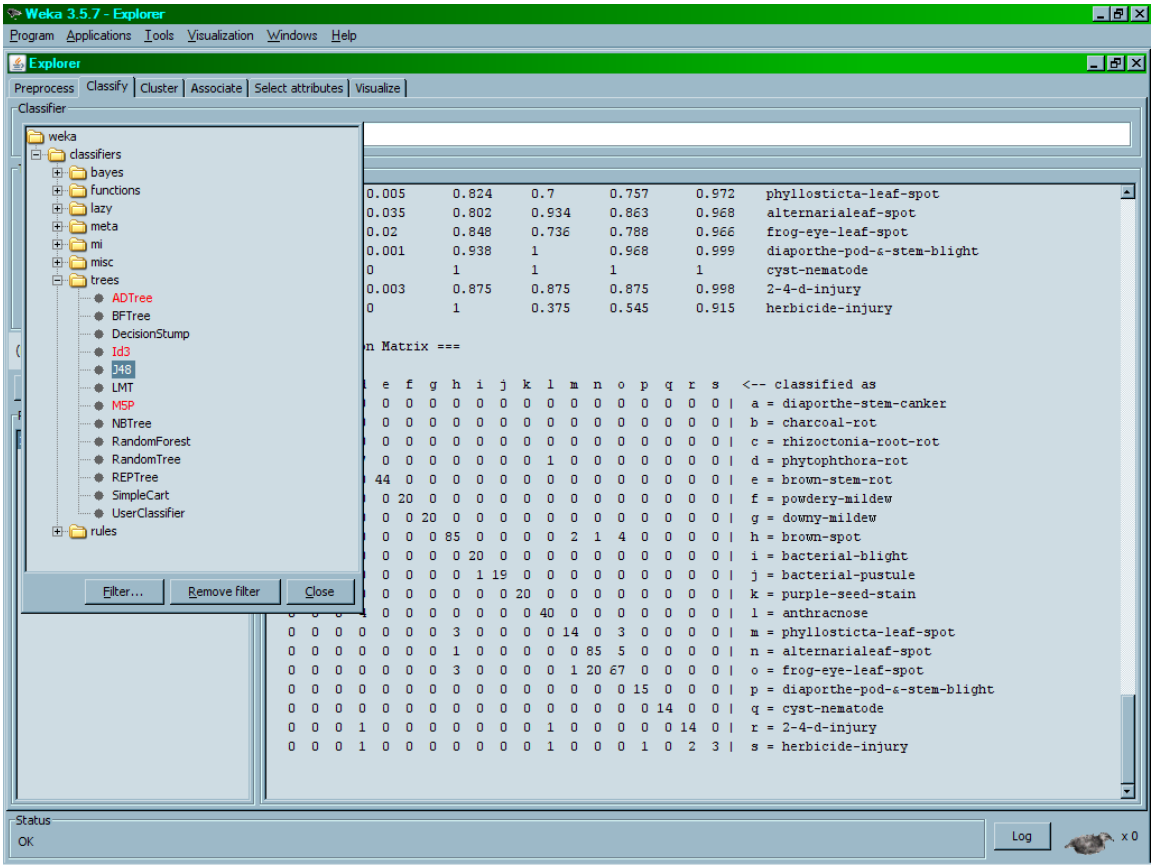


Obrázek č. 6.4: Rozložení hodnot všech atributů (včetně tříd)

Protože cílem je klasifikace do příslušné kategorie choroby dle popsaných příznaků, je nutno zvolit v menu položku *Classify*. Ta nabídne široký výběr různých kategorií klasifikátorů trénovaných aktuálními daty. Jako příklad si lze prostřednictvím položky *Choose* (vyber) v kategorii *Classifier* vybrat kategorii *trees*, což jsou různé algoritmy pro generování rozhodovacích stromů. Velmi často je používaný klasifikátor *J48*, odvozený z poslední nekomerční verze *c4.5* (oprava č. 8) [Witten a Frank, 2005]. Tento generátor stromů je založen na minimalizaci entropie pro měření míry poskytované informace [Quinlan, 1993]. Volbu algoritmu *J48* ukazuje následující ilustrace. Algoritmy, zobrazené červeně, na obr. č. 6.5 nejsou z různých důvodů na aktuální data aplikovatelné – *J48* umějí pracovat se všemi reprezentacemi dat a umějí i nahrazovat chybějící hodnoty, takže je pro daná data použitelný (jeho předchůdce *ID3*, který je také k dispozici; přijímá pouze nominální hodnoty; *ADTree* umí pracovat pouze se dvěma třídami, ale *soybean* má tříd 19).

Vybraný algoritmus zobrazí své implicitně nastavené parametry: **J48** `-C 0.25 -M 2`, což zde znamená, že *konfidenční faktor* C ovlivňuje prořezávání úplného stromu v míře 0.25, kde C = 0.0 je maximální prořezání na základě minimální důvěry v informaci poskytovanou stromem, a naopak C = 1.0 je důvěra maximální, minimalizující prořezání (podrobnosti lze

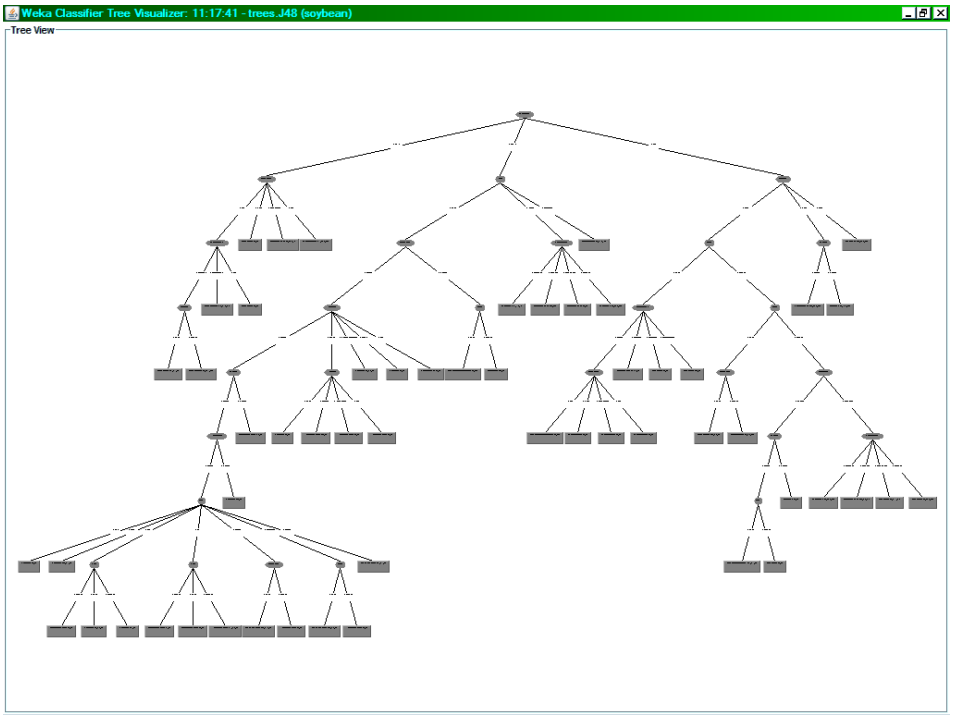
najít v odkazované literatuře). Parametr M říká, kolik instancí může být minimálně v listu stromu – implicitní minimum jsou zde dva trénovací příklady, což umožňuje vyhnout se přeučení stromu, které může být způsobeno tím, že v extrémním případě každý list stromu obsahuje jediný prvek (jednoprvkové množiny), takže se ztratila generalizace; pro některé speciální případy však může být řešení s jednoprvkovými listy naopak žádoucí (např. problémy typu *XOR*).



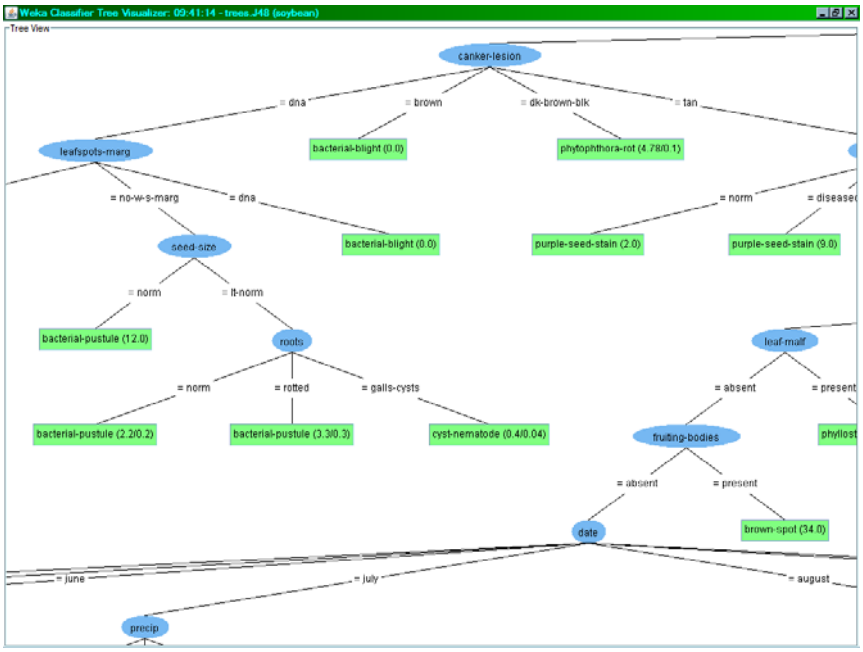
Obrázek č. 6.5: Generování rozhodovacích stromů

Po výběru algoritmu a případné úpravě nastavení parametrů se trénování spustí tlačítkem *Start*. WEKA má implicitně nastavenu volbu *Test options* na desetinásobnou krosvalidaci, ale způsob testování lze ovlivnit – např. zadáním testovacího souboru dat, existuje-li. Po skončení tréninku se zobrazí výpis výsledků v okně *Classifier output*. V okně *Result list* je pro každé trénování jedna položka, která se pravým tlačítkem myši rozvine a lze se podívat na grafické zobrazení stromu *Visualize tree* (viz obr. č. 6.6) nebo na řadu dalších detailů trénování (*Visualize classifier errors,* aj.).

Vizualizace stromu umožňuje zkoumat detailněji i jeho jednotlivé podstromy, jak ukazuje obr. č. 6.7 (v okně *Visualize tree* lze pravým tlačítkem myši vyvolat další okno, v němž lze stanovit umístění stromu, jeho velikost a velikost fontu tak, aby pro velmi malé nebo velké stromy bylo zobrazení vizuálně přijatelné – ne vždy toho lze dosáhnout, takže strom lze studovat i v jeho textovém zobrazení v okně *Classifier output*).



Obrázek č. 6.6: Vizualizace rozhodovacích stromů



Obrázek č. 6.7: Vizualizace rozhodovacích podstromů

Následující ilustrace ukazuje zkrácený výpis výsledků *(Run information)*. Údaje jsou velmi detailní, zahrnují různá vyhodnocení kvality natrénovaného klasifikátoru vzhledem k očekávané budoucí spolehlivosti předpovědi – jde o statistické informace; bližší vysvětlení lze najít např. v [Witten a Frank, 2005]. Prvním vodítkem bývají údaje *Correctly Classified Instances*, kde lze najít absolutní i relativní počty správně a nesprávně klasifikovaných instancí vzhledem ke zvolené trénovací a testovací metodě (*TP Rate* a *FP Rate* informují o poměrech *true* a *false positive):*

