Kikiina Statnicová Směška

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)** – $n(x_1 \ldots x_n)$
    - **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: $Precond+(a) \subseteq s \wedge precond-(a) \cap s = \emptyset$
  - **Aplikace** apl. Akce:
    - Odebereme negativní efekty, přidáme pozitivní efekty:
      $$\gamma(s,a) = (s - effect-(a)) \cup (effect+(a))$$
- **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, S_0, g)
  - Plánovací problém SP s přechody: (Σ, s_0, s_g)
    - $\Sigma = (s, A, \gamma)$
    - Sg = množina cílových stavů
- **Plány a řešení**
  - **Plán** $P = (o, s0, g)$ **– posloupnost akcí** $\pi = [a_1, \ldots, a_n]$, kde a_i = instance operátoru z O
  - Plán je řešením P, jestliže je proveditelný z s_0 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 – $\gamma^{-1}(g, a)$
    - Akce a je relevantní pro cíl g, když:
      - $g \cap effects^+(a) \neq \emptyset$
      - $g^+ \cap effects^-(a) = \emptyset$ a $g^- \cap effects^+(a) = \emptyset$
    - Pro relevantní a vzhledem k g definujeme: $\gamma^{-1}(g, a) = (g - effect(a)) \cup precond(a)$

        - 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:
            1. Přinutit b, aby předcházelo a
            2. Přinutit c, aby předcházelo a
            3. 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: s_g dosažitelný ⇔ obražen v grafu d. s.
        - Relaxovaný: s_g dosažitelný => obsažen v plánovacím grafu d. s.
        - Sjednocení atomů úrovně stromu/grafu dos. Do 1 stavu/uzlu
- **Plánovací graf**
    - Střídání stavových (P_0, ...) a akčních (A_1, ...) vrstev
    - Hrany zpřesňují aproximaci oproti jen sjednocení atomů
    - Hrany z P_i do A_{i+1} – ukazují předpoklady akce
    - Hrany z A_i do P_i – ukazují efekty akce
    - Skrz negativní efekty nemažeme
        - $P_0 \subseteq P_1 \subseteq ...$
        - 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 ⇔

            $Effect^-(a) \cap (precond(b) \cup effect^+(b)) = \emptyset$

            $\wedge$

            $Effect^-(b) \cap (precond(a) \cup effect^+(a)) = \emptyset$
        - 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 A_i jsou vzájemně vyloučené (mutex) = jsou závislé, nebo je a mutex s předpokl. b

    - Atomy p a q z P_i jsou vzájemně vyloučené – každá akce a v A_i, že $p \in effect^+(a)$ je mutex s každou akcí b v A_i, že $q \in effect^+(b)$ a v A_i není akce c, že $p, q \in effect^+(c)$
- 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, $X_t = (x_t, y_t, \theta_t)$, posuvná rychlost v_t , rotační ω_t – 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 $P(X_t \mid z_{1:t}, a_{1:t-1})$ reprezentuje pomocí normálního rozd. $N(\mu_t, \Sigma_t)$
    - Funguje s lineárními (f ah) přechodovými a senzorovými modely
    - Přechodový model $X_{t+1} = f(X_t, v_t, \omega_t)$ interpretujeme jako maticovou operaci, senzorový model $z_{t+1} = h(X_t)$ taky
    - Přesvědčení reprezentované jako normální rozdělení $N(\mu_t, \Sigma_t)$ je po lineárních transformacích opět popsáno normálním rozdělením $N(\mu_{t+1}, \Sigma_{t+1})$
- **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 xt odchýlena od požadované $y(t)$, 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 $S' \cup \{x=d\}$ na celé X
      - S' pro Conf\{x} je pak příčina nemožnosti provést $S'(x)=d$ – $Conf_d$ = Conf \ {x} je konfliktní množina pro hodnotu $d \in D(x)$
    - 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 = $\{y_1 \ldots y_n\}$ 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 x_1 a x_2, D = doména
        1. kontrolujeme orientovanou hranu (x_1, x_2)
        2. pro každou hodnotu $d_1 \in D(x_1)$ hledáme **podporu** $d_2 \in D(x_2)$ vzhledem k podm. c
        3. podpora – $d_2 \in D(x_1)$ taková, že (d_1, d_2) ∈ c
        4. nemá-li d_1 podporu v $D(x_2)$, odebereme d_1 z $D(x_1)$
    - Podpora p*ro 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($x_1 \ldots x_k$) – 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

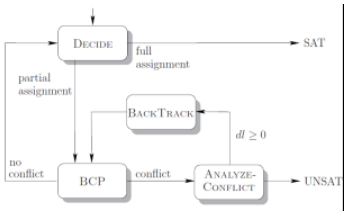
3. Systematické a lokální splňování v logice (DPLL, CDCL, WalkSAT, posílání zpráv). Automatické uvažování, rozhodování v teoriích prvního řádu, obecná rezoluce, princip SAT-modulovaných teorií (SMT). Zpracování přirozeného jazyka.

NI-UMI

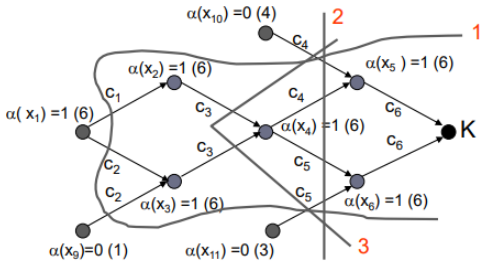
Systematické a lokální splňování v logice

- 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** a_i
    - 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
  - Součásti:
    - **DECIDE** – ohodnotí další neohodnocenou proměnnou
    - **BCP** – jednotková propagace (boolean constraint propagation)
    - **BackTrack**(level) – zruší všechna rozhodnutí na úrovních vyšších než level



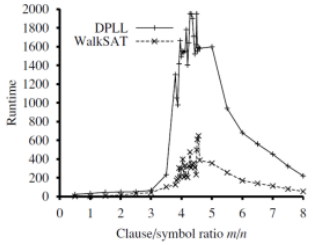
  - Když dojde ke konfliktu, nalezne vhodný řez grafu, z něj vytvoří nogood, ten přidá jako klauzuli
    - **Vhodný řez** – z poslední vrstvy obsahuje pouze prom., u které se rozhodovalo ohodnocení
    - Přidaná klauzule následně vynutí změnu ohodnocení u poslední rozhodované proměnné pomocí UP
    - Dělá **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)



1. $(x_{10} \vee \neg x_1 \vee x_9 \vee x_{11})$
2. $(x_{10} \vee \neg x_4 \vee x_{11})$
3. $(x_{10} \vee \neg x_2 \vee \neg x_3 \vee x_{11})$

    - 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
      (i) Vrchol implikačního grafu různý od K
      (ii) 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
    - Náhodné 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 $\phi(t)$ je konfliktem i pro $\phi(t+1)$

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 $t_1...t_n$ jsou termy, pak $f(t_1...t_n)$ je term
  - **Atomy** – jestliže p je predikátový symbol arity n a $t_1...t_n$ jsou termy, pak $p(t_1...t_n)$ je atomická formule (atom)
  - Formule je slovo (konečná posloupnost symbolů)
    - Atomická formule je formule
    - Jestliže p a q jsou formule, pak $(\neg p)$, $(p \wedge q)$, $(p \vee q)$, $(p \Rightarrow q)$, $(p \Leftrightarrow q)$ jsou formule
    - Jestliže p je formule a x proměnná, $\forall x(p(x))$ a $\exists x(p(x))$ 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) - $T \models \phi$ (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: $f: M \times M \times ... M \to M$
    - Realizace predikátů nad M: $p \subseteq M \times M \times ... M$
- Semi-rozhodování: **rezoluce**
  - Rezoluční metoda – klauzální zamítací dokazování
    - Předp. Dokazování tvrzení C vzhledem k množině axiomů T
    - Uvážíme $T \cup \{\neg C\}$, snažíme se odvodit spor
  - **Rezoluční pravidlo**
    - speciální výroková varianta (řez) – x, y výrokové literály, z výroková proměnná

$$\frac{(x_1 \vee x_2 \vee ... x_n \vee z), (y_1 \vee y \vee ... y_m \vee \neg z)}{(x_1 \vee x_2 \vee ..., \vee x_n \vee y_1 \vee y_2 \vee ... \vee y_m)}$$

    - Obecná varianta pro logiku 1. řádu

$$\frac{(p_1 \vee p_2 \vee ... \vee p_n \vee r), (q_1 \vee q_2 \vee ... \vee q_m \vee \neg s)}{(p_1 \vee p_2 \vee ... \vee p_n \vee q_1 \vee q_2 \vee ... \vee q_m)\theta}$$

      - p, q literály, r,s atomy
      - θ – nejobecnější unifikace r a s s (tedy $r\theta = s\theta$ a každá jiná θ', že $r\theta' = s\theta'$ lze vyjádřit pomocí θ, tedy $\theta' = \theta\rho$)

- 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 $T \models \phi$ 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 $G_{=} = (Var(\phi), E_{=}, E_{\neq})$
  - Vrcholy – proměnné, hrany – $E_{=}$ rovnosti a $E_{\neq}$ nerovnosti
  - Sporná kružnice – cyklus v $G_{=}$, který obsahuje právě jednu nerovnost
- **Zjednodušovací algoritmus** – na vstupu formule ϕ_E 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 $G_{=}^{NP} = (Var(\phi), E_{=} \cup E_{\neq})$ – 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 – $decide_T$ (dokazovací procedura pro konjunktivní fragment teorie T), řešič SATu
  - Spolupráce $decide_T$ a řešiče SATu
    - Řešič vybírá, které literály splnit
    - $Decide_T$ 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
  - $G = (N, T, S, P)$ – terminály, neterminály, počáteční neterminál, konečná množina pravidel
  - **Chomského hierarchie** gramatik a jazyků – rekurzivně spočetné, kontextové, bezk., regulární
  - **Bezkontextová gramatika –** pravidla $A \rightarrow \beta$, A neterminál a β řetězec t. a net.
- **Fáze komunikace**
  - Záměr – sdělit skutečnost P
  - Generování – přesvědčivého sdělené o skutečnosti P → 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 $X \rightarrow YZ$ nebo $X \rightarrow w$
- **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 $\{podmínku\}$ – 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 e z angličtiny do francouzštiny f
  - $P(e|f)$ – **překladový model** – jak pravděpodobné je, že je e anglickým překladem dané francouzské věty f
  - $P(f|e)$ – jak pravděpodobné je, že f je francouzským překladem dané anglické věty e
  - $P(f)$ – **jazykový model** – jak pravděpodobné je, že f je věta ve francouzštině
  - Hledáme řetězec f^*

$$f^* = argmax P(f|e) = argmax P(e|f)P(f)$$

  - Lze počítat podle $P(e|f)P(f)$ – máme-li dobrý jazykový model pro $P(f)$, nebo přímo $P(f|e)$
  - Průběh statistického překladu
    - Rozdělíme vstupní anglickou větu e 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 x_i pro predikci výsledku proměnné Y?
  - Irelevantní příznak X_i: $P(X_i \mid Y = 1) = P(X_i \mid Y = -1)$
  - **T-test** – normálně rozdělené třídy, stejný rozptyl σ^2 neznámý, odhad z dat jako σ^2_{within}
    - Nulová hypotéza: $\mu^+ = \mu^-$
    - Pokud je nul. hyp. pravdivá: $t = (\mu^+ - \mu^-)/(\sigma_{within}\sqrt{1/m^+ + 1/m^-}) \sim$ **Student**$(m^+ + m^- - 2$ d.f.$)$
  - **Korelace** – vzájemný lineární vztah mezi proměnnými
    - +1 přímá závislost, -1 nepřímá závislost
    - **Pearsonův korelační koeficient**

      $$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E\{(X-\mu_X)(Y-\mu_Y)\}}{\sigma_X \sigma_Y}$$

    - **Korelační a kovariační matice**

      Variance (σ^2)

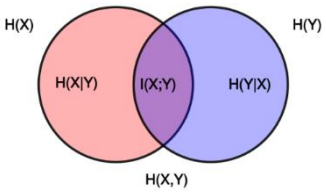
      $$\rho = \begin{bmatrix} 1 & \rho(X_1,X_2) & \cdots & \rho(X_1,X_n) \\ \rho(X_2,X_1) & 1 & & \rho(X_2,X_n) \\ \vdots & & \ddots & \vdots \\ \rho(X_n,X_1) & \rho(X_n,X_2) & \cdots & 1 \end{bmatrix} \quad C = \begin{bmatrix} D(X_1) & C(X_1,X_2) & \cdots & C(X_1,X_n) \\ C(X_2,X_1) & D(X_2) & & C(X_2,X_n) \\ \vdots & & \ddots & \vdots \\ C(X_n,X_1) & C(X_n,X_2) & \cdots & D(X_n) \end{bmatrix}$$

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

        $$\rho = 1 - \frac{6\sum_i (p_i - q_i)^2}{n(n^2-1)}$$

  - **Entropie** – předpokládané množství informace v náhodné proměnné

    $$H = -\sum_i P_i \log P_i$$

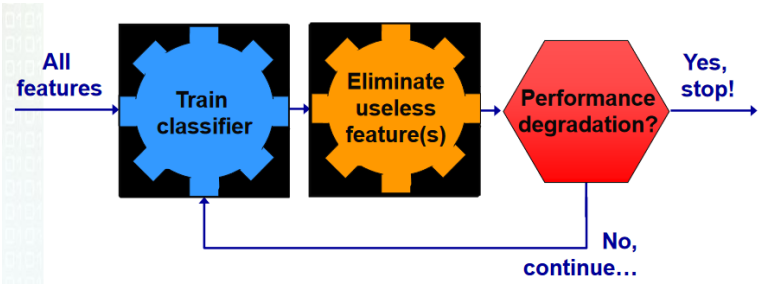
  - **Univarietní závislost**
    - Nezávislost: $P(X,Y) = P(X)P(Y)$
    - **Vzájemná informace** – hodnota vzájemné závislosti mezi dvěma náhodnými proměnnými
      - Kvantifikuje množství informace získané o jedné proměnné zkoumáním jiné náhodné proměnné

        $$\mathrm{MI}(X, Y) = \int \mathrm{P}(X,Y) \log \frac{\mathrm{P}(X,Y)}{\mathrm{P}(X)\mathrm{P}(Y)} \, dX \, dY$$

      - 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

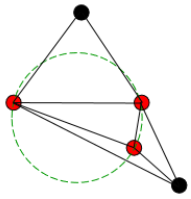
- **Filter metody** – hodnotí příznaky nebo jejich podmnožiny nezávisle na klasifikátoru
  - *Kritéria* – hodnocení relevance příznaků
  - *Hledání* – seřazení příznaků
  - *Hodnocení* – statistické testy
  - Robustní výsledky, ale nemusí vybrat úplně nejlepší příznaky
- **Wrapper metody** – používají 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 k 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
- **Embedded 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

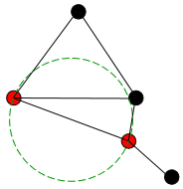
Selektivní/adaptivní metody redukce počtu instancí

- **Condensed Nearest Neighbour** (CNN) – inkrementální, závislá na pořadí, ani minimální ani decision boundary konzistentní
  - $O(n^3)$ 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“
  - Grafická reprezentace dat – zobrazují, jak jsou prvky propojeny na základě vzdáleností/podobnosti
  - **Delaunay**
    - **Delaunayho triangulace** – duální graf Voronoiho diagramu?
      - Voronoi diagram – rozděluje rovinu na oblasti na základě vzdálenosti k bodům (zdroje) – každá oblast obsahuje body, které jsou blíž k danému zdroji než k jakémukoliv jinému
      - Vrcholy Voronoi diagramu jsou rohy trojúhelníků v D.T.

