**Kikiina**

**Státnicová**

**Směska**

**Ni-zi**

**1. Automatické plánování, plánovací graf, kompilace plánování do jiných formalismů jako je SAT nebo CSP, hierarchické plánování, plánování v prostoru plánů. Plánování pohybu a problém lokalizace v robotice.**

NI-UMI

**Automatické plánování**

* Úloha hledání **posloupnosti akcí**, které vedou k dosažení určitého cíle
* **Klasické plánování** – plně pozorovatelné, deterministické, konečné, statické, diskrétní, offline (nejdříve naplánujeme, pak vykonáme)
* **Neklasické plánování** – stochastické, dynamické prostředí
* **Vyjadřovací prostředky**
  + **Atom** – predikátový symbol s dosazenými konstantami
  + **Jazyk** – predikátové symboly a konstanty (nulární funkce)
  + **Stavy** – konečné množiny základních atomů, uzavřený svět (co není zmíněno, to neplatí)
  + **Cíl** – částečná specifikace stavu konečnou množinou literálů
* **Operátory**
  + Trojice o(name(o), precond(o), effects(o))
    - **Name(o)** – n(x1…xn)
    - **Precond(o)** – literály, které musí být splněny
    - **Effect(o)** – množina literálů, které budou platit po provedení operátoru
* STRIPS – dnes v podobě **PDDL**
* **Akce** – základní instance operátoru
  + **Aplikovatelnost** akce:
    - Mějme akci a a stav s

*Precond+(a), precond-(a)*

*Effect+(a), effect-(a)*

* + - Akce je aplikovatelná ve stavu s:
  + **Aplikace** apl. Akce:
    - Odebereme negativní efekty, přidáme pozitivní efekty:
* **Plánovací doména** – jazyk (konstanty a predik. S.) L a operátory O
  + Implicitně určuje množinu stavů a funkci následníka – γ: SxA -> S
* **Plánovací problém** – předp. plánovací doménu: L, O
  + Zadání plánovacího problému: (O, S0, g)
  + Plánovací problém SP s přechody: (Σ, s0, sg)
    - = množina cílových stavů
* **Plány a řešení**
  + **Plán** **– posloupnost akcí** π = [a1, …, an], kde ai = instance operátoru z O
  + Plán je řešením P, jestliže je proveditelný z s0 a dosahuje cíle g
* **Dopředné plánování** – nedeterministický výběr akce do navštíveného stavu, tím se vytvoří nový stav
* **Zpětné plánování**
  + **Inverzní aplikace** – inverzní funkce k funkci následníka, tvořím pod-cíl, ze kterého se dostat do cíle
    - Lze uvažovat jen relevantní akce– ty, které nás posunou směrem ke splnění cíle (splní alespoň 1 cílový literál), a zároveň nezruší žádný ze splněných literálů cíle
    - Cíl g a akce a – γ-1(g, a)
    - Akce a je relevantní pro cíl g, když:
      * + g ∩ effects+(a) ≠ ∅
        + g+ ∩ effects-(a) = ∅ a g- ∩ effects+(a) = ∅
    - Pro relevantní a vzhledem k g definujeme: γ-1(g, a) = (g – effect(a)) ∪ precond(a)
    - Jako u DP se vybírá nedeterministický nový stav z předchozího, ale tentokrát se jde od cíle k počátečnímu stavu
* **Liftování** – zavedení proměnných zmenší větvící faktor
* **Prostor plánů**
  + Postupně opravujeme částečný plán – množina částečně instanciovaných operací/akcí, množina podmínek
  + Oprava – odstranění **kazu**
  + Uzly prohledávání jsou částečné plány
  + Kaz 1: otevřený cíl
    - Máme akci s předpokladem p, o kterém nevíme, jak jej nastavit/splnit
    - Léčba:

1. Najít a, co by mohla produkovat p
2. Instanciovat proměnné a/nebo nastavit vazbové podmínky
3. Nastavit kauzální podmínku
   * Kaz 2: Hrozba
     + Existuje možnost, že si smažeme (c) předpoklad (p) produkovaný pro nějakou akci (b)
     + Léčba:
4. Přinutit b, aby předcházelo a
5. Přinutit c, aby předcházelo a
6. Svázat proměnné tak, aby ke smazání p nedošlo

* Plánování v prostoru plánů (PSP) – vrací částečně uspořádaný plán
* **Analýza dosažitelnosti**
  + Zaměření na větvící faktor, zkoumání obsažitelnosti stavů (cílových)
    - Expanze stavového prostoru
    - Narazíme-li při expanzi na cílový stav, zrekonstruujeme plán
  + Expanze -> strom dosažitelných stavů (O(k^d))
  + **Graf dosažitelných stavů** – sloučení opakujících se stavů (O(k^d))
  + **Relaxace dosažitelnosti**
    - Původní: sg dosažitelný ⬄ obražen v grafu d. s.
    - Relaxovaný: sg dosažitelný => obsažen v plánovacím grafu d. s.
    - Sjednocení atomů úrovně stromu/grafu dos. Do 1 stavu/uzlu
* **Plánovací graf**
  + Střídání stavových (P0, …) a akčních (A1, …) vrstev
  + Hrany zpřesňují aproximaci oproti jen sjednocení atomů
  + Hrany z Pi do Ai+1 – ukazují předpoklady akce
  + Hrany z Ai do Pi – ukazují efekty akce
  + Skrz negativní efekty nemažeme
    - P0 ⊆ P1 ⊆ …
    - Axiom rámce (frame) – atomy přenášíme do další vrstvy
* **Paralelní plány** – určeny plán. Grafem – posl. Množin akcí
  + Převod na **sekvenční plán** – posloupnost množiny uspořádání akcí
  + Nutno zajistit **nezávislost** – mezi paralelně proveditelnými akcemi
    - Dvojice akcí (a, b) je nezávislá ⬄

Effect-(a) ∩ (precond(b) ∪ effect+(b)) = ∅

∧

Effect-(b) ∩ (precond(a) ∪ effect+(a)) = ∅

* + - Množina akcí π je nezávislá, jestliže je v π nezávislá každá dvojice akcí
      * Nezávislou množinu akcí lze aplikovat paralelně – akce si vzájemně neškodí
* Vzájemné vyloučení – **mutex**
  + V plán. Grafu mezi atomy nebo akcemi
  + Zpřesnění aproximace – zákaz nežádoucích dvojic
  + Akce a a b z Ai jsou vzájemně vyloučené (mutex) = jsou závislé, nebo je a mutex s předpokl. b
  + Atomy p a q z Pi jsou vzájemně vyloučené – každá akce a v Ai, že p ∈ effect+(a) je mutex s každou akcí b v Ai, že q ∈ effect+(b) a v Ai není akce c, že p, q ∈ effect+(c)
* Algoritmus **Graphplan** – střídá 2 fáze:

1. **Expanze** plánovacího grafu
2. Pokus o **extrakci** paralelního plánu z plánovacího grafu
   * + Neúspěch -> pokračuje expanze, + úroveň do p. g. – po čase stabilizace
   * Extrakce plánu
     + GP-scarch – chce bez-mutexovou množinu akcí π splňující g
     + Extract – tabulka nogoodů **▽**

▽(i) … zákazové cíle pro úroveň i

* Plánování jako SAT
  + Stavy odpovídají ohodnocení některých výrokových proměnných
  + Nutno dávat pozor na realitu
  + Cíl – vynutíme platnost cíle klauzulí
  + Problém – vývoj v čase -> časový index
    - Časová expanze – naznačuje, jak se postupně mění stavy v čase
  + Zavedení proměnných pro akce (a časové kroky) – pro každou časovou jednotku a každý atom proměnná, znamenající, že atom platí v daném čase
    - Pro každou časovou jednotku a akci další proměnnou – akce se provedla v daném čase
* **SATplan**
  + Využívá **plánovací graf** na určení atomů a akcí, které reprezentovat v časových krocích pomocí proměnných
  + **Mutexy** – zakazují určité kombinace akcí, přirozeně vyplývají z plánovacího grafu, podporují jednotkovou propagaci
  + Splňování formule odpovídá extrakci plánu z plánovacího grafu
* Hierarchické plánování – zkrácení plánů, rozděluje problém na málo velkých akcí, kde každou velkou akci rozděluje na pár menších atd.

**Plánování pohybu a problém lokalizace v robotice**

* **Lokalizace**
  + **Počáteční znalost** 
    - počáteční poloha známá – stopování
    - poč. poloha neznámá – globální lokalizace – změna na stopování hned po objev. polohy
  + **Přechodový model** – pohyb robota v rovině, známe přesnou mapu, Xt = (xt, yt, θt), posuvná rychlost vt , rotační ωt – deterministická stavová predikce
  + skuteční roboti nejsou determinističtí -> použití **Gaussovské distribuce**
  + senzorový model – vzahování k orientačnímu bodu, který senzory detekují (vzdálenost, natočení)
  + lokalizace **Monte Carlo (MCL)** – založeno na částicové filtraci
    - repr. Přesvědčení jako **mračno částic** (vzorků), které odpovídají stavům
    - přechodový a senzorový model na vstupu
    - vrací, jak odpovídá pozorování senzorem simulaci pozorování ve vzorku
* **Kalmanův filtr** – přesvědčení robota P(Xt | z1:t, a1:t-1) reprezentuje pomocí normálního rozd. N(μt ,Σt)
  + Funguje s lineárními (f ah) přechodovými a senzorovými modely
  + Přechodový model Xt+1 = f(Xt ,vt , ωt) interpretujeme jako maticovou operaci, senzorový model zt+1 = h(Xt) taky
  + Přesvědčení reprezentované jako normální rozdělení N(μt,Σt) je po lineárních transformacích opět popsáno normálním rozdělením N(μt+1,Σt+1)
* **Mapování** – otázka vytvoření reprezentace prostředí na základě údajů ze senzoru
  + Kombinace technik – lokalizace, dynam. Vytváření senzor. Modelu, někdy prohledávání prostředí
  + Technické aspekty – reprezentace pomocí mračna bodů, možnost spolupráce robotů, pohybující se objekty v prostředí
  + Aplikace – autonomní vozidla, Henry
* **Plánování pohybu**
  + Typy pohybů – z bodu do bodu, souladný – v kontaktu s překážkou
  + **Konfigurační prostor –** poloha, orientace, natočení kloubů
    - Hledání cesty – v robotice ve spojitém prostoru
  + Spojitost rozklad na buňky, skeletonizace
  + Pro zjednodušení nehledíme na neurčitost, vše deterministické
* **Konfigurační prostor**
  + Reprezentace pomocí pracovních souřadnic – pozice prvků robota
    - Plně popisuje pozici
    - Vhodná na detekci kolizí
    - Ne všechny souřadnice jsou možné
  + Reprezentace pomocí konfigurací – natočení kloubů…
* **Rozklad na buňky**
  + Konečný počet spojitých buněk – hledání cesty v rámci buňky ez
  + Jednoduché na implementaci
  + Komplikje se s rostoucí dimenzí
  + Rozdělování polo-obsazených buněk na 2^dimenze
* **Skeletonizace** – převod hledání cesty na úlohu v jedné dimenzi
  + **Voroného diagram** – množina bodů, které mají stejnou vzdálenost ke 2 a více překážkám
    - Tvoří graf, v němž je problém hledání cesty 1D
    - Vstup i výstup po úsečce
    - Obtížná konstrukce
  + **Pravděpodobnostní mapa** – generuje náhodné konfigurace, spojíme ty, mezi nimiž vede snadná cesta
* **Kinematika**
  + **Dopředná kinematika** – známe otočení kloubů – konfiguraci, chceme určit polohu efektoru – lineární transformace
    - Konfigurace => pracovní souřadnice (z konfiguračního do pracovního prostoru)
  + **Inverzní kinematika** – známe pozici efektoru reprezentovanou pracovními souřadnicemi, chceme určit konfiguraci – složitá transformace
    - Řešení rovnic, víc DOF -> víc řešení (nekonečno) – hledáme řešení minimalizující něco (energii, …)
    - Pracovní souřadnice => konfigurace (z pracovního do konfiguračního prostoru)
* **Dynamika a řízení** – nutno uvažovat nejen kinematický stav (konfiguraci), ale i rychlost – dynamický stav
  + Plánování s dynamickými stavy – hard, řešení diferenciálních rovnic
  + Alternativa – **metoda kontroleru** – je-li hodnota nějaké stavové proměnné odchýlena od požadované , robot ji kompenzuje působením at (síla efektoru)
    - Kontroler P – proporcionální – může oscilovat
    - Kontroler PD – proporciálně derivační – derivace je tlumix, na náhlou velkou změnu pomalá reakce
    - **Kontroler PID** – proporcionálně derivačně integrační – bere v potaz systematické externí působení (zahrnuto v integračním faktoru), nejčastější

**2. Splňování omezení s konečnými doménami (CSP), pokročilé prohledávání (backjumping, dynamický backtracking), filtrace domén a lokální konzistenční techniky, globální omezení, rozhodovací heuristiky.**

NI-UMI

**Splňování omezení s konečnými doménami (CSP)**

* **Stav** v CSP není atomický, má vnitřní strukturu
  + Rozdělen do **proměnných**
  + **Cíl** definován jako **splnění omezení**
* Faktorová reprezentace -> doménově nezávislé heuristiky
* **CSP** = problém splňování podmínek – trojice (X, D, C)

X … konečná množina **proměnných**

D … konečná **doména** (obor hodnot) proměnných

C … množina **podmínek** nad X – omezení pro rozhodnutí

* + - Binární, ternární, n-nární… (podle |X|
* **Řešení** = přiřazení hodnot proměným tak, že všechny podmínky jsou splněny (konjunkce)
* **Prohledávání** v CSP
  + **Stav** = částečné ohodnocení proměnné
    - **Konzistentní** – přiřazené hodnoty neporušují podmínky
  + **Počáteční stav** – prázdné přiřazení
  + **Akce** – přiřazení hodnoty proměnné z její domény
  + **Cílový stav** – všechny p. ohodnoceny + konzistentní
    - Komutativita ohodnocení
* **Chronologický backtracking** – rekurzivní prohledávání do hloubky (DFS)
  + Dopředný chod – přiřazení
  + Nelze přiřadit -> u poslední přiřazené zkoušíme jinou hodnotu
* Složitost: CSP je **NP-těžké**
* Constraint programming – ILOG, Minion, Gecode, MiniZinc, …
* Neefektivita backtrackingu
  + Pasivní využití podmínek
  + Zapomínání již odvedené práce – hodněkrát nastaví pár proměnných na stejné ohodnocení, které je nesplnitelné

**Pokročilé prohledávání**

* **Backjumping** – pokud v průběhu ohodnocování x momentální proměnná nemůže být nastavena na žádnou hodnotu, najdou se všechny nesplněné omezení, z nich se vyberou konfliktní proměnné, následně se odnastaví všechny proměnné až do nejbližší konfliktní proměnné
  + Rozšiřujeme ohodnocení S´ do proměnné x, zbývá ale ohodnotit víc proměnných
    - Přiřazení S´(x)=d nezpůsobí konflikt, ale přesto nejde rozšířit na plné ohodnocení (odhalení konfliktu až při dalším ohodnocování)
    - Nemožnost přiřazení S´(x)=d odhalí prohledávací algoritmus (rekurzivní volání BJ) při pokusu ohodnotit zbylé proměnné
      * Uvažujeme **konfliktní množinu** Conf pro *zbylé proměnné, které v pozdějším běhu nebylo možné ohodnotit* – Conf je příčina nemožnosti rozšířit S’∪{x=d} na celé X
      * S’ pro Conf\{x} je pak příčina nemožnosti provést S’(x)=d – Confd = Conf \ {x} je konfliktní množina pro hodnotu d∈D(x)
    - Nelze-li provést přiřazení S´(x)=d pro žádnou hodnotu d z domény x, BJ skočí zpět na poslední proměnnou z konfliktní množiny
  + **Backjumping** v listu
    - Máme částečné ohodnocení s´, ohodnocujeme poslední proměnnou x. Předpokládejme, že s´ do x rozš. Nelze -> skok zpět na poslední prom. z konfliktní množiny pro x
* **Dynamický backtracking** – při skoku zpět část ohodnocení, která je OK, ponechá
  + Změna pořadí proměnných (-> dynamika) – aby se zachovalo ohodnocení nekonfliktních proměnných
* **Závislosti (nogoody)**

S´…částečné ohodnocení

Conf = {y1…yn} množina konfliktních proměnných k S´

**Nogood** = podmínka zakazující vytvořit nerozšiřitelné přiřazení

* + BJ objevuje nogoody – lze ukládat – učení (lze i zapomínat – šetření pamětí)
  + logický důsledek v daném CSP
  + Specifické lze skládat do obecnějších – rezoluční pravidlo