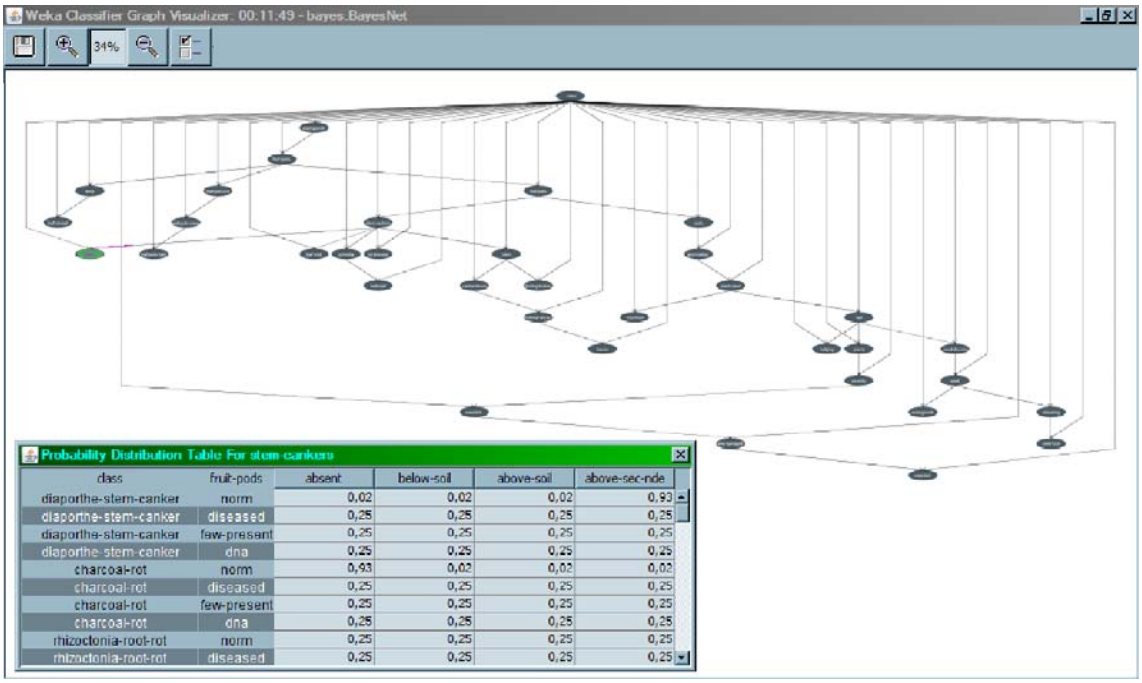
```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     soybean
Instances:    683
Attributes:   36
        .
        . [vynechán výpis názvů atributů]
        .
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------

leafspot-size = lt-1/8
|   canker-lesion = dna
|   |   leafspots-marg = w-s-marg
|   |   |   seed-size = norm: bacterial-blight (21.0/1.0)
|   |   |   seed-size = lt-norm: bacterial-pustule (3.23/1.23)
|   |   leafspots-marg = no-w-s-marg: bacterial-pustule (17.91/0.91)
|   |   leafspots-marg = dna: bacterial-blight (0.0)
|   canker-lesion = brown: bacterial-blight (0.0)
|   canker-lesion = dk-brown-blk: phytophthora-rot (4.78/0.1)
|   canker-lesion = tan: purple-seed-stain (11.23/0.23)
leafspot-size = gt-1/8
|   roots = norm
        .
        . [vynechán zbytek textového popisu stromu]
        .
Number of Leaves  :   61

Size of the tree :    93

Time taken to build model: 0.09 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         625               91.5081 %
Incorrectly Classified Instances        58                8.4919 %
Kappa statistic                          0.9068
Mean absolute error                      0.0135
Root mean squared error                  0.0842
Relative absolute error                 14.0484 %
Root relative squared error             38.4134 %

Total Number of Instances              683

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.95      0.002      0.95      0.95      0.95       0.974    diaporthe-stem-canker
  1         0          1         1         1          1        charcoal-rot
  0.95      0.002      0.95      0.95      0.95       0.974    rhizoctonia-root-rot
  0.989     0.01       0.935     0.989     0.961      0.99     phytophthora-rot
  1         0          1         1         1          1        brown-stem-rot
  1         0          1         1         1          1        powdery-mildew
  1         0          1         1         1          1        downy-mildew
  0.924     0.012      0.924     0.924     0.924      0.995    brown-spot
  1         0.002      0.952     1         0.976      0.999    bacterial-blight
  0.95      0          1         0.95      0.974      0.973    bacterial-pustule
  1         0          1         1         1          1        purple-seed-stain
  0.909     0.005      0.93      0.909     0.92       0.973    anthracnose
  0.7       0.005      0.824     0.7       0.757      0.972    phyllosticta-leaf-spot
  0.934     0.035      0.802     0.934     0.863      0.968    alternarialeaf-spot
  0.736     0.02       0.848     0.736     0.788      0.966    frog-eye-leaf-spot
  1         0.001      0.938     1         0.968      0.999    diaporthe-pod-&-stem-blight
  1         0          1         1         1          1        cyst-nematode
  0.875     0.003      0.875     0.875     0.875      0.998    2-4-d-injury
  0.375     0          1         0.375     0.545      0.915    herbicide-injury

=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s   <-- classified as
```

```
 19  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  a = diaporthe-stem-canker
  0 20  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  b = charcoal-rot
  1  0 19  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  c = rhizoctonia-root-rot
  0  0  0 87  0  0  0  0  0  0  1  0  0  0  0  0  0  0 |  d = phytophthora-rot
  0  0  0  0 44  0  0  0  0  0  0  0  0  0  0  0  0  0 |  e = brown-stem-rot
  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0  0 |  f = powdery-mildew
  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0  0  0 |  g = downy-mildew
  0  0  0  0  0  0  0 85  0  0  0  0  2  1  4  0  0  0 |  h = brown-spot
  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  0  0 |  i = bacterial-blight
  0  0  0  0  0  0  0  0  1 19  0  0  0  0  0  0  0  0 |  j = bacterial-pustule
  0  0  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0 |  k = purple-seed-stain
  0  0  0  4  0  0  0  0  0  0  0 40  0  0  0  0  0  0 |  l = anthracnose
  0  0  0  0  0  0  0  3  0  0  0  0 14  0  3  0  0  0 |  m = phyllosticta-leaf-spot
  0  0  0  0  0  0  0  1  0  0  0  0  0 85  5  0  0  0 |  n = alternarialeaf-spot
  0  0  0  0  0  0  0  3  0  0  0  0  1 20 67  0  0  0 |  o = frog-eye-leaf-spot
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 15  0  0 |  p = diaporthe-pod-&-stem-blight
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 14  0 |  q = cyst-nematode
  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0 14 |  r = 2-4-d-injury
  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  1  0  2  3 |  s = herbicide-injury
```

Další ukázka algoritmu je *PART*, což je jeden z generátorů soustavy pravidel (využívající podobně jako *J48* minimalizaci entropie) jako alternativní reprezentace znalosti:

```
Scheme:       weka.classifiers.rules.PART -M 2 -C 0.25 -Q 1
Relation:     soybean
Instances:    683
Attributes:   36
Test mode:    10-fold cross-validation

PART decision list
------------------

leafspot-size = lt-1/8 AND
canker-lesion = dna AND
leafspots-marg = w-s-marg AND
seed-size = norm: bacterial-blight (21.0/1.0)

int-discolor = none AND
plant-growth = abnorm AND
leaves = abnorm AND
stem = abnorm AND
plant-stand = lt-normal AND
area-damaged = low-areas AND
fruiting-bodies = absent: phytophthora-rot (81.29/0.76)

leafspot-size = lt-1/8 AND
canker-lesion = dna: bacterial-pustule (20.31/1.31)

leafspot-size = gt-1/8 AND
mold-growth = present AND
leaves = abnorm: downy-mildew (20.61/0.61)
        .
        .  [vynechán výpis zbytku pravidel]
        .
Number of Rules  :  40

Time taken to build model: 0.27 seconds
```

Jedním z algoritmů, které využívají pravděpodobnostní přístupy, je např. Bayesova síť (lze zvolit v menu *Choose: Classifiers* > *Bayes* > *BayesNet*). Podle zadaných parametrů lze získat příslušné výsledky včetně grafického zobrazení, např. jak ukazuje obr. č. 6.8:



Obrázek č. 6.8: Výsledky včetně grafického zobrazení

WEKA a jemu podobné softwarové nástroje v současnosti umožňují velmi široké možnosti experimentů, na nichž je dolování znalosti z dat založeno. Nelze zde ve stručnosti probrat ani podstatnou část, protože systém je již tak rozsáhlý a existuje k němu i kniha [Witten a Frank, 2005], která nezahrnuje následné verze – podstata systému, v knize probraná, však umožní z technického hlediska pracovat i s novějšími implementacemi; principy algoritmů je však nutno nastudovat v příslušné literatuře.

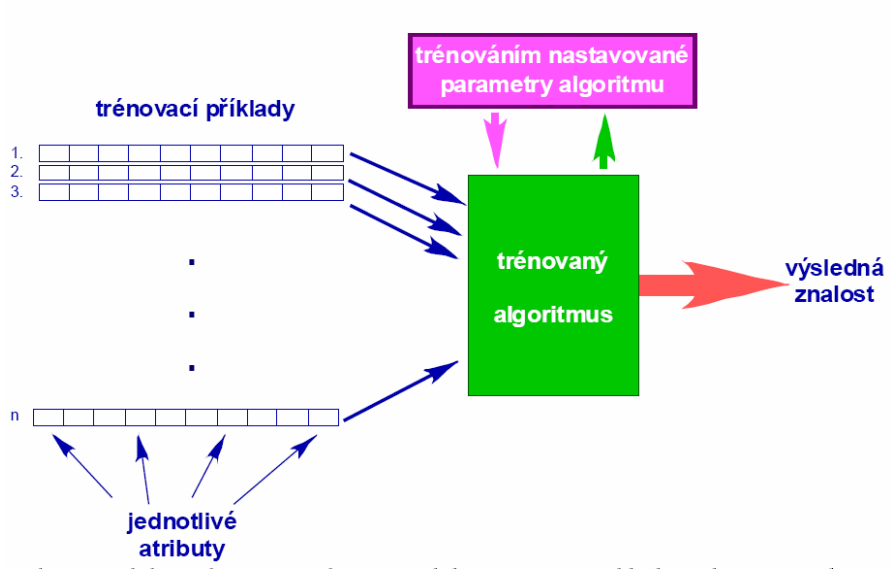
Poznámka: Je nutno zmínit určité varování k používání systému WEKA a dalších obdobných systémů: Tyto softwarové nástroje mohou poskytnout *velmi cennou podporu* při hledání znalosti v libovolných datech, tvořících na vstupu často zcela neprůhledný popis reality. Velké množství různých algoritmů, z nichž mnohé mají řadu parametrů, však *nesmí svádět k* pouhému *povrchnímu zkoušení, co by mohlo fungovat*, i když současné implementace práci velice usnadňují. Především *je nevyhnutelné* dobře a do hloubky *porozumět používaným algoritmům*, jejich výhodám a nevýhodám, a na základě této znalosti s použitím konkrétních dat velmi pečlivě navrhnout experimenty a jejich vyhodnocení. Nástroje obsahují implicitní (*default*) hodnoty parametrů, protože nějak být zadány musejí, ale to v žádném případě neznamená, že pro konkrétní úlohu to jsou ty pravé hodnoty – dolování z dat kromě hledání správného algoritmu musí nastavit optimálně i jeho parametry. Například bez porozumění toho, jak generátory rozhodovacích stromů pracují s parametrem prořezávání (*pruning*), lze získat nesmyslně složité nebo naopak triviální stromy, a tak vlivem přeučení nebo neoprávněné generalizace obdržet chybné výsledky. Stejně tak je nutné dobře znát principy různých metod *vyhodnocování kvality natrénovaných algoritmů*, protože neexistuje jediný univerzální postup a je zapotřebí brát do úvahy *aplikační závislost* disponibilních metod. Obdobné úvahy platí pro všechny parametry používané algoritmy dolování z dat. Při správné aplikaci zmíněných nástrojů na řešený problém však lze očekávat jako odměnu nalezenou znalost, která v původních datech vůbec nemusí být zřejmá a tradiční analytické postupy ji odhalit z nejrůznějších příčin nemohou.

Kapitola 7
Strojové učení
(Jan Žižka)

7.1 Induktivní strojové učení

Induktivní strojové učení je metoda, která má jako vstup *příklady* dat, která jsou často označená, do jaké třídy patří, jakou kategorii představují apod. Každý příklad je popsán *atributy* (vlastnostmi, proměnnými), které mohou obecně nabývat *nominálních, binárních nebo numerických hodnot*. Příslušnost do určité třídy je dána určitou kombinací hodnot atributů. Atributy mohou být navzájem nezávislé, případně mezi nimi může být slabší nebo silnější vzájemná závislost. Mají-li tyto tzv. *trénovací příklady* přispět ke správné generalizaci, musí být *relevantní* vzhledem k řešené úloze. Trénovací objekty popsané v databázi mnoha atributy by měly být předkládány učícímu se algoritmu pouze s relevantními atributy. *Irelevantní atributy* se mohou chovat jako šum, mohou zapříčinit náhodné koincidence, zvyšují výpočetní složitost a znepřehledňují výslednou znalost. Výběr relevantních atributů bývá často obtížným a obvykle je náročnou součástí přípravy dat k učení.

Výstupem metody je objevená znalost v určité formě, kterou představuje natrénovaný algoritmus, jemuž učení nastavilo hodnoty parametrů. Parametry mohou být např. testy a uzly entropického rozhodovacího stromu, váhy spojů umělé neuronové sítě, aposteriorní pravděpodobnosti Bayesova klasifikátoru, instance určující oddělovací hranici mezi třídami pro *support vector machines*, antecedenty a počet generovaných pravidel, složení tzv. chromozomu jako reprezentace optimálního řešení genetického algoritmu, partikulární regresní přímky v regresním rozhodovacím stromu, počet a složení shluků algoritmu *X-means*, a mnoho dalšího.



Obrázek č. 7.1: Schéma induktivního strojového učení, kdy trénovací příklady vedou přes zobecnění konkrétních vzorků dat ke znalosti aplikovatelné (s předpovězenou chybou) na instance odlišné od trénovacích.

Soustava trénovacích příkladů má ve strojovém učení standardně formu tabulky, kde jednotlivé řádky představují příklady a sloupce atributy. Poslední sloupec bývá určen

klasifikační třídě, kde hodnoty představují zařazení do tříd. Zvolený trénovací algoritmus čte příklady předkládané postupně (inkrementální trénování) nebo současně (dávkové trénování) a hledá typické vzory (prototypy) pro jednotlivé třídy. Popsaná metoda je obecná, pro jednotlivé algoritmy se může v detailech lišit: některé algoritmy nemusí transformovat soubor konkrétních příkladů do prototypů (např. jednoduchá metoda nejbližšího souseda, popsaná dále, si předložené vzorky pouze uloží, a pak hledá vzorek, který je neznámému vzorku nejpodobnější), jiné si zase naopak pamatují pouze indukcí získané zobecnění, trénovací příklady si vůbec nepamatují (např. umělé neuronové sítě). Některé algoritmy potřebují projít trénovací příklady cyklicky mnohokrát (umělé neuronové sítě), takže předkládané pořadí není zásadně důležité (oprava parametrů vlivem určitého příkladu může být narušena příkladem následujícím). Jiným algoritmům stačí jediný průchod, což může někdy klást důraz i na pořadí, v němž jsou jednotlivé instance předkládány k tréninku, protože určité pořadí příkladů může způsobit pohyb prostorem jiným směrem než pořadí odlišné – namísto přibližování se ke globálnímu extrému dojde pak jeho vzdalování, a pak třeba k uvíznutí v lokálním extrému.