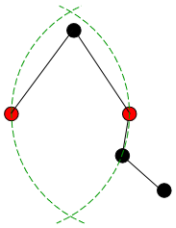


    - Tři body jsou si sousedy, pokud kružnice jim opsaná neobsahuje žádné jiné body
    - Voronoi editing: ponechá body, jejichž sousedi jsou z jiné třídy
    - Rozhodovací hranice je stejná
    - Konzervativní podmnožina, nechává si body navíc, drahá na výpočet ve vysokých dimenzích

- Gabriel
  - Podmnožina delaunayho triangulace
  - Body jsou sousedi, pouze pokud je jejich sféra vlivu prázdná
  - Nezachovává 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
  - Hrany, které spojují vrcholy, které sdílejí společný okraj v diagramu
  - Tomkův spoj
    - E_i, E_j patří do jiných tříd, $d(E_i, E_j)$ je vzdálenost mezi nimi
    - Pár (E_i, E_j) je Tomkův spoj, pokud neexistuje takové E_l, že $d(E_i, E_l) < d(E_i, E_j)$ nebo $d(E_j, E_l) < d(E_i, E_j)$

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
  - k 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 $(Q1 - 1.5 * IQR, Q3 + 1.5 * IQR)$
    - 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 k 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) = průměrná 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
    - E_i, E_j patří do jiných tříd, $d(E_i, E_j)$ je vzdálenost mezi nimi
    - $A(E_i, E_j)$ je Tomkův spoj, pokud neexistuje takové E_l, že $d(E_i, E_l) < d(E_i, E_j)$ nebo $d(E_j, E_l) < d(E_i, E_j)$
  - 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 k nejbližších sousedů z menšinové třídy
    - V závislosti na množství over-samplingu se sousedi z k 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 liší 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 n dimenzního prostoru do k dimenzního prostoru ($k < n$) při zachování nějaké formy informace (např. vzdálenost)
  - Výhoda: vyloučení redundantní informace (korelace)
  - Nevýhoda: fyzický význam nových atributů těžké interpretovat

Lineární projekce

- Jednoduché na výpočet:

$$Y = U \cdot X \quad (b_i = u_i^t a_i)$$

k x l k x n n x l (k<<n)

- **Náhodná projekce** (mapování) – původní d-dimenzní data se projektují do k-dimenzního podprostoru ($k \ll d$) za využití náhodné $k \times d$ matice R, jejíž sloupce mají jednotkové délky
  - Využívá maticovou notaci kde $X_{d \times N}$ je původní množina N d-dimenzních pozorování,

$$X_{k \times N}^{RP} = R_{k \times d} X_{d \times N}$$

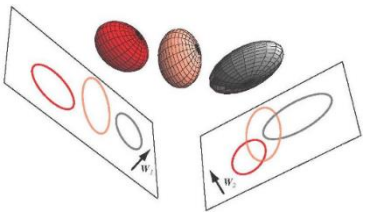
    je projekce dat dov nižšího k-dimenzního podprostoru
  - **Johnson-Lindenstrauss lemma**: pokud jsou body ve vektorovém prostoru projektovány na náhodně vybraný podporostor vhodně velké dimenze, pak jsou vzdálenosti mezi korespondujícími body (v originálním a novém prostoru) zhruba zachovány
  - Technicky vzato ta transformace není projekce, protože R není nutně ortogonální
- **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

$$\min \| x - \hat{x} \| \quad \text{(reconstruction error)}$$

original data reconstructed data (after inverse transformation)

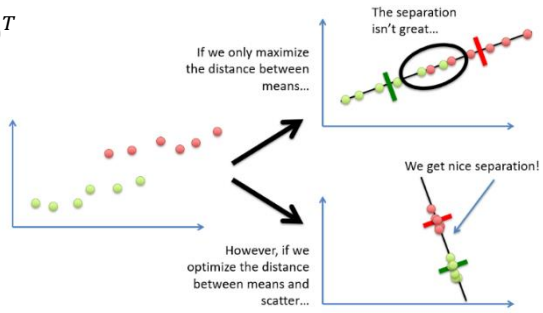
    - PCA zachovává co nejvíce informace (variace) je možno:
  - „Nejlepší“ prostor nízké dimenze má **střed** v průměru vzorku a má **směry** rozhodnuté „nejlepšími“ vektory vlastních čísel kovarianční matice dat x
    - „Nejlepšími“ vektory vlastních čísel se myslí ty odpovídající **největším vlastním 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
    - μ_i průměrový vektor třídy i, $i = 1,2, \dots, C$

- M_i počet vzorků ve třídě i
- $M = \sum_{i=0}^{C} M_i$ celkový počet vzorků
- y_j matice velikostí $C \times M_i$ obsahující vzorky z jednotlivých tříd
- Scatter matice **uvnitř třídy**:



$$S_w = \sum_{i=1}^{C} \sum_{j=1}^{M_i} (y_j - \mu_i)(y_j - \mu_i)^T$$

- Scatter matice **mezi třídami**:



$$S_b = \sum_{i=1}^{C} (\mu_i - \mu)\,(\mu_i - \mu)^T$$

  - S_b má hodnost maximálně $C - 1$
  - Každá pod-matice má hodnost 1 nebo méně – vnější součin dvou vektorů
  - $\mu = \frac{1}{C}\sum_{i=1}^{C} \mu_i$ průměr celého datasetu
- $y = U^T x, U$ je projekční matice
- LDA počítá transformaci, která **maximalizuje scatter mezi třídami** a minimalizuje scatter uvnitř tříd:

$$\max \frac{|U^T S_b U|}{|U^T S_w U|} = \max \frac{|\tilde{S}_b|}{|\tilde{S}_w|}$$ products of eigenvalues

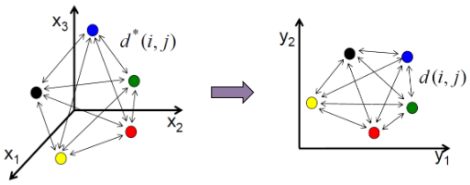
  - $\widetilde{S_b}, \widetilde{S_w}$ … scatter matice projektovaných dat y
- **LDA lineární transformace**:
  - Řešení LDA je dáno vlastními vektory obecného problému vlastních vektorů:

$$S_B u_k = \lambda_k S_w u_k$$

  - Lineární transformace je dána maticí U, jejíž sloupce jsou vlastní vektory problému výše
    - Souřadnice v novém prostoru: $\begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_K \end{bmatrix} = \begin{bmatrix} u_1^T \\ u_2^T \\ \dots \\ u_K^T \end{bmatrix} (x - \boldsymbol{\mu}) = U^T (x - \boldsymbol{\mu})$
  - Jelikož S_b má hodnost nejvýše $C - 1$, je maximální počet vlastních vektorů s nenulovými vlastními čísly $C - 1$ (takže maximální dimenze podprostoru je $C - 1$)
- Při malé trénovací množině je PCA výhodnější než LDA, LDA zas lepší na větších a reprezentativních datech

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 d_{ij}^*, a vzdálenost jejich proj. je d_{ij}
  - Sammonova projekce minimalizuje následující chybovou funkci – Sammon's stress neboli **Sammonova chyba**:

$$E = \frac{1}{\sum_{i<j} d_{ij}^*} \sum_{i<j} \frac{(d_{ij}^* - d_{ij})^2}{d_{ij}^*}$$

  - 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 $vnitřní\ potenciál\ >=\ 0$,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
     a. 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
     b. 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:
    $$L(Y,p)\ =\ -log(p_y).$$
  - 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
  - Pro použití gradientního sestupu pro učení vah musíme najít parciální derivaci Z.F. vůči každé váze – pomocí chain rule pro derivace složených funkcí
    $$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$
  - **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
    4. Gradientní sestup regulovaný learning ratem
    5. 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. Opakování 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

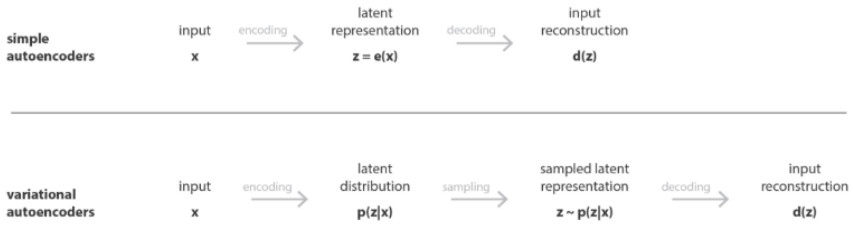
Autoenkodéry

- 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

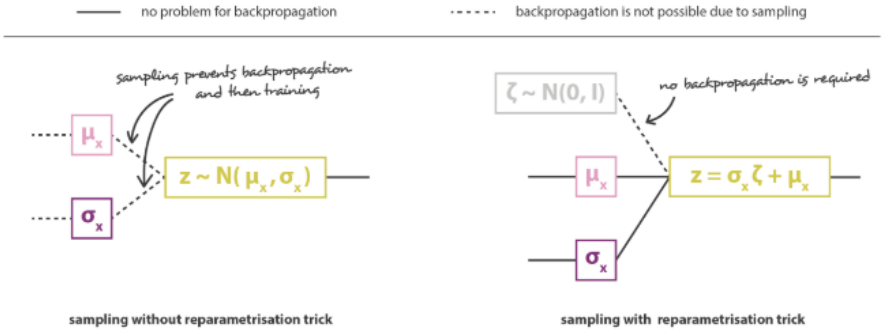
Generativní neuronové sítě

- 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
    - Nedělají 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



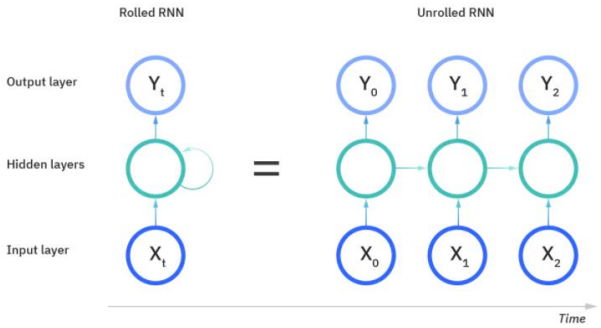
    - Reparametrizace – pro propagaci chyby sítí – samplujeme jednotkové normální rozdělení
- Generative Adversarial Network (GAN)
    - Dvě komponety – **Generátor a Diskriminátor** – hrají spolu **zero-sum hru**, kde mají společný loss a jeden se ho snaží minimalizovat a druhý maximalizovat.
    - **Generátor** – 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.

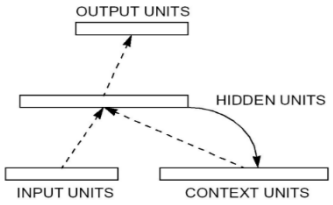
NI-MVI

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
- Vstupy (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 $< a_0, y_0 >, \ldots, < a_{k-1}, y_{k-1} >$, přičemž výchozí hodnota vnitřního stavu x0 musí být explicitně nastavena (typicky nulový vektor)

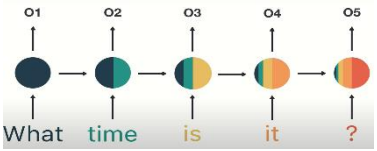


- **Backpropagation through time** – BPTT
  - Způsob trénování RNN – podobný backpropagaci, ale **chyby se napříč časovými kroky sčítají** (protože sdílejí parametry)
    - Exploding/vanishing gradient problém
  - Síť se rozbalí do časových kroků (unroll), chyba kumulativně sečtena pro každý krok, poté 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
- **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

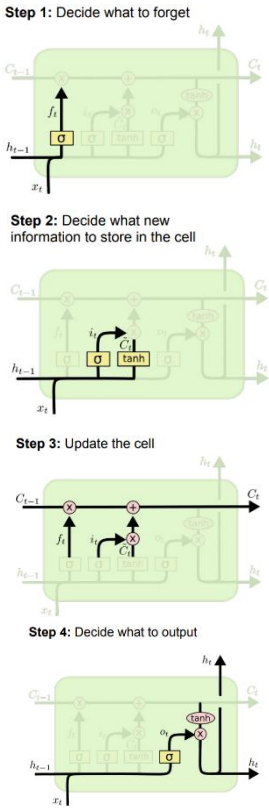


- **Hopfieldova síť** – 1 vrstva n fully connected rekurentních neuronů (každý vstupem každého dalšího, vrstva je zároveň kontext)
  - **Content-addressable memory** – asociativní paměť – do HS si lze ukládat data
    - Asociace – jsou referenční vzory
  - **Hebbovské nesupervizované učení** – když se 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

    - Problém 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
- **LSTM** – long short term memory
    - Informace z prvotních časových kroků se může projevit výrazně později
    - **Forget gate** – **které** informace **ponechat a které zahodit**
        - Vstup 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
        - Podle 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**
        - Současný 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

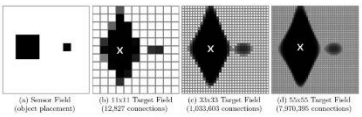
- **Neuroevoluce** – užití evolučních algoritmů ke generování neuronových sítí
    - Kombinuje spojitý a diskrétní prostor – váhy a topologie
    - Síť může mít hodně ekvivalentních stavů, třeba prořezávat SP
    - Topology and Weight Evolving Artifical Neural Networks.
- GNARL rozděloval mutace na parametrickou (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

- Proměnlivá 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

NEAT Overview

1. Initialize population.
2. Compute fitness for all individuals.
3. Speciate by means of Explicit Fitness Sharing.
4. Adjust fitness *f' = f / species_size*.
5. Determine offspring count for all species proportionally to *f'*.
6. Eliminate inferior individuals of current generation.
7. Reproduce – replace current generation by its offspring.
8. While not satisfied, go to 2.

- **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**
  - **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

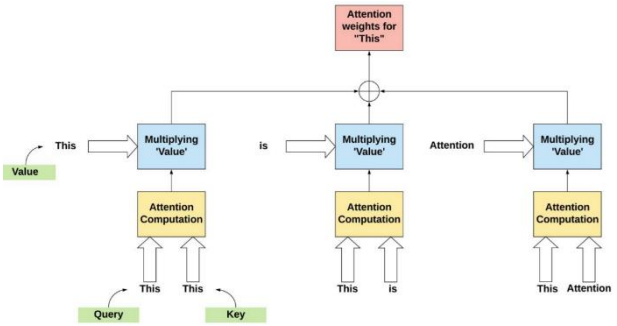


  - Jiný druh – HyperGP – místo neuronové sítě se nepřímo kóduje genetickým programováním
- **Novelty search =** místo abychom hledali něco konkrétního, co už známe, dáme šanci věcem zcela jiným
  - Prostor bývá plný lokálních minim, proto je vhodné se občas vydat zdánlivě odlišnou cestou, abychom dosáhli 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



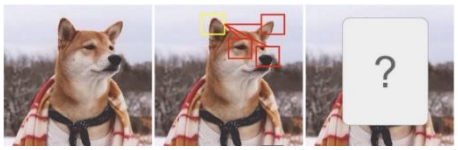
- Zpracovávají vstup ve formě sekvenčních dat, ale všechny najednou – mohou být paralelizovány
- Machine translation, text summarization, image description generation, ...
- Na rozdíl od RNN taky není tak problematický vanishing gradient.
- **Query** – jeden pro daný attention block
- Pro každou value 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:
    - Vstup = hodnoty (values v) – zakódováním slov nebo příznaků v obraze

    - Výstup attentionu – lineární kombinace values s parametry alfa
        - Alfy normalizované na součet 1
    - Pro 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
- **Global/Soft attention** – Bere v potaz celý vstupní prostor, všechny attentiony se předají dál

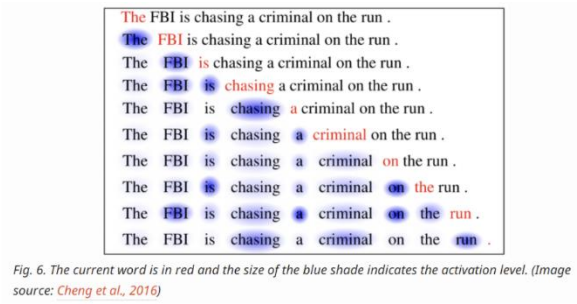
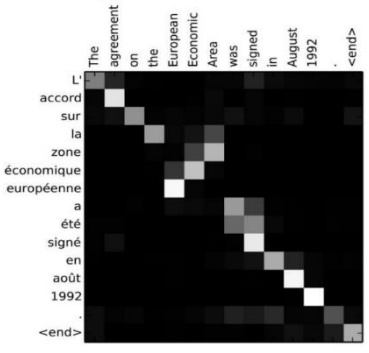


Fig. 6. The current word is in red and the size of the blue shade indicates the activation level. (Image source: Cheng et al., 2016)