**Filtrace domén, konzistence, globální omezení a heuristiky**

* **Filtrace domén** – dopředná kontrola
  + Zavádíme pracovní domény pro proměnné, ze kterých je možné hodnoty vyškrtávat (dynam. datová struktura)
  + Při ohodnocení prom. Kontrola souv. podmínek, případně vyškrtáváme hodnoty z pracovních domén proměnné zasažené podmínkou
* **Backtracking s filtrací** – přiřazení opatříme dopřednou kontrolou (inference)
* Důraznější filtrace – nebudeme čekat na úplné ohodnocení proměnné v podmínkách
  + Propagace podmínek
* **Hranová konzistence** – obecnější druh filtrace pro CSP s binárními podmínkami
  + Výpočet: opakovaně vynucujeme konzistenci na hranách (dokud se mění prac. domény)
  + předpokládejme podmínku c ∈ C nad proměnnými x1 a x2, D = doména
    1. kontrolujeme orientovanou hranu (x1, x2)
    2. pro každou hodnotu d1 ∈ D(x1) hledáme **podporu** d2 ∈ D(x2) vzhledem k podm. c
    3. podpora – d2 ∈ D(x1) taková, že (d1, d2) ∈ c
    4. nemá-li d1 podporu v D(x2), odebereme d1 z D(x1)
  + **Podpora** pro a z domény x: hodnota b je podpora v doméně y, jestliže (x, y) splňuje podmínku (**a nemá podporu, jestliže je nemožné, aby x nabývala hodnoty a, aniž by porušila podmínku**)
* **Heuristiky pro výběr proměnné**:
  + Výběr nejvíce omezené proměnné
  + Výběr „klíčové“ proměnné
  + Výběr tak, abychom co nejdřív narazili na řešení
  + Využití struktury – míra souvislosti grafu CSP je určující
    - Komponenty souvislosti lze řešit nezávisle
    - Nejjednodušší souvislý graf je strom
* **Zobecněná hranová konzistence** (GAC) – zobecněná podpora
  + Podmínky jsou nebinární, pro jednu zvolenou proměnnou v podmínce je podpora tvořena všemi ostatními proměnnými
  + Globální podmínky
    - **allDifferent**(x1…xk) – obvykle uvažována jako jedna podmínka nahrazující všechny dvojice nerovností
    - Modelujeme jako hledání párování – konzistentní ohodnocení odpovídají párování
  + **Propagace** v allDifferent
    - Najdeme nějaké hodnocení proměnných, které splňuje allDifferent, následně se odeberou zbytečné hrany a dle toho se upraví pracovní doména proměnných
      * Najdeme maximální párování (maximální tok) – odstraníme hrany, co nepatří do žádného max. párování
      * Vyřazení neoznačených promítneme zpět do pracovních domén

**3. Systematické a lokální splňování v logice (DPLL, CDCL, WalkSAT, posílání zpráv). Automatické uvažování, rozhodování v teoriích prvního řádu, obecná rezoluce, princip SAT-modulovaných teorií (SMT). Zpracování přirozeného jazyka.**

NI-UMI

**Systematické a lokální splňování v logice**

* φ je splnitelná, jestliže existuje α, že α |= φ
  + NP-úplný problém
  + **α |= φ**: „φ je splněná pro ohodnocení α“
* **CNF** – konjunktivní normální tvar – podobá se CSP
* **Backbone** – podmnožina lit. Φ, které jsou pravdivé v každém splňujícím ohodnocení φ
* Backdoor – podmnožina proměnných φ, po jejichž ohodnocení je vyřešení splnitelnosti pro φ po dosazení v polynomiálním čase
* Autarky (autarkie) – částečné ohodnocení, které splní každou klauzuli, ve které nastavuje nějaký literál
* Unsatisfiable core – je-li φ nesplnitelná, podmnožina jejích klauzulí, která je stále nesplnitelná
* **Cejtinova transformace** – postupujeme podle derivačního stromu
  + Pro každý vnitřní uzel derivačního stromu formule φ zavedeme **pomocnou proměnou** ai
    - Indikátorem splnění pod-formule odp. stromu
    - Listy = původní proměnné
* **Jednotková propagace** – pokud klauzule není splněná a všechny až na jednu proměnnou v ní jsou ohodnocené, poslední proměnná musí splnit danou klauzuli
* **Jednoduché splňování** – **DPLL**
  + Backtracking s jednotkovou propagací čistých proměnných
    - **Čistá proměnná** – buď pouze neg., nebo pouze poz., lze ihned ohodnotit
  + **Důvody konfliktu**
    - Předchůdcovská klauzule (Antecendent(x))
      * Jestliže byl literál l ohodnocen jednotkovou propagací kvůli klauzuli c – c je předchůdcovská pro l
    - **Implikační graf** pro ohodnocení α´
      * Zachycuje vznik α´, umožňuje vypátrat důvod nemožnosti α´ dále rozšířit – příprava na skok zpět (backjumping)
      * Není třeba ho explicitně konstruovat (looking at you, CDCL)
      * Vrcholům je přiřazena informace o ohodnocení, K je speciální vrchol označující konflikt (nemožnost rozšířit dané částečné ohodnocení)
* **Učení klauzulí –** hranový řez v IG, který odděluje K a rozhodovací vrcholy, určuje konfliktní klauzuli c
  + Lze využít k zpětnému skoku, lze si ji zapamatovat
  + Spec. Případ nogoodu
* **Splnitelnost s garancí** – DPLL, CDCL, …
  + Úplnost – vždy skončí a odpoví
  + Mohou poskytnout formální zdůvodnění odpovědí (implikační graf) – Explainable AI
  + Speciální třídy formulí – garantujeme rychlou odpověď
* **Trojice paradigmat** – CSP, SAT, IP (integer programming)
  + CSP

+ prohledávací algoritmy, silná formulace

- Heterogenní omezení

* + SAT

+ homogenní, silné učení

- absence aritmetiky

* + IP

+ dobrá aritmetika

- slabé učení

* **CDCL** algoritmus (Conflict Driven Clause Learning) – využívá jednotkové propagace, staví implikační graf (NE EXPLICITNĚ)
  + Základ systematických řešičů SATu – kombinuje BJ (skoky zpět), UP (jednot. propagaci) a učení
    - Analyzuje implikační graf pro nalezení vhodného konfliktu
  + Obsah obrázku text, snímek obrazovky, diagram, Písmo

    Popis byl vytvořen automatickySoučásti:
    - **DECIDE** – ohodnotí další neohodnocenou proměnnou
    - **BCP** – jednotková propagace (boolean constraint propagation)
    - **BackTrack**(level) – zruší všechna rozhodnutí na úrovních vyšších než level
  + Když dojde ke konfliktu, nalezne vhodný řez grafu, z něj vytvoří nogood, ten přidá jako klauzuli
    - **Vhodný řez** – z poslední vrstvy obsahuje pouze prom., u které se rozhodovalo ohodnocení
    - Přidaná klauzule následně vynutí změnu ohodnocení u poslední rozhodované proměnné pomocí UP
    - Dělá **backjumping** – protože když se dělá řez, skočí se do předposlední rozhodovací vrstvy, která obsahuje proměnnou v udělaném řezu – může přeskočit i nějakou vrstvu, která není na hraně řezu
  + **Analýza konfliktu**
    - **Hranový řez** – odděluje K a rozhodovací uzly v implik. grafu – určuje konfl. klauzuli (nogood)

Obsah obrázku diagram, řada/pruh, Písmo, text

Popis byl vytvořen automaticky Obsah obrázku Písmo, text, rukopis, řada/pruh

Popis byl vytvořen automaticky

* + - Obecné požadavky – krátká konfliktní klauzule (kvůli UP, snadno uložitelná, obecnější)
    - Budeme chtít **VYNUCUJÍCÍ konfliktní klauzuli**
      * Obsahuje jediný literál z aktuální rozhodovací úrovně
      * Po návratu (BackTrack), je-li vynucující klauzule naučena (přidána), dojde skrz ni okamžitě k jednotkové propagaci
      * Nutnost, aby CDCL fungoval – v. k. k. zajistí větvení
  + **Unikátní implikační bod** – UIP
    - Def. Vzhledem k akuální rozhodovací úrovni

1. Vrchol implikačního grafu různý od K
2. Všechny cesty z rozhodovacího vrcholu do K jím procházejí
   * + Vždy existuje, může jich být víc – zajímá nás 1. – nejbíže K
   * **Volba konfliktní klauzule**
     + Vynucení a UIP – obsahuje první UIP jako jediný literál z aktuální rozhodovací úrovně
       - Klauzule bude vynucující a bude krátká
     + Následuje návrat
       - Uložení KK (učení)
       - Návrat na 2. nejvyšší rozhodovací úroveň z KK
       - Singularity
   * Vlastnosti CDCL: Vždy skončí, KK lze zapomínat, krátké klauzule, systematický
   * Implikační graf není třeba explicitně konsturovat
   * Heuristiky pro výběr
     + VSIDS – každý literál má své skóre, literál s nejvyšším skóre ohodnocujeme True
     + Berkmin – proměnné a literály mají VSIDS skóre
       - Konfliktní klauzule na zásobník
       - Nerozhodnutá klauzule se bere ze zásobníku
       - Prázdný zásobník
   * Pomocí restartů se může učit z více stran zároveň

* Jednotková propagace efektivně – rychlé určení jednotkové klauzule
  + 2 sledované literály
    - V každé klauzuli označíme 2 neohodnocené literály (sledované)
    - U každé proměnné známe její sledované výskyty (ve kterých klauz. je sledován její literál)
* SAT řešen lokálně – **GSAT**
  + Hladové procházení úplných ohodnocení a restarty
  + Neúplný – nemůže-li najít splňující ohodnocení, nevíme, jestli je vstupní formule splnitelná, či ne
  + Najde takovou proměnnou, jejímž překlopením co nejvíce zvýším počet splněných klauzulí
  + Účelová funkce – počet splněných klauzulí (náchylný k uváznutí v lokálním maximu)
  + **Restart** = solver vezme zpět každé přiřazení na cestě a udělá novou sérii rozhodnutí – zahodí podstrom, ale neodnaučí se naučené
* Náhodná procházka – **WalkSAT**
  + Procházení úplných ohodnocení s restarty
  + Z lokálních maxim se může dostat náhodnými kroky
  + Střídání hladového kroku (s pravděpodobností p) a náhodného kroku (pst 1-p)
* **Fázový přechod –** poměr mezi počtem klauzulí a počtem proměnných
  + Obsah obrázku diagram, řada/pruh, text, skica

    Popis byl vytvořen automatickyNáhodné k-CNF formule (k=3, 4, 5, …)
  + Snadno splnitelné formule – málo klauzulí (málo omezení)
  + Snadno nesplnitelné formule – mnoho klauzulí (příliš omezené)
  + Ve fázovém přechodu jsou nejtěžší formule někde uprostřed
  + Limitní splnitelnost pro určité hodnoty: c = počet klauzulí / počet proměnných
* **Algoritmy posílání zpráv**
  + Na jednotlivé klauzule formule pohlíží jako na paralelní funkční jednotky – zkoumá, jak změna hodnoty proměnné ovlivní související klauzule
  + Faktorový graf (graf funkčních jednotek) – uzly jsou klauzule a proměnné, hrany jsou že proměnná je v klauzuli
  + **Propagace výstrah** – klauzule posílají proměnným zprávy (výstrahy), jak si přejí proměnnou nastavit, aby klauzule byla splněna
    - Aktualizační pravidlo pro výstrahy
  + **Výstrahami inspirovaná decimace** – postupně zjednodušuje formuli dosazováním
    - Preferovaná hodnota
    - Indikátor sporu
* Systematický CDCL se inspiroval u lokálních
  + **Restarty** v CDCL
  + **Náhodný krok** – s malou pst ignoruje heuristiku a vybere náhodnou hodnotu
* Lokální se inspirovaly u CDCL – 1 sledovaný literál
* **Inkrementální splnitelnost** – konflikt pro φ(t) je konfliktem i pro φ(t+1)

**Automatické uvažování**

* Automatické dokazování vět, uvažování s neurčitostí, …
* ATP – algoritmy ukazující, že zadané matematické tvrzení je logickým důsledkem daných axiomů
* APC – interaktivní verze ATP
* **Formalizace uvažování** – pomocí logiky prvního řádu (first order logic – FOL)
  + Silný vyjadřovací prostředek
  + Platíme nerozhodnutelností – nemůže existovat algoritmus, který by rozhodoval o platnosti zadaného tvrzení v logice 1. řádu
* **Rozhodování** – pro každý vstup skončit a dát správnou odpověď, zda tvrzení platí, či ne
  + Pro logiku prvního řádu máme jen semi-rozhodnutelnost
    - Algoritmus skončí, jestliže tvrzení platí a dokud neskončí, nic nevíme
  + Nebo rozhodnutelnost ve speciálních případech (fragmenty)
* **Nerozhodnutelnost**
  + Příčiny nerozhodnutelnosti
    - Logika 1. řádu je tak silná, že v ní lze popisovat algoritmy
    - Programy a logika se v důsledku toho pak dokáží oklamat
  + Algoritmy neumí rozhodovat tvrzení o algoritmech -> neumí rozhodovat o tvrzeních obecně
* **Logika prvního řádu**
  + Jazyk:
    - Proměnné pro individua – x, y, z, …
    - Spojky – unární, binární (negace, konjunkce, …)
    - Kvantifikátory – všeobecný, existenční
    - Pomocné symboly – závorky, tečka
  + Signatura (nelogické symboly):
    - Symboly pro funkce (transformace individuí) – f, g, h, +, \*… libovolné arity (nulární = konstanty)
    - Symboly pro predikáty (vlastnosti individuí) – R, S, <, = … libovolné arity (nulární obvykle ne)
* **Formule 1. řádu**
  + **Termy –** proměnná je term
    - Jestliže f je funkční symbol arity n a t1…tn jsou termy, pak f(t1…tn) je term
  + **Atomy** – jestliže p je predikátový symbol arity n a t1…tn jsou termy, pak p(t1…tn) je atomická formule (atom)
  + Formule je slovo (konečná posloupnost symbolů)
    - Atomická formule je formule
    - Jestliže p a q jsou formule, pak (¬p), (p ∧ q), (p ∨ q), (p ⇒ q), (p ⇔ q) jsou formule
    - Jestliže p je formule a x proměnná, ∀x(p(x)) a ∃x(p(x)) jsou formule
    - Každá formule vznikne konečným počtem aplikací těchto pravidel
* Pojmy v logice 1. řádu
  + **Teorie** – libovolná množina formulí
  + **Důkaz formule** φ z teorie T – posloupnost formulí končících φ, každá f. patří do T nebo je z nějakých předchozích f. v posloupnosti odvozena odvozovacím pravidlem
  + **Pravdivost** (validita) - T ⊧ φ (φ je pravdivá (validní, platná) v T), model pro T je modelem pro φ
    - Model – struktura, která splňuje všechny formule teorie
    - Sporná teorie nemá model
  + **Struktura** (interpretace, realizace) signatury
    - Nosná množina M (prvky jsou individua)
    - Realizace funkcí nad M: f: M×M×…M →M
    - Realizace predikátů nad M: p⊆ M×M×…M
* Semi-rozhodování: **rezoluce**
  + Rezoluční metoda – klauzální zamítací dokazování
    - Předp. Dokazování tvrzení C vzhledem k množině axiomů T
    - Uvážíme T ∪ {¬C}, snažíme se odvodit spor
  + **Rezoluční pravidlo**
    - speciální výroková varianta (řez) – x, y výrokové literály, z výroková proměnná

Obsah obrázku text, Písmo, řada/pruh, snímek obrazovky

Popis byl vytvořen automaticky

* + - Obecná varianta pro logiku 1. řádu

Obsah obrázku text, Písmo, snímek obrazovky, řada/pruh

Popis byl vytvořen automaticky

* + - * p, q literály, r,s atomy
      * θ – nejobecnější unifikace r a s s (tedy rθ = sθ a každá jiná θ', že rθ' = sθ' lze vyjádřit pomocí θ, tedy θ' = θρ)
* Příprava na rozhodování

1. převod T a C na CNF
   * standardizace proměnných – přejmenovávání
   * skolemizace – odstraňování existenčních kvantifikátorů
   * zahození všeobecných kvantifikátorů
   * distribuce skrz ∧ a ∨
2. eliminace spojek jiných, než ¬, ∧, ∨
3. negace dovnitř až k atomům

* **Unifikační algoritmus** – hledá nejobecnější unifikaci (mgu) θ množiny termů x a y – chceme xθ = yθ
* **ANL loop** – pomocí DFS či BFS výběr klauzule c, vrací splnitelnost
* Holičův paradox – „neexistuje holič, který by holil právě ty, kteří se neholí sami“
  + ANL s rezolučním pravidlem nenajde důkaz
* Pravidlo faktorizace – eliminace unifikovaného literálu -> zúplnění rezoluce pro logiku 1. řádu
* **Resoluce s faktorizací** – zdánlivě úplný algoritmus pro rozhodování o platnosti tvrzení v logice 1. řádu, problém teorie s nekonečně mnoho axiomy
* **Dokazování ve fragmentech** – omezení na podmnožinu jazyka logiky 1. řádu – **fragment**
  + Často bez kvantifikátorů, speciální sada axiomů
  + Speciální algoritmy na dokazování ve fragmentech
    - Poskytují úplnost rozhodovacího procesu
    - Převod otázky T ⊧ φ na splnitelnost
* **Dokazování s rovností**
  + Logika s rovností (fragment s rovností) – z predikátů pouze =, bez funkčních symbolů a kvantifikátorů
  + Zjednodušení – jen spojky ¬, ∧,∨ a převod na NNF (negační normální tvar)
  + V NNF lze hovořit o literálech
* **Graf rovností** – def. Pro formuli φ s rovnostmi v NNF jako G= = (Var(φ), E=, E≠)
  + Vrcholy – proměnné, hrany – E= rovnosti a E≠ nerovnosti
  + Sporná kružnice – cyklus v G=, který obsahuje právě jednu nerovnost
* **Zjednodušovací algoritmus** – na vstupu formule φE s rovnostmi ve tvaru NNF, konstrukce grafu rovností
  + Výstup je kratší ekvivalentně splnitelná formule
* **Výroková kostra** – zachycuje Booleovskou strukturu formule s rovnostmi
  + Pro každou rovnost (atom) zavedeme výrokovou proměnnou, v pův. formuli rovnosti nahradíme zavedenými výrokovými proměnnými
  + Nezachycuje tranzitivitu rovnosti
* **Tranzitivita rovnosti**
  + Nepolární graf rovností G=NP = (Var(φ), E= ∪ E≠) – zapomeneme na polaritu
    - Pro každý cyklus a každou hranu v cyklu přidáme tranzitivní podmínku – zakáže hraně přiřadit False, když jsou všechny ostatní hrany True
* **Líný přístup**
  + Komponenty – decideT (dokazovací procedura pro konjunktivní fragment teorie T), řešič SATu
  + Spolupráce decideT a řešiče SATu
    - Řešič vybírá, které literály splnit
    - DecideT kontroluje, jestli vybrané literály můžou současně platit (ne -> zakázané ohodnocení)
* SAT modulované teorie (**SMT**) – spolupráci SATu a DECIDE lze zabudovat do CDCL
  + Propagace teorií přímo do řešiče satu
  + Na formuli t se kladou další podmínky, aby byla zaručena konečnost algoritmu
* Další metody – přirozená dedukce, metoda tablo, rozhodování s bitovými vektory, ukazateli, poli