7.2 Řízené a neřízené učení

Řízené (*supervised*) **učení**, "s učitelem", se vyznačuje tím, že trénovaný příklad má jako svou součást informaci o sounáležitost do konkrétní třídy. Pokud trénovaný algoritmus zařadí vstupní trénovací příklad správně, nemusí měnit hodnoty svých parametrů (na počátku nastavených náhodně nebo na nejobvyklejší hodnoty). Když však pomocí takto realizované zpětné vazby algoritmus zjistí, že udělal chybu, parametry upraví, aby svou chybovost minimalizoval. Tento způsob induktivního trénování je ideální, zejména když jsou k dispozici dobrá trénovací data vhodně pokrývající řešený problém (např. dodržením rozložení hodnot, neexistencí chybějících hodnot apod.). Učení lze patřičně cílit. Klasifikace budoucích instancí je pak dána výsledkem trénování.

Neřízené (*unsupervised*) **učení**, "bez učitele", tuto zpětnou vazbu postrádá, protože některá data nemají k dispozici označení příslušnosti do třídy. Učení je obtížnější, žák je ponechán „sám sobě". Tento způsob učení je náročnější, protože trénovaný algoritmus nemá možnost si ověřit správnost své odpovědi, odpověď známa není, cílem je přitom obvykle najít, jak by se data dala vhodně do nějakých tříd, jejichž počet také nemusí být znám, rozdělit. Typickými představiteli takto trénovaných algoritmů jsou algoritmy *shlukovací* (*clustering*), kde výsledné shluky jsou považovány za potenciální třídy, a mohou být následně použity pro řízené trénování, třeba stanovení parametrů klasifikátoru typu rozhodovací strom nebo jiného, tj. shlukování může být také součástí předzpracování trénovacích dat.

Různé aplikace vyžadují obecně oba přístupy. Například zpracování environmentálních dat může vyžadovat neřízené učení, kdy velká množství naměřených příkladů je nejprve zapotřebí roztřídit do nějakých "typických" skupin a ty v budoucnu použít pro sledování změn prostředí. Každá typická skupina pokrývá množinu vzorků, jejichž vzájemná podoba je větší než podobnost se vzorky s ostatních shluků – je to zároveň zřejmý příklad dosaženého zobecnění konkrétních příkladů, protože shluk (nyní již třída) je jediná entita, která již trénovací vzorky pro budoucí klasifikace nepotřebuje.

7.3 Podobnost

Míra nějaké vzájemné podobnosti instancí je jednou z nejdůležitějších věcí při dolování z dat. Stanovení podobnosti či odlišnosti trénovacích vzorků lze provést řadou způsobů pro učení řízené i neřízené. Stroje pracují se vzorky převedenými do abstraktního prostoru, kde každý vzorek je představován mnohorozměrným bodem nebo vektorem, přičemž souřadnice jsou dány hodnotami atributů (někdy vhodně transformovanými, např. normalizací).

Míra podobnosti je pak dána například vzdáleností mezi body (navzájem si blízké body jsou si podobnější než vzdálené), nebo úhlem mezi vektory (větší úhel znamená menší podobnost), apod. Vzdálenost l mezi dvěma body (1) a (2) lze měřit různě. Obvyklá je vzdálenost Eukleidova (platná v rovině) pro body o souřadnicích $A_1 [a_1^{(1)},...,a_k^{(1)}]$ a $A_2 [a_1^{(2)},...,a_k^{(2)}]$:

$$l = \sqrt{\sum_{i=1}^{k} (a_i^{(1)} - a_i^{(2)})^2} \,, \tag{7.1}$$

kde k je počet atributů, $a_i^{(1)}$ jsou souřadnice (hodnoty atributu) prvního bodu, a $a_i^{(2)}$ souřadnice druhého bodu. Eukleidova vzdálenost (neboli tzv. L_2 norma) předpokládá, že v rovině se lze pohybovat po libovolné přímce (i šikmé). Pokud je pohyb omezen jen na přímky rovnoběžné s osami, pak se používá tzv. Manhattanská vzdálenost (název inspirován vzájemně rovnoběžnými a kolmými ulicemi vedoucími ze severu na jih a z východu na západ ve stejnojmenné čtvrti města New York):

$$l = \sum_{i=1}^{k} | a_i^{(1)} - a_i^{(2)} | , \tag{7.2}$$

tj. součet vzdáleností (absolutních hodnot z rozdílů) mezi souřadnicemi obou bodů na jednotlivých osách (L_1 norma). Používají se i další typy vzdáleností, např. vycházející z Mahalanobisova vztahu založeného na vzdálenosti mezi vektorem středu μ a vektorem příkladu x:

$$r^2 = (x - \mu)^T \Psi^{-1} (x - \mu) \,, \tag{7.3}$$

kde $(x - \mu)^t$ je transponovaný vektor a Ψ^{-1} je inverzní matice kovariance. Znamená to, že podobné příklady leží uvnitř hyperelipsoidu, jehož osy uvedený vztah určuje; detaily viz např. [Duda, Hart, Stork, 2001]. Podobnost mezi vektory se často určuje pomocí jejich vzájemného úhlu α, resp. $\cos(\alpha)$, kde 1 znamená totožnost a 0 odlišnost, hodnoty uvnitř intervalu pak různou míru podobnosti. Tato metoda se uplatňuje např. při srovnávání podobnosti textových dokumentů, kde jednotlivé souřadnice jsou dány konkrétními *slovy* jako hodnotami atributů. Podobnost se počítá dle vztahu:

$$\cos(\alpha) = \frac{x^T y}{\| x \| . \| y \|} \text{, kde } \| v \| = \sqrt{v^T v} \,.$$

Typickým problémem v dolování z dat je, že nelze obecně spolehlivě předpovědět, který způsob měření vzdáleností je pro danou úlohu správný. Eukleidova vzdálenost poskytuje dobré výsledky velmi často a používá se jako základní východisko, pokud tomu nic nebrání. Z experimentální zkušenosti plyne, že zmíněné alternativní vzdálenosti jsou používány pro speciální druhy problémů, užití výpočtu užitím Eukleidovy vzdálenosti v praxi převažuje.

Pokud jsou atributy nominální, resp. binární, pak se používají jiné metody výpočtu podobnosti, např. tzv. *Hammingova vzdálenost*, d_H, která jako výsledek dává číslo určující počet rozdílných vlastností mezi oběma porovnávanými objekty (takže $d_H = 0$ znamená, že objekty jsou zcela stejné):

1 0 0 0 1 0 1 1 0 1 1 1 1 1 0 0 1
1 0 0 0 1 0 0 1 0 1 1 1 0 0 1 0 1

Oba vektory zde mají mezi sebou vzdálenost $d_H = 4$.

7.4 Vyhodnocení výsledků strojového učení

Vstupní trénovací data jsou dána realitou. Trénovaný algoritmus je nutno zvolit; záleží na konkrétní aplikaci (tzv. aplikační závislost je pro dolování z dat typická). Jsou-li hranice

oddělující jednotlivé třídy nebo hledaný půběh aproximace neznámé funkce nelineární, nemůže např. lineární perceptron nebo entropický rozhodovací strom dát obecně dobré výsledky s nízkou chybou, takže je nutné zkusit např. vícevrstvou sigmoidální síť. Často to nebývá předem známo, a proto je nutno hledat experimentálně i optimální algoritmus. Platí zde matematicky prokázaná skutečnost zvaná *no-free-lunch* ("žádný oběd není zadarmo"), která říká, že žádný algoritmus není obecně lepší než ostatní algoritmy, vezmou-li se do úvahy všechna možná data, která mohou existovat [Wolpert, Macready, 1995; Duda, Hart, Stork, 2001]. Na určité části těchto dat fungují lépe některé algoritmy, na jiné části zase algoritmy odlišné, takže zatímco na nějakých datech může třeba metoda nejbližšího souseda *k*-NN dát vynikající výsledky a umělá neuronová síť zcela selže, pro jiná data tomu může být zcela naopak.

Po volbě algoritmu a adekvátní přípravě trénovacích dat se algoritmus natrénuje, neboli na základě dat se stanoví jeho konkrétní parametry. Samotné trénování poskytne hodnotu chyby, např. kolikrát byla klasifikace trénovacích příkladů provedena špatně a kolikrát správně. Tato tzv. *trénovací chyba* však se vztahuje pouze ke známým datům a může být velmi nízká nebo dokonce nulová – algoritmus se může *přetrénovat* a dokonale pracovat s daty, u nichž však klasifikaci známe, zatímco budoucí data bez známého začlenění do příslušné třídy mohou být klasifikována zcela chybně.

Pro odhad spolehlivější chyby je zapotřebí natrénovaný algoritmus vyzkoušet na datech, která při tréninku neviděl, tzv. *testovací data.* Chyba dosažená na testovacích datech je pak použita pro odhad chyby v budoucnosti. Znamená to, že v okamžiku trénování je zapotřebí část dat oddělit a použít jen pro testování. Způsob výběru testovacích dat závisí např. i na jejich množství, rozložení apod., takže se uplatňují i statistické metody.

K základním a obvyklým metodám vyčlenění testovacích příkladů patří *bootstrap* a *krosvalidace,* i když existuje i řada dalších postupů, popsaných v literatuře, např. v [Duda, Hart, Stork, 2001; Witten, Frank, 2005] a v mnoha odkazech tam uvedených.

- *Bootstrap* je statistická metoda založená na výběru s vracením. Znamená to, že tímto výběrem vzniklá sada příkladů může obsahovat některé příklady vícekrát (takže nelze hovořit o *množině*). Myšlenka vychází z toho, že když některé příklady jsou náhodně vybrány vícekrát, jiné nejsou vybrány nikdy – a právě tyto nevybrané příklady pak vytvoří testovací množinu. Data, mající řekněme n příkladů, jsou podrobena n výběrům s vracením. Šance, že nějaký příklad nebude vybrán v žádném z n pokusů (výběr musí být *náhodný*), je založena na tom, že pravděpodobnost být vybrán je $1/n$ a nebýt vybrán $1–1/n$. Opakuje-li se výběr n-krát, pak vynásobením pravděpodobností vznikne pravděpodobnostní odhad:

  $$(1-\frac{1}{n})^n = e^{-1} \approx 0.368\,,$$ kde $e \approx 2.7183$ ... základ přirozených logaritmů.

  Odhad stanovuje, že v testovací množině bude cca 36.8 % příkladů, zatímco v trénovací množině zůstane 63.2 % – někdy se této metodě výběru říká "0.632 *bootstrap*"; samozřejmě, že ani trénovací ani testovací množina pak *neobsahují* opakované příklady.
- *Krosvalidace* (*crossvalidation*) je alternativní, velmi často používaná metoda odhadu chyby postupem "křížového ověřování". Náhodným výběrem *bez vracení* se původní množina n datových příkladů rozdělí na k podmnožin s přibližně stejným počtem příkladů n/k, přičemž se postupně jedna z podmnožin použije na testování a zbylých $k–1$ na trénování. Celý proces se opakuje k-krát, vždy s jinou testovací podmnožinou,

takže každý příklad je použit pro trénování i testování. Obvyklá hodnota bývá $k = 10$, ale lze použít i jiné hodnoty za předpokladu zvážení možného vlivu. Heuristickou hodnotu $k = 10$ poskytly rozsáhlé testy s mnoha různými daty a algoritmy, přičemž se ukázalo, že 10 je ta "pravá" hodnota pro nejlepší odhad budoucí chyby; existuje literatura popisující také určitou teoretickou podporu hodnotě 10.

- 1-z-*n* (*leave-one-out*) je speciální případ krosvalidace, kdy $k = n$, takže se testuje *n*-krát jedním, pokaždé jiným příkladem. Výhoda je v tom, že téměř všechna data jsou vždy použita k trénování, takže odhad budoucí chyby je přesnější, ale jako obvykle vznikají i nevýhody v tom, že jediný prvek neobsahuje informaci o rozložení hodnot a také se zvýší výpočetní nároky [trénování neproběhne např. (n/10)-krát (*pro* $k = 10$), ale *n*-krát, což u algoritmů typu umělá neuronová síť nebo *support vector machines* může při větším počtu příkladů vést k nepřijatelným dobám výpočtu].

Je nutné vzít do úvahy, že množina dat, z níž byly odebrány příklady pro testování, poskytuje méně trénovacích příkladů, což může vést k vyšší chybovosti natrénovaného algoritmu (jeho parametry mohou být odlišné od výsledku se všemi příklady). Takto zjištěná chyba se považuje za *pesimistický odhad*, protože pro aplikaci se nakonec algoritmus natrénuje s použitím *všech* příkladů, což obvykle dá odhadnutou chybu menší.

Vzhledem k náhodnému vybírání je vhodné popsaný postup pro *bootstrap* a *krosvalidaci* zopakovat vícekrát a jako odhad výsledné chyby použít průměrnou hodnotu z provedených experimentů. Obvykle se provádí opakování desetinásobné (např. *ten-folds ten-times crossvalidation*, čili náhodné rozdělení na 10 částí zopakováno 10 krát – vede to ovšem ke 100 trénování, což může být časově i velmi náročné). Zároveň je dobré sledovat, jak moc se od sebe liší výsledky v jednotlivých krocích odhadu chyby, protože velké rozdíly mohou indikovat nestabilitu algoritmu vůči složení dat, takže je pak dobré vyzkoušet algoritmy alternativní i různé poměry trénovacích a testovacích dat.