- **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 c (podobně jako v logistické regresi)
    - Závěrečná predikce dána n **weak learnery** – hyperparametr n je dané číslo a weak learneři rozhodovací stromy
    - Během trénovací fáze se weak learneři f_k postupně konstruují, aby se zlepšil ensemble model:
      $$F^{(t)} = \sum_{k=1}^{t} f_k = F^{(t-1)} + f_t \text{ pro } n = 1,2,\dots,n$$
    - Závěrečný model je $F^{(n)} = \sum_{k=1}^{n} f_k = F^{(n-1)} + f_n$
    - **Ztrátová funkce** $\boldsymbol{l(y, \hat{y})}$ – 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: $l(y, \hat{y}) = (y - \hat{y})^2$
    - Cíl trénovacího procesu je minimalizovat **objective funkci**
      $$\sum_{i=1}^{N} \ell\left(y_i, \hat{y}_i^{(n)}\right), \qquad \hat{y}_i^{(n)} = F^{(n)}(\mathbf{x}_i) = \sum_{k=1}^{n} f_k(\mathbf{x}_i)$$
      je predikce pro i-tý datový bod a N počet trénovacích datových bodl
    - **Regularizace** – pro předejití overfittingu – s ní je objective funkce: $\sum_{i=1}^{N} \ell\left(y_i, \hat{y}_i^{(n)}\right) + \sum_{k=1}^{n} \Omega(f_k)$
        - Ω je rostoucí **funkce složitosti stromu** f_k
            - Složitost se dá měrit hloubkou, počtem listů, součtem vah listů, …
    - Minimalizujeme $\sum_{i=1}^{N} \ell\left(y_i, \hat{y}_i^{(t)} + f_{t+1}(\mathbf{x}_i)\right) + \Omega(f_{t+1})$
        - Podle volby ztrátové funkce to nemusí být možné
        - Zjednodušení – **aproximace** pomocí **Taylorova polynomu**
            - V XGBoostu polynom druhého stupně – aproximace v bodě $x_0 + d$
              $$f(x_0 + d) \approx f(x_0) + f'(x_0)d + \frac{1}{2}f''(x_0)d^2$$
        - Cílem je **minimalizace** ne y, 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 Y a vektory příznaků X: $Y = w_0 + w_1x_1 + \cdots + w_px_p + \varepsilon$
  - Vektor vah w a náhodná proměnná ε
  - Odhadujeme vektor vah $\widehat{w}$ a pomocí něj predikce Y: $\hat{Y} = \widehat{w}^{\,T}x$
- **OLS – ordinary least squares**
  - Při tréninku minimalizace reziduální sumy čtverců $RSS(w) = \|Y - Xw\|^2$
  - Minimum pomocí **normální rovnice**: $X^TY - X^TXw = 0$ (odpovídá $\nabla RSS(w) = 0$)
  - Pokud je X^TX regulární, existuje právě jedno řešení: $\widehat{w}_{OLS} = (X^TX)^{-1}X^TY$
- **Hřebenová regrese**
  - Minimalizujeme regularizovaný reziduální součet čtverců: $RSS_\lambda(w) = \|Y - Xw\|^2 + \lambda\sum_{i=1}^{p} w_i^2$
    - λ je prostě debilní parametr $\lambda \geq 0$
  - Matice I' - na diagonále kromě 1. prvku jedničky, jinak všude nuly
    - $RSS_\lambda(w) = \|Y - Xw\|^2 + \lambda w^TI'w$
  - To nám předělává normální rovnici a nové řešení (a zjistíme, že pro $\lambda = 0$ máme normální OLS)
    - $X^TY - X^TXw - \lambda I'w = 0$ (to co odčítáme je regulární pro nezáporné λ)
    - $\widehat{w}_\lambda = (X^TX + \lambda I')^{-1}X^TY$

Lineární bázová expanze

- Zobecnění, aby bylo možno použít pro nelineární funkce – nahrazení původních přízn. za transformované
  - $\mathcal{X}$... prostor obsahující všechny možné hodnoty vektoru příznaků $X = (X_1, \ldots, X_p)^T$, typ. $\mathcal{X} = \mathbb{R}^p$
  - $\varphi_1 \ldots \varphi_M$ ... **bázové funkce** = lineárně nezávislé funkce z $\mathcal{X}$ do $\mathbb{R}$
    - Představují **transformace** pův. příznaků $X_1, \ldots, X_p$ do **nového M-dimenzního prost. příznaků**
    - Model pro vysvětlovanou proměnnou je teď v tom novém prostoru lineární
- **Model lineární bázové expanze**
  - Vysvětlovaná proměnná Y v hodnotě $x = (x_1, \ldots, x_p)^T$ z X je dána jako:
    $$Y = w_1\varphi_1(x) + \cdots + w_M\varphi_M(x) + \varepsilon = w^T\varphi(x) + \varepsilon$$
    - φ: $\mathcal{X} \to \mathbb{R}^M$ je definována jako $\varphi(x) = (\varphi_1(x) \ldots \varphi_M(x))^T$ pro všechna $x \in \mathcal{X}$
    - $w = (w_1, \ldots, w_M)^T$,... vektor neznámých parametrů
    - ε ... náhodná proměnná s $E\varepsilon = 0$
- **Odhad lineární bázové expanze**
  - **Model** pro tréninkový vzorek daný N dvojicemi (Y_1, x_1) ... i (Y_N, x_N) v maticové formě:
    $$Y = \Phi\mathrm{w} + \varepsilon$$
    kde
    $$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\varphi}(\boldsymbol{x}_1)^T \\ \vdots \\ \boldsymbol{\varphi}(\boldsymbol{x}_N)^T \end{pmatrix} = \begin{pmatrix} \varphi_1(\boldsymbol{x}_1) & \cdots & \varphi_M(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \varphi_1(\boldsymbol{x}_N) & \cdots & \varphi_M(\boldsymbol{x}_N) \end{pmatrix}, \quad \boldsymbol{Y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_N \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{pmatrix}$$
  - **Minimalizace** (podobně jako v hřebenové regresi):
    $$RSS_\lambda(w) = \|Y - \Phi w\|^2 + \lambda w^Tw$$
  - **Normální rovnice**:
    $$\Phi^TY - \Phi^T\Phi w - \lambda w = 0$$
  - Pro existuje právě jedno **řešení**:
    $$\widehat{w}_\lambda = (\Phi^T\Phi + \lambda I)^{-1}\Phi^TY$$
  - **Predikce** Y v x:
    $$\hat{Y} = \widehat{w}_\lambda^{\,T}\varphi(x)$$
- Různé volby bázové funkce zajišťují flexibilitu modelu
- Často skončíme s velkým počtem bázových funkcí (větším než tréninkový vzorek), takže je nutná regularizace

Jádrový trik

- Duální reprezentace
  - V duální reprezentaci jsou bázové funkce dány implicitně skrze **jádrovou funkci**
  - Máme hodnoty w ve tvaru $w = \Phi^T \alpha$, kde $\alpha \in \mathbb{R}^N$
    - **Omezení w** na podprostor $\mathbb{R}^M$ generovaný vektory $\varphi(x_1) \dots \varphi(x_N)$
    - Dosazení do $RSS_\lambda(w)$:
      $$RSS_\lambda(\alpha) = \|Y - \Phi\Phi^T\alpha\|^2 + \lambda\alpha^T\Phi\Phi^T\alpha$$
    - Protože jsme w omezili, dostaneme:
      $$\min_\alpha RSS_\lambda(\alpha) = \min_\alpha RSS_\lambda(w(\alpha)) \geq \min_w RSS_\lambda(w)$$
  - Pokud $\lambda > 0$, pak
    $$\min_\alpha RSS_\lambda(\alpha) = \min_w RSS_\lambda(w)$$
    Navíc, pokud w^* minimalizuje $RSS_\lambda(w)$, potom $\alpha^* = \frac{1}{\lambda}(Y - \Phi w^*)$ minimalizuje $RSS_\lambda(\alpha)$
    Pokud α^* minimalizuje $RSS_\lambda(\alpha)$, potom $w^* = \Phi^T\alpha^*$ minimalizuje $RSS_\lambda(w)$
  - Tím pádem jsou úlohy minimalizace $RSS_\lambda(\alpha)$ a $RSS_\lambda(w)$ **ekvivalentní**
- Jádrová funkce k: $\mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$
  $$k(x, y) = \varphi(x)^T\varphi(y)$$
- Gramova matice: $\boldsymbol{G} = \Phi\Phi^T$
  - $N \times N$ symetrická matice
  - **Pozitivně semidefinitní**:
    $$\alpha^T G\alpha = \alpha^T\Phi\Phi^T\alpha = (\Phi^T\alpha)^T(\Phi^T\alpha) = \|\Phi^T\alpha\|^2 \geq 0$$
  - Pomocí ní se dá $RSS_\lambda(\alpha)$ vyjádřit jako:
    $$RSS_\lambda(\alpha) = \|Y - G\alpha\|^2 + \lambda\alpha^T G\alpha$$
  - (i, j)-tá komponenta Gramovy matice:
    $$G_{i,j} = (\boldsymbol{\Phi}\boldsymbol{\Phi}^T)_{i,j} = \sum_{\ell=1}^{M} \Phi_{i,\ell}\Phi_{j,\ell} = \sum_{\ell=1}^{M} \varphi_\ell(\boldsymbol{x}_i)\varphi_\ell(\boldsymbol{x}_j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$
    - Gramova matice je celá dána jádrovou funkcí
- Predikce v duální reprezentaci
  - Minimalizujeme $\widehat{w} = \Phi^T\hat{\alpha}$
  - Predikce Y v x:
    $$\hat{Y} = \hat{\boldsymbol{w}}^T\boldsymbol{\varphi}(\boldsymbol{x}) = \hat{\boldsymbol{\alpha}}^T\boldsymbol{\Phi}\boldsymbol{\varphi}(\boldsymbol{x}) = \sum_{i=1}^{N}\sum_{j=1}^{M} \hat{\alpha}_i\Phi_{i,j}\varphi_j(\boldsymbol{x}) = \sum_{i=1}^{N} \hat{\alpha}_i k(\boldsymbol{x}_i, \boldsymbol{x}) = \hat{\boldsymbol{\alpha}}^T\boldsymbol{k}(\boldsymbol{x})$$
  - Kde $\boldsymbol{k}(\boldsymbol{x}) = (k(\boldsymbol{x}_1, \boldsymbol{x}), \dots, k(\boldsymbol{x}_N, \boldsymbol{x}))^T$
  - Tím pádem nám na vyjáření účelové funkce $RSS_\lambda(\alpha)$ i predikce $\hat{Y}$ stačí jádrová funkce (a stejně tak pro $\hat{\alpha}$)
- Odhad α
  - Řešení uzavřené formy minimalizačního problému
    - Gradient $\nabla RSS_\lambda(\alpha) = -2G(Y - G\alpha) + 2G\lambda\alpha$
    - Položení rovno nule → normální rovnice $G(Y - G\alpha - \lambda\alpha) = 0$
    - Kvůli pozitivní semidefinitnosti Gramovy matice je $(G + \lambda I)$ pozitivně definitní
  - Dostáváme odhad $\hat{\alpha} = (G + \lambda I)^{-1}Y$
    - Taky se dá vyjádřit jenom pomocí jádrové funkce
- Jádrový trik
  - Stvořili jsme ekvivalentní duální reprezentaci celého modelu spolu s účelovou funkcí, která je formulovaná tak, že vstupní vektor x vstupuje do výpočtu jako forma skalárních součinů
  - Jádrový trik = náhrada skalárních součinů za jádrovou funkci
  - Rozšíření – začne se s jádrem bez explicitní specifikace bázové funkce
    - Implicitně tak můžeme používat prostory příznaků vysokých, i nekonečných dimenzí
- Lineární jádro – pro všechny x platí $\varphi(x)$:
  $$k(x, y) = x^T y$$
  - Zobecnění – polynomiální jádro:
    $$k(x, y) = (x^T y + 1)^n$$

Jádrové (kernelové) stroje

- Předpokládáme, že skutečný model vysvětlované proměnné Y v x je $Y = f(x) + \varepsilon$, kde f je neznámá funkce a ε je náhodná proměnná s nulovou střední hodnotou
- V lineární bázové expanzi použijeme odhad:

$$f(\boldsymbol{x}) = w_1\varphi_1(\boldsymbol{x}) + \dots w_M\varphi_M(\boldsymbol{x})$$

- **Jádrový stroj** – model, kdy začínáme z jádrové funkce k:

$$f(\boldsymbol{x}) = \sum_{j=1}^{K} \alpha_j k(\boldsymbol{x}, \boldsymbol{\mu}_j)$$

  - $\mu_1 \dots \mu_K \in \mathcal{X}$ jsou středy
  - Jádrový stroj odpovídá **lineární bázové expanzi** s $\varphi_j(\cdot) = k(\cdot, \mu_j)$
    - Proto se $\boldsymbol{\varphi}(\boldsymbol{x}) = (k(\boldsymbol{x}, \boldsymbol{\mu}_1), \dots, k(\boldsymbol{x}, \boldsymbol{\mu}_K))^T$ jmenuje kernelizovaný vektor příznaků
- **Vektorové stroje**
  - Speciální případ jádrových strojů, kde **středy jsou dány body z trénovacího datasetu**:

$$f(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j k(\boldsymbol{x}, \boldsymbol{x}_j)$$

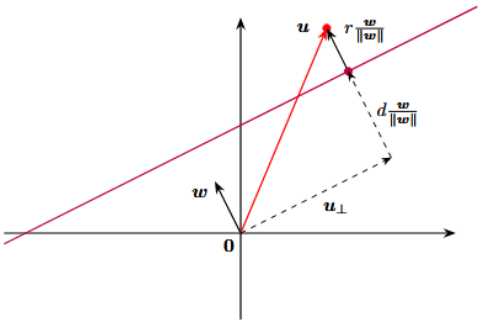
  - Vektorový stroj můžeme dostat tím, že uděláme **jádrový trik na modelu lineární bázové expanze**
  - Zajímají nás **řídké vektorové stroje**, kde je pro hodně bodů platí $\alpha_j = 0$

Diskriminativní funkce

- Zaměřujeme se teď na binární klasifikaci, kde predikujeme $Y \in \{-1, 1\}$ v bodě $x \in \mathcal{X}$
- $f(x) = w^T\varphi(x) + w_0$
  - Pokud $f(x) > 0$, pak $\hat{Y} = 1$, jinak -1:

$$\hat{Y}(\boldsymbol{x}) = \operatorname{sgn}(f(\boldsymbol{x})) = \operatorname{sgn}(\boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x}) + w_0)$$

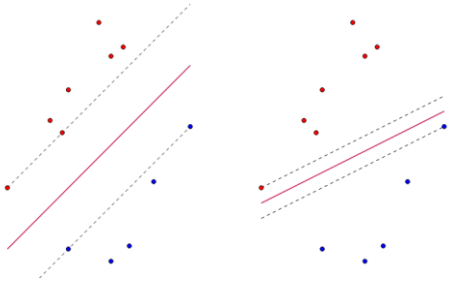
- **Rozhodovací hranice** v prostoru příznaků $\mathbb{R}^M$
  - Definovaná **podmínkou $\boldsymbol{w^Tu + w_0 = 0}$**
  - Řešení tvoří hyperrovinu v $\mathbb{R}^M$ = rozdělovací hyperrovina (separating hyperplane)



    - Normála je dána vektorem w
    - Vzdálenost od 0 ve směru w: $d = -\frac{w_0}{\|w\|}$
    - Libovolný bod u lze **rozložit na dvě části** – $u_\perp$ je kolmá na w a $u_\parallel$ je s ním rovnoběžná
    - $r = u_\parallel - d$ … **vzdálenost $\boldsymbol{u}$ od rozhodovací hranice** ve směru w
    - Můžeme dosadit do podmínky a získat $w^Tu + w_0 = r\|w\|$, z čehož můžeme vyjádřit r a dosadit do diskriminativní funkce
    - Tím dostaneme, že **vzdálenost $\boldsymbol{\varphi(x)}$** od rozhodovací hranice je dána $r = \frac{f(x)}{\|w\|}$
    - Ta vzdálenost je **pozitivní, když $\boldsymbol{f(x) > 0}$** a negativní, když $f(x) < 0$
    - **Znaménko $\boldsymbol{f(x)}$ separuje prostor příznaků** na dva podprostory
      - Pozitivní je ve směru w, všechny predikce jsou 1
      - Negativní ve směru $-w$, predikce jsou -1

