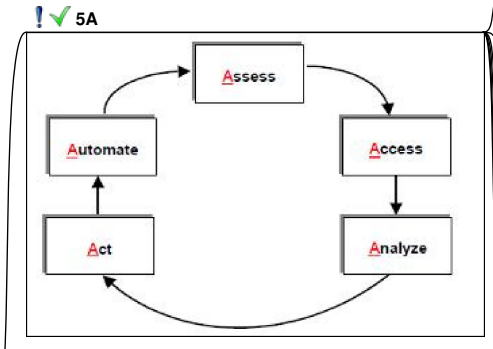
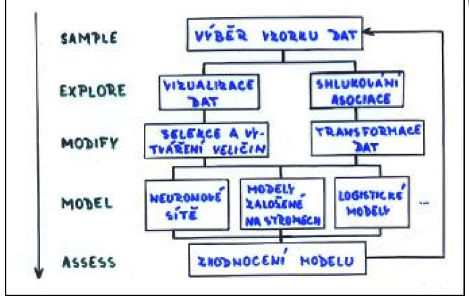
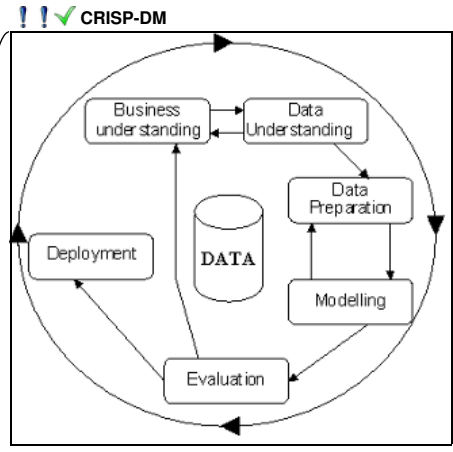
- Data-Mining
  - poskytnout uživatelům jednotný rámec pro řešení různých úloh z oblasti dobývání znalostí
  - 1. Metodiky
    - vyvinuté firmami (závislé na sw)
      - ! ✓ 5A



        - **ASSESS** – posouzení potřeb projektu
          - Stanovení kontextu – cílů, strategií a procesů
        - **ACCESS** – shromážděnípotřebných dat a jejich příprava
        - **ANALYZE** – provedení analýz
          - Data se přeměňujína informace a znalosti
          - → použít vícero metod a porovnat jejich výsledky a efektivitu
        - **ACT** – přeměna znalostí na akční znalosti
          - Doporučení, dodatečné otázky a následná rozhodnutí
          - → nalezené výsledky by měly být prezentovány jasně a srozumitelně
        - **AUTOMATE** – převedenívýsledkůanalýzy do praxe
          - Může zahrnovat např. i vytvoření praktického rozhraní pro snadné použití
          - Umožnit aktualizaci modelůpodle nových výsledků
        - vyvinula firma SPSS (nyní IBM)
          - používal ji systém SPSS Clementine
          - přestala se používat v roce 2001
          - nahradila ji CRISP-DM
          - podobá se hodně SEMMA
      - ! ✓ SEMMA



        - **SAMPLE** – výběr vhodných objektů
        - **EXPLORE** – vizuálníexplorace a redukce dat
        - **MODIFY** – seskupováníobjektůa hodnot atributů, datovétransformace
        - **MODEL** – analýza dat
          - Neuronovésítě, rozhodovacístromy, statistické techniky, asociace a shlukování
        - **ASSESS** – porovnánímodelů a interpretace
          - Srozumitelnost pro uživatele
        - vyvinula firma SAS
          - na svých stránkách tvrdí že SEMMA neni metoda dataminingu, ale logická organizace klíčových úkolů data-miningu pro SAS Enterprise Miner :o)
          - používá ji systém SAS Enterprise Miner
      - vyvinuté konsorciem (nezávislé na sw)
        - ! ! ✓ CRISP-DM