**Zpracování přirozeného jazyka**

* **Formální jazyk** – množina řetězců, řetězec posloupnost terminálů
* **Přirozený jazyk** – nemá pevnou definici (přístup k nim jako by byly formální)
* **Gramatika** – sada pravidel specifikující jazyk
  + – terminály, neterminály, počáteční neterminál, konečná množina pravidel
  + **Chomského hierarchie** gramatik a jazyků – rekurzivně spočetné, kontextové, bezk., regulární
* **Fáze komunikace**
  + Záměr – sdělit skutečnost
  + Generování – přesvědčivého sdělené o skutečnosti věta
  + Syntéza – sdělení, hlasový syntetizátor zvuk
  + Vnímání – rozpoznání hlasu věta
  + Analýza – věta derivační strom
  + Odstranění dvojsmyslů – určení správné interpretace, pravděpodobnostně
  + Začlenění – budeme věřit či ne
* **Analýza** (parsing)
  + Pravděpodobnostní derivační strom – vnitřním uzlům přiřazena pravděpodobnost
  + Algoritmus CYK – předpokládá bezkontextovou (pravděpodobnostní) gramatiku v Chomského normální formě (pravidla nebo
* **Učení PCFG**
  + Učení pravděpodobností pro PCFG – podle banky stromů (treebank), korpus korektně vytvořených derivačních stromů vzhledem k dané bezkontextové gramatice
* **Rozšířené gramatiky**
  + **Lexikalizované PCFG**
    - Pravděpodobnost pravidel závisí na vztahu slov v derivačním stromu, ne jen na jejich sousedství ve větě
    - Vztah všech dvojic je komplikované počítat, vybereme nejvýznamější slovo z daného spojení – hlava (head)
    - Stále mnoho dvojic, kro které potřebujeme pravděpodobnost
  + **DCG** – definite-clause grammars (gramatiky s určenou klauzulí)
    - Používáme navíc – pravidlo platí, pokud je podmínka splněná
    - Problém – skloňování/pády, shoda přísudku s podmětem
* **Sémantika** – výrazy
  + Aritmetické výrazy, lambda notace
  + Kvantifikace – používá se přechodná forma mezi derivačním stromem a logickou interpretací
    - **Kvazi-logická forma** – akceptace ústupků od přísných pravidel ve prospěch zachycení kontextu, modelování významových relací mezi větami
  + Další jevy
    - Časy – kalkulus událostí (event calculus)
    - Nejednocnačnost – lexikální, syntaktická (více derivací), doslovné x obrazné
* **Statistický překlad** – překládáme větu z angličtiny do francouzštiny
  + – **překladový model** – jak pravděpodobné je, že je anglickým překladem dané francouzské věty
  + – jak pravděpodobné je, že je francouzským překladem dané anglické věty
  + – **jazykový model** – jak pravděpodobné je, že je věta ve francouzštině
  + Hledáme řetězec
  + Lze počítat podle – máme-li dobrý jazykový model pro nebo přímo
  + Průběh statistického překladu
    - Rozdělíme vstupní anglickou větu na slovní spojení
    - Vyhledáme slovní spojení v anglické části korpusu
    - Najdeme odpovídající překlad slovních spojení ve francouzské části korpusu
    - Sestavíme výsledný překlad z nalezených francouzských překladů slovních spojení

**4. Metody pro hodnocení a výběr příznaků (univarietní/multivarietní metody, filtrační/wrapper/embedded metody). Selektivní/adaptivní metody redukce počtu instancí: Condensed Nearest Neighbor (CNN), Delauney/Gabriel/RNG grafy, Wilsonova editace, Multi-edit metoda, Tomkovy spoje.**

NI-PDD

**Univarietní / multivarietní metody**

* **Univarietní** – zvažuje každý příznak zvlášť
  + Jak relevantní je proměnná pro predikci výsledku proměnné ?
  + Irelevantní příznak :
  + **T-test** – normálně rozdělené třídy, stejný rozptyl neznámý, odhad z dat jako
    - Nulová hypotéza:
    - Pokud je nul. hyp. pravdivá:
  + **Korelace** – vzájemný lineární vztah mezi proměnnými
    - +1 přímá závislost, -1 nepřímá závislost
    - **Pearsonův korelační koeficient**

Obsah obrázku text, Písmo, řada/pruh, bílé

Popis byl vytvořen automaticky

* + - **Korelační a kovariační matice**

Obsah obrázku text, Písmo, řada/pruh, snímek obrazovky

Popis byl vytvořen automaticky

* + - **Spearmanův koeficient pořadové korelace**
      * Máme 2 náhodné proměnné X a Y s neznámými rozděleními
      * Seřadíme hodnot a podle velikosti a přiřadíme jim pořadová čísla a . Hodnota koeficientu je:

Obsah obrázku Písmo, text, bílé, řada/pruh

Popis byl vytvořen automaticky

* + **Obsah obrázku kruh, snímek obrazovky, diagram, Písmo

    Popis byl vytvořen automatickyObsah obrázku Písmo, bílé, Grafika, text

    Popis byl vytvořen automatickyEntropie** – předpokládané množství informace v náhodné proměnné
  + **Univarietní závislost**
    - Nezávislost:
    - **Vzájemná informace** – hodnota vzájemné závislosti mezi dvěma náhodnými proměnnými
      * Kvantifikuje množství informace získané o jedné proměnné zkoumáním jiné náhodné proměnné

Obsah obrázku text, Písmo, řada/pruh, bílé

Popis byl vytvořen automaticky

* + - * Empirická vzájemná informace – z dat kontingenční tabulka, z ní empirická pravděpodobnost, ta je vstupem pro vzájemnou pravděpodobnost
* **Multivarietní** – zvažuje podmnožiny příznaků najednou

**Filtrační / wrapper / embedded metody**

* **Filter metody** – hodnotí příznaky nebo jejich podmnožiny nezávisle na klasifikátoru
  + *Kritéria* – hodnocení relevance příznaků
  + *Hledání* – seřazení příznaků
  + *Hodnocení* – statistické testy
  + Robustní výsledky, ale nemusí vybrat úplně nejlepší příznaky
* **Wrapper metody** – používají klasifikátor k hodnocení příznaků / podmnožin
  + Rozdělení dat na tréninkové, validační a testovací
  + Problém přeučení, ale najde nejlepší příznaky
  + *Kritéria* – měření užitečnosti podmnožiny
  + *Hledání* – prohledávání prostoru podmnožin příznaků
  + *Hodnocení* – **Křížová validace** – rozdělení do  skupin
    - Trénink na všech skupinách kromě 1
    - Výběr těch příznaků s nejlepším skóre podle poslední skupiny
    - Opakování, dokud nejsou všechny skupiny použity pro testování
    - Zprůměrovat
* **Obsah obrázku text, Písmo, snímek obrazovky, diagram

  Popis byl vytvořen automatickyEmbedded metody** – jako wrapper, hledání ovládáno algoritmem konstruujícím klasifikátor
  + *Kritéria* – měření užitečnosti podmnožin
  + *Hledání* – naváděno učícím procesem
  + *Hodnocení* – křížová validace
  + Výsledky podobné wrapperům, ale menší riziko přeučení a míň výpočetně náročné

**Selektivní/adaptivní metody redukce počtu instancí**

* **Condensed Nearest Neighbour** (CNN) – inkrementální, závislá na pořadí, ani minimální ani decision boundary konzistentní
  + pro brute-force metodu
  + Lze navázat na reduced NN
  + Postup:
    - Inicializace podmnožiny jedním trénovacím případem
    - Klasifikace většiny vzorků pomocí té podmnožiny a přesun všech špatně klasifikovaných vzorků do podmnožiny
    - To se opakuje, dokud buď nedochází k přesunu nebo je podmnožina plná
* **Proximity grafy** – poskytují různé definice „souseda“
  + Obsah obrázku řada/pruh, kruh

    Popis byl vytvořen automaticky**Delaunay**
    - Delaunayho triangulace – duální graf Voronoiho diagramu?
    - Tři body jsou si sousedy, pokud kružnice jim opsaná neobsahuje žádné jiné body
    - Voronoi editing: ponechá body, jejichž sousedi jsou z jiné třídy
    - Rozhodovací hranice je stejná
    - Obsah obrázku kruh, řada/pruh

      Popis byl vytvořen automatickyKonzervativní podmnožina, nechává si body navíc, drahá na výpočet ve vysokých dimenzích
  + **Gabriel**
    - Podmnožina delaunayho triangulace
    - Body jsou sousedi, pouze pokud je jejich sféra vlivu prázdná
    - Obsah obrázku kruh, umění

      Popis byl vytvořen automatickyNezachovává identickou rozhodovací hranici, ale změny se odehrávají jen vně komvexního obalu datových bodů
    - Spoje mezy sousedy = Tomek links
  + **RNG** – Relative Neighbourhood Graph
    - Podmnožina Gabrielova gradu
    - Dva body jsou sousedi, pokud „luna“ definovaná průsečíkem jejich radiálních sfér je prázdná
    - Dále zmenšuje počet sousedů
    - Rozhodovací hranice bývají drastické a nemusí být konzistentní s trénovací množinou
* **Editing** – vyjmutí šumových bodů a tvorba hladkých rozhodovacíh hranic
  + Často se zachovají body daleko od hranice a vzniknou homogenní shluky bodů
  + **Wilson editing** – vyjmutí bodů, které mají jinou třídu než většina jejich k nejbližších sousedů
  + **Multi-edit** – opakovaně opakuje Wilsonův editing na náhodné části, klasifikuje pomocí 1-NN pravidla
    - **Difúze** – rozdělení dat do N >= 3 náhodných podmnožin
    - **Klasifikace** – klasifikace S\_i pomocí 1-NN za použití S\_(i + 1)modN jako trénovací set
    - **Editing** – zrušení všech špatně klasifikovaných vzorků
      * 1-NN pravidlo – klafikace X podle jeho nejbližšího souseda z trénovacích bodů
    - **Konfúze** – přidání všech vzorků do nové množiny
    - **Terminace** – pokud posledních I iterací nemělo za následek žádný editing, konec, jinak od začátku
  + Kombinace editingu a condensingu – první editing na odstranění šumu a hladkou hranici, potom kondenzace pro získání menší podmnožiny
* **Tomkovy spoje** – odstranění šumu a hraničních případů
  + Tomkův spoj
    - patří do jiných tříd, je vzdálenost mezi nimi
    - je Tomkův spoj, pokud neexistuje takové , že nebo

**5. Algoritmy pro nahrazování chybějících hodnot. Detekce a ošetření odlehlých hodnot. Vyvažování skupin dat (undersampling/oversampling metody).**

NI-PDD

**Nahrazování chybějících hodnot**

* Nelze rozlišit mezi chybějící hodnotou a prázdnou buňkou
* **Způsoby nahrazení:**
  + **Nic nedělání** – použije se relevantní reprezentace tak, aby šel aplikovat model
    - Např. použití -1, když jsou všechny ostatní hodnoty kladné
  + **Vymazání**
    - **Listwise** – vymaže všechny data pro případ, který á jednu či více chybějících hodnot
    - **Pairwise** – listwise ale až v pozdějším stádiu, kdy jsou určeny důležité příznaky – zaměření na opravdu potřebná data
    - **Příznaky s chybějícími daty** – odstraní celý příznak, pokud chybí většina dat
    - Užitečné jen pokud málo záznamů obsahuje chybějící data, nebo příznak obsahuje většinu chybějících dat
  + **Imputace** – proces nahrazení chybějících dat substituovanými hodnotami
    - Neexistuje ultimátně nejlepší způsob
* **Mechanismy chybějících dat**
  + **MCAR** = *missing completely at random* – pokud je pravděpodobnost, že příznak chybí nezávislá na hodnotě příznaku a hodnotách jiných příznaků
    - Chybějící hodnoty jsou nezávislé na datech
  + **MAR** = *missing at random* – pokud je pravděpodobnost, že příznak chybí nezávislá na hodnotě příznaku, ale může záviset na hodnotách dalších příznaků
    - *Na otázku o platu častěji odpoví muži než ženy*
  + **MNAR** = *missing not at random* – pokud je pravděpodobnost, že příznak chybí závislá na hodnotě příznaku
    - *Chybí údaj o platu, protože na otázku člověk neodpověděl kvůli jeho výši*
* **Imputace**
  + **Průměr/medián** pomocí přítomních hodnot daného příznaku
  + **nejbližších sousedů** – vhodné pouze pro datasety nízkých dimenzí kvůli prokletí dimenzionality
    - Využití data encodingu – one-hot/dummy
  + Evaluace – pomocí korelace

**Odlehlé hodnoty**

* **Odlehlé hodnoty** = anomálie = extrémní hodnoty, které se odchylují od ostatních pozorování
  + Detekce anomálií – podvodné transakce, podezřelí cestující, zdravotní problémy
* **Detekce** – nesupervizované metody, hlavně clustering
  + Díky absenci labelů – outlieři nejsou předem známí
  + **Jeden příznak – rule of thumb** – outlieři jsou mimo interval
    - Q1 = první kvartil
    - Q3 = třetí kvartil
    - IQR = inter quartile range = Q3 – Q1 (prostředních 50 %)
  + **Více příznaků** – více-dimenzionální pohled
    - **Shlukovací přístupy** závislé na vzdálenostech (k-means, local outlier factor)
    - **Density-based** přístupy (SVM)
    - **Modelové** přístupy (neuronky – autoenkodéry, LSTM)
* Po identifikaci nutná analýza původu – Lidská chyba, Chyba měření, Experimentální chyba (extrakce dat), Samplingová chyba, Přírodní úkaz (novoty)
* **LOF = local outlier factor**
  + Hledání anomálií měřením lokální odchylky daného datového bodu vzhledem k jeho sousedům
  + Založeno na konceptu lokální hustoty, kde lokálnost je dána nejbližšími sousedy, jejichž vzdálenost je použita na odhad hustoty
  + Porovnáním lokální hustoty datového bodu s lokálními hustotami jeho sousedů lze identifikovat oblasti s podobnou hustotoou
  + Body, které mají podstatně nižší hustotu, než jejich sousedi jsou anomálie
  + **LOF(A)** = průměřná lokální hustota sousedů A dělená lokální hustotou A (relativní)
    - Hodnota blízká 1 znamená, že je objekt srovnatelmý se sousedy (není odchylka)
    - Hodnota pod 1 indikuje hustší oblast (inliner), hodnota hodně nad 1 indikují odchylky
  + Výhody:
    - LOF umí identifikovat anomálie v části datasetu takové, které by v jiné části stejného datasetu nebyly považovány za anomálie
    - Dobrá geometrická intuice pro málo rozměrné vektorové prostory
    - Mnoho variant a odvětví použitelnosti
  + Nevýhody:
    - Výsledné hodnoty se těžko interpretují
* Další metody detekce – **Isolation forest**, metody používající meta-learning (MetaOD)