Princip velkého okolí nebo jak já už nevím, jak tahat z paty ty překlady (Large margin principle)

- Pokud máme diskriminativní funkci f, predikce v prostoru příznaků jde dobře geometricky interpretovat
  - Chceme nějak natrénovat neznámé parametry w a w_0
  - Trénovací vzorky N párů ve tvaru $(Y_1x_1), \dots, (Y_Nx_N)$
- **Lineárně separabilní případ** – máme trénovací body v prostoru příznaků $\varphi(x_1) \dots \varphi(x_N)$ a existuje alespoň jedna volba parametrů w a w_0 taková, že $f(x_i) > 0$ pro body kde $Y_i = 1$ a $f(x_j) < 0$ pro body kde $Y_j = -1$ (binární klasifikace)
    - Řešení může být nekonečno



- **Margin** = okolí = **nejmenší vzdálenost** mezi rozhodovací hranicí a kterýmkoliv ze vzorků
  - Toto se snažíme maximalizovat
- Vzdálenost i-tého trénovacího bodu $\varphi(x_i)$ od rozhodovací hyperroviny je dána $r_i = \frac{f(x_i)}{\|w\|}$
  - Takže **hledáme w a w_0** takové, aby minimální r_i bylo co největší, a zároveň aby každý bod byl na správné straně hranice: $Y_i f(x_i) > 0$

$$\max_{\boldsymbol{w},w_0} \min_i \frac{Y_i(\boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x}_i) + w_0)}{\|\boldsymbol{w}\|}$$

  - w a w_0 jdou libovolně přeškálovat, takže můžeme zvolit, že pro bod nejblíže k rozhodovací hranici bude platit $Y_j(\boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x}_j) + w_0) = 1$
    - Tím pádem pro všechny body musí platit $Y_i(\boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x}_i) + w_0) \geq 1$
    - Hledáme **minimum** $1/\|w\|$ – maximalizace $\|w\|^{-1}$, ekvivalentní minimalizaci $\|w\|^2$
- Řešíme následující **optimalizační problém**:

$$\min_{\boldsymbol{w},w_0} \frac{1}{2}\|\boldsymbol{w}\|^2 \quad \text{subject to} \quad Y_i(\boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x}_i) + w_0) \geq 1 \quad \text{for all } i$$

  - Omezení říká, že všechny body jsou na správné straně hranice s okolím minimálně 1 (vhodné škálování)
  - Příklad kvadratického programování – minim. kvadratické funkce za podmínky lineární nerovnice
  - **Řešení** ve formátu jádrové funkce je **support vector machine**

Support vector machine

- Pokud **nemáme lineárně separabilní** data, nemůžeme tu nerovnici dobře vyřešit pro všechna i
  - Musíme **zrelaxovat podmínky** a povolit, ať jsou některé body na špatné straně hranice
  - Zavedení **penalizace** se zvyšující se vzdáleností od hranice ve špatném směru
- **Volné proměnné**: $\xi_i \geq 0,\ i = 1, \dots, N$ takové, že $\xi_i = 0$, pokud je bod na nebo vně správného okolí na správné straně rozhodovací hranice, jinak $\xi_i = |Y_i - f(x_i)|$
  - Pokud $\xi_i = 0$, pak $Y_i f(x_i) \geq 1$
  - Pokud $0 < \xi_i \leq 1$, bod leží uvnitř okolí, ale na správné straně hranice a $Y_i f(x_i) > 1$
  - Pokud $\xi_i > 1$, bod leží na špatné straně hranice a je špatně klasifikován – $Y_i f(x_i) < 0$
- Cíl SVM – díky volným proměnným můžeme **uvolnit hard podmínky** $Y_i f(x_i) \geq 1$ **soft margin podmínkami**[1]:

$$Y_i f(x_i) \geq 1 - \xi_i \ \ \forall i = 1 \dots N$$

  - **Nový optimalizační problém**:

$$\min_{\boldsymbol{w},w_0,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

    S podmínkami:

$$\xi_i \geq 0 \quad \text{and} \quad Y_i(\boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x}_i) + w_0) + \xi_i - 1 \geq 0 \quad \text{for all } i.$$

  - Protože $\xi_i > 1$ znamená že bod i je špatně klasifikovaný, můžeme $\Sigma_i \xi_i$ chápat jako **horní hranici** počtu **špatně klasifikovaných** bodů
  - C … **regularizační parametr** – kontroluje počet chyb, které můžeme tolerovat v trénovací množině
    - Často $C = \frac{1}{\nu N}$, kde $0 < \nu \leq 1$ ovládá zlomek špatně klasifikovaných bodů během trénovací fáze - $\boldsymbol{\nu - SVM}$ klasifikátor
- Pomocí Lagrangeových multiplikátorů a principu duality můžeme **maximalizovat ekvivalentní Lagrangeovský duální problém**:

$$\tilde{L}(\boldsymbol{a}) = \sum_{i=1}^{N} a_n - \frac{1}{2}\sum_{i,j=1}^{N} a_i a_j Y_i Y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

  - Vzhledem k novým proměnným a s podmínkami:

$$0 \leq a_i \leq C \quad \text{and} \quad \sum_{i=1}^{N} a_i Y_i = 0$$

  - $k(x_i, x_j)$ … jádrová funkce, která má k původní bázové funkci vztah $k(x, y) = \varphi(x)^T\varphi(y)$

[1] Sry mě už to překládání sere, fakin anglické slajdy s obskurními pojmy

- Souvislost s KKT[2][3]
  - Abychom vyřešili (minimalizovali) duální problém, musíme maximalizovat Lagrangian s ohledem na w a w_0 – vytvoříme parciální derivace, položíme je rovny 0 a získáme dvě rovnice:
    - Parc. derivace w: $w = \sum a_i Y_i x_i$
    - Parc. derivace w_0: $\sum a_i Y_i = 0$
  - Dosazení rovnic do optim. problému dostaneme Lagrangeovský duální problém (výše)
    - Jelikož $a \geq 0$, bude minimum v $a = 0$
  - Pro bod, ve kterém se nachází minimum Lagrangeovy funkce L platí $\nabla L = 0$ a KKT podmínky
  - $a = 0$ pro všechny ne-support vektory (z KKT podmínek prý)
- Řešení = **support vector machine (SVM)** ve formě:

$$f(\boldsymbol{x}) = \sum_{j=1}^{N} a_j Y_j k(\boldsymbol{x}, \boldsymbol{x}_j) + w_0$$

- **Predikce** v bodě x:

$$\hat{Y}(x) = sgn(f(x))$$

- Vlastnosti řešení
  - Řešení je **vektorový stroj**

$$f(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j k(\boldsymbol{x}, \boldsymbol{x}_j) + w_0$$

    kde $\alpha_j = a_j Y_j$
  - Je **řídké** - $a_j = 0$ pro všechny správně klasifikované body ležící vně okolí hranice
  - **Support vectory** = body, kde $a_j > 0$
    - Datové body, které leží nejblíž k rozhodovací rovině/hyperplane/hranici
    - Datové body, které je **nejtěžší klasifikovat**
    - Mají přímý vliv na polohu optimální rozhodovací hranice
    - Rozhodovací funkce je plně specifikovaná malou podmnožinou trénovacích vzorků = support vektorů
  - Body s $a_j < C$ leží na hranici okolí
  - Body s $a_j = C$ jsou buď správně klasifikované uvnitř okolí, nebo špatně klasifikované
  - Čím větší je regularizační parametr C, tím řidší řešení dostaneme
- **Vstup a výstup SVM**
  - Vstup – množina (vstup, výstup) párů trénovacích vzorků (vstup příznaky x, výstup y)
  - Výstup – množina vah w, jedna pro každý příznak, jejichž lineární kombinace predikuje hodnotu y
  - Používá se optimalizace na maximalizaci marginu (okolí, taková „šířka ulice“), abychom zmenšili počet vah, které jsou nenulové, na jenom pár, které odpovídají důležitým příznakům, na kterých záleží při stanovování rozhodovací hranice
    - Nenulové váhy odpovídají support vectorům („podporují rozhodovací hranici“)
- Support vectory – **lineárně separabilní** případ
  - Prvky trénovacího datasetu, které kdybychom odstranili, tak by se změnila pozice rozhodovací hranice
  - Jsou to kritické prvky trénovacího datasetu
  - Problém nalezení optimální hyperroviny je optimalizační problém
    - Řešení pomocí Lagrangeových multiplikátorů

[2] Holeňova a Vašatova oblíbená otázka, MUSÍME umět
[3] Je to popsáno v NI-PON skriptech, ale je to absolutní peklo a nevím, jak moc podrobně to potřebujeme

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:
  - **Top-N** – cílem je najít n věcí, které jsou pro uživatele nejvíc relevantní
    - Dobré doporučovat pouze položky, které uživatel nezná
  - **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 k 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 k může prudce ovlivnit kvalitu doporučování, volí se empiricky vzkledem k výkonu systému
- **Scoring function** – predikce na základě hodnotící funkce – skóre $U \times I \rightarrow \mathbb{R}$ hodnotí jednotlivé itemy na základě kontextu (uživatelů)
- **Hodnocení**
  - **Cumulative gain –** hodnotí, jak relevantních je prvních k položek

$$CG_k = \sum_{i=1}^{k} rel_i$$

    - **Discounted CG –** hodnota relevance logaritmicky proporcionálně zmenšena podle pozice výsledku
  - **Precision** – kolik z doporučených ho skutečně zajímá

$$precision = \frac{|relevantní \cap doporučené|}{doporučené}$$

  - **Recall** – kolik z položek, které ho zajímají, jsme mu doporučili
  - F1 skóre, AUC (plocha pod křivkou ROC), RMSE pro spojité hodnoty

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
  - Rozptyl měřen pomocí ztrátové funkce jako **MSE**:

$$\underset{U,V}{\operatorname{argmin}} \sum_{(i,j)\in\Omega} (r_{i,j} - \mathbf{u}_i^T \mathbf{v}_j)^2 + \lambda \left(\sum_x ||\mathbf{u}_x||^2 + \sum_y ||\mathbf{v}_y||^2 \right)$$

  - 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
    - Např. matice P reprezentuje interakční matici a matice C indikuje konfidenční hladiny interakcí:

$$P = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \quad C = \begin{bmatrix} 0.85 & 0 & 0 & 0.34 & 0 & 0.98 \\ 0 & 0.37 & 0.10 & 0 & 0.63 & 0.01 \\ 0.45 & 0.42 & 0.43 & 0 & 0 & 0.23 \\ 0 & 0 & 0.26 & 0 & 0 & 0.88 \end{bmatrix}$$

  - Metody kolaborativního filtrování jako **Vážená Regularizovaná Maticová Faktorizace** (WRMF) řídí implicitní feedback přiřazením různých konfidenčních hladin pozorovaným a nepozorovaným interakcím
    - Optimalizační problém:

$$\min_{U,V} \sum_{i,j} C_{ij} (P_{ij} - \mathbf{u}_i^\top \mathbf{v}_j)^2 + \lambda ||\mathbf{u}_i||^2 + \lambda ||\mathbf{v}_j||^2$$

- 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

- **Apriorní znalost** = počáteční znalost o studované proměnné před zohledněním dat (pozorování, měření)
  - Vyjádřena **apriorní distribucí** – pravděpodobnost (u diskrétních) nebo hustota (u spojitých)
  - Zahrnutí nových informací → **aposteriorní znalost**
  - Apriorní i aposteriorní kvantifikují **míru neurčitosti**
- **Vícerozměrné distribuce** – veličiny X, Y
  - **Sdružené hustoty** $f(x,y) = f(y,x)$
  - **Marginální hustoty** $f(x) = \int f(x,y)dy$, $f(y) = \int f(x,y)dx$
  - **Podmíněné hustoty** $f(x|y) = \frac{f(x,y)}{f(y)}$ a naopak
  - **Řetězové pravidlo** $f(x,y) = f(x|y)f(y) = f(y|x)f(x)$
- **Bayesova věta** – máme náhodné veličiny x, θ s hustotami $f(x|\,\theta)$ a $\pi(\theta)$:

$$\pi(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{f(x)}, \quad f(x) > 0$$

  - $\pi(\theta|x)$ … **aposteriorní podmíněná hustota** X
  - $\pi(\theta)$ … **apriorní hustota**
  - $f(x|\theta)$ … **model / věrohodnost** (likelihood) dat
  - $f(x)$ … marginální hustota x / normální faktor / evidence
    - Jmenovatel je **normalizační faktor** nezávislý na odhadovaném θ a dostaneme ho jednoduše vysčítáním nebo integrací čitatele, $f(x) = \int f(x|\theta)\pi(\theta)d\theta$, zapisujeme často jen proporcionalitu:

$$\pi(\theta|x) \propto f(x|\theta)\pi(\theta)$$

- **Beta distribuce** - $p \in [0,1] \rightarrow p \sim B(a,b)$
  - Hustota $\pi(p|a,b) = \frac{1}{B(a,b)}p^{a-1}(1-p)^{b-1}$
  - Střední hodnota $E[p] = E[p|a,b] = \frac{a}{a+b}$
  - Beta funkce $B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$
    - Je v ní obsažená gamma funkce
  - Aposteriorní distribuce je zároveň beta distribuce

Bayesovské odhadování

- y_t … pozorovaná veličina v okamžiku $t = 0, 1, 2$
- Vektor hodnot $y_{0:t-1}[y_0 \dots y_{t-1}]$
- y_t determinováno veličinou x_t a konstantním parametrem θ
- y_t nezávislé, stejně rozdělené
- **Apriorní hustota** $\pi(\theta|x_0, y_0)$ – vhodná znalost
  - x_0, y_0 … pseudodata
- $f(y_t|x_t, \theta)$ je hustota a $\pi(\theta|x_{0:t-1}, y_{0:t-1})$ apriorní hustota pro θ, pak aposteriorní hustota pro **jednokrokový update** je:

$$\begin{aligned}\pi(\theta|y_{0:t}, x_{0,t}) &= \frac{f(y_t|x_t,\theta)\,\pi(\theta|x_{0:t-1}, y_{0:t-1})}{\int f(y_t|x_t,\theta)\,\pi(\theta|x_{0:t-1}, y_{0:t-1})d\theta} \\ &= \frac{f(y_t|x_t,\theta)\,\pi(\theta|x_{0:t-1}, y_{0:t-1})}{f(y_t|x_t)} \\ &\propto f(y_t|x_t,\theta)\,\pi(\theta|x_{0:t-1}, y_{0:t-1}).\end{aligned}$$

  - Bez normalizující hutoty ve jmenovateli
- Sekvenční update po jednom datu je stejný, jako update více daty najednou

Sekvenční odhad

- Ke stávající informaci přidáváme pouze nejnovější data = **update**
- **Sekvenční bayesovský update v krocích**:
  1. Použijeme apriorní distribuci
  2. Přidáme nové pozorování
  3. Dostaneme aposteriorní distribuci
  4. Tu opět použijeme jako apriorní pro další pozorování

$$\pi(\theta|x_0,y_0) \underset{\text{Bayes}}{\overset{x_1,y_1}{}} \pi(\theta|x_{0:1},y_{0:1}) \underset{\text{Bayes}}{\overset{x_2,y_2}{}} \pi(\theta|x_{0:2},y_{0:2}) \to \cdots \underset{\text{Bayes}}{\overset{x_t,y_t}{}} \pi(\theta|x_{0:t},y_{0:t}) \to \cdots$$

- **Bodový odhad** θ (neznámý důležitý parametr)
  - Střední hodnota $E[\theta] = E[\theta|x_{0:t},y_{0:t}]$
  - Modus (MAP odhad), medián
- **Věrohodnost** odhadu – rozptyl $var\theta$
- Problém bayesovského modelování = odvození aposteriorní distribuce a jejích vlastností
- **Exponenciální třída distribucí**

$$f(y|x,\theta) = h(y,x)g(\theta)\exp[\eta^T T(y,x)]$$

  - $\eta \equiv \eta(\theta)$ … přirozený parametr
  - $T(y,x)$ … **suficientní statistika** fixního rozměru
    - Obsahuje všechny informace o parametrech modelu
  - $h(y,x)$ … známá funkce
  - $g(\theta)$ … **normalizační funkce**
  - Pokud $\eta(\theta) = \theta$, třída je **kanonická**
- **Konjugovaná apriorní distribuce**
  - $y|x,\theta$ má rozdělení z **exponenciální třídy distribucí** → apriorní distribuce θ s hyperparametry ξ a ν je k němu konjugovaná, pokud:

$$\pi(\theta) = q(\xi,\nu)g(\theta)^{\nu}\exp\,[\eta^T\xi]$$

    - ξ má stejný rozměr jako $T(y,x)$
    - $\nu \in \mathbb{R}^+$
    - $q(\xi,\nu)$ … známá funkce
    - $g(\theta)$ … normalizační funkce
  - Hyperparametry = parametry apriorna
- **Příklady konjugované distribuce**:

Model	Použití	Konjugované apriorno
Normální, známý rozptyl	Všue možně	Normální
Normální, neznámý rozptyl	Všude možně	Normální inverzní – gamma
Bernoulli	Úspěch – neúspěch	Beta
Binomický	Úspěch – neúspěch	Beta
Poissonův	Řídké jevy (částice)	Gamma
Multinomický	Klasifikace do k tříd	Dirichletovo

Bayesovský odhad s konjugovaným apriornem

- Hyperparametry ξ_{t-1}, ν_{t-1} → **Bayesův update**

$$\pi(\theta|y_{0:t},x_{0,t}) \propto f(y_t|x_t,\theta)\,\pi(\theta|x_{0:t-1},y_{0:t-1})$$

  je jen triviální **součet**

$$\xi_t = \xi_{t-1} + T(y_t,x_t)$$
$$\nu_t = \nu_{t-1} + 1.$$

- Update pro více dat najednou – taky jen součet
- **Bayesova věta** se při použití konjugovaného apriorna **ztriviální na přičtení suficientní statistiky** k hyperparametru ξ_{t-1} a **inkrementaci** hyperparametru ν_{t-1}

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

Stavový model

- Charakterizován 3 veličinami:
  - **Stav x_t** – nemůžeme pozorovat, ale můžeme odhadovat
    - Stav je parametr celého modelu, ale vyvíjí se
  - **Vstup = řídící veličina u_t** – známá
  - **Výstup z_t** – pozorovatelná veličina determinovaná x_t a u_t
- Model diskrétní (časové okamžiky) nebo spojitý (derivace) – tady jen diskrétní
- **Zápis modelu** – stochastické funkce stavů a vstupů:

$$x_t = f_t(x_{t-1}, u_t)$$
$$z_t = g_t(x_t)$$

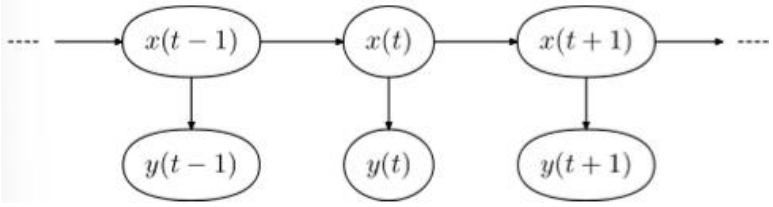
- **Lineární systém**:

$$x_t = A_t x_{t-1} + B_t u_t + v_t$$
$$z_t = H_t x_t + w_t$$

  - x_t … stav
  - u_t … řídící veličina
  - w_t a v_t … šum stavu a šum měření
  - A_t, B_t, H_t … matice

Skrytý markovský model – HMM

- **Markovská vlastnost** – markovský proces (nebo markovský model 1. řádu) je model, v němž závisí pouze na stavu předchozím
  - Popsán pravděpodobnostmi:
    - $p(x_t|x_{t-1})$ … prvděpodobnost přechodu
    - $p(x_0)$ … počáteční stav
- Skrytý markovský proces – není přímo pozorovatelný, ale lze na něj nahlížet prostřednictvím jiné pozorované veličiny y_t (nebo z_t)
  - Předpokládá spojitý proces



Kalmanův filtr

- Uvažujme **časově invariantní stavový model**:

$$x_t = Ax_{t-1} + Bu_t + v_t$$
$$z_t = Hx_t + w_t$$

  - Obě šumové proměnné nezávislé a centrované v 0

$$v_t \sim N(0, Q)$$
$$w_t \sim N(0, R)$$

  - Z normality:
    - $x_t \sim \mathcal{N}(Ax_{t-1} + Bu_t, Q)$ s hustotou $p(x_t|x_{t-1}u_t)$
    - $z_t \sim \mathcal{N}(Hx_t, R)$ s hustotou $f(z_t|x_t)$
- **Apriorní distribuce** pro x_t – model y_t je normální → konjugované apriorno bude též normální se střední hodnotou x_{t-1}^+ a kovarianční maticí P_{t-1}^+

$$\pi(x_t|z_{0:t-1}, u_{0:t-1}) = \mathcal{N}(x_{t-1}^+, P_{t-1}^+)$$

- Kalmanův filtr má 2 kroky – **predikce stavu a update stavu**
  - Predikce využívá 1. stavovou rovnici
  - Update nutný kvůli šumu
  - Predikovaný stav je apriorní informace do Bayesovy věty, kde je pomocí měření zkorigován
- **Predikce**
  - Užití modelu pro vývoj stavu a **předpověď na základě odhadu stavu v předchozím okamžiku**, jaký bude stav nyní
    - Řešíme časový vývoj stavu $x_{t-1} \rightarrow x_t$
  - Použijeme **odhad reprezentovaný posledním aposteriornem**, nyní apriornem, pro další časový krok a ten proženeme modelem vývoje:

$$\pi(x_t|z_{0:t-1}, u_{0:t}) = \int p(x_t|x_{t-1}, u_t)\, \pi(x_{t-1}|z_{0:t-1}, u_{0:t-1})\, \mathrm{d}x_{t-1}$$

  - **Násobíme 2 normální distribuce a marginalizujeme** → máme zas normální distribuci $\mathcal{N}(x_{t-1}^-, P_{t-1}^-)$ s hyperparametry:

$$x_t^- = Ax_{t-1}^- + Bu_t$$
$$P_t^- = AP_{t-1}^- A^T + Q$$

    - Toto **vyčíslí rovnici** pro stav
    - Odhad stavu x_t^- - dosazení do rovnice
    - Kovariance odhadu P_t^- - **míra neurčitosti odhadu**
  - Byla použita lineární transformace, platí

$$E[AX + B] - AE[X] + B$$
$$var(AX + B) = A^2 var(X) = Avar(X)A$$

- **Update**
  - **Korekce** – opraví predikovaný odhad novými pozorováními y_t
    - Pomocí Bayesovy věty:

$$\pi(x_t|z_{0:t}, u_{0:t}) \propto f(z_t|x_t)\pi(x_t|z_{0:t-1}, u_{0:t})$$

  - Model se přepisuje do tvaru distribuce z **exponenciální třídy** + vhodné apriorno
    - Pak je bayesovský update jen **součet hyperparametrů a suficientní statistiky**
  - **Kalmanovo zesílení** – zmenšuje střední kvadratickou chybu (optimální ji minimalizuje)
    - Čím větší je, tím větší je důraz na nová měření

$$K_t = P_t^- H^\intercal (R + HP_t^- H^\intercal)^{-1}$$

- **Rovnice KF se dají rozepsat**:

$$\begin{aligned}
\hat{z}_t^- &= H\hat{x}_t^-, && \text{(predikce měření)}\\
\nu_t &= z_t - \hat{z}_t^-, && \text{(innovace, chyba predikce } z_t)\\
S_t &= HP_t^- H^\intercal + R, && \text{(kovariance inovace } \nu_t)\\
K_t &= P_t^- H^\intercal S_t^{-1}, && \text{(kalmanovské zesílení - gain)}\\
x_t^+ &= \hat{x}_t^- + K_t\nu_t, && \text{(aposteriorní odhadu stavu } x_t)\\
P_t^+ &= (I - K_t H)P_t^-. && \text{(aposteriorní kovariance)}
\end{aligned}$$

Nelineární stavové modely

- Zase ve tvaru $x_t = f_t(x_{t-1}, u_t)$ a $z_t = g_t(x_t)$, ale jedna nebo obě **funkce nejsou lineární**
- Možná řešení v závislosti na míře linearity:
  - **Slabá nelinearita** → lokální **linearizace** derivacemi → **rozšířený Kalmanův filtr** (EKF)
  - **Velká** nelinearita → **Unscented Kalmanův Filtr** (UKF)
  - Vždy lze **brute force** – **Monte Carlo** vzorkování stavů ze souvisejícího stavového prostoru
    - Vygenerování hromady náhodných vektorů x_t
    - Vložení n. v. do stavového modelu
    - Zjištění, které jsou nejpravděpodobnější pro z_t
    - Nejpravděpodobnějším se přiřadí vysoké váhy a hledá se v jejich okolí
    - Např. částicový filtr (**particle filter**)

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

Monte Carlo integrace

- Integrál funkce $f(x)$ na $[a, b]$ je vlastně součet malých obdélníků s výškou $f(x_i)$ a délkou strany Δx
- Uvažujeme shodná Δx → můžeme je vytknout před sumu, rovnoměrně jimi rozdělit interval $[a, b]$ na N pod-intervalů a vlastně se přesunout ke střední hodnotě funkce

$$\int \dots = \frac{b-a}{N} \sum_{i=1}^{N} f(x_i) = (b-a)\langle f \rangle$$

  - $\langle f \rangle$ … odhad střední hodnoty
  - Pro M-rozměrný případ analogicky, integrál je roven $V\langle f \rangle$, kde $V = b - a$ je „objem“ množiny V
- MC integrace – využívá princip výše s tím, že body x_i nebere jako deterministicky stanovené body v rovnoměrné síti, ale vybírá tyto body náhodně z rovnoměrného rozdělení, středování i objem zachovány
  - Zákon velkých čísel zajišťuje konvergenci ke skutečné hodnotě při $N \to \infty$
    - Pro $N \to \infty$ konverguje chyba odhadu $var(V\langle f \rangle)$ k 0

Rejection sampling

- „accept-reject algoritmus“, využívá se, když nemůžeme použít konjugované apriorno
- Využívá proposal distribuci – vhodná distribuce, z níž umíme snadno vzorkovat (nemusí být normovaná)
- $f(x)$ … hustota, z níž chceme vzorkovat
- Předpoklad algoritmu:

$$f(x) = \int_0^{f(x)} du = \int_0^{f(x)} \underbrace{1_{0<u<f(x)}}_{=f(x,u)} du$$

  - Na f je možno nahlížet jako na marginální hustotu sdružené distribuce

$$(x, u) \sim \mathcal{U}\{(x, u): 0 < u < f(x)\}$$

  - Základni teorém vzorkování – vz. $x \sim f(x)$ je ekvivalentní k vz. - $(x, u) \sim \mathcal{U}\{(x, u): 0 < u < f(x)\}$
    - Lze využít oklikou – navzorkovat (x, u) z větší množiny a vybrat takové dvojice, pro něž je podmínka $0 < u < f(x)$ splněná
- Algoritmus
  - Předpokládáme $\int_a^b f(x)dx = 1, m > f(x) \forall x \in [a, b]$
  - Vzorkujeme $(x', u) \sim \mathcal{U}(0 < u < m)$ tak, že:
    I. Navzorkujeme $x' \sim \mathcal{U}(a, b)$
    II. Navzorkujeme $u|x = x' \sim \mathcal{U}(0, m)$
    III. Vzorek přijmeme, pokud $0 < u < f(x')$
  - Lze vylepšit, bo lze vzorkovat rovnou přes množinu:

$$\mathcal{L} = \{(x', u): 0 < u < m(x')\}, m(x) \geq f(x')$$

    - $m(x)$ není hustota, můžeme ale využít vhodný proposal $g(x)$, který hustotou bude, a vhodné kladné číslo $M > 0$ a nastavit $m(x) = Mg(x)$
  - Takhle dostaneme lepší algoritmus:
    I. Navzorkujeme $x' \sim g$
    II. Navzorkujeme $u \sim \mathcal{U}(0,1)$
    III. Vzorek x' přijmeme, když $u \leq \frac{f(x')}{Mg(x')}$
    - Kritérium přijetí je stejné, jako v základní metodě
    - Pravděpodobnost přijetí je $1/M$ – čím blíže je M k vzorkované hustotě, tím vyšší je četnost přijatých vzorků
- Rejection sampling je efektivnější než MC, ale problémy s efektivitou v oblastech, kde je hustota koncentrovaná na malou podmnožinu, nebo tam, kde nabývá moc nízkých hodnot a většina vz. není přijata (a taky nepotřebujeme integrovat)

Importance sampling

- **Problém** s efektivitou **rejection samplingu** – kompenzuje fakt, že k vzorkování používá jinou hustotou tím, že **přijímá či zahazuje** vzorky
- **Importance sampling** – kompenzuje jinou vzorkovací hustotu **přidělováním vah všem vzorkům**
- $f(x)$ cílová (komplikovaná) hustota a $g(x)$ proposal hustota, pak:

$$\int f(x)dx = \int g(x)\frac{f(x)}{g(x)}dx = \int g(x)w(x)dx$$

$$=w(x)$$

  - Podmínka $g(x) > 0$ tam, kde $f(x) > 0$
  - Výpočet vah – vložení hodnoty x do známých funkcí $f()$ a $g()$
  - Zobecnění – pro střední hodnotu $E[x]$ při $f(x)$:

$$\mathbb{E}_f[x] = \int x \cdot f(x)dx = \int x \cdot \frac{f(x)}{g(x)}g(x)dx = \int x \cdot w(x)g(x)dx = \mathbb{E}_g[x \cdot w(x)]$$

    - Podmínka $g(x) > 0$ tam, kde $xf(x) \neq 0$
- **Odhad střední hodnoty** = vážený průměr vzorků:

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i w(x_i)$$

- **Normované váhy** – protože váhy z předchozí rovnice nejsou normované a nesčítají se do 1
  - **Normování** zde zastupuje faktor $1/N$ a mohli bychom tedy uvažovat:

$$w'(x_i) = \frac{w(x_i)}{N} \qquad \text{kde} \qquad \sum_{i=1}^{N} w'(x_i) \neq 1$$

  - Z praktických důvodů často použití **normované varianty vah**

$$W(x_i) = \frac{w(x_i)}{\sum_{i=1}^{N} w(x_i)}$$

  - **Odhad střední hodnoty** je pak vážený průměr v podobě:

$$\hat{\mu} = \sum_{i=1}^{N} x_i W(x_i)$$

    - Odhad sice vychýlený, ale může mít nižší varianci a nezávisí na normalizační konstantě
- **Algoritmus IS** – je o **určování vah**, které pak používáme např. při výpočtu integrálů
  - Nagenerujeme N vzorků z proposal hustoty $x_i \sim g(x)$
  - Spočteme hodnotu hustoty $f(x_i)$
  - Spočteme hodnotu hustoty $g(x_i)$
  - Spočteme váhy $w(x_i) = \frac{f(x_i)}{g(x_i)}$
  - Váhy normalizujeme, buď $W(x_i) = w(x_i)/\sum w(x_i)$, nebo $w'(x_i) = w(x_i)/N$

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

$$A = QR$$

- **Ortogonální matice** = matice, jejíž sloupce jsou navzájem ortogonální a mají jednotkovou velikost
- Platí $Q^T A = R$ – matice R vznikne ze stejně velké matice A vynásobením ortogonální Q^T, která vynuluje v matici A všechny složky matice A pod diagonálou
  - Složité → nacházíme ortogonální matice $Q_1 \dots Q_k$, které postupně zvětšují počet nul pod diagonálou A:
    - $Q_1 A$ má aspoň o 1 nulu pod diagonálou víc než A
    - $Q_2 Q_1 A$ má aspoň o 1 nulu pod diagonálou víc než $Q_1 A$
    - …
    - $Q_k \dots Q_2 Q_1 A$ má pod diagonálou samé 0 a je rovna R
  - Tzn. Hledáme ortogonální matice Q_i – součin o. matic je o. matice
- **Givensovy rotace** – matice Q_i jsou jednoduché – jen na 4 místech se liší od jednotkové matice – rotace 2 složek vektoru, roztažení rotace ve 2 dimenzích
  - Hledáme ortogonální matici $S \in \mathbb{R}^{2,2}$, která otočí nenulový vektor $x = (a, b)^T$ tak, že výsledný vektor leží na ose x → 2. složka je 0
  - 1. sloupec matice S: $(\alpha, \beta)^T$
  - Celá matice S:

$$S = \begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix}$$

  - Platí $\alpha^2 + \beta^2 = 1$ → vlastnost rotace na osu x znamená, že musí pro vektor x platit:

$$\begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \pm||x|| \\ 0 \end{pmatrix} = \begin{pmatrix} \pm\sqrt{a^2 + b^2} \\ 0 \end{pmatrix}$$