          - **CR**oss-**I**ndustry **S**tandard **P**rocess for **D**ata **M**ining
            - pořadí fází není přesně určeno
            - Výsledky získané v jedné fázi ovlivňují volbu dalších kroků
            - Některé kroky a fáze je třeba provádět opakovaně
          - **Businessunderstanding**
            - Pochopení cílů úlohy a požadavků na řešení (formulovaných z pohledu manažera)
            - Manažerskou formulaci je nutné převést na zadání úlohy pro dobývání znalostí z databází
            - "Revize" zdrojů(datových, výpočetních i lidských)
              - Hodnotí se možná rizika, náklady a přínos
            - Stanoví se předběžný plán prací
          - **Data understanding**
            - Prvotní sběr dat
            - Získání základní představy o datech
              - Posouzení kvality dat, vytipování zajímavých podmnožin záznamů v databázi, …
            - Výpočet deskriptivních charakteristik dat
              - Četnost atributů, průměrné hodnoty, …
            - Výhodou jsou vizualizačnítechniky
          - **Data preparation**
            - Vytvoření datového souboru, který bude zpracováván jednotlivými analytickými metodami
            - Data by měla obsahovat údaje podstatnépro danou úlohu a měla by být ve tvaru vyžadovaném algoritmy pro analýzu
            - Přípravadatzahrnuje:
              - Selekci dat, čištěnídat, transformaci dat, vytvářenídat, integrovánídat, formátovánídat, …
            - Jednotlivé úkony se obvykle provádějíopakovaně a v nejrůznějším pořadí
          - **Modeling**
            - Použití analytických metod pro dobývání znalostí
              - Z možných metod vybrat ty nejvhodnější a adekvátněnastavit jejich parametry
            - Iterativníčinnost
              - Opakovaná aplikace algoritmů s různými parametry
            - Může vést k potřeběmodifikovat data
            - Ověřenínalezených znalostí
          - **Evaluation**
            - Z pohledu manažerů
            - Byly splněny cíle formulované při zadání úlohy?
            - Rozhodnutíozpůsobuvyužitívýsledků
          - **Deployment**
            - Upravit získané znalosti do podoby použitelné pro zákazníka (manažera, zadavatele)
            - Zákazník musí pochopit, co je třeba učinit pro efektivnívyužitídosažených výsledků!
            - Implementaceklasifikačníhoalgoritmuvuser-friendlypodobě
            - Přípravauživatelskéhomanuálu
            - Instalace programu na pobočkách banky a zaškolení uživatelů
            - Změnametodikyposkytováníúvěrůapříslušnázměnavnitřníchpředpisůbanky
          - dnes prakticky standard
            - používá např. systém SPSS Modeller (dříve Clementine)
            - vyvinulo konzorcium: SPSS, NCR(Teradata), Daimler AG, OHRA(pojistovna)
  - ✓ 2. Předzpracování dat
  - ✓ 4. Rozhodovací stromy
  - ✓ 5. Asociacni pravidla
  - ✓ 6. Bayesovské modely
  - ✓ 7. Neuronové sítě
  - ✓ 8. ELM-sítě
  - ✓ 9. Genetické algoritmy
  - ✗ 10. Pokrocile předzpracování dat

Data-Mining

1. Metodiky

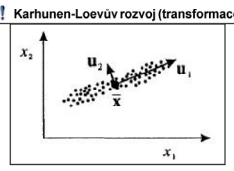
Výběr příznaků

Cíl: máme zbytečně moc příznaků => chceme vybrat jenom ty s největším přínosem

selekce
- z puvodni množiny ponechámejen n nejvíce informativních příznaků
- může snížit cenu měření příznaků

extrakce
- transformace příznaků na menší počet jiných příznaků
- od výsledku předpokládáme stejnou informační přínos
- musime měřit všechny příznaky

Karhunen-Loevův rozvoj (transformace)



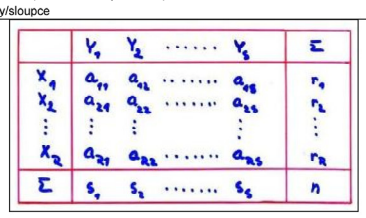
princip
- analýza hlavních komponent (PCA)
- něco jako neperiodický Fourier, vliv prvků rozvoje se s indexem monotonně snižuje
- cílem metody je transformace V: X^m -> Y^p vektorů z *m*-dimenzionálního příznakového prostoru vektorů X^m do *p*-dimenziálního prostoru vektorů Y^p, kde $p \leq m$, ktery minimalizuje střední kvadratickou odchylku
- máme: { xi∈R^K, i=1...m }
- hledáme { ei∈R^K, i=1...p, p < m } pro aproximaci vektorů xi:
  - vzorec je lin.kombinace vektorů ci
  - tak aby kvadrát odchylky ε byl minimální:

$$y_k - \sum_{i=1}^{p} c_{ki} e_i \qquad \varepsilon_k^2 = ||x_k - y_k||^2$$

vlastnosti
- při daném počtu členů rozvoje poskytuje ze všech možných aproximaci **nejmenší střední kvadratickou odchylku** od pův.vzorů
- při použití disperzní matice jsou nové transformované příznakové proměnné **nekorelované**
- členy rozvoje **nepřispívají k aproximaci rovnoměrně**
  - vliv každého z členů uspořádané posloupnosti aproximace se zmenšuje s jeho pořadím určeným velikostí odpovídajících vlastních čísel
- **velikost chyby aproximace neovlivňuje strukturu rozvoje**
  - změna požadavků na velikost střední kvadratické odchylky nevyžaduje přepočítat celý rozvoj, je třeba pouze změnit počet jeho členů