**Vyvažování skupin dat**

* **Baseline metody**
  + Random over-sampling
    - Náhodná replikace příkladů z menšinových tříd
    - Zvyšuje pravděpodobnost overfittingu
  + Random under-sampling
    - Náhodná eliminace příkladů z většinových tříd
    - Může zničit potenciálně užitečná data, která by mohla být důležitá pro indukční proces
* **Under-sampling metody**
  + **Tomek links**
    - Odstranění šumu i hraničních případů
    - patří do jiných tříd, je vzdálenost mezi nimi
    - je Tomkův spoj, pokud neexistuje takové , že nebo
  + **Condensed Nearest Neighbour Rule** (CNN) – výběr bodů blízko hranice mezi třídami
    - Algoritmus:
      * E je původní trénovací množina
      * E´ obsahuje všechny pozitivní vzorky z S a jeden náhodně vybraný negativní vzorek
      * Klasifikace E s 1-NN pravidlem za použití vzorků z E´
      * Přesun všech špatně klasifikovaných vzorků z E do E´
    - Citlivý na šum – šumové vzorky se pravděpodobně špatně klasifikují a hodně jich bude přidáno do trénovací množiny
  + **One-Sided Selection (OSS)**
    - Tomek links + CNN (v tomto pořadí)
  + **CNN + Tomek links**
    - Nalezení Tomkových spojů je výpočetně náročné, bylo by levnější, kdyby se provádělo na redukovaném datasetu
  + **NCL = neighborhood cleaning rule**
    - Pro odstranění vzorků z většinové třídy
    - Více k čištění dat než redukci
    - Algoritmus:
      * Nalezení 3 nejbližších sousedů pro každý vzorek Ei z trénovací množiny
      * Pokud Ei patří do většinové třídy a 3 nejbližší sousedi do menšinové, Ei se odstraní
      * Pokud Ei patří do menšinové třídy a 3 nejbližší sousedi do většinové, odstraní se ti sousedi
* **Over-sampling – SMOTE** = Synthetic Minority Over-Samling Technique
  + Tvorba nových vzorků z menšinové třídy interpolací mezi několika vzorky z minoritní třídy které jsou si blízko
  + V prostoru příznaků spíš než v prostoru dat
  + Algoritmus:
    - Pro každý vzorek z minoritní třídy:
    - Vytvoř syntetické vzorky podél úseček spojujících nějaké/všechny z  nejbližších sousedů z menšinové třídy
  + V závislosti na množství over-samplingu se sousedi z  vybírají náhodně
  + **Generování syntetických vzorků**:
    - Vezme se rozdíl mezi vektorem příznaků, který se zvažuje a jeho nejbližší soused
    - Rozdíl se vynásobí náhodným číslem od 0 do 1
    - Toto se přičte k uvažovanému vektoru příznaků
* **Kombinace under-samplingu a over-samplingu**
  + **Smote + Tomek links**
    - Problém se Smote – může vyrobit umělou minoritní třídu moc hluboko v prostoru většinové třídy
    - Tomek links: čištění dat
    - Namísto odstranění pouze vzorků z většinové třídy které formují Tomek links, vzorky z obou tříd jsou odstraněny
  + **Smote + ENN**
    - ENN – extended nearest neighbour – odstraní vzorky, jejichž label se liěí od třídy alespoň dvou z jeho tří nejblížších sousedů
    - ENN odstraní víc vzorků než Tomek links
    - ENN odstraňuje vzorky z obou tříd
  + Over-sampling bývá lepší než under-sampling, ale kombinace může být ještě výhodnější

**6. Lineární projekce dat do prostoru o méně dimenzích: metoda hlavních komponent (PCA), lineární diskriminační analýza (LDA). Nelineární metody redukce dimensionality (Sammonova projekce).**

NI-PDD

* **Projekce dat** – nalezení transformace projektující data z  dimenzního prostoru do dimenzního prostoru ( při zachování nějaké formy informace (např. vzdálenost)
  + Výhoda: vyloučení redundantní informace (korelace)
  + Nevýhoda: fyzický význam nových atributů těžké interpretovat

**Lineární projekce**

* Jednoduché na výpočet:

Obsah obrázku Písmo, text, rukopis, bílé

Popis byl vytvořen automaticky

* **Náhodná projekce** (mapování) – původní -dimenzní data se projektují do -dimenzního podprostoru ( za využití náhodné matice , jejíž sloupce mají jednotkové délky
  + Obsah obrázku Písmo, text, typografie, bílé

    Popis byl vytvořen automatickyVyužívá maticovou notaci kde je původní množina -dimenzních pozorování,

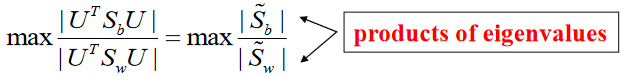
je projekce dat dov nižšího -dimenzního podprostoru

* + **Johnson-Lindenstrauss lemma**: pokud jsou body ve vektorovém prostoru projektovány na náhodně vybraný podporostor vhodně velké dimenze, pak jsou vzdálenosti mezi korespondujícími body (v originálním a novém prostoru) zhruba zachovány
  + Technicky vzato ta transformace není projekce, protože není nutně ortogonální
* Obsah obrázku text, Písmo, snímek obrazovky, řada/pruh

  Popis byl vytvořen automaticky**Metoda hlavních komponent (PCA)** – cíl je snížit dimenzionalitu dat a zároveň **zachovat co nejvíce variace** v datasetu (koresponduje množství informace)
  + **Informační ztráta** – je implikována redukcí dimenzionality
    - PCA zachovává co nejvíce informace (variace) je možno:
  + „Nejlepší“ prostor nízké dimenze má **střed** v průměru vzorku a má **směry** rozhodnuté „nejlepšími“ vektory vlastních čísel kovarianční matice dat
    - „Nejlepšími“ vektory vlastních čísel se myslí ty odpovídající **největším vlastním číslům = hlavní komponenty**
    - Protože je kovarianční matice reálná a symetrická, tyto vektory jsou ortogonální a formují množinu bázových vektorů
  + Možný výpočet – prolnutí přímky těžištěm, zkusit přímku rozovat a tím najít nejlepší úhel – PCA transdormuje osy
  + Cíl je transformace, nalezení nového souřadného systému, který zachovává max. míru rozptylu/informace
    - Hlavní komponenty = osy v cílovém prostoru
    - Pořadí hlavních komponent má význam
    - Hlavní komponenty jsou na sebe kolmé
* **Lineární diskriminační analýza (LDA)** = Fisherova projekce – snížení dimenzionality za **zachování co** **nejvíce diskriminační informace** (chyba špatné klasifikace)
  + Nalezení směrů, ve kterých jsou třídy **nejlépe separovány**
  + Bere v úvahu rozptyl uvnitř tříd, ale také rozptyl mezi třídami (= **scatter**)
  + Notace:
    - C tříd
    - průměrový vektor třídy ,
    - počet vzorků ve tříddě
    - celkový počet vzorků
    - matice velikostí obsahující vzorky z jednotlivých tříd
    - Obsah obrázku diagram, text, snímek obrazovky, origami

      Popis byl vytvořen automatickyObsah obrázku míč, design

      Popis byl vytvořen automatickyScatter matice **uvnitř třídy**:
    - Scatter matice **mezi třídami**:
      * má hodnost maximálně
      * Každá pod-matice má hodnost 1 nebo méně – vnější součin dvou vektorů
      * průměr celého datasetu
  + je projekční matice
  + LDA počítá transformaci, která **maximalizuje scatter mezi třídami** a minimalizuje scatter uvnitř tříd:



* + - … scatter matice projektovaných dat
  + **LDA lineární transformace**:
    - Řešení LDA je dáno vlastními vektory obecného problému vlastních vektorů:



* + - Lineární transformace je dána maticí , jejíž sloupce jsou vlastní vektory problému výše
      * Obsah obrázku text, Písmo, diagram, řada/pruh

        Popis byl vytvořen automatickySouřadnice v novém prostoru:
    - Jelikož má hodnost nejvýše , je maximální počet vlastních vektorů s nenulovými vlastními čísly (takže maximální dimenze podprostoru je )
* Při malé trénovací množině je PCA výhodnější než LDA, LDA zas lepší na větších a reprezentativních datech

Obsah obrázku diagram, řada/pruh, origami

Popis byl vytvořen automaticky**Nelineární redukce dimenzionality**

* **Sammonova projekce** – mapuje vysoko-dimenzní prostor na prostor nižší dimenze při **zachování mezi-bodových vzdáleností** z vysoko-dimenzního prostoru v nízko-dimenzním prostoru
  + Netransformuje souřadnice, místo toho reorganizuje pozice vzorů v novém prostoru
  + Vzdálenost mezi i-tým a j-tým objektem v původním prostoru je , a vzdálenost jejich proj. je
  + Obsah obrázku Písmo, řada/pruh, rukopis, bílé

    Popis byl vytvořen automatickySammonova projekce minimalizuje následující chybovou funkci – Sammon’s stress neboli **Sammonova chyba**:
  + Minimalizace se provádí např. gradientním sestupem

**7. Učení dopředných neuronových sítí, konvoluční neuronové sítě a jejich regularizace.**

NI-MVI

**Dopředné neuronové sítě**

* **Perceptron** –nejjednodušší neuronová sít, binární klasifikátor
  + Mapuje vstupní vektor na výstup pomocí lineární kombinace
  + 1 neuron, který vypočte lineární kombinaci vstupů a vah, přičte bias a tuto hodnotu vnitřního potenciálu prožené nelineární aktivační funkcí, jejíž výstup je zároveň výstupem celého perceptronu
  + Aktivační funkce – často step function: 1 pro ,jinak
  + Perceptron nelze použít pro problémy, které nejsou lineárně separabilní (např. XOR).
* **Perceptronový algoritmus**

1. Náhodně zvolený **vektor vah** kolmý na rovinu, která dělí 2 třídy (BINÁRNÍ klasifikátor)
2. Opakovaně se vybírá náhodný prvek z trénovacích dat
   1. Je z třídy P, ale hodnota wx je záporná – nový vektor vah w se vypočte přičtením vektoru x k vektoru vah w – posune se do pozitivní nadroviny určené vektorem w
   2. Je z třídy N, ale wx je kladná – x se od ve odečte, bod se posune do negativní poloroviny
3. **Vektor vah je operacemi 2. posouván k optimálnímu rozdělení tříd**
   * Prvotní nastavení vah – průměr všech kladných vektorů minus průměr všech záporných

* **Gradient learning** – konverguje i pro lineárně nesaparabilní data a maximalizuje rozdělení tříd
  + **Ztrátová funkce** (cost function) – např. **sum of squared errors**
  + Funkce je minimalizována krokem proti směru gradientu – ovládán ještě parametrem learning rate
    - Potřeba parciální derivace ztrátové funkce vůči každé váze vektoru w
* **Cross-entropy loss (log loss)** – ztrátová funkce používaná pro klasifikaci c tříd
  + Hodnota pro třídu Y a vektor p pravděpodobností, že bod na vstupu náleží do tříd:

* + Hodnota stoupá, čím jistější si je model klasifikací špatné třídy – nejvíc trestá predikce, kterými si je model jistý a jsou špatné
* **Multilayer perceptrony (MLP)** – více vrstev s více neurony, výstup každého neuronu vede do každého neuronu další vrstvy
  + Nelineární aktivační funkce – **sigmoida, tanh, ReLU**,… - musí být diferencovatelná
  + I na lineárně neseparabilní data
  + Vrstva MLP = funkce -> neuronová síť je složená funkce
  + A picture containing text

    Description automatically generatedPro použití gradientního sestupu pro učení vah musíme najít parciální derivaci Z.F. vůči každé váze – pomocí chain rule pro derivace složených funkcí
  + **Backpropagation** – zpětné šíření chyby

1. Pro všechna trén. Data spočítáme výstup sítě
2. Vyčíslíme hodnotu cost funkce – např. průměr cross entropy
3. **Postupně od poslední vrstvy ke vstupní** počítáme parc. Derivace vah vůči cost funkci

-> gradient

1. Gradientní sestup regulovaný learning ratem
2. Opakování do splnění terminálního kritéria

* **Momentum setrvačnosti** – Při updatu vah kromě aktuálního momentu přičítáme část gradientu z předchozího kroku
  + Urychlení konvergence, únik lokálním optimům, zamezení přehnané reakce na vzorek
* **ADAM** – **Adaptive moment estimation**
  + **Optimizátor** využívající exponentially decaying moving average gradientu akožto momentum
  + Adaptivní learning rate parametry pro jednotlivé váhy – penalizace vah způsobujících velké oscilace – zamezení přeučení
* **Kohonen’s self-organizing maps (SOM)** – typ neuronové sítě trénované nesupervizovaným učením
  + Produkce méně dimenzionální (2D) diskretizované reprezentace vstupního prostoru = mapa -> pro **redukci dimenzionality**
  + **Neighbourhood funkce** – snaha zachovat topologické vlastnosti prostoru

1. Inicializace vah na malé hodnoty
2. Vybrán vstup a neuron, který je mu nejblíže (např. euklidovsky, manhattansky)
3. Neuron s jeho okolím jsou updatovány a posunuty směrem ke vstupu
4. A picture containing athletic game

   Description automatically generatedOpakování 2-3 a neurony a okolí updatovány a posunovány k datovým bodům, dokud nejsou rozprostřeny tak, že zachycují nějakou strukturu
5. Váhy neuronů – pro popis dané oblasti

**Konvoluční neuronové sítě**

* Kromě fully connected vrstev konvoluční a pooling vrstvy
* **Fully connected vrstvy** – všechny neurony spojeny se všemi neurony v předchozí vrstvě
  + Každý neuron má vlastní váhu
  + Nepředpokládá nic o povaze dat, funguje zcela obecně, ale velmi výpočetně náročné
* **Konvoluční vrstvy** – neurony spojeny jen s hrstkou neuronů v předchozí vrstvě, které patří do nějakého okolí
  + Stejná sada vah použita pro všechny neurony ve vrstvě
  + Každý neuron si ze svého regionu (např. 3x3 grid) **extrahuje příznak** podle stejného předpisu (bo stejné váhy)
    - Dává smysl jen v případě, že data jsou prostorová a příznaky se vyskytují lokálně, a ne na jakékoliv pozici
  + **Konvoluční neuron** – má **filtr** definovaný váhami, ten aplikuje konvolucí na svůj region
    - **Padding** – pro místa, kde filter zasahuje mimo rozsah konvoluce
    - **Stride** – pro délku kroku při přechodech konvoluce
    - Aktivace = vizuální stimulace
    - Extrakce nejdřív obecných příznaků – rohy, hrany, čáry, později textury atd
  + **Pooling vrstvy** – Následují konvoluční vrstvy – **redukce dimenzionality**
    - Sjednocení extrahovaných příznaků (max pooling, average pooling, …)
* **Regularizace**
  + **Dropout** – během trénování **vypíná náhodné neurony** (= nastavuje váhy na 0) = regularizace
    - Síť se nenaučí spoléhat na malou sadu důležitých neuronů
  + Další forma regularizace například L1 a L2, augmentace dat, batch normalizace (standardizace vstupů do vrstvy).
* Typy předurčených konvolučních sítí
  + **AlexNet** – stackované konvoluční vrstvy
  + **GoogLeNet** – později Inception sítě – inception moduly s méně parametry, místo poslední FC vrstvy average pooling
  + **VGG** – velmi hluboká a velká síť
  + **ResNet** – reziduální skip connections – pomoc s vanishing gradient problémem, zajištění information flow, batch normalization
  + DenseNet
  + MobileNet

**8. Autoencodery a generativní neuronové sítě.**

NI-MVI

**Autoenkodéry**

* Feedworward NN, **na výstupu se snaží napodobit vstup**, který je v průběhu dopředného chodu propuštěn malým bottleneckem
* Kódovací část do malé dimenze = **encoder**
* Část rekonstruující vstup = **decoder**
* Vizualizace v menší dimenzi, učení abstraktních příznaků, komprese, rekonstrukce zašuměných či porušených dat
* **Stacked AE**
  + Řada sparse autoencodérů jdoucích po sobě – každý bere na vstupu výstup předchozího
  + U posledního encodéru odstranění decodéru – výstup jde do klasifikátoru/prediktoru
    - Poslední vrstva trénovaná supervizovaně na nějakou úlohu
  + Každá vrstva naučena na greedy extrakci nejlepších příznaků – víme, že extrahujeme smysluplné příznaky
* Porovnání s PCA
  + PCA je lineární transformace kdežto AE umí modelovat i komplexní nelineární funkce.
  + Příznaky nalezené PCA jsou lineárně nekorelované, protože jsou ortogonální. Autoencoded příznaky mohou být korelované.
  + PCA je rychlejší a výpočetně levnější.
  + Jednovrstvý AE s lineární aktivační funkcí funguje velmi podobně jako PCA.
  + AE je náchylnější na overfitting kvůli velkému množství parametrů.
* **Regularizace**
  + **Dropout, L1 regularizace**
  + Reprezentace nižší dimenze se musí naučit extrahovat důležité příznaky
  + Přidání regularizačního termu – podpora hledání smysluplných příznaků
    - **Sparsity regularisation** – k loss funkci přičteme počet aktivních neuronů
  + **Denoising AE** – vstupní data zašuměny, takže se AE musí naučit podstatné příznaky pro rekonstrukci vstupu

**Generativní neuronové sítě**

* Snaha **namodelovat distribuci** trénovacích dat a **generování dat** ze stejného rozdělení, aby nebylo poznat, že jsou syntetická
  + Generování obrázků, textu, dogenerování dat pro jiné metody
* **Gaussian mixture models (GMM)**
  + **Modelování distribuce** – vážený součet Gaussovských distribucí
  + Trénink váh a parametrů rozdělení pomocí expectation maximization algoritmu
    - EM algoritmy – nejdřív náhodně zvolíme parametry modelu, pak upřesňujeme – k-means
    - Odhad parametrů jednotlivých rozdělení
  + Využití autoenkodéru – zakódujeme jím vstupy do latentního prostoru nízké dimenze, tam použití GMM k naučení distribuce a pomocí naučeného rozdělení generujeme nová data -> dekodérem převod do původních dimenzí
* **Autoregresivní generativní modely**
  + **Predikce budoucího chování** na základě známých historických dat
  + **Forecasting časových řad**
  + Trénovací data musí obsahovat **autokorelaci** – pravidlo určující korelaci signálu a jeho časového zpoždění (např. obrázek – přechod pozadí – pixely se postupně mění – pozvolný přechod = vysoká autokorelace) – můžeme pozorovat **postupné změny**
  + **PixelRNN** – predikce **pixelů** jako časových řad
    - V obr. Jdeme od rohu a predikujeme, nebo jdeme proti sobě ze dvou směrů
    - Díky paměti lze hledat i netriviální sekvence
    - Pomalé, nelze paralelizovat
  + **PixelCNN** – pomocí **konvolučních filtrů** výběr oblastí obrázků, podle kterých se dopočítávají pixely
    - Lze více paralelizovat
    - Postupné generování částí obrázku a konvolucí extrakce příznaků, podle ní pak dobarvení dalších pixelů
    - Super-resolution – nejdřív modelování obecných tvarů a postupně doplnění detailů
* **Variational autoencoders (VAE)**
  + **Autoenkodér**, jehož **rozdělení** latentního prostoru (embeddingů) je **regularizováno** tak, aby mělo **dobré vlastnosti pro generování** nových dat
  + Text