    - 1. složka je $\pm||x||$, protože výsledný vector musí mít stejnou normu jako původní
      - Otáčíme na kladnou nebo zápornou část osy x, podle toho znaménko
  - → podmínka na parametry α, β: $a\beta + b\alpha = 0 \quad \Leftrightarrow \quad a\beta = -b\alpha$, což vede (i z definice) na soustavu:

$$\alpha^2\beta^2 = b^2\alpha^2$$
$$\alpha^2 + \beta^2 = 1$$
$$\rightarrow \alpha = \frac{a}{\sqrt{a^2+b^2}} = \boldsymbol{cos\varphi}, \beta = -\frac{b}{\sqrt{a^2+b^2}} = \boldsymbol{-sin\varphi}$$

- **QR rozklad pomocí rotací** – hledáme QR rozklad $A \in \mathbb{R}^{5,3}$ a chceme matici, která vynuluje A_{21}

$$\mathbf{A} = \begin{pmatrix} a_1 & \bullet & \bullet \\ b_1 & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix}$$

  - Výpočet parametrů podle vzorců výše → získáme ortogonální Q_1, která vynuluje b_1:

$$\underbrace{\begin{pmatrix} \alpha_1 & -\beta_1 & 0 & 0 & 0 \\ \beta_1 & \alpha_1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{Q_1}} \begin{pmatrix} a_1 & \bullet & \bullet \\ b_1 & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix} = \begin{pmatrix} \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix} = \underbrace{\begin{pmatrix} a_2 & \bullet & \bullet \\ 0 & \bullet & \bullet \\ b_2 & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix}}_{\mathbf{Q_1 A}}$$

    - Oranžová = prvek byl vynásobením změněn

- V $Q_1 A$ provedeme zase stejný výpočet parametrů a podle toho zvolíme Q_2:

$$\underbrace{\begin{pmatrix} \alpha_2 & 0 & -\beta_2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \beta_2 & 0 & \alpha_2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{Q_2}} \underbrace{\begin{pmatrix} a_2 & \bullet & \bullet \\ 0 & \bullet & \bullet \\ b_2 & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix}}_{\mathbf{Q_1 A}} = \begin{pmatrix} \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix} = \underbrace{\begin{pmatrix} a_3 & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & \bullet & \bullet \\ b_3 & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix}}_{\mathbf{Q_2 Q_1 A}}$$

  - Analogicky Q_3, Q_4 – vynulování prvků b_3, b_4:
  - 1. sloupec hotový – v 2. sloupci nulujeme b_2:
    - Q_5 mění jen 2. a 3. řádek – v 1. sloupci jsou 0, takže nevyrobíme ne0 tam, kde už byly

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \alpha_5 & -\beta_5 & 0 & 0 \\ 0 & \beta_5 & \alpha_5 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{Q_5}} \underbrace{\begin{pmatrix} \bullet & \bullet & \bullet \\ 0 & a_5 & \bullet \\ 0 & b_5 & \bullet \\ 0 & \bullet & \bullet \\ 0 & \bullet & \bullet \end{pmatrix}}_{\mathbf{Q_4 Q_3 Q_2 Q_1 A}} = \begin{pmatrix} \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & 0 & \bullet \\ 0 & \bullet & \bullet \\ 0 & \bullet & \bullet \end{pmatrix} = \underbrace{\begin{pmatrix} \bullet & \bullet & \bullet \\ 0 & a_6 & \bullet \\ 0 & 0 & \bullet \\ 0 & b_6 & \bullet \\ 0 & \bullet & \bullet \end{pmatrix}}_{\mathbf{Q_5 Q_4 Q_3 Q_2 Q_1 A}}$$

  - Postupně nulujeme další b, dokud nedostaneme:

$$\mathbf{Q_9 Q_8 \cdots Q_2 Q_1 A} = \begin{pmatrix} \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & 0 & \bullet \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \mathbf{R}$$

  - To je už ten QR rozklad: $\mathbf{A} = \mathbf{QR}, \quad \text{kde} \quad \mathbf{Q} = \mathbf{Q}_1^T \mathbf{Q}_2^T \cdots \mathbf{Q}_8^T \mathbf{Q}_9^T$
- Celkový odhad složitosti = $O(mn^2)$
- 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. $\mathbb{R}^3$- hledáme 2 rozměrné zrcadlo, které nastavíme vektoru tak, aby odraz ležel na ose x
  - Tím se vynulují všechny složky kromě 1. – rovna normě vektoru
  - Hledáme **ortogonální matice P**, pro které:

$$\mathbf{P}\boldsymbol{x} = \mathbf{P}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \pm\|\boldsymbol{x}\| \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- **Zrcadlo v n-dimenzionálním prostoru** = nadrovina procházející počátkem souřadnic
  - Popsatelná pomocí normálového vektoru u, který je na ni kolmý (ortogonální), předp. $||u|| = 1$
  - Rovina = všechny vektory x ortogonální s u: $x^T = 0$
    - Lineární rovnice pro n neznámých – množina řešení je podprostor dimenze $n - 1$
- **Zrcadlení**
  - Vzdálenost x od u = $d \rightarrow$ bod na zrcadle nejbližší vektoru x = vektor $x - du$
  - **Zrcadlový obraz x** získáme posunutím o stejnou vzdálenost za zrcadlo:

$$z.o.x = \tilde{x} = x - 2du$$

  - Vzdálenost $d = u^T x \rightarrow \tilde{\boldsymbol{x}} = x - 2u^T xu = x - 2uu^T x = (\boldsymbol{I} - \boldsymbol{2uu^T})\boldsymbol{x}$
    - $\boldsymbol{P = I - 2uu^T}$
    - P reprezentuje zrcadlení v zrcadle s normálovým jednotkovým vektorem u
    - P je rovná svojí transpozici → je symetrická → je ortogonální

- Zrcadlení nemění velikost – když aplikujeme zrcadlo 2x, dostaneme se tam, kde jsme začali

- Volba **vektoru $\boldsymbol{u}$, aby zrcadlil $\boldsymbol{x}$ na osu** odpovídající 1. souřadnici – aby platilo

$$\mathbf{P}\boldsymbol{x} = (\mathbf{I} - 2\boldsymbol{u}\boldsymbol{u}^T)\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \alpha \underbrace{\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{\boldsymbol{e}_1}$$

  - $\alpha = \pm||x||$

$$\boldsymbol{x} - 2\boldsymbol{u}\boldsymbol{u}^T\boldsymbol{x} = \alpha\boldsymbol{e}_1 \quad \underbrace{\Longrightarrow}_{2\boldsymbol{u}^T\boldsymbol{x} \text{ je číslo}} \quad \boldsymbol{u}(2\boldsymbol{u}^T\boldsymbol{x}) = \boldsymbol{x} - \alpha\boldsymbol{e}_1.$$

  - = vektor $\boldsymbol{u}$ je násovek vektoru $\boldsymbol{x} - \boldsymbol{\alpha e}$
  - Velikost $\boldsymbol{u}$ = 1 → musí platit

$$\boldsymbol{u} = \pm\frac{\boldsymbol{x} - \alpha\boldsymbol{e}}{\|\boldsymbol{x} - \alpha\boldsymbol{e}\|} = \pm\left(\frac{x_1-\alpha}{\|\boldsymbol{x}-\alpha\boldsymbol{e}\|} \quad \frac{x_2}{\|\boldsymbol{x}-\alpha\boldsymbol{e}\|} \quad \cdots \quad \frac{x_n}{\|\boldsymbol{x}-\alpha\boldsymbol{e}\|}\right)^T$$

    - Možnost vybrat si znaménko = zrcadlo má 2 opačné normálové vektory

$$\|\boldsymbol{x}-\alpha\boldsymbol{e}\| = \sqrt{(x_1-\alpha)^2 + x_2^2 + \cdots + x_n^2} = \sqrt{\alpha^2 - 2\alpha x_1 + \underbrace{x_1^2 + x_2^2 + \cdots + x_n^2}_{=\|\boldsymbol{x}\|^2=\alpha^2}} = \sqrt{2\alpha(\alpha - x_1)}.$$

$$\mathbf{P}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & & & & \\ 0 & & \mathbf{I}_4 - 2\boldsymbol{u}_2\boldsymbol{u}_2^T & & \\ 0 & & & & \\ 0 & & & & \end{pmatrix}$$

  - → složky $\boldsymbol{u}$:

$$u_1 = \frac{x_1 - \alpha}{\sqrt{2\alpha(\alpha - x_1)}} = -\frac{\sqrt{\alpha - x_1}}{\sqrt{2\alpha}}$$

$$u_i = \frac{x_i}{\sqrt{2\alpha(\alpha - x_1)}} = \left\{\text{platí } \sqrt{\alpha - x_1} = -\sqrt{2\alpha}u_1\right\} = -\frac{x_i}{2\alpha u_1}.$$

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

$$\mathbf{A} = \begin{pmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix} = \begin{pmatrix} x_1^{(1)} & \bullet & \bullet \\ x_2^{(1)} & \bullet & \bullet \\ x_3^{(1)} & \bullet & \bullet \\ x_4^{(1)} & \bullet & \bullet \\ x_5^{(1)} & \bullet & \bullet \end{pmatrix} \quad \rightarrow \quad \mathbf{P}_1\boldsymbol{x}^{(1)} == \begin{pmatrix} \alpha_1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

  - $\alpha_1 = \pm\left|\left|x^{(1)}\right|\right|$

$$\rightarrow \quad \mathbf{P}_1\mathbf{A} = \mathbf{P}_1\begin{pmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix} = \underbrace{\begin{pmatrix} \alpha_1 & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & \bullet & \bullet \\ 0 & \bullet & \bullet \end{pmatrix}}_{\mathbf{P}_1\mathbf{A}} = \begin{pmatrix} \bullet & \bullet & \bullet \\ 0 & x_1^{(2)} & \bullet \\ 0 & x_2^{(2)} & \bullet \\ 0 & x_3^{(2)} & \bullet \\ 0 & x_4^{(2)} & \bullet \end{pmatrix}$$

  - 2. krok – vektor $x^{(2)}$ má jen 4 složky → P_2 bude taková, aby se k 1. souřadnici chovala jako jednotková matice a zrcadlila pouze zbylé 4
  - Vektor $u_2 \in \mathbb{R}^4$ tak, aby matice uvnitř zrcadlila $x^{(2)}$ na $(\alpha_2, 0, 0, 0)^T$
  - 3. krok:

$$\mathbf{P}_2\mathbf{P}_1\mathbf{A} = \mathbf{P}_2\underbrace{\begin{pmatrix} \bullet & \bullet & \bullet \\ 0 & x_1^{(2)} & \bullet \\ 0 & x_2^{(2)} & \bullet \\ 0 & x_3^{(2)} & \bullet \\ 0 & x_4^{(2)} & \bullet \end{pmatrix}}_{\mathbf{P}_1\mathbf{A}} = \underbrace{\begin{pmatrix} \bullet & \bullet & \bullet \\ 0 & \alpha_2 & \bullet \\ 0 & 0 & x_1^{(3)} \\ 0 & 0 & x_2^{(3)} \\ 0 & 0 & x_3^{(3)} \end{pmatrix}}_{\mathbf{P}_2\mathbf{P}_1\mathbf{A}}$$

- 4. krok - P_3 – vyrobí poslední 2 nuly analogicky
- Finální QR rozklad:

$$\mathbf{A} = \mathbf{QR}, \quad \text{kde } \mathbf{R} = \mathbf{P}_3\mathbf{P}_2\mathbf{P}_1\mathbf{A} \quad \text{a} \quad \mathbf{Q} = \mathbf{P}_1^T\mathbf{P}_2^T\mathbf{P}_3^T.$$

- Matic P_i potřeba tolik, kolik je sloupců matice A = n

- Složitost = $O(m^2n)$ (# 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 $A \in \mathbb{R}^{n,n}$ a hledáme její vlastní čísla
  1. Najdeme QR rozklad: $A = QR$
     - Matice A, R si nejsou podobné, ale podobné jsou si A a RQ
  2. $A = QR \Rightarrow R = Q^TA \Rightarrow RQ = Q^TAQ$
     - Díky ortogonalitě $Q^T = Q^{-1} \rightarrow A$ a RQ jsou podobné → stejná vlastní čísla
  3. Přes 2. konstruuji **posloupnost matic podobných matici A**:

$$A_0 = A = Q_0R_0$$
$$A_i = R_{i-1}Q_{i-1} = Q_iR_i$$

  → konstruujeme posloupnost matic $(A_i)_{i\geq 0}$, které jsou si navzájem podobné

     - Často konvergence k horní trojúhelníkové matici
     - **Zrychlení konvergence** pomocí Hessenbergovy formy
- **Hessenbergova forma matice** – matice $A \in \mathbb{R}^{n,n}$ je v H. formě, pokud pro $1 \leq i, j \leq n$ platí

$$i > j + 1 \Rightarrow A_{i,j} = 0$$

  - Matice A je navíc **tridiagonální**, jestliže platí

$$(i > j + 1 \; nebo \; i < j - 1) \Rightarrow A_{i,j} = 0$$

$$\begin{pmatrix} \bullet & \bullet & 0 & 0 \\ \bullet & \bullet & \bullet & 0 \\ 0 & \bullet & \bullet & \bullet \\ 0 & 0 & \bullet & \bullet \end{pmatrix}$$

- K libovolné matici $A \in \mathbb{R}^{n,n}$ existuje matice $B \in \mathbb{R}^{n,n}$ v Hessenbergově formě, která je A podobná. Matici podobnosti P lze navíc volit ortogonální a symetrickou:

$$B = PAP$$

  - Pokud je matice A symetrická, můžeme navíc B volit tridiagonální.
  - Matice P vznikne součinem $n - 2$ matic P_i provádějících Householderovy reflexe
- **Výpočet** singulárních hodnot – vlastních čísel symetrické matice A^TA, $A \in \mathbb{R}^{m,n}$
  1. Matici A^TA převedeme pomocí $n - 2$ matic P_i provádějících Householderovy reflexe na podobnou matici B, která je tridiagonální.
  2. Na matici $B_0 = B$ aplikujeme QR algoritmus.
  3. Najdeme QR rozklad $B_0 = Q_0R_0$, ten můžeme najít pomocí $n - 1$ Givensových rotací, které nulují $n - 1$ prvků pod diagonálou. Násobení maticí provádějících Givensovu rotaci vždy ovlivňuje pouze 6 prvků matice a je tedy výpočetně nenáročné
  4. Posloupnost matic $(B_i)_{i\geq 0}$:

$$B_i = R_{i-1}Q_{i-1}, \text{ její rozklad označíme } B_i = Q_iR_i$$

     - Matice B_i 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 $A \in \mathbb{R}^{m,n}$ – nehledáme její vlastní čísla, ale vl. č. matice A^TA
  - A^TA je čtvercová $n \times n$, **pozitivně semidefinitní, symetrická**
  - Symetrická → **diagonalizovatelná**, má reálná (nezáporná) vlastní čísla a z vlastních vektorů lze vytvořit ortonormální bázi prostoru $\mathbb{R}^n$
  - $h(A) = r$, hodnost nesmí být větší než počet sloupců/řádků ($r \leq \min\{m,n\}$). $h(A^TA) = r$
  - A^TA má n vlastních čísel. Jelikož hodnost je r, je 0 vlastní číslo právě když $r < n$. V takovém případě má vlastní číslo 0 násobnost $n - r$.
  - Matice má tedy r **kladných vlastních čísel**, které seřadíme od největšího po nejmenší a označíme σ_i^2: $\sigma_1^2 \geq \sigma_2^2 \geq \cdots \geq \sigma_r^2 > 0$.
  - Ke každému vlastnímu číslu umíme najít lineárně nezávislých (+ ortonormálních) vlastních vektorů, kolik je násobnost daného vlastního čísla. Získáme soubor pro všechna $i = 1, 2, \ldots r$

$$(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r), \quad \text{kde} \quad \mathbf{A}^T\mathbf{A}\boldsymbol{v}_i = \sigma_i^2\boldsymbol{v}_i$$