Porovnání výsledků různých metod s různými parametry pro řešení téhož problému vyžaduje pečlivé statistické vyhodnocení, např. získaných průměrných hodnot a jejich rozptylů; často se používá *t*-test. Popis statistických metod vyhodnocení experimentů však překračuje zaměření i rozsah tohoto textu, ale existuje mnoho literatury popisující tato vyhodnocení, aby byl pro budoucí aplikaci vybrán a implementován co nejspolehlivější algoritmus.

7.5 Základní algoritmy induktivního strojového učení

Algoritmů jsou k dispozici desítky, s modifikacemi stovky. Velké množství modifikací souvisejí s relativně silnou aplikační závislostí, protože aplikací strojového učení v nejrůznějších oblastech existují tisíce, počínaje elektronickou poštou (učení se rozeznávat *spam* z příkladů) přes robotiku, lékařství (bioinformatika) až po aplikace vojenské aj.

Zde jsou zmíněny jen některé algoritmy pro uvedení do problematiky, avšak typické pro danou oblast. Je zapotřebí mít na mysli, že je to právě praxe, která si vynucuje rozvoj metod umělé inteligence a dolování z dat, a že problémy reálného světa předbíhají možnosti jejich řešení.

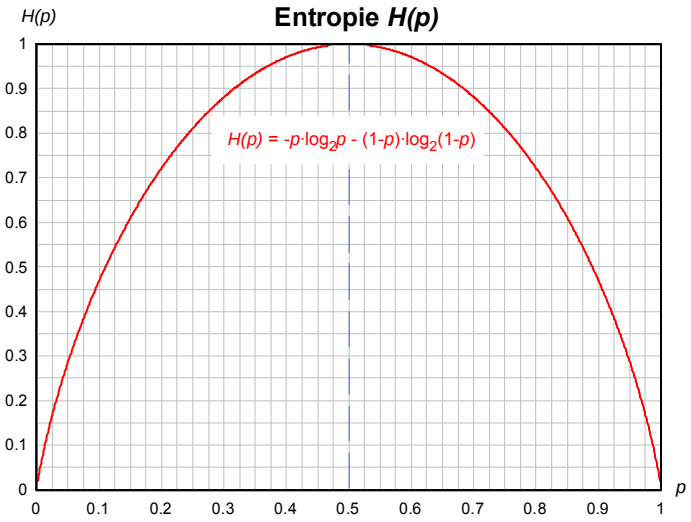
7.5.1 Rozhodovací stromy

Rozhodovací stromy patří k účinným algoritmům, i když mívají svá principiální omezení. Jednou z nejužívanějších metod automatického generování stromu je minimalizace entropie [Quinlan, 1993]. Výchozí množina tréninkových příkladů se pomocí testů na hodnoty atributů dělí na podmnožiny, které jsou více homogrnní z hlediska obsahu členů jednotlivých tříd; v

ideálním případě obsahují pouze členy jediné třídy (nulová entropie), takže není nutno prvky takové množiny nijak zkoumat, neboť je dosaženo ideálního zobecnění – při splnění testů dospěje dotaz do soupisu, který mu sdělí, kam náleží příslušná instance, jejíž zařazení bylo neznámé. Entropie $H(X)$ je součástí teorie informace a je popsána vztahem:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i ,$$

kde X je náhodná proměnná, pro niž je známo n hodnot s pravděpodobnostmi p_i. Logaritmus se základem rovným dvěma se používá tehdy, když se poskytovaná informace vyjadřuje v *bitech* (pozn.: zde může výsledek dávat i desetinná čísla, tj. části bitů). Pro jev, který může nabývat dvou hodnot (např. *ano* a *ne*), lze pomocí entropie spočítat homogenitu množiny. Maximální nehomogenita je při rozdělení instancí z obou tříd 50 %: 50 %, maximální homogenita pro 100 % : 0 % (a naopak):



Obrázek č. 7.2: Entropie H(p)

Z toho plyne, že je nutno hledat takové testy (dotazy na určité atributy), které povedou ke snížení entropie vůči výchozímu stavu. Algoritmus hledá atribut, který nejlépe rozdělí původní směs příkladů do podmnožin. Poté zkouší stejným způsobem dělit vzniklé podmnožiny až do konce daného nemožností získat lepší výsledek. Je nutno zabránit ztrátě generalizace a přeučení se na trénovací příklady, kdy by mohly vzniknout podmnožiny obsahující pouze jeden prvek (a tedy s nulovou entropií). Strom by pak dosahoval dokonalého výsledku na trénovacích datech, ale špatného na testovacích, a zejména na budoucích reálných, dosud neznámých instancích. Takto vybudovaný strom umí rozdělit prostor pomocí nadrovin rovnoběžných s osami, což pro šikmé nelineární hranice mezi třídami nemusí dát dobrý výsledek.

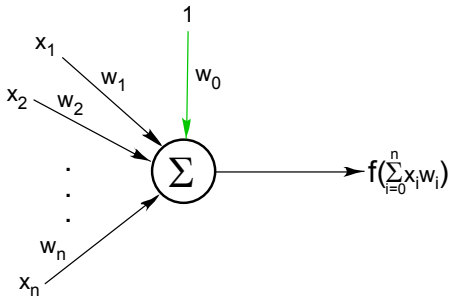
Strom má také výhodu v tom, že se do testů nemusí některé atributy vůbec dostat, protože nepřispívají ke snížení entropie, takže tímto způsobem lze zkoumat relevantnost a irelevantnost atributů, což je velmi cenné. Vybudovaný strom lze snadno převést na soubor pravidel, protože v principu každá větev z kořene do listu představuje jedno pravidlo (list je *konsekvent*, testovací uzly *antecedent*). Takto získaná pravidla lze (vlastně jako jinou reprezentaci znalosti) ještě optimalizovat i vzhledem k jejich pořadí, takže mohou být některé testy (nebo i pravidla) vyřazeny.

7.5.2 Nejbližší soused

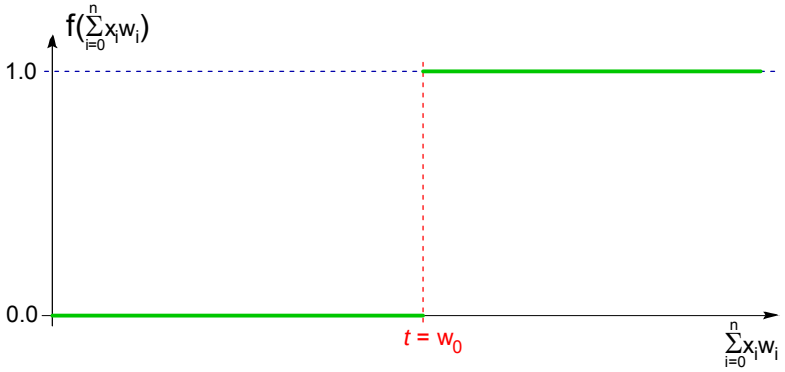
Algoritmus nejbližšího souseda *k*-NN (*k* nearest neighbors) si jednoduše uloží do databáze poskytnuté příklady, o nichž je známo, do jaké třídy patří. Je-li mu předložen neznámý případ, pak hledá, která z uložených instancí je nejblíže, tj. je nejpodobnější, a ta pak určí zařazení neznámého případu; to je 1-NN. Někdy dává lepší výsledky srovnání *s* k > 1 nejbližšími sousedy, např. pro $k = 5$ mohou být v okolí zkoumaného případu dva sousedi pozitivní a tři negativní, takže pak rozhoduje většina. Hodnotu *k* bývá vhodné zjistit experimentálně vzhledem k nejmenší chybě dosahované na testovacích datech. Podobnost se dá měřit různě, nejčastěji to je Eukleidova vzdálenost (7.1). Algoritmus je citlivý na šum a na irelevantní atributy. Má několik modifikací, obecně náleží do skupiny zvané *instance-based learning, IBL* (*instance-based learning,* učení založené na instancích), kde lze jeho nedostatky odstranit nebo potlačit.

7.5.3. Umělé neuronové sítě

Inspirací pro umělé neuronové sítě jsou biologické neuronové struktury. Umělý neuron [Gallant, 1994; Hassoun, 1995] na obr. č. 7.3 je v principu jednotka, do níž vstupují hodnoty atributů x_i, jejichž význam je modifikován váhováním (vynásobením reálným číslem w_i) a výstupem je signál daný součtem vstupních váhovaných hodnot::



Obrázek č. 7.3: Umělý neuron

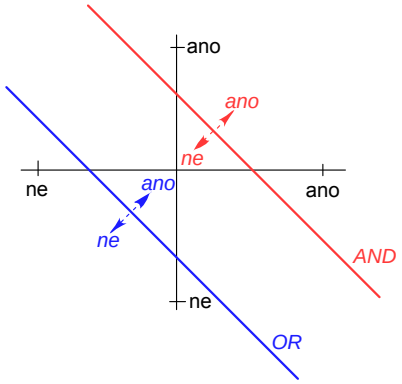


Obrázek č. 7.4: Skokový přechod hodnoty výstupu umělého neuronu po překonání prahové hodnoty *t*

Neuron je aktivován, pokud jeho výstup překročí určitou **prahovou hodnotu** *t* (*threshold*) pro *n* hodnot vstupních atributů: $\sum_{i=1}^{n} x_i w_i > t$, což znamená, že pro aktivaci neuronu je nutno najít váhy w_i a práh *t*. Určité zjednodušení jak to provést přináší úvaha, že jde-li formálně nahradit práh jednotkovým vstupem váhou hodnotou prahu $t = w_0$, pak se hledají pouze váhy:

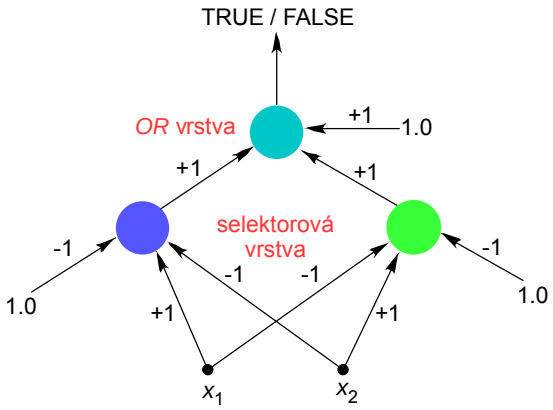
$\sum_{i=1}^{n} x_i w_i > 0$, takže aktivace je dosažena při kladné hodnotě výstupu. Obr. č. 7.4 ukazuje skokový přechod hodnoty výstupu umělého neuronu po překonání prahové hodnoty *t* součtem váhovaných vstupů:

Neuron tedy poskytuje lineární kombinaci váhovaných vstupů, což nemusí realitě vždy vyhovovat, protože třídy mohou být odděleny nelineární hranicí. Lineární neuron umožňuje realizaci základních logických funkcí *AND, OR, NOT,* ale selhává např. na *XOR* (*buď'— anebo*, *vylučovací nebo*). Kombinace vstupů *ano/ne* a odpovědí pro *AND* a *OR* ilustruje obr. č. 7.5.



Obrázek č. 7.5: Kombinace vstupů *ano/ne* a odpovědí pro *AND* a *OR neuronu*

Složitější libovolné binární funkce lze realizovat více neurony zapojenými do vrstvené sítě, například pro zmíněnou funkci *XOR* lze použít následující síť se dvěma vrstvami:



Obrázek č. 7.6: Vrstvená síť neuronů

Na obr. č. 7.6 je použita alternativa *ne* = –1 rovnocenná s *ne* = 0; *ano* = 1 vždy (pro *ne* = 0 by zde bylo nutno nastavit jinak váhy; někdy se dává přednost *ne* = –1, protože násobení nulou dá už navždy jen nulu, takže ani velká váha již nemůže změnit zrušený vliv příslušného propojení). Na výstupu je TRUE pro vstupy x_1 a $x_2 \in \{1, -1\}$ nebo $\{-1, 1\}$, jinak FALSE pro $\{1, 1\}$ nebo $\{-1, -1\}$.

Jednotky s lineárními přenosovými funkcemi, propojené do vrstvené sítě, mohou dát jako celkový výstup opět jen lineární kombinaci vstupů. Aby bylo možno realizovat v praxi se vyskytující nelineární závislosti, je nutno použít nelineární přenosové funkce. Nejčastěji se

používá tzv. sigmoidální prahová funkce, *sigmoida*, která odstraňuje nespojitost danou skokovým prahem:

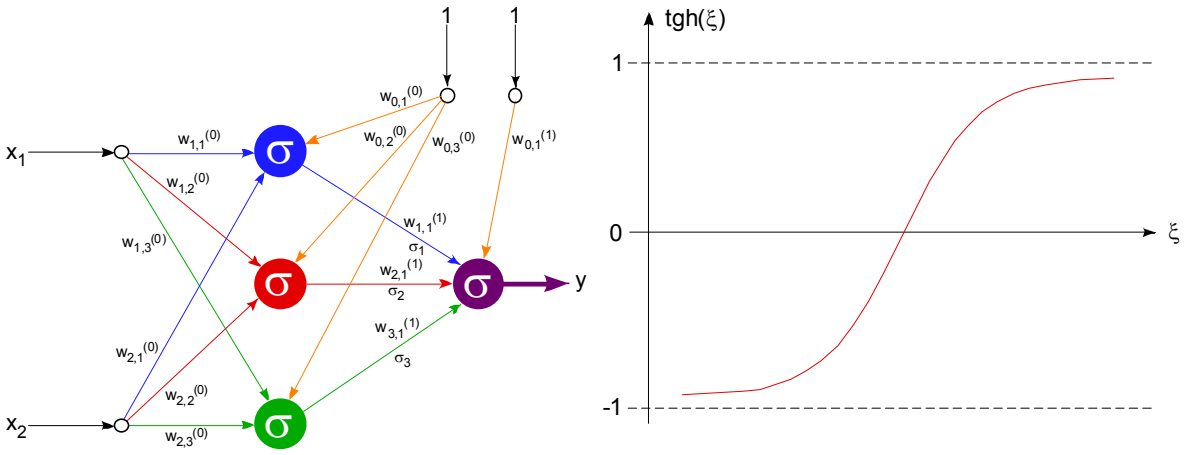
$$\sigma(x) = \frac{1}{1 - e^{-\beta x}}, -\infty \le x \le \infty, 0 \le \sigma(x) \le 1,$$

kde *e* je základ přirozených logaritmů a konstanta β ovlivňuje strmost funkce. Alternativou může být hyperbolický tangens $-1 \le \text{tgh}(\beta x) \le +1$, pokud je zapotřebí mít průběh výstupu od −1 do +1 místo od 0 do +1. Nelineární jednotka rovněž napřed spočítá lineární kombinaci svých váhovaných vstupů, a pak na ni aplikuje prahovou funkci. Obecně má nelineární prahová funkce být diferencovatelnou nelineární funkcí vstupů. Diferencovatelnost je nutná pro hledání sestupného gradientu, tj. směru vedoucího k minimální chybě při aproximaci neznámé funkce na základě znalosti konečného počtu vstupních a jim odpovídajících výstupních hodnot. Z hlediska klasifikace jde o nelineární hyperplochu oddělující od sebe prvky jednotlivých tříd v mnohorozměrném prostoru, kde výsledné parametry umělé neuronové sítě jako klasifikátoru jsou iterativně nastaveny trénovacími daty. Trénování je řízené (*supervised*).