    Description automatically generatedNedělají encoding vstupu na jeden bod latentního prostoru, ale **encodují ho jako distribuci** na latentním prostoru, ze které je pak vzorkován bod – dekódován a porovnán se vstupem.
  + VAE loss zakomponuje dva typy chyb:
    - Reconstruction error původního vstupu (least squares stejně jako AE)
    - Také jak moc se latentní distribuce liší od jednotkového normálního rozdělení
  + Graphical user interface, application

    Description automatically generated with medium confidenceReparametrizace – pro propagaci chyby sítí – samplujeme jednotkové normální rozdělení
* **Generative Adversarial Network (GAN)**
  + Dvě komponety – **Generátor a Diskriminátor** – hrají spolu **zero-sum hru**, kde mají společný loss a jeden se ho snaží minimalizovat a druhý maximalizovat.
  + **Generátor** – z náhodného vektoru v latentním prostoru **generuje upsamplingem** (dekonvoluce) obrázek
  + **Diskriminátor** se snaží **rozeznat skutečné obrázky** od těch uměle vygenerovaných
  + Postupně se takhle navzájem vylepšují ve svých činnostech.
  + Nestabilní trénování, často problém, že jeden model se moc rychle zlepší a druhý ho nemá šanci dohnat.
  + Problém v počtu vygenerovaných objektů, problém s perspektivou,... Protože diskriminátor může hledat jen nějaké featury ale je mu jedno kolik jich je.
  + Problémy:
    - Výstupy generují z náhodného šumu, takže pokud bychom chtěli generovat na základě nějakých konkrétních příznaků, tak nelze jednoduše určit zdrojový šum
    - Diskriminátor se učí jen rozpoznat skutečné a vygenerované vstupy, ale už neřeší, jestli ty vstupy vypadají tak jak mají

**9. Rekurentní neuronové sítě a jejich učení, neuroevoluce.**

NI-MVI

**Rekurentní neuronové sítě**

* **NN s pamětí** – alespoň 1 cyklus, mezi časovými kroky předáván vstup – zachování dřívější informace, větší důraz na nedávné vstupy
  + Aktivace neuronu i v případě, že v daném časovém kroku není žádný vstup
* Překlad, NLP, speech recognition, image captioning
* Diagram

  Description automatically generatedVstupy (např. slova) různé délky, které se vždy zakódují na stejnou velikost pro zpracování stejnou architekturou
* Sdílení parametrů mezi časovými kroky – lze předpokládat, že určité příznaky se mohou vyskytovat v různých částech sekvence
* Trénovací data pro RNN mají podobu sekvence k input-output párů , přičemž výchozí hodnota vnitřního stavu x0 musí být explicitně nastavena (typicky nulový vektor)
* **Backpropagation through time** – BPTT
  + Způsob trénování RNN – podobný backpropagaci, ale **chyby se napříč časovými kroky sčítají** (protože sdílejí parametry)
    - Exploding/vanishing gradient problém
  + Síť se rozbalí do časových kroků (unroll), chyba kumulativně sečtena pro každý krok, poté složení sítě a aktualizace vah
    - Může vést k exploding/vanishing gradient problému
  + Truncated Backpropagation in Time – kumulativní chyby pro podsekvence k kroků, prováděno postupně pro všechny podsekvence
* Diagram

  Description automatically generated**Elmanova síť** – Vanilla RNN
  + Feedforward síť s částečnou rekurencí
  + Architektura – 4 vrstvy – vstup, skrytá vrstva, kontextová vrstva, výstup
  + **Kontextová vrstva** – krátkodobě pamatuje výstupy skryté vrstvy – každý neuron má paměťovou buňku – detekce časově proměnlivých příznaků
  + Využití BPTT – rekurence se rozbalí a počítá se jako dopředná síť
* **Hopfieldova síť** – 1 vrstva n fully connected rekurentních neuronů (každý vstupem každého dalšího, vrstva je zároveň kontext)
  + **Content-addressable memory** – asociativní paměť – do HS si lze ukládat data
    - Asociace– jsou referenční vzory
  + **Hebbovské nesupervizované učení** – když se 2 neurony aktivují spolu, posilují se mezi nimi synapse
    - Pro dvojici neuronů součet násobku vstupů – váha synapse
  + Lze modelovat jako energetickou funkci – pohyb v prostoru chyby sítě
    - Stabilní stavy – vyšší váhy
    - Snaha o minimalizaci energie
  + Omezená kapacita
  + Optimalizace, auto-asociace
* **Echo state networks** (reservoires)
  + Modelování složitých nelineárních dynamických systémů pomocí rekurentního modulu
  + Signál vstupuje do RNN, tam se šíří a způsobuje oscilace
  + Predikce časových řad
  + Různé reakce neuronů na excitaci – nehomogenní, výstupní neurony tuto dynamiku převádí na cílový model dynamiky
  + Minimalizace čtverců chyb
  + A picture containing text, clock

    Description automatically generatedProblém s krátkodobou pamětí (short term memory – způsobeno vanishing gradientem) – při zpracování vstupů problém udržovat informace z dřívějších (proto LSTM a GRU)
* Čisté RNN rychlejší, vhodné pro krátké sekvence a krátkodobé vztahy
* **Hradla** – tensor operace, které se učí, jaké informace přidávat/odebírat ze skrytého stavu
  + Umožňují dlouhodobé závislosti u GRU a LSTM
* Diagram

  Description automatically generated**LSTM** – long short term memory
  + Informace z prvotních časových kroků se může projevit výrazně později
  + **Forget gate** – **které** informace **ponechat a které zahodit**
    - Diagram

      Description automatically generatedVstup v daném kroce a vnitřní stav předchozího projdou sigmoidou, pokud je hodnota blízko 0, informace bude zapomenuta
  + **Input gate** – určuje, **které informace ze současného vstupu budou uloženy** do LSTM paměťové buňky
    - Vnitřní stav a vstup do sigmoidy a tanh, výstupy vynásobeny
  + **Cell state** – **stav** LSTM paměťové buňky
    - A picture containing text, clock

      Description automatically generatedPodle výsledku Forget gate **stav buňky buď zapomenut nebo ponechán**
    - Výsledek sečten s výstupem Input gate – aktuální vstup ovlivněný předchozím stavem
    - Stav buňky přenášen jako kontext do příštího kroku + použit v aktuálním kroku pro výstup
  + **Output gate** – rozhodne **hodnotu následujícího vnitřního stavu**
    - A picture containing text, clock

      Description automatically generatedSoučasný vstup a předchozí vnitřní stav do sigmoidy, nově pozměněný stav buňky do tanh, pak vynásobení – rozh., která část se přenese do příštího vnitřního stavu
* **GRU** – Gated recurrent unit
  + Méně parametrů, než LSTM
  + Nemá stav buňky – používá vnitřní stav k přenosu informací
  + **Update gate** – rozhoduje, **jak moc** předchozí informace **předat** do budoucna
    - Co z minulých informací předávat dál, co ze současného vstupu
  + **Reset gate** – rozhoduje, **jaké** informace už **můžeme zapomenout**
* **Stack RNN** – realizace dlouhodobé paměti pomocí externí paměti – **zásobník, fronta**
  + Elmanova síť, ale místo kontextuálních buněk je zásobník
  + Může být i více paralelních zásobníků

**Neuroevoluce**

* **Neuroevoluce** – užití evolučních algoritmů ke generování neuronových sítí
  + Kombinuje spojitý a diskrétní prostor – váhy a topologie
  + Síť může mít hodně ekvivalentních stavů, třeba prořezávat SP
  + Topology and Weight Evolving Artifical Neural Networks.
* GNARL rozděloval mutace na parametrickou (váhy mutucí gaussovským šumem) a strukturální (přidávání a odebírání neuronů a spojů).
* **SANE** = **Symbiotic, Adaptive Neuro-Evolution**
  + Princip **koevoluce** – v jedné populaci se šlechtí váhy, v další neurony – **společná fitness** funkce
  + Vývoj neuronů jakožto vah spojů vstupujících do neuronu
  + Fitness = fitness 5 nejlepších sítí, ve kterých se daný neuron objevil
  + Blueprint se vyvíjí formou spojování neuronů do sítě – fitness se počítá jako fitness celé sítě
* **NEAT** = **Neuro-Evolution of Augmenting Topologies**
  + **Komplexifikace** – začíná se z malých topologií, evolucí se komplikují a propojují
  + Param. mutace (Gaussovské zašumění) a strukturální (přidávání neuronů a spojů, vypínání s., …)
  + Chromozom = objekt, který má v sobě neurony a spoje
  + Text, letter

    Description automatically generatedProměnlivá délka genomu
  + **Mutace**:
    - **Přidáme spoj** mezi dva nespojené neurony
    - **Rozpojíme spoj a přidáme tam neuron** – propojuje původní dva sousedy
  + Při tvorbě nových neuronů se změna označuje inovačním číslem – pořadí genetických změn
  + **Křížení** – seřadíme genotypy rodičů podle inovačních čísel a postupně slučujeme do potomka
    - Potomek strukturálně komplexnější
* **Niching –** Populace je rozdělena na druhy – sítě **podobných vlastností** se **vyvíjí zvlášť** od jiných druhů
  + Protože v populaci je spoustu sítí různých velikostí, menší se učí rychleji – přidáním genu dočasně snížíme fitness, ale kdybychom pořád chtěli jen vyšší fitness, nebudeme zvětšovat sítě a hledat komplexnější topologie
  + Výpočet podobnosti sítí přes vzdálenostní metriku
  + Dostatečně odlišný jedinec zakládá nový niche
  + **Fitness sharing** – fitness vydělen počtem jedinců v nichi – šance pro zajímavé slabší jedince se vyvinout – zabraňuje přemnožení zatím dobrých řešení
* **Přímé kódování** – všechny linky (spoje) jsou reprezentovány dedikovaným genem
* **Nepřímé kódování** – optimalizuje nějaké DNA (předpis), ze kterého se síť pak postaví
  + HyperNEAT, HyperGP
  + Stavění sítí, které mají nějaké symetrie
* **HyperNEAT**
  + NEAT není použit k vývinu topologie, ale k **vývinu jejího nepřímého kódování** stylem CPPN
    - **CPPN** – síť, která **přiřazuje neuronům váhy**
  + A picture containing text, crossword puzzle

    Description automatically generated**Substrate** – prostor, ve kterém jsou rozmístěny neurony – např. rovina (neurony mají souřadnice) – CPPN dostane souřadnice, podle nějaké funkce určí váhu
    - Definuje možnosti propojování neuronů
    - Lze škálovat substrate density – hustota systému souřadnic
  + Jiný druh – HyperGP – místo neuronové sítě se nepřímo kóduje genetickým programováním
* **Novelty search =** místo abychom hledali něco konkrétního, co už známe, dáme šanci věcem zcela jiným
  + Prostor bývá plný lokálních minim, proto je vhodné se občas vydat zdánlivě odlišnou cestou, abychom dosáhli skutečného optima
  + Každé individuum při evoluci **odměněno**, pokud **objeví něco nového**
  + **Curiosity driven learning** – agent v prostředí získává **odměnu** za to, že se dostane do bodu, **kde předtím nebyl**
* **Covarience Matrix Adaptation Evolution Strategy (CMA-ES)**
  + **Strategie** pro složitou numerickou nelineární nekonvexní **black box optimalizaci**
  + Kombinuje vlastnosti evolučních algoritmů i gradient. technik – dobré překonávání lokálních minim
* **Ant Colony Optimization (ACO)**
  + Mravenci komunikují pomocí **feromonů** – algoritmus se toto snaží napodobit – agenti nanášejí v prostoru tolik „feromonu“, jak dobré je současné řešení
    - Čím více feromonu je někde naneseno, tím více agentů tudy bude chodit.
    - Často uvázne v lokálním minimu, takže často kombinováno s tabu searchem.
    - Často používáno pro **hledání cest v grafech** – nejvíce feromonu se na hrany nanáší, když je daná cesta dlouhá.
* **Particle Swarm Optimization (PSO)**
  + Částice mají **rychlostní vektor** udávající směr pohybu
  + Každý jedinec si pamatuje svoje lokální nejlepší řešení a snaží se k němu vrátit
  + Zároveň si celé **hejno** pamatuje globální nejlepší řešení a všichni míří k tomuto řešení.
  + Lokální a globální optima se balancují pomocí parametrů, postupně tak prohledávají prostor řešení
  + Záleží na počáteční rychlosti (možná oscilace)

**10. Transformery, pozornostní mechanismy, transfer a meta learning.**

NI-MVI

**Transformery**

* Diagram

  Description automatically generated Zpracovávají vstup ve formě sekvenčních dat, ale všechny najednou – mohou být paralelizovány
* Machine translation, text summarization, image description generation, …
* Na rozdíl od RNN taky není tak problematický vanishing gradient.
* **Query** – jeden pro daný attention block
* Pro každou value se query vynásobí odpovídajícím klíčem key – tak získáme hodnotu parametru, kterým value násobíme -> získáme attention
* Poziční kódování – kvůli tomu, že jsou modelu předložena všechna data najednou – není info o pořadí slov
  + Přičte ke každému vektorovému embeddingu **časový vektor**
    - Stejná slova budou zakódována jinak kvůli jiné pozici
* **Self-attention** – porovnání tokenů všech slov ve větě mezi sebou a převážení těchto embeddingů tak, že reprezentace zachycuje kontext
  + **Keys = queries = values**
  + Proces nezávislý na velikosti vstupu, nevyžaduje učení
  + Skalární součin query a key, normalizace výsledku -> váhy pro daný query, 1 váha odpovídá 1 value – pronásobíme, sečteme -> kontextový vektor pro slovo použité jako query
    - Zopakování tak, že se jako query vystřídají všechna slova
  + Parametrizované matice z keys, values a queries – trénovatelný attention
* **Multi head attention** – v rámci vstupů může být několik kontextů (jedno slovo má kontexty s více slovy) – je potřeba zachytit víc pozorností
  + Několik vrstev lineárních vstupů k, v, q, každý má vliv na nezávislé trénování jednoho attentionu
  + Natrénuje několik kontextových vektorů – spojení pomocí concatetation a napojení do dense vrstev
* **Masked attention**
  + Při použití **maskingu** má model přístup jen ke vstupům, které byly před bodem, který se snaží predikovat
  + Aby model **nemohl podvádět** a použít budoucí slova k predikci současného
  + V dekodéru
* **Cross attention** – keys a values generovány z jiného výstupu než queries
  + U dekodéru, keys a values se berou z enkodéru a queries z předchozího výstupu stackovaného dekodéru
* Transformer má enkodér a dekodér které oba používají multi head attention a mohou být stackovány
* V dekodéru se používá ještě masked attention a cross attention
* Z návrhu lze vidět hodně reziduálních spojení aby se potlačil gradient vanishing, takže před vstupem do attentionu se embeddingy ještě odpojí a znovu připojí po výstupu attentionu.
* Na výstupu decoder je pak Linera vrstva, která z výstupu dekóduje vektor velikosti slovníku a softmaxem určí, které slovo je nejpravděpodobnější.
* **BERT**
  + **Masked language model** – self-supervizovaně se učí **doplňovat zamaskovaná (chybějící) slova** do textu
  + **Next sentence prediction** – predikuje, **zda věta B následuje za větou A** v daném kontextu
  + BERT trénován, aby minimalizoval loss obou úkolů
  + 340M parametrů
  + Seq2seq model využívající transformer architekturu se stackem enkodérů
  + Použitelný jako jazykový základ, lze fine-tunovat na konkrétní úkoly
* Problém transformerů – vysoké paměťové nároky, rozšiřitelnost na dlouhé sekvence, složitá implementace, složitost attention vrstev (kvadratická vzhledem k délce sekvence)
* **Performer** – Obchází problém se škálovatelností attention vrstev – místo query a key matice dosazuje **aproximované matice**
  + Lineární složitost vzhledem k délce sekvence
* **Synthesizer** – query a key nahrazeny a attention matice sestavena trénovatelnou sítí, až pak pronásobena values
  + Random synthesizer – náhodná matice
* **Reformer** – lokální sémantické hashování – sousedé z podobných kontextů skončí s podobným hashem -> předpočítání podobností
  + Rychlejší, ale náročnější na paměť

**Pozornostní mechanismy**

* **Attention** mechanismus **napodobuje kognitivní pozornost** – důležité části dat zvýrazněny, méně důležité potlačeny
* **Zpracování textu** – důraz na důležitá a více vypovídající slova
* **Zpracování obrazu** – zaměření na regiony v obraze, které obsahují důležité příznaky – nedůležité je např. pozadí
* Postup:
  + Graphical user interface, text

    Description automatically generated with medium confidenceVstup = hodnoty (values v) – zakódováním slov nebo příznaků v obraze
  + Výstup attentionu – lineární kombinace values s parametry alfa
    - Alfy normalizované na součet 1
  + A dog wearing a garment