  - Je-li $r = n$, tvoří tento soubor bázi $\mathbb{R}^n$. Je-li $r < n$, můžeme k němu připojit $n - r$ **vlastních vektorů** $\boldsymbol{v}$ příslušejících k vlastnímu číslu 0 a vyrobit tak ortonormální bázi i tak
- **Odvození** SVD rozkladu
  - Pro každé $i = 1,2, \ldots, r$ definujeme vektor u_i:

$$u_i = \frac{1}{\sigma_i} A v_i$$

  → soubor vektorů, o kterém platí:
    - Je ortonormální:

$$u_i^T u_j = \begin{cases} 0 & pro\ i \neq j \\ 1 & pro\ i = j \end{cases}$$

    - Pro každé $i = 1,2, \ldots, r$:

$$AA^T u_i = \sigma_i^2 u_i$$

  → $\boldsymbol{u_i}$ **je vlastní vektor** čtvercové, symetrické a pozitivně semidefinitní matice AA^T příslušející jejímu kladnému vlastnímu číslu σ_i^2
  → A^TA a AA^T mají stejná kladná vlastní čísla
  - Ze souborů $(v_1 \ldots v_n)$ a $(u_1 \ldots u_m)$ vyrobíme matice $V \in \mathbb{R}^{n,n}, U \in \mathbb{R}^{m,m}$ tak, že tyto soubory napíšeme jako jejich sloupce.
  - Jelikož jsou oba soubory ortonormální, jsou obě matice ortogonální:

$$V^TV = I_n, U^TU = I_m$$

  - $\Sigma \in \mathbb{R}^{m,n}$ matice, která má na **diagonále postupně vlastní č.** $\boldsymbol{\sigma_1 \ldots \sigma_r}$, takto seřazená podle velikosti
    - Je-li $r < \min\{m,n\}$, **doplníme na diagonálu nuly** (doplnění nejmenšího vlastního čísla 0)
    - Např. Pokud $m > n = r + 1$ ($\rightarrow h(A) = n - 1$), bude matice Σ:

$$\begin{pmatrix} \sigma_1 & 0 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & 0 & \cdots & \sigma_r & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{m,n}$$

  → s tímto značením a jelikož pro $i = 1,2, \ldots, r$: $Av_i = \sigma_i u_i$ a pro $i = r+1, r+2, \ldots, n$: $Av_i = \theta$ dostaneme:

$$AV = U\Sigma$$

- **Definice SVD rozkladu** (fucking finally) – Buď $A \in \mathbb{R}^{m,n}$ s hodností r. Potom existují ortogonální matice $V \in \mathbb{R}^{n,n}, U \in \mathbb{R}^{m,m}$ a diagonální matice $\Sigma \in \mathbb{R}^{m,n}$, která má na diagonále kladná čísla

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$$

doplněná příslušným počtem nul, takové, že

$$\boldsymbol{A = U\Sigma V^T}$$

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

  - Singulární hodnoty σ_i jsou odmocniny vlastních čísel matic A^TA a AA^T (proto ta nezápornost, AHA)
  - Sloupce matic V, U 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 V. Sloupce v U se pak dopočítají
- Supr trik – v maticích U, V, Σ lze uvažovat jen prvních r sloupců a řádků, ostatní jsou nulové
  → $\boldsymbol{A = U_r\Sigma_r V_r^T}$
  → A lze zrekonstruovat z těchto r sloupců
  - Dokonce platí, že podprostor generovaný sloupci matice A = lineární obal souboru $(u_1 \ldots u_r)$

Aproximace maticí s nižší hodností

- Když vynásobíme matice $m \times d$ a $d \times n$, dostaneme matici $m \times n$ – lze využít ke kompresi
- Mějme matici $A \in \mathbb{R}^{m,n}$, jaké je nejmenší d takové, že existují $B \in \mathbb{R}^{m,d}$ a $C \in \mathbb{R}^{d,n}$ pro které

$$A = BC \text{ ?}$$

- Hodnost součinu nemůže být vyšší, než hodnost jednotlivých matic, $h(A) = r$ a $h(B) = h(C) = d$, musí být $d \geq r$ (kdyby ne, BC by mělo nižší hodnost než A a nefungovalo by to). B a C lze najít pro $d = r$:

$$B = U_r, C = \Sigma_r V_r^T$$

→ nejmenší $\boldsymbol{d}$ je rovno $\boldsymbol{r = h(A)}$

- Máme-li zadané d, jaké jsou matice $B \in \mathbb{R}^{m,d}$ a $C \in \mathbb{R}^{d,n}$ takové, že jejich **součin $\boldsymbol{BC}$ je co nejblíže $\boldsymbol{A}$** $\in \mathbb{R}^{m,n}$?
  → SVD
- **Vzdálenost** 2 stejně velkých matic = norma jejich rozdílu: $||A - \bar{A}||$
  - **Frobeniova norma** matice $A \in \mathbb{R}^{m,n}$ = suma čtverců odchylek v jednotlivých složkách

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}$$

  - Souvislost s SVD – je to odmocnina ze součtu kvadrátů singulárních hodnot

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{r} \sigma_i^2}$$

- **Eckart-Young-Mirskyho** věta – Mějme matici $A \in \mathbb{R}^{m,n}$ hodnosti r a buď $U\Sigma V^T$ její SVD rozklad. Označme U_k, Σ_k, V_k sloupcové matice, které vznikly z U, Σ, V tak, že jsme vzali pouze jejich prvních k sloupců. Potom pro každé kladné přirozené číslo $d \leq r$ je řešením následující úlohy s neznámou $\bar{A} \in \mathbb{R}^{m,n}$

$$minimalizujte\ ||A - \bar{A}||_F, kde\ h(\bar{A}) \leq d$$

matice

$$A_d^* = U_d \Sigma_d {V_d}^T$$

- Neboli:

$$\underset{h(\overline{\mathbf{A}}) \leq d}{\operatorname{argmin}} \|\mathbf{A} - \overline{\mathbf{A}}\|_F = \mathbf{U}_d \boldsymbol{\Sigma}_d \mathbf{V}_d^T = \sum_{i=1}^{d} \sigma_i u_i v_i^T,$$

→ d největších singulárních hodnot a k nim příslušné sloupce matic U, V definují nejlepší aproximaci maticí o hodnosti d

- Zbylých $r - d$ singulárních hodnot říká, jak dobrá je to aproximace:

$$\|\mathbf{A} - \mathbf{A}_d^*\|_F = \sqrt{\sum_{i=d+1}^{r} \sigma_i^2}.$$

- Místo Frobeniovy normy můžeme použít klasickou L_2 vektorovou normu:

$$\|\mathbf{A}\|_2 = \sup_{x \in \mathbb{R}^n} \frac{\|\mathbf{A}x\|_2}{\|x\|_2}$$

= **spektrální norma**

- Rovná se největší singulární hodnotě matice $A \|\mathbf{A}\|_2 = \sigma_1$

PCA a SVD

- SVD u PCA pro **redukci dimenzionality**
- PCA se počítá pro matici $X \in \mathbb{R}^{N,p}$ (hodnost r), o které předpokládáme, že je **vycentrovaná** = průměry příznaků ve sloupcích jsou 0

$$0 = \frac{1}{N}\sum_{i=1}^{N} X_{ij} \quad pro\ j = 1, \dots, p$$

  - Docílení **posunutím** dat o **průměry příznaků**
  - Pro takovou matici lze spočítat matici kovariancí příznaků:

$$\frac{1}{p-1} X^T X$$

  - **Hlavní komponenty** jsou dány **vlastními vektory** této matice a jejich **rozptyly** jsou příslušná **vlastní čísla**
- Souvislost se SVD rozkladem $X = U\Sigma V^T$:

$$\frac{1}{p-1} X^T X = \frac{1}{p-1} V\Sigma^T \mathrm{U}\mathrm{U}\Sigma V^T = \frac{1}{p-1} V\Sigma^T \Sigma V^T$$

- Platí, že:

$$\mathbf{\Sigma}^T\mathbf{\Sigma} = \begin{pmatrix} (\sigma_1)^2 & 0 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & (\sigma_2)^2 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 0 & (\sigma_3)^2 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & 0 & \cdots & (\sigma_r)^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{p,p}$$

  je čtvercová diagonální matice
  → dosazením jsme dostali **spektrální rozklad kovarianční matice** (vztah podobnosti matice s diagonální maticí)
- Vyplývá vztah mezi PCA a SVD:
  - **Hlavní komponenty** matice X odpovídají **prvním r sloupcům** matice V
  - Jejich **rozptyly**, navíc sestupně seřazené, najdeme jako **kvadráty singulárních hodnot** na diagonále $\Sigma^T\Sigma$, které stačí vydělit číslem $p - 1$

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

NI-PON

- **Totální derivace**
  - Mějme funkci $f: D \to \mathbb{R}^m$, kde $D \subset \mathbb{R}^n$ je otevřená množina. Řekneme, že f je **(totálně) diferencovatelná** v bodě $a \in D$, pokud existuje lineární zobrazení $\lambda: \mathbb{R}^n \to \mathbb{R}^m$ takové, že:
$$\lim_{h \to 0} \frac{\|f(a+h) - f(a) - \lambda(h)\|}{\|h\|} = 0.$$
  - Pokud je f diferencovatelná v a, pak takové lineární zobrazení existuje právě jedno **= (totální) derivace** funkce f bodě a = $\boldsymbol{Df(a)}$ nebo $f'(a)$.
  - **Jacobiho matice** = matice lineárního zobrazení $Df(a)$
  - Mějme funkci $f: D \to \mathbb{R}^m$, kde $D \subset \mathbb{R}^n$ a $a \in D$. Pokud $\partial_j f_i(a)$ existuje a je spojitá na otevřeném okolí bodu a pro všechna $i, j: 1 \leq i \leq m, 1 \leq j \leq n$ potom $Df(a)$ existuje
    = **spojitě diferencovatelná** funkce
- **Řetězové plavidlo**
  - Mějme funkci $g: D_g \to \mathbb{R}^m$, kde $D_g \subset \mathbb{R}^n$, a $f: D_f \to \mathbb{R}^q$, kde $D_f \subset \mathbb{R}^m$. Je-li funkce f diferencovatelná v $a \in D_g$ a funkce f diferencovatelná v $g(a) \in D_f$, potom je funkce $f \circ g: D_g \to \mathbb{R}^q$ diferencovatelná v a a platí: $D(f \circ g)(a) = Df(g(a)) \circ Dg(a)$
  - Speciálně pro $n = q = 1$:
$$D(f \circ g)(a) = \mathrm{D}f(g(a)) \circ \mathrm{D}g(a) = \nabla f(g(a))^T \cdot \begin{pmatrix} \partial g_1(a) \\ \vdots \\ \partial g_m(a) \end{pmatrix}$$
- **Gradient** – směr největšího růstu – na derivaci ve směru s funkce $f: D_f \to \mathbb{R}$ v bodě $a \in D_f \subset \mathbb{R}^n$ můžeme nahlížet jako na derivaci jednorozměrné funkce $h(t) = f(a + ts)$ v bodě 0, $g(t) = a + ts$. Pak
$$Dh(t) = D(f \circ g)(t) = \nabla f(g(t))^T \cdot \begin{pmatrix} \partial g_1(t) \\ \vdots \\ \partial g_m(t) \end{pmatrix} = \nabla f(a + ts)^T \cdot \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix}$$
  → pro bod 0:
$$\nabla_s f(a) = Dh(0) = \nabla f(a)^T \cdot s.$$
- **Optimalizace** → hledáme
$$\operatorname*{argmin}_{x \in D \subset \mathbb{R}^n} f(x)$$
  pro f (dvakrát) spojitě diferencovatelnou
  - Řešení iteračními metodami → konstrukce posloupnosti aproximací x_0, x_1, x_2 konvergující k bodu, který je vhodným kandidátem na lokální minimum funkce f (typicky nulový gradient)
  - **Trust region** přístup – na okolí bodu x_k vytvoříme aproximaci funkce f, označené m_k, a hledáme minimum této aproximace m_k na okolí bodu x_k
$$p_k = \operatorname*{argmin}_p \{m_k(x_k + p) \mid \|p\| \leq \Delta_k\}$$
    - Další aproximace je $x_{k+1} = x_k + p_k$
  - **Line search –** aproximaci hledáme ve směru p_k:
$$x_{k+1} = x_k + \alpha_k p_k$$
    kde $\alpha_k = argmin_{\alpha \geq 0} f(x_k + \alpha_k p_k)$ nebo nějaká aproximace řešení této podúlohy
- Line search – **směr spádu** $x_{k+1} = x_k + \alpha_k p_k$
  kde p_k je zvolený směr a α_k je délka kroku
  - Obecná volba směru = **směr spádu** (descent) = $\boldsymbol{\nabla f(x_k)^T \cdot p_k < 0}$
  - Časté volby směru: $\boldsymbol{p_k = -B_k^{-1} \nabla f(x_k)}$:
    - Metoda největšího spádu → $\boldsymbol{B_k = I}$ (steepest descent)
    - Newtonova metoda $\boldsymbol{B_k = \nabla^2 f(x_k)}$
    - Kvazi-Newtonova metoda $\boldsymbol{B_k \approx \nabla^2 f(x_k)}$
    - Je-li B_k pozitivně definitní, jedná se o spádový směr
- Ideální volba α_k do $x_{k+1} = x_k + \alpha_k p_k$ je minimem funkce
$$\phi(\alpha) = f(x_k + \alpha p_k) \quad \text{pro } \alpha \geq 0$$
  = **exact** line search

- Výpočetně složité → hledání vhodné délky kroku, která zajistí dostatečný pokles funkční hodnoty f = **inexact** line search

- **Armijova podmínka**:

$$\phi(\alpha_k) \leq \phi(0) + \alpha_k c \phi'(0).$$

kde $0 < c < 1$

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

  - Klasický algoritmus:

Algorithm 3.1: ARMIJOSTEPBACKTRACK(ϕ, γ_0, c)

$i = 0$
while $\phi(\gamma_i) \geq \phi(0) + \gamma_i c \phi'(0)$
do $\begin{cases} \gamma_{i+1} = \rho \gamma_i \\ i = i + 1 \end{cases}$
return (γ_i)

  - $\rho \in (0, 1)$

- **Goldsteinova podmínka**:

$$\phi(0) + \alpha_k(1 - c)\phi'(0) \leq \phi(\alpha_k) \leq \phi(0) + \alpha_k c \phi'(0)$$

kde $0 < c < 1/2$

→ funkční hodnota má ležet **mezi zvolenými přímkami**

→ omezuje se nevhodná možnost, že nalezený krok bude příliš malý

  - Nevýhoda – můžeme minout optimální hodnotu pro α_k

- **Wolfeho podmínky**:
  - Slabá podmínka:

$$\phi(\alpha_k) \leq \phi(0) + \alpha_k c_1 \phi'(0)$$
$$\phi'(\alpha_k) \geq c_2 \phi'(0)$$

kde $0 < c_1 < c_2 < 1$

  - Silná podmínka:

$$\phi(\alpha_k) \leq \phi(0) + \alpha_k c_1 \phi'(0)$$
$$|\phi'(\alpha_k)| \leq c_2 |\phi'(0)|$$

kde $0 < c_1 < c_2 < 1$

  - Přidání absolutní hodnoty → derivace $\phi'(\alpha_k)$ nebude mít moc velké hodnoty
  - Pro spojitě diferencovatelnou a sdola omezenou f vždy existuje α_k splňující Wolfeho podmínky

- **Metoda největšího spádu**
  - $x_{k+1} = x_k + \alpha_k p_k$ a $\alpha_k = argmin_{\alpha \geq 0} \phi(\alpha)$
  - **Největší spád:** $\boldsymbol{p_k = -\nabla f(x_k)}$ (ajo my vlastně jdeme proti směru gradientu, co? PROTI SMĚRU NEJVĚTŠÍHO RŮSTU)
    - $(B_k = I)$
  - Platí $p_{k+1}^T \cdot p_k = 0$

Věta 3.1 Nechť f je dvakrát spojitě diferencovatelná a předpokládejme, že algoritmus největšího spádu konverguje k x^*, ve kterém je $\nabla^2 f(x^*)$ pozitivně definitní.