Následující obr. č.7.7 ukazuje síť se dvěma vstupy x_1 a x_2, s jednou skrytou vrstvou obsahující tři sigmoidální jednotky, a s výstupní vrstvou s jednou sigmoidální jednotkou. Trénováním se hledají hodnoty vah $w_{i,j}^{(k)}$ u propojení jednotek, kde *i* je pořadové číslo jednotky v dané vrstvě, *j* označuje číslo propojení pro každou jednotku v nižší vrstvě směrem k vrstvě vyšší, a (*k*) udává, v které vrstvě se jednotka nachází ($k = 0$ je pro vstupní vrstvu, $k = 1$ pro skrytou). Jednotky s $i = 0$ jsou náhrady prahů *t* váhami a jednotkovými vstupy. Iterativní nastavování hodnot vah (na začátku jsou to náhodná malá reálná čísla, např. v rozsahu −0.01…+0.01) se provádí vhodným algoritmem, velmi často se používá tzv. metoda zpětného šíření chyb (*backpropagation of errors*), která ovšem chyby nešíří, ale postupně směrem od výstupní vrstvy zpět ke vstupu opravuje hodnoty vah tak, aby zmenšila dosahovanou chybu. Proces se opakuje mnohokrát, tj. každý tréninkový příklad je síti předložen opakovaně a po určitém počtu kroků je dosažena nějaká, dále se již nesnižující hodnota chyby na výstupu (může být i nulová). V principu se minimalizuje čtverec chyby dané rozdílem mezi tím, co síť poskytuje na výstupu a požadovanou hodnotou výstupu pro danou kombinaci vstupních hodnot:

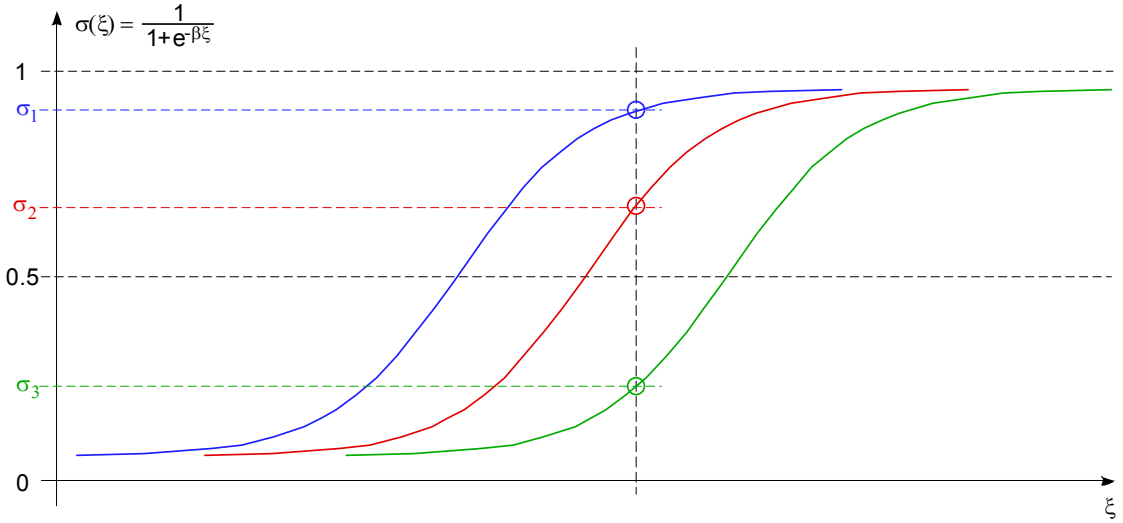
$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k} (t_{kd} - y_{kd})^2,$$

kde *k* označuje jednotku. Počítá se tedy celková chyba přes všechny výstupy jednotek; t_{kd} je požadovaný výstup *(true value)* a y_{kd} je skutečný výstup *k*-té jednotky pro příklad *d*. Metoda je citlivá na uvíznutí v lokálních extrémech, protože reálný prostor je transformován do umělého silně nelineárního prostoru, kde jednotlivými dimenzemi jsou váhy, jejichž počet rychle narůstá s počtem jednotek i vrstev (v obou případech není jejich množství omezeno a je dáno návrhem architektury sítě). Definovat správnou architekturu sítě pro konkrétní aplikaci není snadná úloha, obvykle je nutno provést řadu experimentů, resp. použít pro návrh sítě např. nějaký optimalizátor typu *genetické algoritmy* (jsou popsány v další kapitole). Popisovaný typ sítě je se vyznačuje šířením signálů pouze od vstupu k výstupu (*feedforward*). Existují i složitější architektury se zpětnými vazbami. Obr. č. 7.7 vedle sítě zobrazuje průběh funkce *hyperbolický tangens* jako alternativu k sigmoidě.



Obrázek č. 7.7: Síť se dvěma vstupy x_1 a x_2, s jednou skrytou vrstvou a průběh funkce *hyperbolický tangens* jako alternativu k sigmoidě

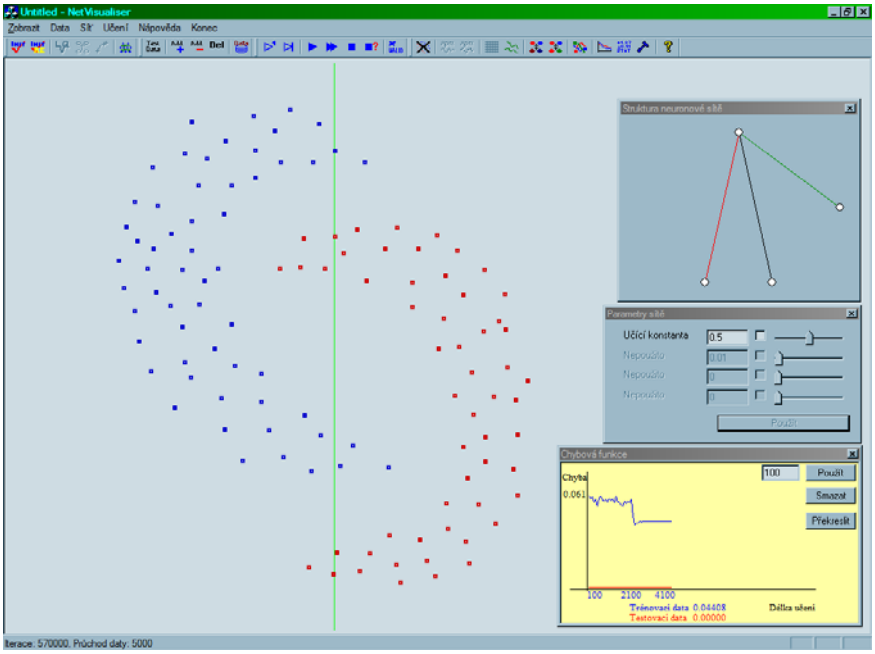
Obr. č. 7.8 ilustruje sigmoidy pro jednotlivé neurony skryté vrstvy, včetně odezvy na vstupní hodnotu.



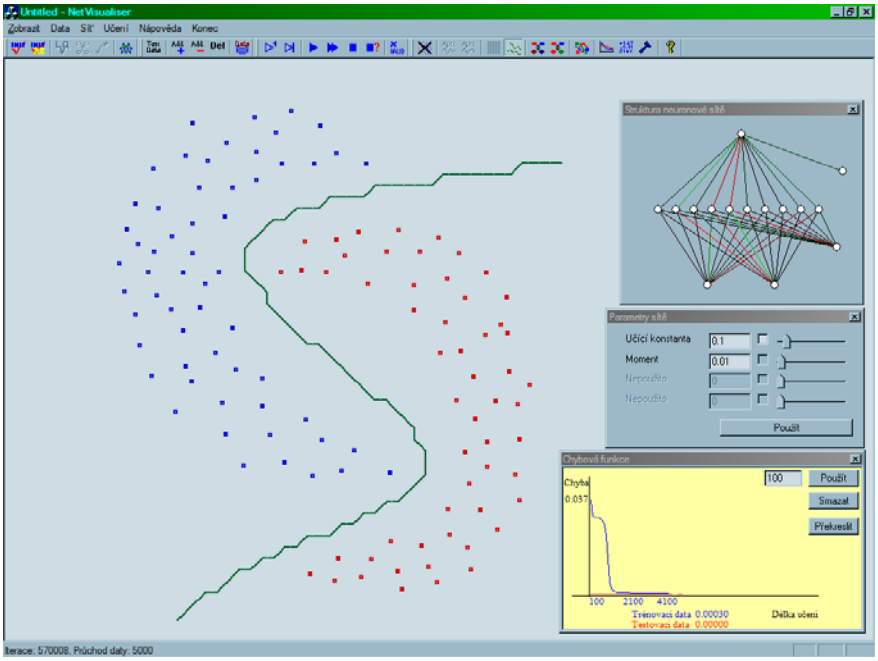
Obrázek č. 7.8: Sigmoidy pro jednotlivé neurony skryté vrstvy

Následující obr. č. 7.9 a 7.10 ukazují výsledek jednak pro lineární jednotku, jednak pro sigmoidální síť pro tatáž data, která představují dvě třídy oddělené nějakou nelineární hranicí (testovací úloha na oddělení dvou skupin dat představujících do sebe zaklíněné dva půlměsíce v dvourozměrném reálném prostoru – modré a červené body). Lineární jednotka našla (zelenou) přímku oddělující obě třídy ne zcela dokonale (nelze je lineárně oddělit), zatímco sigmoidální síť, trénovaná zpětným šířením chyb, navrhla (pro daná trénovací data) jedno z možných nelineárních oddělení ve tvaru "S" křivky. V obou případech síť prošla opakovaně všemi trénovacími daty pět tisíckrát. Poté byl trénink zastaven, protože chyba už se dále nesnižovala. Obrázky demonstrují architekturu sítí i průběh poklesu chyby v závislosti na počtu trénovacích kroků. (Program pro ukázku činnosti různých typů sítí včetně vizualizace viz [Marťán, 2002].)

Popisovaný typ nelineárních sítí je teoreticky schopen aproximovat libovolné funkce v případě trojvrstvé architektury. Není však známo, jak spolehlivě navrhnout počty jednotek ve vrstvách.