    Description automatically generated with medium confidencePro získání alfy – key a query
    - Každá value má s sebou spojený klíč **key**
    - **Query** je jeden pro daný attention mechanismus
  + Funkce, která zkombinuje i-tý key a query tak, že výsledkem je alfa pro i-tou value
    - Dot-product attention – skalární součin query a key, na výsledek aplikace nelineární transformace (třeba tanh), potom normalizace přes softmax
  + **Alfa udává důležitost vstupu**
  + **Attention** mechanismus je třeba trénovat – matice trénovatelných vah, které se lineárně kombinují s values, keys a queries
* **Attention block v dekodéru** – vnitřní stav enkodéru se pošle do attention bloku, projde FC vrstvou a softmax výstup určuje důležitost vstupů do dekodéru
* Pro seq2seq překlady – alignment model – tvorba **kontextového vektoru** – větší kontext = větší alfa
* Případ, kdy values = keys
  + Hledáme **relevanci různých pozic** jedné sekvence
  + Pokud v rámci jedné věty hledáme slova, která odkazují na stejný objekt
* **Table

  Description automatically generated**Qr code

  Description automatically generated**Global/Soft attention** – Bere v potaz celý vstupní prostor, všechny attentiony se předají dál
* **Local/Hard attention** – Bere v potaz jen lokální region (např. patch v obrázku), pravděpodobnost že se zaměří na konkrétní část může být dána její relevancí (attention score)
* Výhoda: model se může soustředit na části vstupu, které nejlépe pomohou splnit úkol
* Nevýhoda: výpočetní složitost

**Meta learning**

* Stavba high-level **systému, který je přenositelný** na různé bottom-level AI problémy
* Ovlivňuje, jak vypadá daný podproblém, aplikovatelný na řadu úloh
* Modely se automatizovaně **učí se učit**
* Neuroevoluce je meta learning:
  + meta-level = evoluční algoritmus modelující topologii sítě
  + bottom-level = samotná neuronová síť
* Hyper Networks:
  + Meta-level = malá síť
  + Bottom-level = velká síť
* Použití LSTM k zapamatování sekvence updatů vah v neur. sítě – určuje nastavování vah bottom-level sítě
* **Optimalizace hyperparametrů (HPO)**
  + Bottom-level AI = model, jehož hyperparametry optimalizujeme
  + Meta-level = algoritmus, který provádí optimalizaci hyperparametrů
  + **Grid search** – systematické prohledávání parameter grid (tabulka parametrů) – kombinace všech povolených hodnot
  + **Random search** – náhodně prohledává parameter grid, překvapivě funkční
  + **Bayesovská optimalizace** hyperparametrů – informované prohledávání, bere v potaz minulé ohodnocení pro vyzkoušené kombinace hyperparametrů
    - Funkce hodnotící výběr hyperparametrů podle přínosnosti
  + **HyperBand = bandit-based** přístup k optimalizaci hyperparametrů
    - Kombinace **alokace zdrojů** a brzkého zastavení k prohledávání parameter gridu

1. Volba většího množství n-tic hyperparametrů
2. Přiřazení n-ticím budget několik iterací
3. Po iteracích ohodnocení n-tic (trénink a ohodnocení modelu), zahození horší poloviny
4. Opět rozdělí budget na iterace mezi n-tice, po doběhnutí zase zahodí – tak iteruje a realokuje prostředky na trénování, dokud nepřežije jen jeden

* **Metadatabáze = databáze pro uchování metadat** o trénování různých problémů
  + Informace, jaké metody se používají na jaké podproblémy a s jakými výsledky – může pak **doporučovat** vhodné algoritmy
  + Podle extrahovaných atributů vypočteme podobné problémy, podle toho volba algoritmu
  + Problém s kompatibilitou algoritmů a problémů – šum, neinformativita
* I pro rekomendační systémy – postupné upřesňování nastavení
* **Model Agnostic Meta Learning (MAML)**
  + Meta learning přístup k **nastavování parametrů nezávisle na modelu a problému**.
  + vybere vzorek ze sady problémů, problémy ohodnotí a provede update parametrů modelu pomocí gradientního sestupu
* **Neural architecture search (NAS)**
  + Automatické učení a evoluce topologií neuronových sítí.
  + **Prohledávání stavového prostoru** – přechody mezi stavy = různé operace jako konvoluční vrstvy, fully connected, pooling, ..., hyperparametry a propojování tak, aby tvořily validní neuronovou síť
  + Stavový prostor bývá velmi velký a ohodnocení stavů (fitness) je výpočetně náročné
  + Příklad – neuroevoluce
* Automated Machine Learning (AutoML)
  + Předzpracování dat – extrakce a předzpracování příznaků
  + Kritéria AutoML: Performance, speed, explainability, simplicity.
* **Few-shot learning**
  + Učení na velmi **malých datasetech**
  + Trénovací data = support set, podle kterého se model musí naučit se učit
  + Využití předchozích znalostí o podobných případech, které už model zná
  + Využití znalostí o samotném učení – pomocí omezení nutíme model generalizovat
  + Využití znalostí o samotných datech a jejich distribuci a variabilitě

**11. Ensemble metody: rozdíl mezi základními metodami (např. Bagging, Boosting, XGBoost).**

NI-ADM

* **Ensemble metody** – více modelů použito k dosažení lepšího prediktivního výkonu v porovnání s použitím jednoho modelu
  + Bagging snižuje rozptyl, boosting snižuje bias

**Bagging**

* **Bagging** = bootstrap aggregating – trénování **několika modelů nezávisle** na sobě a potom **kombinování** jejich predikcí
  + Každý model natrénován na jiné podmnožině dat vytvořené **samplingem s nahrazením** (bootstrapping) z originálního datasetu
  + Výhody:
    - Snižuje rozptyl – průměrováním predikcí z několika modelů
    - Snižuje overfitting – použitím několika modelů trénovaných na různých podmnožinách dat
  + Nevýhody:
    - Zvyšuje složitost – využitím více algoritmů se zvyšuje složitost a výpočetní náročnost
    - Malý vliv na bias
* **Náhodné lesy** – algoritmus, který konstruuje množinu rozhodovacích stromů při tréninku a vyhodí třídu, která je **modus tříd** (klasifikace) nebo **průměr predikce** (regrese) jednotlivých stromů
  + Každý **rozhodovací strom** stavěn na různé podmnožině originálních dat
    - Konstrukce např. pomocí ID3 algoritmu – pro každý příznak se spočítá kritérium (entropie/gini index), pak se zkusí rozdělit podle těch příznaků a počítá se kritérium, vybere se nejlepší rozdělení a opakuje se
  + V každém uzlu stromu se k dělení používá pouze **náhodná podmnožina** příznaků
  + Konečná predikce je **průměr (regrese) nebo většinový hlas (klasifikace)** napříč stromy
  + Zvyšují prediktivní přesnost a kontrolují over-fitting zavedením náhody do ensemblu
    - Rozptyl se sníží bez zvýšení biasu
  + **Hyperparametry**:
    - Počet stromů
    - Velikost bootstrapu
    - Počet příznaků

**Boosting**

* Kombinuje několik weak learnerů, které vytvoří strong learner
  + **Weak learner** – algoritmus strojového učení, který klasifikuje s přesností o něco málo lepší než náhodné hádání
    - Decision stumps, naivní bayes, k-nejbližších sousedů
  + **Strong learner** – má nízkou míru chyby
  + Weak learneři se trénují v sekvenci, každá se snaží napravit předchůdce
* **AdaBoost** = Adaptive boosting
  + Speciální případ Gradient boostingu
  + Přiřazuje stejné váhy všem vzorkům a vybírá slabý klasifikátor, který minimalizuje chybu
  + Pro evaluaci prvního learnera AdaBoost zvýší váhy špatně klasifikovaným vzorkům, aby tvořily větší část trénovací množiny pro následující klasifikátor
  + Proces se opakuje, pokaždé přiřadí větší váhy špatně klasifikovaným vzorkům
  + Závěrečná predikce je **vážený hlas** (v klasifikaci) nebo **vážená suma** (v regresi) predikcí tvořených jednotlivými learnery
  + AdaBoost je **adaptivní** ve smyslu že následující weak lerneři jsou upravování ve prospěch instancí špatně klasifikovaných předchozími kroky
    - Cíl je nastavit váhy a trénovat data tak, aby správně předpovídali neobvyklé pozorování
  + Někdy se používají **Decision stumps** – jako stromy v rozhodovacím lese, ale mají jen jeden uzel a dva listy
  + Algoritmus:
    - **Inicializace vah** (stejné hodnoty)
    - **Postaví se weak learner** – trénink základního modelu
    - **Výpočet chyby** – suma vah špatně klasifikovaných bodů
    - **Výpočet důležitosti** learneru – na základě chyby – menší chyba značí větší důležitost
    - **Update vah** – zvětší se váhy špatně klasifikovaným bodům
    - To se **opakuje** dokud se chyba nepřestane zlepšovat
    - **Formace finálního klasifikátoru** – vážený hlas mezi slabými klasifikátory
  + Výhody:
    - **Všestrannost** – je robustní a všestranná metoda, pro mnoho praktických aplikací
    - **Komplexní klasifikace** – dobře zvládá špatně klasifikovatelné body z komplexních problémů
    - **Přesnost**
  + Nevýhody
    - Riziko **přeučení** – hlavně na datasetech s vysokým šumem, ale dá se zlepšit regularizací
    - **Složitost** výběru a ladění – výběr slabého klasifikatorů a ladění hyperparametrů vyžaduje expertízu
* **XGBoost**
  + **Spojitá cílová proměnná** – může být pravděpodobnost, že Y patří do třídy (podobně jako v logistické regresi)
  + Závěrečná predikce dána **weak learnery**– hyperparametr je dané číslo a weak learneři rozhodovací stromy
  + Během trénovací fáze se weak learneři postupně konstruují, aby se zlepšil ensemble model:

pro

* + Závěrečný model je
  + **Ztrátová funkce**  – měří kvalitu predikce
    - Pro gradient boosting obecně musí mít všude první derivace, a v **XGBoostu musí mít všude druhé derivace**
    - Derivace existují pro obvyklou volbu čtverce reziuí:
  + Obsah obrázku text, Písmo, bílé, kaligrafie

    Popis byl vytvořen automatickyCíl trénovacího procesu je minimalizovat **objective funkci**

Obsah obrázku Písmo, bílé, text, typografie

Popis byl vytvořen automatickyje predikce pro i-tý datový bod a N počet trénovacích datových bodl

* + **Regularizace** – pro předejití overfittingu – s ní je objective funkce:
    - je rostoucí **funkce složitosti stromu**
      * Složitost se dá měrit hloubkou, počtem listů, součtem vah listů, …
  + Obsah obrázku Písmo, text, rukopis, bílé

    Popis byl vytvořen automatickyMinimalizujeme
    - Podle volby ztrátové funkce to nemusí být možné
    - Zjednodušení – **aproximace** pomocí **Taylorova polynomu**
      * V XGBoostu polynom druhého stupně – aproximace v bodě



* + - Cílem je **minimalizace** ne , ale **„délky kroku“ ve směru daného derivací** – gradientu, kvůli tomu jsou to gradientové metody duh
    - Dalšími úpravami můžeme dostat míru kvality stromu a předpis pro váhy
      * Stále NP-těžké konstruování stromů, lze použít greedy algoritmus jako ID3

**12. Jádrové metody: jádrová regrese, bázové funkce, Support Vector Machine (SVM): separabilní a neseparabilní případ.**

NI-ADM

**Připomenutí základů**

* **Lineární regrese –** vysvětlovaná proměnná a vektory příznaků :
  + Vektor vah a náhodná proměnná
  + Odhadujeme vektor vah a pomocí něj predikce :
* **OLS – ordinary least squares**
  + Při tréninku minimalizace reziduální sumy čtverců
  + Minimum pomocí **normální rovnice**: (odpovídá )
  + Pokud je regulární, existuje právě jedno řešení:
* **Hřebenová regrese**
  + Minimalizujeme regularizovaný reziduální součet čtverců:
    - je prostě debilní parametr
  + Matice - na diagonále kromě 1. prvku jedničky, jinak všude nuly
  + To nám předělává normální rovnici a nové řešení (a zjistíme, že pro máme normální OLS)
    - (to co odčítáme je regulární pro nezáporné )

**Lineární bázová expanze**

* **Li**

**Jádrový trik**

* **Duální reprezentace**
* **Gramova matice**
* **Jádrový trik**

**Jádro**

**Jádrové stroje**

**Discriminant function**

**Large margin principle**

**Support vector machines**

**13. Algoritmy pro doporučování: základní přístupy a způsob vyhodnocení kvality, faktorizační metody pro doporučování.**

NI-ADM

**Základní principy**

* **Interakce uživatele a položky** – systémy berou v potaz historické interakce mezi uživateli a položkami
  + Explicitní – hodnonocení filmů, implicitní – historie nákupů
* **Hodnocení podobnosti** – založeny na principu podobnosti – uživatelé se sdílenými zájmy pravděpodobně předvádějí stejné chování, a položky, které jsou často vybírány společně jsou nejspíš nějak spojeny
* **Personalizace** – systémy nabízejí personalizované návrhy spíš než univerzální množinu doporučení, na základně jednotlivých preferencí
* Typy úloh:
  + **Kolaborativní filtrování** – predikce o uživateli na základě sběru informací od hodně uživatelů
    - Pokud se dva uživatelé shodnou na jedné věci, nejspíš se shodnou i na dalších
  + **Obsahové filtrování** – systém doporučuje položky podobné tomu, co si uživatel v minulosti oblíbil, v závislosti na vlastnostech položky
  + **Hybridní metody** – kombinace kolaborativního a obsahového filtrování
  + **Demografické doporučovací metody** – podle demografiky uživatele se doporučuje
  + **Utilitní doporučovací úlohy** – tvorba utilitní funkce pro každého uživatele
* **k nejbližších sousedů** pro doporučovací systémy
  + **User-based** kolaborativní filtrování – kNN identifikuje uživatele podobné danému uživateli
    - Hodnotí nejpodobnějších uživatelů a jejich preference použije k doporučování
  + **Item-based** kolaborativní filtrování – kNN identifikuje položky podobné těm, o které uživatel v minulosti projevil zájem
    - Systém doporučí položky nejpodobnější těm, které uživatel hodnotil kladně
    - Výběr může prudce ovlivnit kvalitu doporučování, volí se empiricky vzkledem k výkonu systému
* **Hodnocení**
  + **Precision** – kolik z doporučených ho skutečně zajímá
  + **Recall** – kolik z položek, které ho zajímají, jsme mu doporučili
  + F1 skóre, AUC (plocha pod křivkou ROC), RMSE pro spojité hodnoty
  + **Dodat detaily**

**Faktorizační metody pro doporučování**

* **Maticová faktorizace** – rozkládá velkou, často řídkou user-item interaction matici na dvě menší matice
  + **Odvozené matice** ukládají latentní atributy jak uživatelů, tak položek, a zjednodušují tak jejich komplexní vztah
  + Např. doporučování filmů – latentní faktory můžou být preference žánru, režiséra nebo herce a korelace filmu s těmito faktory – predikce uživatelova hodnocení filmu může být zobrazena pomocí skalárního součinu vektoru latentních faktorů pro toho uživatela a položku
  + **Optimalizace** v maticové faktorizaci
    - Chceme **minimalizovat rozpor** mezi původní a rekonstruovanou maticí hodnocení
    - Obsah obrázku Písmo, text, bílé, kaligrafie

      Popis byl vytvořen automatickyRozptyl měřen pomocí ztrátové funkce jako **MSE**:
    - Často pomocí **stochastického gradientního sestupu** a **alterujících nejmenších čtverců** (ALS) za použití regularizace pro předejití overfittingu
  + **Řešení řídkosti** – uživatel obvykle interaguje jen s malou podmnožinou položek – maticová faktorizace to řeší zobrazením uživatelského hodnocení na všechny položky, včetně těch nepřítomných v trénovacích datech, a doplní tak nepozorované preference uživatele
* **Alterující nejmenší čtverce (ALS)** – technika zaměřená na minimalizaci mezery mezi pozorovanými a predikovanými hodnoceními
  + Alteruje mezi opravami uživatelské matice a položkové matice a zároveň optimalizuje tu druhou
  + **ALS algoritmus**:
    - Inicializace uživatelské (U) a položkové (V) matice náhodně
    - Opakování do konvergence:
      * Oprava V a minimalizace ztrátové funkce U
      * Oprava U a redukce ztrátové funkce V
  + Může zahrnovat **implicitní zpětná vazba** – kliknutí, historie nákupů, historie prohlížení
    - Nutné modifikace ztrátové funkce, aby byly zahrnuty konfidenční hladiny odvozené z množství interakce mezi uživatelem a položkami
    - Implicitní feedback = informace o preferencích, které nelze získat z explicitních akcí, jako jsou třeba hodnocení nebo recenze
* **Modelování implicitní zpětné vazby**
  + Potřeba když nejsou k dispozici explicitní informace (obvykle vzácné) – použití historie prohlížení apod.
    - V maticové faktorizaci je **přítomnost interakce** interpretovaná jako pozitivní signál, zatímco absence znamená nedostatek informace, ne přítomnost nezájmu
    - Obsah obrázku text, Písmo, bílé, algebra

      Popis byl vytvořen automatickyNapř. matice P reprezentuje interakční matici a matice C indikuje konfidenční hladiny interakcí:
  + Metody kolaborativního filtrování jako **Vážená Regularizovaná Maticová Faktorizace** (WRMF) řídí implicitní feedback přiřazením různých konfidenčních hladin pozorovaným a nepozorovaným interakcím
    - Obsah obrázku Písmo, bílé, kaligrafie, rukopis