Nechť r je číslo, pro které platí

$$r \in \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}, 1 \right),$$

kde λ_n je největší a λ_1 nejmenší vlastní číslo matice $\nabla^2 f(x^*)$. Pro dostatečně velké k platí

$$f(x_{k+1}) - f(x^*) \leq r^2 \left(f(x_k) - f(x^*) \right)$$

  - $\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} = \frac{\frac{\lambda_n}{\lambda_1} - 1}{\frac{\lambda_n}{\lambda_1} + 1} = \frac{\kappa(\nabla^2 f(x^*)) - 1}{\kappa(\nabla^2 f(x^*)) + 1}$ poslední rovnost platí při použití mat. normy indukované E. normou
  - Obecně vyžaduje vyšší počet kroků ke konvergenci, pro hůř podmíněnou je moc pomalá – při použití nepřesné volby délky kroku nebude rychlejší

- Newtonova metoda
  - Máme aproximaci x_k a posunutí p. Pro $f(x_k + p)$ platí:
$$f(x_k + p) \approx f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 f(x_k) p = m_k(p)$$
  - Optimální p → **minimalizace** funkce m_k:
$$p_k = \operatorname*{argmin}_p m_k(p)$$
  - Po parciální derivaci 1. rovnice:
$$\nabla m_k(p) = 0 \iff \nabla f(x_k) + \nabla^2 f(x_k) p = 0$$
$$p_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$
  - Pokud není $\nabla^2 f(x_k)$ **pozitivně definitní**, pak p_k nemusí být spádový směr!

> **Věta 3.2** Nechť f je dvakrát spojitě diferencovatelná, $\nabla^2 f(x)$ je Lipschitzovsky spojitá na okolí H bodu minima x^* ($\exists L, \forall x_0, x_1 \in H, \|\nabla^2 f(x_1) - \nabla^2 f(x_0)\| \leq L\|x_1 - x_0\|$), $\nabla f(x^*) = 0$ a $\nabla^2 f(x^*)$ je pozitivně definitní. Pro Newtonovu metodu s $\alpha_k = 1$ platí
>
> 1. je-li x_0 dostatečně blízko x^*, potom $(x_k)_{k=0}^{+\infty}$ konverguje k x^*;
> 2. $(x_k)_{k=0}^{+\infty}$ konverguje kvadraticky;
> 3. posloupnost $(\|\nabla f(x_k)\|)_{k=0}^{+\infty}$ konverguje kvadraticky k nule.

  - Poblíž řešení bude délka kroku $\alpha_k = 1$ 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 $\nabla^2 f(x_k)$ (= Hessián) nemusí být pozitivně definitní → nemusí existovat inverze
    - **Inverzi chceme**, abychom mohli vyřešit $B_k p_k = -\nabla f(x_k)$ → modifikace
    - Nastavení matice $B_k = \nabla^2 f(x_k) + E_k$ tak, aby B_k byla **pozitivně definitní**:
      - Změna záporných vlastních čísel
      - Přičtení kladné diagonální matice
      - Modifikace nějakého rozkladu matice $\nabla^2 f(x_k)$
    - Pokud jsou čísla podmíněnosti matice B_k omezená, pak lze dokázat, že modifikace bude konvergovat
- Stochastický gradientní sestup
  - Minimalizovaná funkce ve tvaru sumy:
$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$
    f_i ... ztrátová funkce pro i-tý vzorek
    x ... parametry modelu
  - **SGD** (Stochastic gradient descent) – začíná s aproximací x_0 a iteruje podle schématu:
$$x_{k+1} = x_k - \eta \nabla f_{i_k}(x_k)$$
    i_k ... náhodné z uniformního rozdělení
    η ... **míra učení** (learning rate)
  - Výhoda = nízká výpočetní složitost, pro konvexní funkce konvergence skoro jistá
  - **Modifikace SGD** – většinou modifikace míry učení nebo implementace setrvačnosti (vliv předchozích změn)
    - **Adagrad**:
$$s_0 = 0,$$
$$s_k = s_{k-1} + \left(\nabla f_{i_k}(x_k)\right)^2,$$
$$x_{k+1} = x_k - \eta \frac{\nabla f_{i_k}(x_k)}{\sqrt{s_k + \varepsilon}},$$
      - Vektorové operace po složkách
      - ε, abychom nedělili 0
    - **Momentová metoda**:
$$x_{k+1} = x_k - \left(\eta \nabla f_{i_k}(x_k) + \gamma(x_k - x_{k-1})\right)$$
      - γ ... 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 x_t zíykaných v konkrétních časových okamžicích t
  - Buď $(\Omega, \mathcal{F}, P)$ pravděpodobnostní prostor a T množina indexů interpretovaných jako čas. Časovou čadou nazýváme množinu $\{x_t, t \in T\}$, kde x_t jsou náhodné veličiny z $(\Omega, \mathcal{F}, P)$
  - Pokud je t 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
    - Y_t … pozorovaná veličina v čase t
    - T_t … hodnota trendu
    - S_t … sezónní složka
    - E_t … nevysvětlitelná složka
  - 2 modely:
    - **Aditivní**: $Y_t = T_t + S_t + E_t$
      - Amplituda sezónních složek je cca stejná
    - **Multiplikativní** = $Y_t = T_t \cdot S_t \cdot E_t$
      - S rostoucím trendem se zvyšuje i sezónní amplituda
  - **Výběr modelu** – minimalizace součtu čtverců hodnot **autokorelační funkce** reziduí E_t – říká, jaká míra korelace v E_t 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 $R(s,t)$ pro časy s, t
- **Náhodný proces** = posloupnost náhodných veličin x_t, kde t je z vhodné množiny indexů
- **Momenty** – popisují časové řady
  - **Střední hodnota**: $\mu_t = E[x_t]$
  - **Variance**: $\sigma_t^2 = var\ x_t$
  - **Autokovariance**: $\gamma(t_1, t_2) = cov(x_{t1}, x_{t2}) = E[(x_{t1} - \mu_{t1}) \cdot (x_{t2} - \mu_{t2})]$

Stacionarita

- **Striktní stacionarita** (silná) – řada je striktně s., pokud sdružená distribuce $x_{t1}, \dots, x_{tn}$ je stejná, jako sdružená distribuce $x_{t1+\tau}, \dots, x_{tn+\tau}$ pro všechna $t_1, \dots, t_n, \tau$
  - Libovolný posun o čas τ nemá vliv na sdruženou distribuci a ta tedy závisí jen na časech $t_1, \dots, t_n$ pro libovolná n
- **Slabá stacionarita** – řada je slabě s., pokud je invariantní vůči posunům v čase pouze v rámci momentů rozdělení do druhého řádu:

$$E[x_t] = \mu$$
$$cov(x_t, x_{t+\tau}) = \gamma(\tau)$$

- Trend → řada není stacionární
- Typy procesů podle stacionarity:
  - Stacionární – bez trendu a jednotkových kořenů, silně či slabě
    - Pokud je $\{x_t, t \in T\}$ slabě stacionární proces, potom autokorelace závisí pouze na zpoždění mezi časy t a s: $R(s,t) = R(s-t) = R(\tau)$
  - 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:

$$H_0: proces\ má\ jednotkový\ kořen$$
$$H_A: proces\ nemá\ jednotkový\ kořen\ (ale\ kořeny\ vně\ jednotkové\ kružnice)$$

  - KPSS – neparametrický test
    - Hypotézy:

$$H_0: proces\ je\ trend-stacionární$$
$$H_A: proces\ není\ stacionární\ (má\ jednotkový\ kořen)$$

    - Proces H_0 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:
  - $E[x_t] = 0$
  - $var(x_t) = \sigma^2 < \infty$
  - $cov(x_t, x_{t+\tau}) = \gamma(\tau) = 0$
  - Normální bílý šum:
  - Zvuková syntéza, generátory náhodných čísel
- Náhodná procházka – uvažujeme diskrétní bílý šum $z_t \sim \mathcal{L}(0, \sigma^2)$. Proces $\{x_t\}$ je náhodný proces, pokud:

$$x_0 = 0$$
$$x_t = x_{t-1} + z_t$$
$$\rightarrow\ x_t = \sum_{i=1}^{t} z_t$$

  - $E[x_t] = 0$
  - $var(x_t) = t \cdot \sigma^2$
  - 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

$$\rho_{XY} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$$

    - $\rho \in [-1, 1]$
    - $\rho_{XY} = 0 \Rightarrow X$ a Y jsou nezávislé (opačně neplatí)
    - σ_{XY} = kovariance
  - Výběrový korelační koeficient – pomocí výběrových variancí
  - Parciální korelační koeficient
    - 2 náhodné veličiny X a Y, mezi kterými existuje závislost
    - X a Y ovlivněny třetí n-rozměrnou náhodnou veličinou Z
    - Pro měření korelace X a Y je potřeba je od vlivu Z očistit
    - Nalezneme regresní přímky:

$$\hat{X} = a + b^T Z$$
$$\hat{Y} = c + d^T Z$$

      - $\hat{X}, \hat{Y}$ … nejlepší lineární přiblížení k X a Y
      - $e_X = (X - \hat{X}), e_Y = (Y - \hat{Y})$ … rezidua očištěná od vlivu Z
    - Jejich korelační koeficient = parciální korelační koeficient mezi X a Y při daném $\boldsymbol{Z}$ – $\rho_{XY \cdot Z}$
    - 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 $R(s,t)$ pro časy s a t
    - Pokud je $\{X_t, t = 1,2, \dots\}$ slabě stacionární proces (= existují časově invariantní první 2 momenty μ, σ^2), potom autokorelace závisí pouze na zpoždění mezi t a s:

$$R(s,t) = R(s-t) = R(\tau)$$

  - Parciální autokorelační funkce
    - Parciální autokorelace zpoždění $k \geq 1 - \alpha(k)$
    - (auto)korelace mezi X_t a X_{t+k} s odstraněním lineárního vlivu mezilehlých hodnot $X_{t+1}, \dots, X_{t+k-1}$
- Informační kritéria
  - AIC – Akaikeho informační kritérium

$$AIC = 2k - 2\ln \mathcal{L}$$

    - k … počet odhadovaných parametrů
    - $\mathcal{L}$ … maximální hodnota věrohodnosti při daném modelu
    - Asymptoticky ekvivalentní ke křížové validaci
  - BIC – Bayesovské informační kritérium

$$BIC = k\ln(n) - 2\ln \mathcal{L}$$

    - n … počet pozorování
  - Hodnotu AIC/BIC chceme minimalizovat
- Operátor zpoždění – lag operator B nebo L – pro zjednodušení zápisu

$$BX_t = X_{t-1}$$
$$B^{-1}X_t = X_{t+1}$$
$$B^k X_t = B \cdot B \cdot \dots \cdot BX_t = X_{t-k}$$

(B je tam k-krát)

Autoregresní modely (AR)

- Umožňují popis náhodného procesu na základě jeho předchozích realizací
- **Autoregresní model řádu p**:

$$x_t = c + \Phi_1 x_{t-1} + \Phi_p x_{t-p} + \varepsilon_t = c + \sum_{\tau=1}^{p} \Phi_\tau x_{t-\tau} + \varepsilon_t$$

  - ε_t … **bílý šum**
  - $\Phi = [c, \dots, \Phi_p]^T$ … vektor **regresních koeficientů**
  - Zápis pomocí **operátoru zpoždění**:

$$\varepsilon_t = \left(1 - \sum_{i=1}^{p} \Phi_i B^i\right) x_t$$

- p může být libovolné, ale vyšší řády nemusí dávat smysl – jak hlubokou minulost potřebujeme?
- AR procesy **nemusí být slabě stacionární –** kořeny charakteristické rovnice $1 - \Phi_1 z - \dots - \Phi_p z^p = 0$ musí **ležet vně jednotkové kružnice** (tzn. $\forall z_i \in \mathbb{C}: |z_i| > 1$)
- **Odhad parametrů** metodou nejmenších čtverců – odhad regresních koeficientů c a Φ_1
  - X návrhová matice, Y vektor měření, ε šum
  - $Y = X\Phi + \varepsilon \quad \rightarrow \quad \hat{\Phi} = (X^T X)^{-1} X^T Y$
- **Odhad řádu** AR modelu
  - **ACF** – postupně klesá k nule, popř. klesá shora i zdola
    - Má mnoho významných lagů (stacionární AR procesy lze konvertovat do $MA(\infty)$)
  - **PACF** – vrcholy do hodnoty řádu modelu, pak jdou strmě k nule
  - Obecný příklad (při bílém šumu ε_t) $x_t = \Phi_1 x_{t-1} + \varepsilon_t$
    - Abs. hodnoty **ACF** budou zřejmě postupně klesat, protože x_t je přímo ovlivněno x_{t-1} atd.
    - Absolutní hodnoty **PACF** budou indikovat silnou korelaci mezi x_t a x_{t-1}, 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 q:

$$x_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q}$$

  - $\varepsilon_t \sim iid\ N(0, \sigma^2)$
  - θ … váha toho, jak se šum propaguje
  - Zápis pomocí operátoru zpoždění:

$$x_t = \left(1 + \sum_{i=1}^{q} \theta_i B^i\right) \varepsilon_t$$

- Prostřednictvím **bílého šumu** vystihuje **náhodné šoky** – nezávislé a stejně rozdělené
- $E[x_t] = c, var x_t = \sigma^2 \sum_{i=0}^{q} \theta_t^2$
- Veličiny x_t a x_{t+k} jsou pro k větší než řád modelu q nekorelované, korelace x_t a x_{t+k} pro $k <$ řád nenulová
- Prohodíme-li v kovarianci pořadí, změní se znaménko u k
- **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** $MA(q)$:

$$\theta(x) = 1 - \theta_1 x - \theta_2 x^2 - \dots - \theta_q x^q$$

  - Jsou-li kořeny char. polynomu vně jednotkové kružnice, potom je MA proces invertibilní - $AR(\infty)$
- **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 ε_t) $x_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1}$
    - x_t není ovlivněna $x_{t-1}, x_{t-2}, \dots$
    - Absolutní hodnoty **ACF** budou vysoké pro $k = 1$, 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)**

$$x_t = c + \varepsilon_t + \sum_{i=1}^{p} \Phi_i x_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

  - Neznámé $c, \Phi, \theta, \sigma^2$
  - 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):

$$\Phi X_t = \theta \varepsilon_t$$

- **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**:
  - p – řád **autoregresní** části (AR)
  - q – řád modelu **klouzavých průměrů** (MA)
  - d – řád **diferencování**
- Zápis pomocí **operátoru zpoždění** (bez konstanty c):

$$\Phi(1-B)^d X_t = \theta \varepsilon_t$$

$$\Phi = 1 - \sum_{i=1}^{p} \Phi_i B^i, \qquad \theta = 1 + \sum_{i=1}^{q} \theta_i B^i$$

- **Operátor diference ∇**:

$$\nabla X_t = X_t - X_{t-1} = (1-B) X_t$$
$$\nabla^k X_t = (1-B)^k X_t$$

- Diferencování řady → **odstranění trendu**
  - Počítáme veličiny $Y_t^{'} = Y_t - Y_{t-1}$
  - Diferencováním náhodné procházky dostaneme gaussovský bílý šum ($Y_t^{'} = \varepsilon_t$)
    - Gaussovská náhodná procházka:

$$Y_0 = 0$$
$$Y_t = Y_{t-1} \varepsilon_t$$

- 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 $\boldsymbol{d}$:
  - $d \leq 2$, 2. diference málokdy, 3. je výjimka
  - Při $d = 2$ nepoužíváme v ARIMA modelu konstantu c
- Role konstanty v ARIMA modelu:
  - $d = 0$ – konstanta zavádí nenulovou střední hodnotu a vyplatí se ji zkusit
  - $d = 1$ – konstanta zavádí nenulový „průměrný“ trend, může se vyplatit
  - $d = 2$ – 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 c: 1. $\Phi_1 = 0 \rightarrow$ bílý šum 2. $\Phi_1 = 0, c = 0 \rightarrow$ náhodná procházka 3. $\Phi_1 = 0, c \neq 0 \rightarrow$ náhodná procházka s driftem 4. $\Phi_1 < 0 \rightarrow$ řada oscilující mezi klad. a záp. hodnotami

- Obecná pravidla
  - $p = 0$ nebo $q = 0$
  - $\boldsymbol{p + q \leq 3}$ – složitější modely jsou řídké
  - Koeficienty Φ_i blízké θ_i 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 p, q 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 :(