Obrázek č. 7.9: Výsledek pro lineární jednotku



Obrázek č. 7.10: Výsledek pro sigmoidální síť

7.5.4 Genetické algoritmy

Genetické algoritmy, (zkráceně GA), hledají optimum pomocí inspirace Darwinovou teorii o vývoji druhů: přežívají (tj. mají potomstvo) ti lepší, přizpůsobenější [Goldberg, 1989]. Myšlenka použití GA vychází z představy, že na počátku v tzv. nulté generaci se náhodně

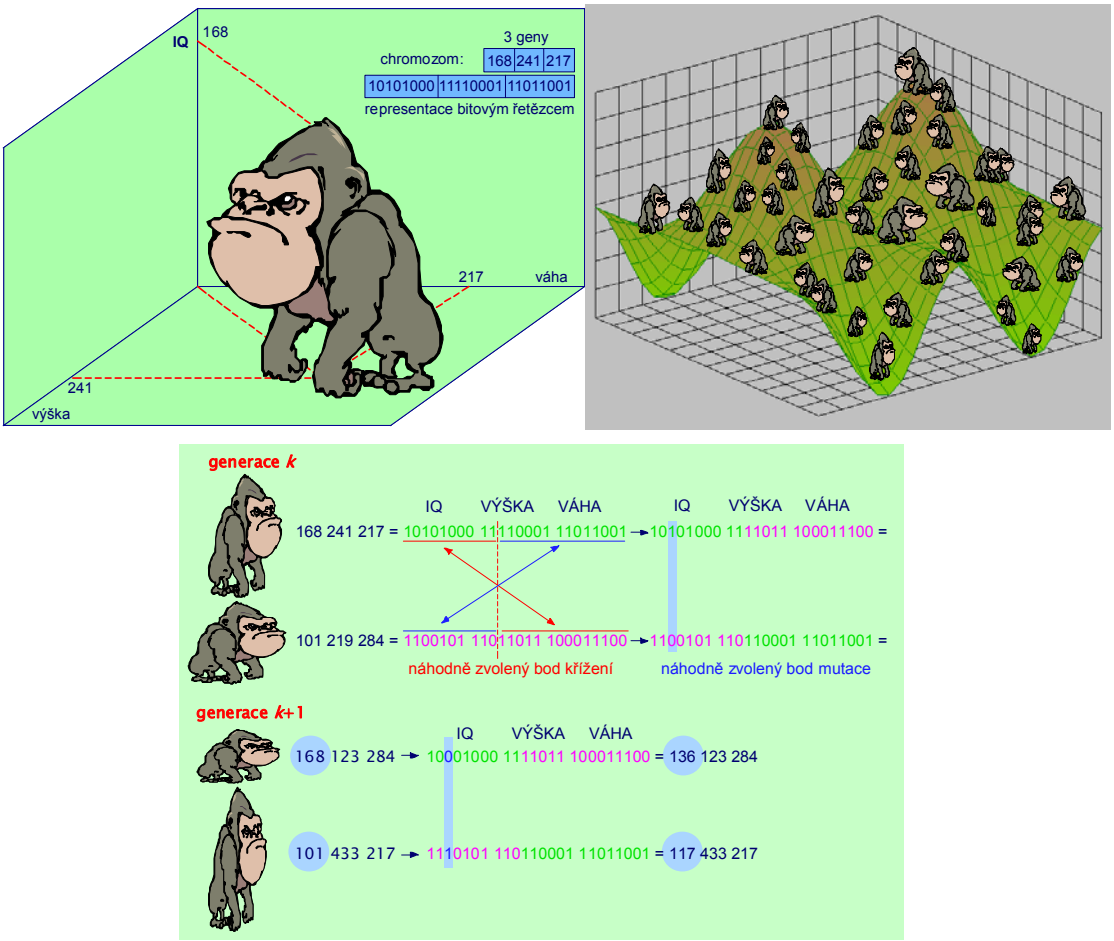
vygeneruje dostatečné množství různých řešení (velikost populace musí být dostatečně velká, jinak dojde k degeneraci). Některá řešení jsou lepší, jiná horší – kvalita jedinců v populaci se stanoví prostřednictvím **funkce přizpůsobenosti** (*fitness function*). „Lepší“ jedinci dostanou vyšší pravděpodobnost pro to, aby byli náhodně vybráni ke *křížení,* a tím k vytvoření nových jedinců pro následující populaci. „Horší“ jedinci tu šanci mají menší, úměrnou své kvalitě, ale neměli by být zcela vyřazeni, protože některou vlastnost mohou mít vynikající, i když zbytek vlastností jim to kazí. Základní princip GA lze popsat následovně:

Každý jedinec je reprezentován prostřednictvím *chromosomu*, který se skládá z genů. *Geny* představují proměnné, jejichž hodnoty v kombinaci určují *kvalitu jedince*. Cílem je dosáhnout křížením optimálního jedince, který odpovídá optimálnímu řešení.
Objasníme to na příkladu mnohorozměrné hornaté krajiny, kde na nejvyšší horu (globální maximum) vyleze nejlepší jedinec, zatímco horší jedinci se rozptýlí v krajině a obsadí nižší hory (lokální extrémy). Na počátku se krajina zabydlí populací jedinců generovanou náhodně, neboli souřadnice polohy jedinců jsou vytvořeny tak, aby rovnoměrně pokryly celou oblast. Někteří jedinci jsou náhodou umístěni blíže optimu, jiní jsou od něho vzdáleni.
Algoritmus řešení spočívá v tom, že pro každé dva jedince se vezmou jejich souřadnice a vyberou se ty souřadnice, které umístí jedince blíže k optimu. Tak se vytvoří souřadnice kvalitnějšího potomka, který se v další generaci posune blíže ke globálnímu extrému, který se hledá a jehož blízkost lze odhadovat z kvality jedinců.

Takto vygenerované souřadnice lze mezi sebou jenom kombinovat, proto je vhodné zavést alespoň ještě jednu další operaci, rovněž inspirovanou přirozeným vývojem druhů společenstev: *mutaci*.
Operace mutace umožňuje náhodně občas změnit existující hodnoty jedince, čímž zabraňuje předčasné jeho degeneraci, která se může projevit např. uvíznutím v lokálním extrému (nižší hoře), protože nikdo nemá geny obsahující hodnoty umožňující posun jedince alespoň podél jedné osy pryč z lokálního extrému. Uvážlivě používaná operace mutace náhodně vybere jedince, v něm náhodně některý gen a ten náhodně změní o nepříliš velkou hodnotu. Nevhodná mutace, která by příliš měnila hodnoty (viz výše souřadnice v krajině), by totiž řešení problému směrovala do spíše náhodného prohledávání krajiny, přičemž cílem je získávat elitní jedince, kteří se ubírají správným směrem k nejvyšší hoře.

Křížení jedinců (výběr dvojic jedinců v algoritmu) je vhodné provádět tak, aby byla zachována co největší různorodost jedinců, protože platí, že je dobré být přizpůsobený, ale také je dobré být také odlišný. Jedinci jsou proto posuzováni dvojrozměrně: jednak podle kvality (míry přizpůsobenosti) a jednak podle odlišnosti. Dobrý jedinec má vysokou kvalitu a odlišuje se od průměru.

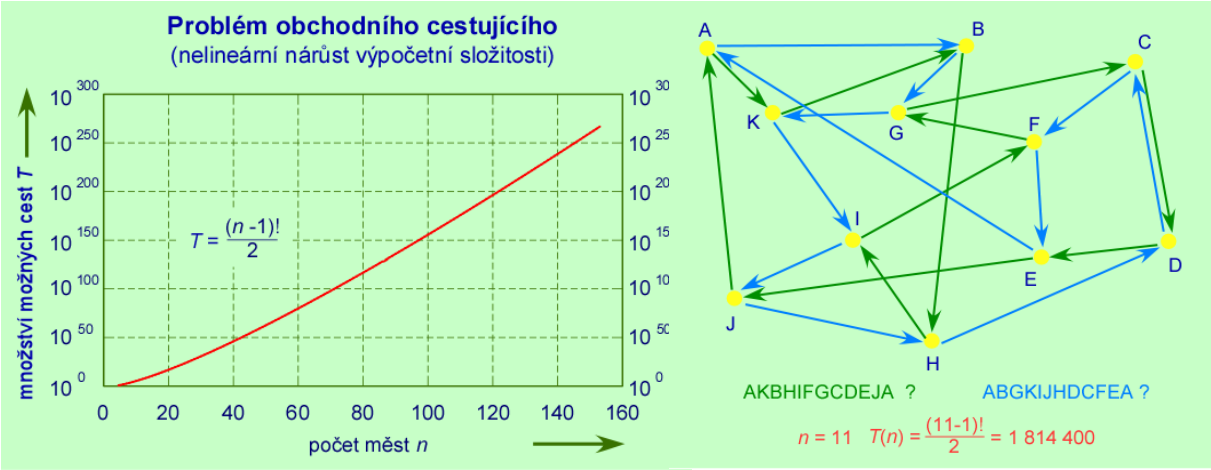
Populace jedinců se udržuje v konstantním množství v každé generaci a „putuje“ krajinou, dokud dochází ke zlepšování některých jedinců nebo dokud není dosaženo limitu v počtu generací. Proces je výpočetně náročný, často je nutno hledat kompromis mezi velikostí populace a přijatelnou dobou výpočtu. GA patří mezi obecné optimalizační techniky a dosáhly řady úspěchů v řešení úloh nezvládnutelných analyticky nebo jinými metodami. Velkou režii u GA představuje nutnost transformace jedinců z reálného světa do abstraktního prostoru vyhledávání a zpět, protože kvalita se určuje dle parametrů jedince v reálném světě – jeho vlastnosti jsou v GA světě kódovány do bitových řetězců případně jiným způsobem.



Obrázek č. 7.11: Transformace a mutace jedinců

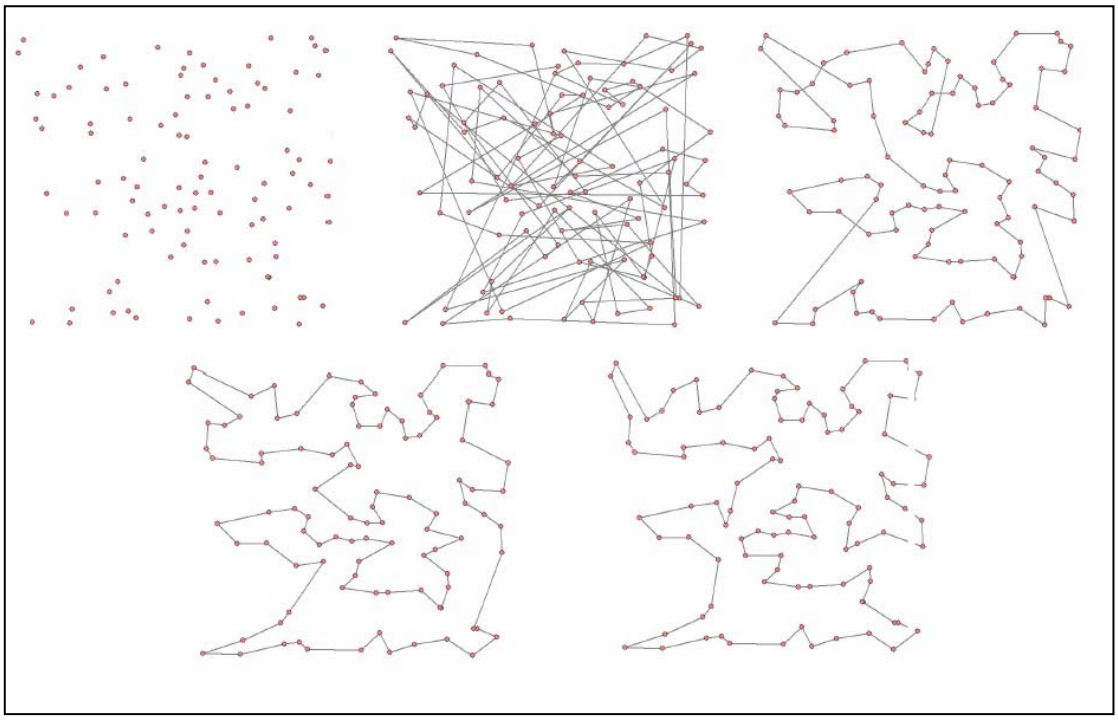
Transformace jedinců, zabydlení vyhledávacího prostoru a křížení s mutací ilustrují tři obrázky na obr. č. 7. 11, kde jsou jedinci pro jednoduchost popsáni jen třemi atributy: *IQ, váha, výška.* Ideálem je zde docílit optimálního jedince, který má vyváženy tělesné rozměry s intelektem vzhledem k tomu, co má dělat za činnost – to se zjišťuje jeho transformací do reálného prostoru a otestováním, které dá k dispozici skóre určující kvalitu.

Na obr. č. 7.12 je ilustrována aplikace GA na jedné typické úloze: **problém obchodního cestujícího** *(traveling salesman problem, TSP*). Obecně jde o nalezení optimální (nejkratší) cesty, která postupně spojí *n* měst tak, že do každého lze vstoupit jen jednou (graf bez smyček); začít lze z libovolného. Přesné analytické řešení není kvůli vysoké výpočetní složitosti, dané velkým počtem možných kombinací, reálně uskutečnitelné. Pokud by byla k dispozici všechna možná řešení, bylo by snadné vybrat optimální, ale takový přístup nelze prakticky uskutečnit. Složitost roste s *n* silně nelineárně: Na pravém obrázku obr. č. 7.12 jsou uvedeny dva možné chromosomy (AKBHIFGCDEJA a ABGKIJHDCFEA) neboli dvě z možných řešení, kterých je pro 11 měst celkem 1 814 400.



Obrázek č. 7.12: Problém obchodního cestujícího

Další pětice obrázků na obr. č. 7.13 ukazuje složitější případ pro 100 měst, řešený pomocí GA. Velikost populace byla 200. Lze sledovat počáteční, náhodně vygenerovanou situaci a dále řešení dosažená po 2, 100, 500 a 1000 generacích (po 2000 byl algoritmus zastaven, protože již od 833. generace nedocházelo k dalšímu zlepšování):



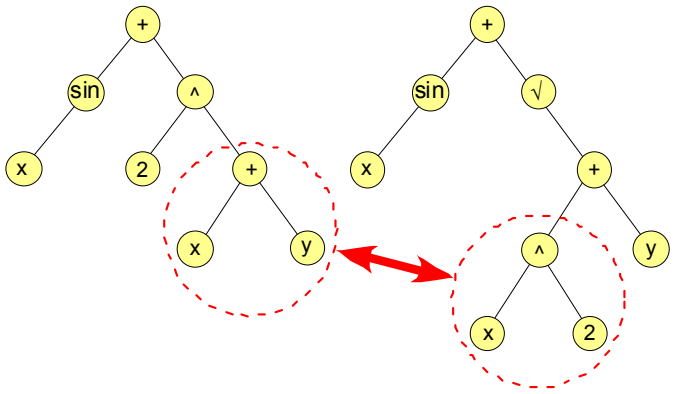
Obrázek č. 7.13: Problém obchodního cestujícího pro 100 měst

Počáteční generace č. 0 dala nejlepší náhodné propojení délky 44 824. Druhá generace zlepšila řešení na 36 805 (horní obrázek uprostřed), přičemž průměr všech 200 řešení byl 45 449. Stá generace již dávala nejlepší řešení 9 206, průměr byl 10 370 (přeživší jedinci již byli všichni poměrně kvalitní). Po pětisté generaci to bylo 8 547 (průměr 9 228), a po 1000 generacích 8 340 (průměr 10 197), viz poslední spodní obrázek vpravo. Nelze říci, zda nakonec došlo k uvíznutí v lokálním extrému nebo k dosažení optima, protože optimum není známo a nelze ho v přijatelné době zjistit. Přesto říci lze, že bylo dosaženo dobré řešení, kdy

Darwinovským vývojem došlo ke snížení délky trasy v poměru cca 1 : 5.37. Typické pro tento typ vývoje je, že na začátku dochází k prudkému poklesu délky trasy, která se pak dále zlepšuje již relativně pomalu a nakonec osciluje kolem nějaké dosažené hodnoty. Je vhodné pro tatáž data zkusit více náhodně vytvořených počátečních generací, resp. více různých kombinací hodnot parametrů GA a výsledky porovnat, jinak je obtížné usoudit, zda dosažené řešení ještě lze či nelze vylepšit.