      Popis byl vytvořen automatickyOptimalizační problém:
* Omezení a výzvy ALS
  + **Cold Start** – ALS se musí potýkat s novými uživateli nebo položkami, které nemají historii dat o interakci
  + **Extrémní řídkost** – výkon ALS může klesnout, když je interakční matice příliš řídká
  + **Problém se škálováním** – příliš vysoký výkon u velkých systémů kvůli potřebě maticové inverze
  + I přesto je ALS důležitá metoda díky efektivnosti a adaptabilitě

**14. Principy bayesovského modelování – pojmy model, apriorní a aposteriorní distribuce. Exponenciální třída distribucí, konjugovaná apriorna a jejich význam v bayesovském odhadu. Příklad konjugovaného apriorna.**

NI-BML

**15. Stavové modely: rovnice pro vývoj stavu a rovnice měření, rozdíly mezi nimi. Bayesovský sekvenční odhad stavových modelůa jejich vliv na apriorní distribuci (znalost). Možnosti odhadu stavů v případě nelinearity (pouze vyjmenovat).**

NI-BML

**16. Rejection sampling (RS) a importance sampling (IS): důvody používání RS a IS, jejich základní principy a rozdíly, efektivita práce se vzorky. Stanovení vah v IS a možnosti jejich normování.**

NI-BML

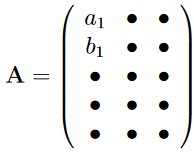
**17. QR rozklad: metody výpočtu, použití při výpočtu odhadu metodou nejmenších čtverců, QR algoritmus pro hledání vlastních čísel.**

NI-PON

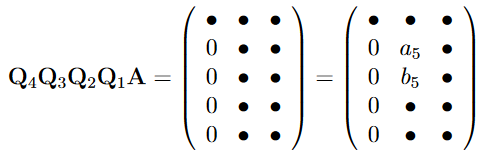
**Givensovy rotace**

* **QR rozklad** = přepis matice A na součin ortogonální matice Q a horní trojúhelníkové matice R
* **Ortogonální matice** = matice, jejíž sloupce jsou navzájem ortogonální a mají jednotkovou velikost
* Platí – matice R vznikne ze stejně velké matice A vynásobením ortogonální , která vynuluje v matici A všechny složky matice A pod diagonálou
  + Složité nacházíme ortogonální matice , které postupně zvětšují počet nul pod diagonálou A:
    - má aspoň o 1 nulu pod diagonálou víc než
    - má aspoň o 1 nulu pod diagonálou víc než
    - …
    - má pod diagonálou samé 0 a je rovna R
  + Tzn. Hledáme ortogonální matice – součin o. matic je o. matice
* **Givensovy rotace** – matice jsou jednoduché – jen na 4 místech se liší od jednotkové matice – rotace 2 složek vektoru, roztažení rotace ve 2 dimenzích
  + Hledáme ortogonální matici , která otočí nenulový vektor tak, že výsledný vektor leží na ose x 2. složka je 0
  + 1. sloupec matice S:
  + Celá matice S:
  + Platí vlastnost rotace na osu znamená, že musí pro vektor platit:
    - 1. složka je , protože výsledný vector musí mít stejnou normu jako původní
      * Otáčíme na kladnou nebo zápornou část osy x, podle toho znaménko
  + podmínka na parametry : , což vede (i z definice) na soustavu:

,

* **QR rozklad pomocí rotací** – hledáme QR rozklad a chceme matici, která vynuluje
  + Obsah obrázku přepěťová ochrana

    Popis byl vytvořen automatickyVýpočet parametrů podle vzorců výše získáme ortogonální , která vynuluje :
    - Oranžová = prvek byl vynásobením změněn
  + Obsah obrázku přepěťová ochrana

    Popis byl vytvořen automaticky se střední mírou spolehlivostiV provedeme zase stejný výpočet parametrů a podle toho zvolíme
  + Analogicky – vynulování prvků :
  + Obsah obrázku Písmo, text

    Popis byl vytvořen automaticky1. sloupec hotový – v 2. sloupci nulujeme :
    - mění jen 2. a 3. řádek – v 1. sloupci jsou 0, takže nevyrobíme ne0 tam, kde už byly
  + Obsah obrázku text, Písmo

    Popis byl vytvořen automatickyPostupně nulujeme další , dokud nedostaneme:



* + To je už ten QR rozklad:
* Celkový odhad složitosti =
* Je to vlastně jako GEM, ale s numerickou stabilitou díky ortogonálním operacím

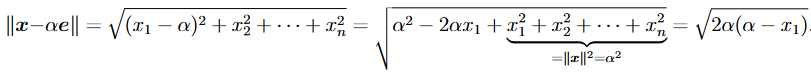
**Householderovy reflexe**

* **Zrcadlení vektoru** v zrcadle, které reprezentuje nadrovina procházející počátkem souřadnice
  + Např. - hledáme 2 rozměrné zrcadlo, které nastavíme vektoru tak, aby odraz ležel na ose x
  + Tím se vynulují všechny složky kromě 1. – rovna normě vektoru
  + Obsah obrázku Písmo, diagram, číslo, snímek obrazovky

    Popis byl vytvořen automatickyHledáme **ortogonální matice P**, pro které:
* **Zrcadlo v n-dimenzionálním prostoru** = nadrovina procházející počátkem souřadnic
  + Popsatelná pomocí normálového vektoru , který je na ni kolmý (ortogonální), předp.
  + Rovina = všechny vektory ortogonální s :
    - Lineární rovnice pro neznámých – množina řešení je podprostor dimenze
* **Zrcadlení**
  + Vzdálenost od = bod na zrcadle nejbližší vektoru = vektor
  + **Zrcadlový obraz**  získáme posunutím o stejnou vzdálenost za zrcadlo:
  + Vzdálenost
    - reprezentuje zrcadlení v zrcadle s normálovým jednotkovým vektorem
    - je rovná svojí transpozici je symetrická je ortogonální
    - Zrcadlení nemění velikost – když aplikujeme zrcadlo 2x, dostaneme se tam, kde jsme začali
* Obsah obrázku text, Písmo, snímek obrazovky, řada/pruh

  Popis byl vytvořen automatickyObsah obrázku kresba, skica, diagram, typografie

  Popis byl vytvořen automatickyVolba **vektoru , aby zrcadlil na osu** odpovídající 1. souřadnici – aby platilo
  + = vektor je násovek vektoru
  + Velikost = 1 musí platit
    - Obsah obrázku Písmo, rukopis, diagram, text

      Popis byl vytvořen automatickyMožnost vybrat si znaménko = zrcadlo má 2 opačné normálové vektory
  + složky :

Obsah obrázku Písmo, text, řada/pruh, číslo

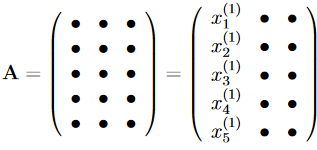
Popis byl vytvořen automaticky

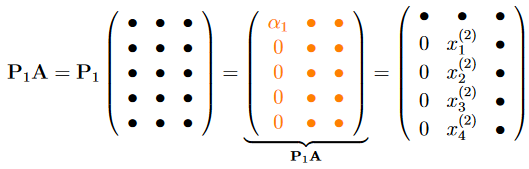
Obsah obrázku Písmo, text, řada/pruh, typografie

Popis byl vytvořen automaticky

* **QR rozklad pomocí zrcadlení** – analogický k ortogonálním rotacím

Obsah obrázku Písmo, design

Popis byl vytvořen automatickyk



* + 2. krok – vektor má jen 4 složky bude taková, aby se k 1. souřadnici chovala jako jednotková matice a zrcadlila pouze zbylé 4
  + Vektor tak, aby matice uvnitř zrcadlila na
  + Obsah obrázku text, diagram

    Popis byl vytvořen automaticky3. krok:
  + 4. krok - – vyrobí poslední 2 nuly analogicky
  + Finální QR rozklad:
  + Matic potřeba tolik, kolik je sloupců matice A =
* Složitost = (# řádků = m, # sloupců = n)

**QR algoritmus**

* **QR algoritmus** = algoritmus pro hledání vlastních čísel
  + Využití toho, že podobné matice mají stejná vlastní čísla
  + Konstruujeme posloupnost matic, které jsou podobné matici, jejíž vlastní čísla hledáme
* **Algoritmus**:
  + Máme čtvercovou a hledáme její vlastní čísla

1. Najdeme QR rozklad:

* Matice si nejsou podobné, ale podobné jsou si a
* Díky ortogonalitě a jsou podobné stejná vlastní čísla

1. Přes 2. konstruuji **posloupnost matic podobných matici** :

konstruujeme posloupnost matic , které jsou si navzájem podobné

* Často konvergence k horní trojúhelníkové matici
* **Zrychlení konvergence** pomocí Hessenbergovy formy
* **Hessenbergova forma matice** – matice je v H. formě, pokud pro platí
  + Matice je navíc **tridiagonální**, jestliže platí

Obsah obrázku kostky

Popis byl vytvořen automaticky se střední mírou spolehlivosti

* K libovolné matici existuje matice v Hessenbergově formě, která je A podobná. Matici podobnosti lze navíc volit ortogonální a symetrickou:
  + Pokud je matice A symetrická, můžeme navíc volit tridiagonální.
  + Matice vznikne součinem matic provádějících Householderovy reflexe
* **Výpočet** singulárních hodnot – vlastních čísel symetrické matice ,

1. Matici převedeme pomocí matic provádějících Householderovy reflexe na podobnou matici , která je tridiagonální.
2. Na matici aplikujeme QR algoritmus.
3. Najdeme QR rozklad , ten můžeme najít pomocí Givensových rotací, které nulují prvků pod diagonálou. Násobení maticí provádějících Givensovu rotaci vždy ovlivňuje pouze 6 prvků matice a je tedy výpočetně nenáročné
4. Posloupnost matic :

, její rozklad označíme

* Matice zůstávají tridiagonální

**18. Maticové faktorizace pomocí SVD, její výpočet, vlastnosti a použití ve strojovém učení: souvislost s metodou hlavních komponent (PCA)**

NI-PON

**SVD rozklad**

* SVD = Singular value decomposition
* **Hlavní myšlenka** SVD rozkladu – máme matici – nehledáme její vlastní čísla, ale vl. č. matice
  + je čtvercová , **pozitivně semidefinitní, symetrická**
  + Symetrická **diagonalizovatelná**, má reálná (nezáporná) vlastní čísla a z vlastních vektorů lze vytvořit ortonormální bázi prostoru
  + , hodnost nesmí být větší než počet sloupců/řádků ().
  + má vlastních čísel. Jelikož hodnost je , je 0 vlastní číslo právě když . V takovém případě má vlastní číslo 0 násobnost .
  + Matice má tedy  **kladných vlastních čísel**, které seřadíme od největšího po nejmenší a označíme :
  + Ke každému vlastnímu číslu umíme najít lineárně nezávislých (+ ortonormálních) vlastních vektorů, kolik je násobnost daného vlastního čísla. Získáme soubor

pro všechna

* + Je-li , tvoří tento soubor bázi . Je-li , můžeme k němu připojit **vlastních vektorů**  příslušejících k vlastnímu číslu 0 a vyrobit tak ortonormální bázi i tak
* **Odvození** SVD rozkladu
  + Pro každé definujeme vektor

soubor vektorů, o kterém platí:

* Je ortonormální:
* Pro každé :

**je vlastní vektor** čtvercové, symetrické a pozitivně semidefinitní matice příslušející jejímu kladnému vlastnímu číslu

a mají stejná kladná vlastní čísla

* + Ze souborů a vyrobíme matice tak, že tyto soubory napíšeme jako jejich sloupce.
  + Jelikož jsou oba soubory ortonormální, jsou obě matice ortogonální:
  + matice, která má na **diagonále postupně vlastní čísla ,** takto seřazená podle velikosti
    - Je-li , **doplníme na diagonálu nuly** (doplnění nejmenšího vlastního čísla 0)
    - Obsah obrázku text, snímek obrazovky, číslo, Písmo

      Popis byl vytvořen automatickyNapř. Pokud (), bude matice :

s tímto značením a jelikož pro a pro dostaneme:

* **Definice SVD rozkladu** (fucking finally) – Buď s hodností . Potom existují ortogonální matice a diagonální matice , která má na diagonále kladná čísla

doplněná příslušným počtem nul, takové, že

Tomuto rozkladu říkáme **SVD rozklad** a číslům pak **singulární hodnoty** matice A.

* + Singulární hodnoty jsou odmocniny vlastních čísel matic a (proto ta nezápornost, AHA)
  + Sloupce matic jsou tvořeny příslušnými vlastními vektory tvořícími ortonormální bázi
* SVD rozklad určen jednoznačně až na:
  + Singulární hodnoty určeny jednoznačně, ale u vl. vektorů si můžeme vybrat znaménko (bo normované na jedničku)
  + Pokud má singulární hodnota více vlastních vektorů (jako vlastní číslo má vyšší násobnost než 1), můžeme libovolně měnit pořadí těchto vektorů v matici . Sloupce v  se pak dopočítají
* Supr trik – v maticích lze uvažovat jen prvních sloupců a řádků, ostatní jsou nulové

lze zrekonstruovat z těchto sloupců

* + Dokonce platí, že podprostor generovaný sloupci matice A = lineární obal souboru

**Aproximace maticí s nižší hodností**

* Když vynásobíme matice a , dostaneme matici – lze využít ke kompresi
* Mějme matici , jaké je nejmenší takové, že existují a pro které

?

* Hodnost součinu nemůže být vyšší, než hodnost jednotlivých matic, a , musí být (kdyby ne, BC by mělo nižší hodnost než A a nefungovalo by to). B a C lze najít pro :

**nejmenší je rovno**

* Máme-li zadané , jaké jsou matice a takové, že jejich **součin je co nejblíže** ?

SVD

* **Vzdálenost** 2 stejně velkých matic = norma jejich rozdílu:
  + Obsah obrázku Písmo, text, snímek obrazovky, diagram

    Popis byl vytvořen automaticky**Frobeniova norma** matice  = suma čtverců odchylek v jednotlivých složkách
    - Obsah obrázku Písmo, design, snímek obrazovky, řada/pruh

      Popis byl vytvořen automatickySouvislost s SVD – je to odmocnina ze součtu kvadrátů singulárních hodnot
* **Eckart-Young-Mirskyho** věta – Mějme matici hodnosti a buď její SVD rozklad. Označme sloupcové matice, které vznikly z  tak, že jsme vzali pouze jejich prvních sloupců. Potom pro každé kladné přirozené číslo je řešením následující úlohy s neznámou

Obsah obrázku text, Písmo, bílé, řada/pruh

Popis byl vytvořen automatickymatice

* Neboli:

největších singulárních hodnot a k nim příslušné sloupce matic definují nejlepší aproximaci maticí o hodnosti

* + Obsah obrázku Písmo, text, bílé, řada/pruh

    Popis byl vytvořen automatickyZbylých singulárních hodnot říká, jak dobrá je to aproximace:
* Obsah obrázku text, Písmo, snímek obrazovky, design

  Popis byl vytvořen automatickyMísto Frobeniovy normy můžeme použít klasickou vektorovou normu:

= **spektrální norma**

* + Rovná se největší singulární hodnotě matice :
* **SOUVISLOST S PCA**

**19. Hladká optimalizace (bez vazeb), spádové metody, volba směru a délky kroku.**

NI-PON

* **Totální derivace**
  + Obsah obrázku Písmo, text, řada/pruh, typografie

    Popis byl vytvořen automatickyMějme funkci , kde je otevřená množina. Řekneme, že je **(totálně) diferencovatelná** v bodě , pokud existuje lineární zobrazení takové, že:
  + Pokud je diferencovatelná v , pak takové lineární zobrazení existuje právě jedno **= (totální) derivace** funkce bodě =nebo .
  + **Jacobiho matice** = matice lineárního zobrazení
  + Mějme funkci , kde a . Pokud existuje a je spojitá na otevřeném okolí bodu pro všechna potom existuje

= **spojitě diferencovatelná** funkce

* **Řetězové plavidlo**
  + Obsah obrázku text, Písmo, bílé, typografie

    Popis byl vytvořen automatickyMějme funkci , kde , a , kde . Je-li funkce diferencovatelná v a funkce diferencovatelná v , potom je funkce diferencovatelná v  a platí:
  + Speciálně pro :
* Obsah obrázku Písmo, text, řada/pruh, typografie

  Popis byl vytvořen automaticky**Gradient** – směr největšího růstu – na derivaci ve směru funkce v bodě můžeme nahlížet jako na derivaci jednorozměrné funkce v bodě 0, . Pak

pro bod 0:

* Obsah obrázku Písmo, text, typografie, Grafika

  Popis byl vytvořen automaticky**Optimalizace** hledáme

pro (dvakrát) spojitě diferencovatelnou

* + Řešení iteračními metodami konstrukce posloupnosti aproximací konvergující k bodu, který je vhodným kandidátem na lokální minimum funkce (typicky nulový gradient)
  + **Trust region** přístup – na okolí bodu vytvoříme aproximaci funkce , označené , a hledáme minimum této aproximace na okolí bodu
    - Další aproximace je
  + **Line search –** aproximaci hledáme ve směru :

kde nebo nějaká aproximace řešení této podúlohy

* Line search – **směr spádu**

kde je zvolený směr a je délka kroku

* + Obecná volba směru = **směr spádu** (descent) =
  + Časté volby směru: **:**
    - Metoda největšího spádu (steepest descent)
    - Newtonova metoda
    - Kvazi-Newtonova metoda
    - Je-li pozitivně definitní, jedná se o spádový směr
* Ideální volba do je minimem funkce