Hledání optimální cesty *TSP* lze aplikovat na mnoho nejrůznějších problémů, např. na hledání minimálního počtu co nejlevnějších vyšetření v pokud možno nejkratším čase, na hledání co nejlepšího přenosu ve složité počítačové síti, apod. Existuje mnoho literatury popisující různé metody a přístupy k řešení, kromě [Goldberg, 1989] také [Holland, 1992] nebo [Fogel, 2006].

Hledání optima vývojem vedlo také k jednomu zajímavému směru, zvanému *genetické programování, GP*, viz např. [Koza, 1992]. Tam se problém převede na reprezentaci pomocí mnoha náhodně vygenerovaných stromů (funkcí), a křížení se provádí výměnami náhodně vybraných podstromů. Lze takto hledat neznámé funkce složené nějakou kombinací z funkcí jednodušších, např. pro nelineární regresi, kde jsou k dispozici naměřená mnohorozměrná data a cílem je těmito body optimálně proložit analyticky neznámou funkci. Je nutno definovat, které operátory, konstanty a jednodušší funkce mohou být použity a pak se pokusit vývojem nalézt vhodné řešení. Rovněž toto odvětví zaznamenává v praxi velké množství úspěšných aplikací. Křížení pro genetické programování ilustruje obr. č. 7.14, kde jsou kříženy dva chromosomy reprezentované stromovou strukturou:



Obrázek č. 7.14: Křížení dvou chromosomů reprezentované stromovou strukturou

Levý strom vyjadřuje funkci $\sin(x) + 2^{(x+y)}$ a pravý strom funkci $\sin(x) + \sqrt{(x^2+y}$. Přehozením zakroužkovaných podstromů vzniknou dva noví jedinci pro následující generaci, kteří vyjadřují nové funkce $\sin(x) + 2^{x^2}$ a $\sin(x) + \sqrt{x+2y}$. Přizpůsobenost jedinců se otestuje na numerických datech, zda např. nebylo dosaženo lepšího koeficientu regrese, nižší chyby dané součtem čtverců odchylek, aj. Mutaci lze provádět změnami hodnot konstant, operátorů i funkcí, např. z 2.0 na 1.98, ze $\sin(x)$ na $0.71 \cdot \sin(x)$, $\log_2(x)$ na $\log_{10}(x)$, apod. Mutace by neměly ani zde být příliš velké, ale jejich stanovení je jako parametr dáno uživatelem, takže je možné např. mutací náhodně vybrat uzel nějakého stromu a funkci $\cos(x^2)$ v daném uzlu nahradit např. $\ln(x)$, nebo přičtení odečtením, atd. Funkce (stromy) se špatnou přizpůsobeností získají nízkou pravděpodobnost být vybrány ke křížení a snadno mohou zaniknout – vývoj probíhá obdobně jako u GA.

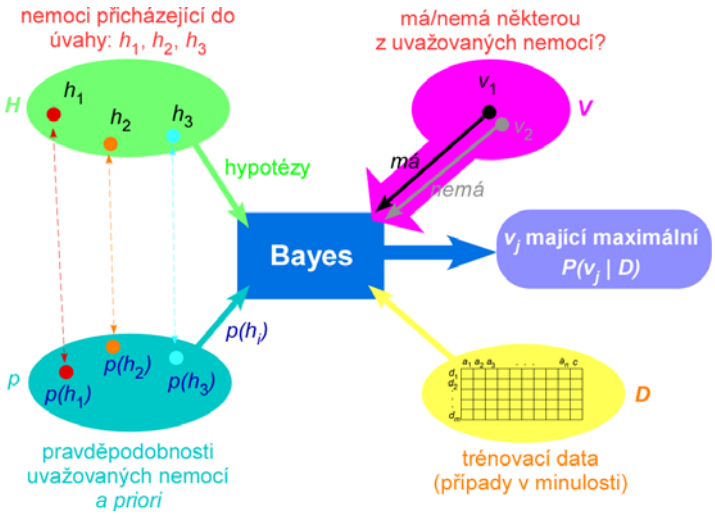
7.5.5 Bayesovské učení

Bayesova metoda inference je založena na použití teorie pravděpodobnosti. Strojové učení poskytuje kvantitativní přístup ke zvažování důkazů podporujících alternativní hypotézy. Základem je Bayesův teorém:

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)},$$

kde $P(h)$ je apriorní pravděpodobnost platnosti hypotézy h před tím, než byla získána trénovací data D, $P(D)$ je apriorní pravděpodobnost pozorování dat D bez jakéhokoliv vztahu k nějaké hypotéze h, $P(D|h)$ pravděpodobnost zpozorování dat D ve světě, kde platí hypotéza h, a $P(h|D)$ je aposteriorní pravděpodobnost hypotézy h za předpokladu pozorování dat D. Důležité je, že $P(h)$ platí bez ohledu na jakákoliv data, která jsou k dispozici, zatímco $P(h|D)$ je ovlivněno daty, která se podařilo pozorovat. $P(h|D)$ klesá s rostoucí $P(D)$, protože roste-li pravděpodobnost pozorování dat D bez vztahu k h, pak D poskytují stále menší podporu hypotéze h (zvyšuje se vzájemná nezávislost). Logicky také lze očekávat, že $P(h|D)$ poroste jak s $P(h)$, tak s $P(D|h)$.

Postavení Bayesovy metody ilustruje obr. č. 7.15, kde vstupem je svět H uvažovaných hypotéz h_i (např. o kterých chorobách má lékař uvažovat při stanovení diagnózy na základě údajů od pacienta (a případně laboratorních vyšetření)), apriorní nepodmíněné pravděpodobnosti $p(h_i)$ jednotlivých h_i , získané statisticky z minulosti (např. jaké procento populace průměrně onemocní danou chorobou), nepodmíněná pravděpodobností $P(D)$, dále *aposteriorní* podmíněné pravděpodobnosti $P(D|h_i)$ dávající do vztahu jednotlivé hypotézy a pozorovaná data, a svět V možných klasifikací v_j (např. pacient má či nemá některou z uvažovaných nemocí). Bayesův vzorec stanoví pro každou hypotézu její pravděpodobnost $P(h_i|D)$ za předpokladu dat D. Žádná hypotéza není ani zcela potvrzena, ani zcela zamítnuta. Pravděpodobnosti hypotéz se mění s tím, jak jsou postupně získávána reálná data, která hypotézy více či méně podporují aposteriorními pravděpodobnostmi.



Obrázek č. 7.15: Postavení Bayesovy metody

Základním problémem při dolování z dat je stanovit, jaká je nejpravděpodobnější klasifikace dané instance, jsou-li dána určitá trénovací data. Není to však totéž jako určení nejvyšší pravděpodobnosti pro jednu z hypotéz: Nechť existují tři hypotézy s pravděpodobnostmi (vzhledem k tréninkovým datům D) $P(h_1|D) = 0.4$, $P(h_2|D) = 0.3$, a $P(h_3|D) = 0.3$. Nejvíce je tedy podporovaná hypotéza h_1. Objeví-li se v budoucnu instance klasifikovaná pozitivně od h_1

a negativně od h_2 a h_3, pak podpora být pozitivní je 0.4, ale být negativní je 0.6, což převažuje od h_2 a h_3. Může-li klasifikace nabýt libovolné hodnoty $v_j \in V$, pak pravděpodobnost korektní klasifikace v_j pro data D je dána vztahem:

$$P(v_j \mid D) = \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D).$$

Optimální Bayesova klasifikace nové instance je pak taková hodnota v_j, pro niž má $P(v_j|D)$ maximální hodnotu:

$$v_j = \arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D).$$

Výsledek klasifikace tedy nebude pozitivní (i když h_1 má podporu větší než jsou podpory ostatním individuálním hypotézám h_2 nebo h_3), nýbrž negativní ($h_2 + h_3$ tedy převáží nad h_1): 0.3+0.3 > 0.4.

Neexistuje žádná jiná klasifikační metoda, která bude (za předpokladu použití téhož prostoru hypotéz a téže apriorní znalosti) dávat lepší výsledky (mohou být stejně dobré). Tato metoda maximalizuje pravděpodobnost, že nové instance budou klasifikovány korektně za předpokladu, že jsou k dispozici určitá data pro natrénování (tj. stanovení aposteriorních pravděpodobností), prostor hypotéz a apriorní pravděpodobnosti nad těmito hypotézami.

Naivní Bayesův klasifikátor (NBK) je jednou z vysoce praktických metod strojového učení. Vychází z popsaného optimálního Bayesova klasifikátoru (OBK) a umožňuje *snížit výpočetní složitost* za předpokladu teoreticky ne zcela korektního zjednodušení – cenou je možné snížení přesnosti, ale pragmatický přínos je velmi výrazný pro úlohy popsané mnoha atributy (desítky a mnohem více). NBK může dosáhnout i stejné přesnosti jako OBK, nebo se výsledkům OBK dostatečně přiblížit, jak ukazují výsledky tisíců aplikací v reálném světě, protože praktická realita většinou do značné míry vyhovuje teoretickým požadavkům. Samozřejmě, že výsledky NBK mohou být špatné, pokud je odchylka od teoretického předpokladu velká; pak nelze NBK použít.

NBK je použitelný, když lze každou instanci popsat jako *konjunkci hodnot atributů* a kde cílová funkce může nabýt libovolné hodnoty z nějaké konečné množiny V. NBK je založen na zjednodušujícím předpokladu, že hodnoty atributů a_i jsou navzájem *podmíněně nezávislé* za předpokladu dané cílové hodnoty v_j. Jinými slovy: Pravděpodobnost NBK pro výskyt určité konjunkce hodnot atributů je dána pouze součinem pravděpodobností výskytu hodnot individuálních atributů:

$$P(a_1, a_2, ..., a_n \mid v_j) = \prod_i P(a_i \mid v_j),$$

což není (obecně a teoreticky) korektní předpoklad. Použije-li se uvedené zjednodušení, vznikne následující vztah pro klasifikaci naivním Bayesovým klasifikátorem:

$$v_{NBK} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$$

V NBK je počet různých termů $P(a_i \mid v_j)$, které je nutno odhadnout pomocí trénovacích dat, roven právě počtu různých hodnot atributů násobeno počtem různých cílových hodnot (výpočetní složitost daná kombinací hodnot). To je *výrazně menší* počet, než kdyby se musely odhadovat pravděpodobnosti $P(a_1, a_2, ..., a_n \mid v_j)$ bez uvažovaného zjednodušení za předpokladu nezávislosti – to však bývá v praxi velmi často alespoň pro část atributů splněno, takže narušení teoretických předpokladů *většinou* příliš výsledky nenarušuje (ale někdy samozřejmě může).

Příkladem aplikace naivního Bayesova klasifikátoru je zařazování nestrukturovaných textů do příslušných kategorií. Typické je např. filtrování zpráv v elektronické poště, kde uživatel může jednoduše označit nechtěnou zprávu jako tzv. *spam* (tj. stanoví třídu *negativní*; implicitně platí třída *pozitivní*). Slova ve zprávě jsou použita jako konjunkce hodnot charakterizujících konkrétní klasifikační třídu – vzájemné pořadí slov se nebere v NBK do úvahy.

7.5.6 Roje a mravenci

Velmi moderní algoritmy nazývané **optimalizace rojem částic** (*Particle Swarm Optimization, PSO*) patří k metodám umělé inteligence pro prohledávání prostoru. Jsou příbuzné s genetickými algoritmy. Algoritmy PSO vycházejí z existence určité populace částic, které tvoří roj směřující ke globálnímu extrému. Inspirace pochází například od hejna ryb, které se drží pohromadě a směřují k nalezení co největšího množství co nejlepší potravy. Roj či hejno se pohybuje určitým nenáhodným směrem, který je částečně narušován nepředvídatelnými odbočováními do stran. Směr pohybu je v každém časovém kroku dán v principu dvěma složkami:

- momentálním globálním optimem a
- optimem nalezeným momentálně tou nejlepší z částic.

Podobnost s genetickými algoritmy (GA) spočívá ve vytvoření počáteční populace náhodných řešení, optimum se hledá aktualizací generace. Na rozdíl od GA však PSO nepoužívá evoluční operátory typu křížení, mutace, apod. Částice, představující potenciální řešení, se pohybují prohledávaným prostorem vždy směrem udávaným momentálně nejlepší částicí z roje (rybou z hejna). PSO je jednoduché na implementaci, nemá mnoho parametrů, je ovšem náročné na výkon paměti i procesoru. PSO lze používat tam, kde GA, a naopak, i když samozřejmě výsledky nemusejí být totožné a zaručené. V principu jde o stochastickou optimalizační techniku, vycházející z odvětví **umělý život** *(artificial life*). Tyto techniky byly původně vyvíjeny pro simulaci a studium živých organismů a jevů s nimi spojených, ale obecně je lze aplikovat na nejrůznější výpočetní problémy. PSO je zaměřeno na "společenské" biologické systémy, kde se zkoumá chování jedinců tvořících součásti kolektivu, jejich vzájemné interakce a interakce s okolím (existuje název *swarm intelligence*, tj. inteligence roje/hejna/davu). Používá se například ke zkoumání nepředvídatelné dynamiky společenského chování různých živých organismů (ptáci, ryby, mravenci, atd.). Nyní existují dvě hlavní skupiny: optimalizace mravenčí kolonie (souboru mravenců) ACO (*ant colony optimization*) a PSO (*particle swarm optimization*). Používají se pro diskrétní optimalizační problémy.

Princip PSO inspirovaný hejnem ptáků

Skupina ptáků náhodně hledá potravu v určité oblasti, v níž je pouze jeden kus potravy. O potravě vědí jen někteří ptáci, ne všichni. Jako nejlepší strategie se používá následování ptáka, jemuž je ta potrava nejblíže. Každá částice v roji představuje jednoho ptáka v prohledávaném prostoru a má určitou hodnotu své kvality, kterou stanovuje **funkce přizpůsobenosti** (*fitness function*). Kromě toho má každá částice nějakou rychlost, s níž se pohybuje a která udává i směr pohybu. Částice se pohybují prostorem tak, že následují tu, která je momentálně nejlepší. Hledání optima se provádí iterativně. V každém iteračním kroku je každá částice aktualizována pomocí dvou "optimálních" hodnot:

- doposud nalezeným nejlepším řešením každé částice p_{best} (*particle best*), a
- doposud nejlepším řešením populace g_{best} (*global best*).

Po nalezení obou nejlepších hodnot p_{best} a g_{best} jsou částice upravovány z hlediska jejich rychlosti a polohy pomocí dvou jednoduchých vztahů:

- $\mathbf{v} \leftarrow \mathbf{v} + c_1 \cdot r \cdot (p_{best} - p_{present}) + c_2 \cdot r \cdot (g_{best} - p_{present})$, a
- $p_{present} \leftarrow p_{present} + \mathbf{v}$,

kde **v** je rychlost částice, $p_{present}$ je její současná kvalita, a $0.0 \leq r \leq 1.0$ je náhodné číslo. Konstanty c_1 a c_2 jsou tzv. akcelerační konstanty, ovlivňující míru aktualizace částic, např. 0.5, 1.0, nebo 2.0. Algoritmus spočítá pro každou částici hodnotu jejího přizpůsobení *f*, a pro $f > p_{best}$ ji použije jako novou p_{best}. Nejlépe přizpůsobená částice určí g_{best}, spočítá se její rychlost **v** a aktualizuje se její pozice $p_{present}$. Celý postup se pak opakuje tak dlouho, dokud není vyčerpán počet kroků optimalizace nebo dosaženo zadaného minima chyby. Pro aplikaci PSO je nutno vhodně reprezentovat řešený problém a navrhnout funkci přizpůsobenosti. PSO pracuje s reálnými čísly a má tyto parametry:

- počet částic – řádově desítky, obvykle 20-40 (málo částic snižuje výpočetní složitost, ale zároveň i možnost nalézt optimum; hodně částic vede k opačným výhodám a nevýhodám – někdy se používají stovky částic);
- počet dimenzí částic – je dán optimalizovaným problémem;
- rozsah hodnot částic – je dán optimalizovaným problémem;
- v_{max} – ovlivňuje maximální možnou změnu rychlosti částice při jedné iteraci;
- konstanty učení c_1 a c_2 – obvykle mívají hodnotu 2.0, experimenty používají nejčastěji rozsah hodnot z heuristického intervalu [0, 4];
- podmínka ukončení – minimální předem zadaná chyba požadovaná uživatelem (funkce přizpůsobenosti se použije pro výpočet chyby v každé iteraci), nebo stanovení max. počtu iteračních kroků (stovky, tisíce).

Princip ACO - optimalizace kolonií mravenců

Optimalizace kolonií mravenců (*ant colony optimization, ACO*) vznikla studiem společenství mravenců, kteří tvoří velmi jednoduché složky, avšak jako celek se společenství chová velmi inteligentně z hlediska schopnosti úspěšně přežít [Dorigo, Stützle, 2004]. Společenství představuje vysoce strukturovanou organizaci, která umožňuje dosáhnout složitých výsledků navzdory primitivnosti svých členů. Mravenci jsou inspirací pro racionální agenty, kteří nemusí být individuálně složití, ale musí být schopni mezi sebou nějak komunikovat. Potom třeba agent (malý jednoduchý robot), který má přemístit nějaký objekt, začne volat další agenty, když zjistí, že ač tlačí na daný objekt sebevíc, nepohne s ním. Agenti, nacházející se poblíž, se začnou přisunovat a přidají se k tlačení objektu. Pokud s ním stále nemohou pohnout, volají další agenty, až nakonec úkol splní.

Metoda ACO se dá aplikovat na velmi obtížné problémy, např. problém obchodního cestujícího (*TSP, traveling salesman problem*), který má složitost NP–úplnou. Obdobně ACO pomáhá strojovému učení při generování klasifikačních pravidel, hledání optimálních shluků dat, regresi, aj. Mravenci mohou např. začít s prázdným pravidlem, do jehož části s předpoklady (*antecedent*) iterativně přidávají testy, a postupně přidávají i další pravidla. Pravidla jsou testována vzhledem ke své schopnosti zařazovat trénovací příklady správně do zadaných tříd. Činnost mravenců pokračuje tak dlouho, dokud např. není dosaženo přijatelně malé klasifikační chyby nebo dokud neskončí zlepšování se výsledků klasifikace. Přizpůsobenost pravidla (*fitness*) je dána počtem jeho chyb na testovacích datech, a pokud přidání nového testu sníží chybovost, nahradí nové pravidlo to původní. Agenti – mravenci byly použiti úspěšně např. při vytváření fuzzy pravidel, kde na obtížných testovacích problémech (*benchmarks*) prokázali lepší výsledky, dosažené navíc rychleji, při srovnání s alternativními postupy.

Jedním z důvodů, proč se používají mravenčí kolonie, je skutečnost, že řada problémů má distribuovaný charakter, takže jeden – byť chytrý – mravenec na řešení nestačí. Reálnou úlohou je např. hledání cesty sítí, což je typické třeba pro telekomunikace (optimální spojení) nebo WWW. Počet mravenců je nutno hledat experimentálně, ale dosavadní zkušenosti ukazují, že ASO je dostatečně robustní vzhledem k tomuto parametru.

ACO byla použita pro řešení TSP na základě znalosti, že není nutno hledat optimální cestu v celém složitém grafu, ale že dobré a rychlé výsledky dává prohledávání subgrafů, kdy se bere do úvahy pouze okolí se sousedními městy. Tak se vytvoří seznam kandidátů pro propojení s východiskovým městem a mravenec pak z každého města zkoumá kandidáty, pokud ještě existují; když ne, musí subgraf rozšířit.

ACO prokázala velmi dobré schopnosti při řešení dynamických problémů (kdy se během řešení mění hodnoty rozhodovacích parametrů, cílových funkcí, omezení) a stochastických problémů (známy jsou jen pravděpodobnosti hodnot cílových funkcí), což jsou problémy typické pro současnost např. v oblasti počítačových sítí, Internetu, apod. Jednoduchost ACO algoritmů je kompenzována nevýhodou velkých výpočetních nároků, takže pro zmiňovanou metodu se velmi hodí možnost využívat paralelismus.

Kapitola 8
Literatura

[1] Cvrčková F. (2006) Úvod do praktické bioinformatiky, Academia.
[2] de Vries G., Hillen T., Lewis M., Miller J., Schönfisch B. (2006) A Course in Mathematical Biology: Quantitative Modeling with Mathematical & Computational Methods, Softcover.
[3] Dorigo M., Stützle T. (2004) Ant colony optimization. The MIT Press.
[4] Duda R. O, Hart P. E., Stork D. G. (2001) Pattern classification. Second edition. John Wiley and Sons, Inc.
[5] Fogel D. B. (2006) Evolutionary computation: Toward a new philosophy of machine intelligence. IEEE Press, NJ. Wiley-Interscience, John Wiley and Sons, Inc.
[6] Fowkes N.D., Mahony J.J. (1994) An introduction to mathematical modelling. John Wiley and Sons, Inc.
[7] Gallant S. I. (1994) Neural network learning and expert systems. The MIT Press.
[8] Gander W., Hřebíček J. (2005) Solving Scientific Problems Using Maple and MATLAB. 4th exp. and rev. ed. Springer.
[9] Goldberg D. E. (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley.
[10] Golub G., Ortega J. M. (1993) Scientific Computing: An Introduction with Parallel Computing. Academic Press.
[11] Gonnet, G., Baeza-Yates, R. (2007) Handbook of Algorithms and Data Structures. http://www.dcc.uchile.cl/~rbaeza/handbook/
[12] Gupta S., Anderson R. M., and May R.M. (1993) Mathematical Models and the design of public health policy: HIV and Antiviral therapy. SIAM Reviw, Vol 35, No. 3, pp. 1-16.
[13] Hassoun M. H. (1995) Fundamentals of artificial neural networks. The MIT Press.
[14] Heath M. T. (2002) Scientific Computing: An Introductory Survey. Second edition. McGraw-Hill.
[15] Holland J. H. (1992) Adaptation in natural and artificial systems. Second edition, Cambridge, MA. MIT Press.
[16] Horová I., Zelinka J. (2004) Numerické metody. 2. vyd. MU Brno.
[17] Hřebíček J., Škrdla M. (2006) Úvod do matematického modelování. MU Brno.
[18] Hřebíček J., Žák V. (2007) Nové možnosti systému Maple 11 ve výuce. In Sborník čtvrtého ročníku konference o e-learningu --- SCO 2007. Brno : vydavatelství Masarykovy university, s. 105-110.
[19] Hutchinson A. (1994) Algorithmic learning. Oxford University Press.
[20] ICS-UCI (2007) <http://www.ics.uci.edu/~mlearn/databases/>.
[21] Koza J. R. (1992) Genetic programming. Cambridge, MA. MIT Press.
[22] Kruse R., Gebhardt J., Klawonn F. (1994) Foundations of fuzzy systems. John Wiley and Sons, Inc.
[23] Lacko, L. (2003) Databáze : datové sklady, OLAP a dolování dat s příklady v Microsoft SQL Serveru a Oracle. Computer Press.
[24] Langley, P. (1996) Elements of machine learning. Morgan Kaufmann Publishers.
[25] Liu J.S. (2002) Monte Carlo Strategies in Scientific Computing. Second edition. Springer
[26] Marťán P. (2002) NetVizualizér: Diplomová práce na FI MU, <http://neuralnets.czweb.org>.
[27] Matyska L. (2006) Enabling Grids for E-sciencE - The EU EGEE Project. Sborník Znalosti 2006, FEI VŠB-Technická univerzita Ostrava, str. 233-236.

[28] Mitchell T. M. (1997) Machine learning. McGraw-Hill.
[29] Quinlan J. R. (1993) C4.5: Programs for machine learning. Morgan Kaufmann Publishers.
[30] Peter H. (2008) Python Scripting for Computational Science. Third edition. Springer.
[31] Press W. H., Teukolsky S. A., Vetterling W. T. (2007) Numerical Recipes: The Art of Scientific Computing. Third edition. Cambridge University Press.
[32] RapidMiner (2007) <http://www.rapidminer.com>.
[33] Ressler M. a kol (2006): Informační věda a knihovnictví: Výkladový slovník české terminologie z oblasti informační vědy a knihovnictví. Výběr z hesel v databázi TDKIV. Vydavatelství VŠCHT Praha.
[34] Russel S., Norvig P. (2003) Artificial intelligence: A modern approach. Second edition. Prentice Hall.
[35] Shapiro S. C. a kol. (1991) Encyclopedia of artificial intelligence. Second edition. John Wiley and Sons, Inc.
[36] Sklenák V. a kol. (2001) Data, informace, znalosti a internet. C. H. Beck.
[37] Sonnenburg S., Braun M. L., Ong Ch. S., Bengio S., Bottou L., Holmes G., LeCun Y., Müller K.-R., Pereira F., Rasmussen C. E., Rätsch G., Schölkopf B., Smola A., Vincent P., Weston J., Williamson R. C. (2007) The need for open source software in machine learning. Journal of machine learning research (JMLR), 8 (2007), pp. 2443-2466, <http://jmlr.csail.mit.edu/papers/>.
[38] Strang G. (2007) Computational Science and Engineering. Wellesley-Cambridge Press. <http://www-math.mit.edu/cse/>
[39] Turing A. M. (1950) Computing machinery and intelligence. Mind, 59, pp. 433-460. Do-stupné na <http://cogprints.org/499/0/turing.html>.
[40] WEKA (2007) <http://www.cs.waikato.ac.nz/ml/weka>.
[41] Wikipedia (2007) <http://cs.wikipedia.org/wiki/Hlavn%C3%AD_strana>
[42] Witten I. H., Frank E. (2005) Data mining: Machine learning tools and techniques. Second edition. Elsevier/Morgan Kaufmann Publishers.
[43] Wolfram S. (2002) A New Kind of Science. Wolfram Media. <http://www.wolframscience.com/>
[44] Wolpert D. H., Macready W. G. (1995) No free lunch theorems for search. Technical report SFI-TR-05-010. <ftp://ftp.santafe.edu/pub/dhwftp/nfl.search.TR.ps.Z>, Santa Fe Institute.
[45] Zadeh L. A. (1965) Fuzzy sets. Information and control, Vol. 8, No. 3, June 1965, 338-353.