= **exact** line search

* + Výpočetně složité hledání vhodné délky kroku, která zajistí dostatečný pokles funkční hodnoty

= **inexact** line search

* **Armijova podmínka**:

kde

funkční hodnota má ležet **pod zvolenou přímkou**, nalezený krok by ale neměl být příliš malý

* + Obsah obrázku text, Písmo, snímek obrazovky, bílé

    Popis byl vytvořen automatickyKlasický algoritmus:



* **Goldsteinova podmínka**:

kde

funkční hodnota má ležet **mezi zvolenými přímkami**

omezuje se nevhodná možnost, že nalezený krok bude příliš malý

* + Nevýhoda – můžeme minout optimální hodnotu pro
* **Wolfeho podmínky**:
  + Obsah obrázku Písmo, typografie, rukopis, kaligrafie

    Popis byl vytvořen automatickySlabá podmínka:

kde

* + Obsah obrázku Písmo, typografie, text, kaligrafie

    Popis byl vytvořen automatickySilná podmínka:

kde

* + Přidání absolutní hodnoty derivace nebude mít moc velké hodnoty
  + Pro spojitě diferencovatelnou a sdola omezenou vždy existuje splňující Wolfeho podmínky
* **Metoda největšího spádu**
  + a
  + **Největší spád:** (ajo my vlastně jdeme proti směru gradientu, co? PROTI SMĚRU NEJVĚTŠÍHO RŮSTU)
    - ()
  + Obsah obrázku text, Písmo, řada/pruh, snímek obrazovky

    Popis byl vytvořen automatickyPlatí
  + 
  + Obecně vyžaduje vyšší počet kroků ke konvergenci, pro hůř podmíněnou je moc pomalá – při použití nepřesné volby délky kroku nebude rychlejší
* **Newtonova metoda**
  + Obsah obrázku Písmo, text, řada/pruh, typografie

    Popis byl vytvořen automatickyMáme aproximaci a posunutí . Pro platí:
  + Obsah obrázku Písmo, text, typografie, design

    Popis byl vytvořen automatickyOptimální **minimalizace** funkce :
  + Po parciální derivaci 1. rovnice:



* + Pokud není **pozitivně definitní**, pak nemusí být spádový směr!
  + Obsah obrázku text, Písmo, řada/pruh, číslo

    Popis byl vytvořen automaticky
  + Poblíž řešení bude délka kroku splňovat Wolfeho podmínky s výchozí délkou kroku blízko 1 budou nalézat podobné kroky a kvůli výpočetní složitosti se délku nevyplatí optimalizovat
  + **Modifikace Newtonovy metody**
    - Matice (= Hessián) nemusí být pozitivně definitní nemusí existovat inverze
    - **Inverzi chceme**, abychom mohli vyřešit modifikace
    - Nastavení matice tak, aby byla **pozitivně definitní**:
      * Změna záporných vlastních čísel
      * Přičtení kladné diagonální matice
      * Modifikace nějakého rozkladu matice
    - Pokud jsou čísla podmíněnosti matice omezená, pak lze dokázat, že modifikace bude konvergovat
* **Stochastický gradientní sestup** 
  + Obsah obrázku Písmo, číslo, text, řada/pruh

    Popis byl vytvořen automatickyMinimalizovaná funkce ve tvaru sumy:

…ztrátová funkce pro -tý vzorek

… parametry modelu

* + **SGD** (Stochastic gradient descent) – začíná s aproximací a iteruje podle schématu:

… náhodné z uniformního rozdělení

… **míra učení** (learning rate)

* + Výhoda = nízká výpočetní složitost, pro konvexní funkce konvergence skoro jistá
  + **Obsah obrázku text, Písmo, řada/pruh, bílé

    Popis byl vytvořen automatickyModifikace SGD** – většinou modifikace míry učení nebo implementace setrvačnosti (vliv předchozích změn)
    - **Adagrad**:
      * Vektorové operace po složkách
      * , abychom nedělili 0
    - **Momentová metoda**:
      * … faktor zapomínání menší než 1 (typicky 0.9)

**20. Časové řady: aditivní a multiplikativní dekompozice, momenty (střední hodnota, rozptyl, autokovariance). Druhy stacionarity a rozdíl mezi nimi. Základní vlastnosti náhodné procházky a bílého šumu.**

NI-SCR

**Časové řady**

* **Časová řada** = soubor pozorování zíykaných v konkrétních časových okamžicích
  + Buď pravděpodobnostní prostor a množina indexů interpretovaných jako čas. Časovou čadou nazýváme množinu , kde jsou náhodné veličiny z
  + Pokud je z celých čísel, je do řada s diskrétním časem (tyhle tu teď řešíme), jinak se spojitým
* **Variabilita vývoje**
  + **Trend** – dlouhodobý vývoj střední hodnoty
  + **Sezónnost** – periodicky se opakující pravidelný vývoj časové řady
    - Zjištění periody – autokorelační funkce (ACF)
    - Pro analýzu sezónnosti nutno odstranit trend
  + **Cyklické změny** – nepravidelné fluktuace – např. ekonomické cykly – ne pevná perioda
  + Další **nepravidelné fluktuace**
* **Rozložení časové řady na složky**
  + Sloučení cyklických změn a nepravidelných fluktuací do jednoho
    - … pozorovaná veličina v čase t
    - … hodnota trendu
    - … sezónní složka
    - … nevysvětlitelná složka
  + 2 modely:
    - **Aditivní**:
      * Amplituda sezónních složek je cca stejná
    - **Multiplikativní** =
      * S rostoucím trendem se zvyšuje i sezónní amplituda
  + **Výběr modelu** – minimalizace součtu čtverců hodnot **autokorelační funkce** reziduí – říká, jaká míra korelace v zbyla
    - Od pozorované veličiny lze pro různé aplikace odečítat složku trendu nebo sezónnosti
    - Autokorelační koeficient – lineární korelace hodnot časové řady v různých časových okamžicích
      * Umožňuje odhalit opakující se vývoj řady
      * Značení pro časy
* **Náhodný proces** = posloupnost náhodných veličin , kde je z vhodné množiny indexů
* **Momenty** – popisují časové řady
  + **Střední hodnota**:
  + **Variance**:
  + **Autokovariance**:

**Stacionarita**

* **Striktní stacionarita** (silná) – řada je striktně s., pokud sdružená distribuce je stejná, jako sdružená distribuce pro všechna
  + Libovolný posun o čas nemá vliv na sdruženou distribuci a ta tedy závisí jen na časech pro libovolná
* **Slabá stacionarita** – řada je slabě s., pokud je invariantní vůči posunům v čase pouze v rámci momentů rozdělení do druhého řádu:
* Trend řada není stacionární
* Typy procesů podle stacionarity:
  + **Stacionární** – bez trendu a jednotkových kořenů, silně či slabě
    - Pokud je slabě stacionární proces, potom autokorelace závisí pouze na zpoždění mezi časy a :
  + **Trend-stacionární** – stacionární po odstranění lineárního/nelineárního trendu
  + **Stacionární po diferencování** (=procesy s jednotkovým kořenem) – stacionární po tolika diferencích, kolik mají jednotkových kořenů, ale jen za předpokladu, že nemají jiné kořeny uvnitř jednotkové kružnice
  + **Nestacionární**
* Testy stacionarity
  + **ADF** (Augmented Dickey-Fuller)
    - Uvažuje hypotézy:
  + **KPSS** – neparametrický test
    - Hypotézy:
    - Proces je **mean-reverting** = po šoku se vrací ke střední hodnotě
  + Výsledky testů: (významný – V = zamítám)

|  |  |  |
| --- | --- | --- |
| ADF | KPSS | Pravděpodobná vlastnost |
| V | N | stacionární |
| N | V | Nestacionární, ex. Jednotkový kořen |
| N | N | Nedostatek evidence, možná trend-stacionární |
| V | V | Heteroskedasticita, strukturální změna, … |

**Náhodná procházka a bílý šum**

* **Bílý šum**:
  + Normální bílý šum:
  + Zvuková syntéza, generátory náhodných čísel
* **Náhodná procházka** – uvažujeme diskrétní bílý šum . Proces je náhodný proces, pokud:
  + Modelování cen akcií, odhad velikosti webu

**21. Autoregresní modely (AR) a modely klouzavých průměrů (MA): základní vlastnosti modelů/procesů, jejich stacionarita. Zápis AR a MA, včetně zápisu pomocí operátoru zpoždění. Identifikace řádů AR a MA z autokorelačních funkcí a pomocí informačních kritérií.**

NI-SCR

* **Autokorelace**
  + **Lineární korelační koeficient** – určuje míru lineární nezávislosti mezi veličinami
    - a jsou nezávislé (opačně neplatí)
    - = kovariance
  + **Výběrový korelační koeficient** – pomocí výběrových variancí
  + **Parciální korelační koeficient**
    - 2 náhodné veličiny a , mezi kterými existuje závislost
    - a ovlivněny třetí -rozměrnou náhodnou veličinou
    - Pro měření korelace a je potřeba je od vlivu očistit
    - Nalezneme regresní přímky:
      * … nejlepší lineární přiblížení k  a
      * … rezidua očištěná od vlivu
    - Jejich korelační koeficient = **parciální korelační koeficient** mezi a **při daném**  –
    - Projeví se v sezónnosti v časové řadě – vyšší hodnoty korelace v periodách sezón
  + **Autokorelační koeficient** = lineární korelace hodnot časové řady v různých časových okamžicích
    - Umožňuje odhalit opakující se vývoj řady
    - Značení pro časy a
    - Pokud je slabě stacionární proces (= existují časově invariantní první 2 momenty ), potom autokorelace závisí pouze na zpoždění mezi a :
  + **Parciální autokorelační funkce**
    - **Parciální autokorelace zpoždění**
    - (auto)korelace mezi  a s odstraněním lineárního vlivu mezilehlých hodnot
* **Informační kritéria**
  + **AIC – Akaikeho informační kritérium**
    - … počet odhadovaných parametrů
    - … maximální hodnota věrohodnosti při daném modelu
    - Asymptoticky ekvivalentní ke křížové validaci
  + **BIC – Bayesovské informační kritérium**
    - … počet pozorování
  + Hodnotu AIC/BIC chceme minimalizovat
* **Operátor zpoždění –** lag operator nebo – pro zjednodušení zápisu

( je tam -krát)

**Autoregresní modely (AR)**

* Umožňují popis náhodného procesu na základě jeho předchozích realizací
* **Autoregresní model řádu** :
  + … **bílý šum**
  + … vektor **regresních koeficientů**
  + Zápis pomocí **operátoru zpoždění**:
* může být libovolné, ale vyšší řády nemusí dávat smysl – jak hlubokou minulost potřebujeme?
* AR procesy **nemusí být slabě stacionární –** kořeny charakteristické rovnice musí **ležet vně jednotkové kružnice** (tzn. )
* **Odhad parametrů** metodou nejmenších čtverců – odhad regresních koeficientů a
  + návrhová matice, vektor měření, šum
* **Odhad řádu** AR modelu
  + **ACF** – postupně klesá k nule, popř. klesá shora i zdola
    - Má mnoho významných lagů (stacionární AR procesy lze konvertovat do )
  + **PACF** – vrcholy do hodnoty řádu modelu, pak jdou strmě k nule
  + Obecný příklad (při bílém šumu )
    - Abs. hodnoty **ACF** budou zřejmě postupně klesat, protože je přímo ovlivněno atd.
    - Absolutní hodnoty **PACF** budou indikovat silnou korelaci mezi a , ale další hodnoty už by byly 0 kvůli očištění

**Modely klouzavých průměrů (MA)**

* Předchozí hodnota je **konstanta, šumová složka se propaguje –** nebere v potaz předchozí měření
* **Model klouzavých průměrů** řádu :
  + … váha toho, jak se šum propaguje
  + Zápis pomocí operátoru zpoždění:
* Prostřednictvím **bílého šumu** vystihuje **náhodné šoky** – nezávislé a stejně rozdělené
* Veličiny a jsou pro větší než řád modelu nekorelované, korelace a pro řád nenulová
* Prohodíme-li v kovarianci pořadí, změní se znaménko u
* **Odhad** MA modelu je složitý numerická optimalizace
* **Invertibilita MA procesů** – u MA nás zajímá místo stacionarity
  + MA proces 1. Řádu je ekvivalentní AR procesu řádu
  + **Charakteristický polynom** :
  + Jsou-li kořeny char. polynomu vně jednotkové kružnice, potom je MA proces invertibilní -
* **Odhad řádu** MA modelu
  + ACF – vrcholy do hodnoty řádu modelu, pak jdou strmě k 0
  + PACF – mnoho významných lagů (kvůli invertibilitě) – jako ACF u AR modelů
  + Obecný příklad (při bílém šumu )
    - není ovlivněna
    - Absolutní hodnoty **ACF** budou vysoké pro , ostatní nulové (šum iid)
    - Absolutní hodnoty **PACF** budou klesat k 0

**22. Smíšené modely ARIMA: základní vlastnosti modelů/procesů, integrování a diferencování. Zápis ARIMA, včetně zápisu pomocí operátorů zpoždění a diference, speciální případy podle hodnot p, d, q. Problém redundance parametrů.**

NI-SCR

* **Smíšené modely ARMA (p, q)**
  + Neznámé
  + Populární – flexibilní, ale mnoho neznámých
  + Většinou předpokládána normalita šumu
  + Ekonometrie – ceny akcií – MA šokové změny + AR vývoj na základě minulých cen
  + Rozšíření NARMA, ARIMA, SARIMA, VARMA, …
  + Předpoklad = **slabá stacionarita** – co když je porušena?
    - Odstranění trendu
    - Metoda diferencí – pokud nepomůžou 1. diference, pomohou často 2.
      * ARMA model, který tohle dělá, je ARIMA
      * ARIMA (3,2,1) je ARMA (3,1) se dvěma diferencováními
  + Zápis pomocí operátoru zpoždění (bez c):
* **Box-Jenkinsův přístup** k ARMA modelům

1. **Identifikace modelu** – posouzení stacionarity a sezónnosti časové řady, odhalení přítomnosti AR a MA části a jejich řádů (ACF/PACF), popř. transformace pro zajištění stacionarity
2. **Odhad parametrů** – MLE
3. **Ověření modelu** – posouzení nekorelovanosti reziduí a jejich slabé stacionarity v čase (ACF/PACF)

**Smíšené modely ARIMA (p, d, q)**

* **Parametry**:
  + – řád **autoregresní** části (AR)
  + – řád modelu **klouzavých průměrů** (MA)
  + – řád **diferencování**
* Zápis pomocí **operátoru zpoždění** (bez konstanty ):
* **Operátor diference** :
* Diferencování řady **odstranění trendu**
  + Počítáme veličiny
  + Diferencováním náhodné procházky dostaneme gaussovský bílý šum ()
    - Gaussovská náhodná procházka:
* Časové řady s **trendem nebo náhodnou procházkou** jsou vždy **silně pozitivně autokorelované**
  + V ACF vidíme velké korelace
  + V PACF typicky v 1. lagu a blízkou 1
* Časové řady s ACF v 1. lagu s hodnotami -0.5 a méně mohou být **přediferencované**
* Modely s d=1 typicky předpokládají **konstantní průměrný trend** (náhodná procházka s driftem)
* **AR** charakteristika může značit **poddiferencovanost** časové řady, **MA** pak **přediferencovanost**
* **Volba** :
  + , 2. diference málokdy, 3. je výjimka
  + Při nepoužíváme v ARIMA modelu konstantu
* **Role konstanty** v ARIMA modelu:
  + – konstanta zavádí nenulovou střední hodnotu a vyplatí se ji zkusit
  + – konstanta zavádí nenulový „průměrný“ trend, může se vyplatit
  + – konstanta by měla význam „trendu v trendu“, obvykle nechceme
* **Běžné ARIMA modely**:

|  |  |
| --- | --- |
| ARIMA (0, 0, 0) + c | Konstantní model |
| ARIMA (0, 1, 0) | **Model náhodné procházky** |
| ARIMA (0, 1, 0) + c | **Náhodná procházka s driftem** |
| ARIMA (1, 0, 0) + c | AR (1) |
| ARIMA (2, 0, 0) + c | AR (2) |
| ARIMA (1, 1, 0) + c | AR (1) na 1x diferencovaných datech |
| ARIMA (2, 1, 0) + c | AR (2) na 1x diferencovaných datech |
| ARIMA (0, 1, 1) | Jednoduché exponenciální vyhlazování = MA (1) na 1x dif. Datech |
| ARIMA (0, 1, 1) + c | MA (1) na 1x diferencovaných datech s konstantním lineárním trendem |
| ARIMA (1, 1, 2) | Lineární exponenciální vyhlazování s tlumeným trendem |
| ARIMA (0, 2, 2) | Zobecněné lineární exponenciální vyhlazování |
| ARIMA (1, 0, 0) + c | Podle toho, jaké parametry má drift :   1. **bílý šum** 2. **náhodná procházka** 3. **náhodná procházka s driftem** 4. řada oscilující mezi klad. a záp. hodnotami |

* **Obecná pravidla**
  + nebo
  + – složitější modely jsou řídké
  + Koeficienty blízké mohou pracovat proti sobě
  + Obecně volíme co **nejjednodušší** modely
  + Modely lze porovnávat kritérii **AIC, BIC**
  + Můžou existovat i složitější modely porušující pravidla
    - I tak by měly být malé
    - Problém složitých modelů – **redundance parametrů**
      * AR a MA části modelu pracují proti sobě – zvýšíme-li řády u obou, mohou se navzájem vyrušit zvyšuje se složitost :(