Statistickétesty

Kontingenční tabulka - pro dva atributy matice, pro každou kombinaci hodnot četnost, marginály se sumamy přes řádky/sloupce



- a_{kl}...četnost (frekvence) kombinace (X = X_k) ∧(Y = Y_l)
- r_k,s_l...řádkové, sloupcové součty (tzv. marginální hodnoty)

!! χ^2-test - jsou atributy nezávislé? stupně volnosti (skoro počet kombinací), hladina významnosti (pravděpodobnost závěru)

$$\chi^2 = \sum_i \sum_j \frac{(a_{ij} - e_{ij})^2}{e_{ij}}$$

$e_{kl} = r_k s_j / n$ očekávaná četnost kombinace (X = X_k) ∧(Y = Y_l) při nezávislosti X a Y

- počet stupňů volnosti = (počet řádků - 1) * (počet sloupců - 1)
- Laicky: Měříme, jak daleko je očekávaná a reálná četnost, vážíme očekávanou četnost
- Je-li χ^2 hodnota **větší** než "tabulková" hodnota χ^2-distribuce, atributy jsou závislé
- χ^2-test lze použít jen v případě dostatečně velkých četností: pro $(r_k \cdot s_l) / n \geq 5$ ∀k, l

Fisherův test - máme málo dat a jen dva booleovské atributy (= čtyřpolní kontingenční tabulka)

$$P = \sum_{i=0}^{a_{11}} \frac{r_1!\, r_2!\, s_1!\, s_2!}{n!\,(a_{11}-i)!\,(a_{12}+i)!\,(a_{21}+i)!\,(a_{22}-i)!}$$

n je celkový součet

Je-li **P≤α**, zamítne se nulová hypotéza o nezávislosti na hladině významnosti α

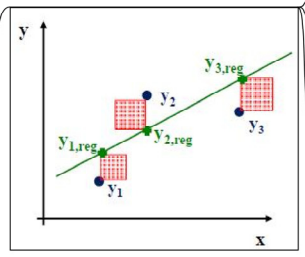
Regresní analýza - určit, jaký vztah má proměnná Y k jedné anebo vícero jiným proměnným X1,...,Xn

Jednorozměrná regrese

!! Lineární regrese - jaké parametry má lin.závislost mezi 2 veličinami

- typy závislostí 2 proměnných
  - funkční vztah - 2 závislé proměnné, určité x odpovídá jediná hodnota y
  - korelace - 2 náhodné (nezávislé) proměnné, síla vztahu popisovaná kor.koef. R
  - **regrese** - vztah náhodné (nezávislé, vysvětlující) proměnné x a závislé (vysvětlované) proměnné y, má pravděpodobnostní rozdělení
- pro x a y hledáme přímku (jejní parametry) vystihující průběh jejich závislostí = je nejblíže všem bodům
- závislost y na x můžeme popsat (aproximovat) pomocí **regresní rovnice (přímky)**: y = q1x + q0 + ε
  - ε je regresni odchylka
  - alternativně používaný zápis: y = bx + a
- **metodou nejmenších čtverců** - minimalizace rozdílů mezi skutečnou a očekávanou hodnotou
  - nejpoužívanější metoda pro odhad parametrů qi do regresního modelu
  - hledáme:

$$min \sum \blacksquare = min \sum (y_i - f(x_i))^2 = min \sum (y_i - q_1 x_i - q_0))^2$$



- **regresní koeficienty**
  - **odvození** - sumu zderivujeme podle všech qi a získáme rovnice rovné 0 (jako při hledání minima)
  - q1 - směrnice $q_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}$ — tan úhlu s osou x
  - q0 - absolutní člen $q_0 = \frac{\sum y_i - q_1 \sum x_i}{n}$

Mnohorozměrná regrese

! Lineární

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \cdot \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

řešení metodou nejmenších čtverců $\vec{q} = (X^T X)^{-1} X^T \vec{y}$

Nelineární
- předpokládáme složitější funkční závislost mezi y a vektorem **x** (kvadratickou, exponenciální)
- **Logistická regrese**
  - předpokládáme, že závislá veličina y je kategoriální, např. dvouhodnotová
  - modelujeme pravděpodobnost, že y má konkrétní hodnotu v závislosti na kombinaci hodnot nezávislých veličin vektoru **x**
  - podmíněná šance: P(y|x)/(1-P(y|x))

Diskriminační analýza
- v podstatě zkoušíme vyrobit perceptron, měříme chybu
- klasifikace příkladů do zadaných tříd

Shlukováni metodou k-středů (k-means)

Lze pozorované vzory rozdělit do skupin (shluků) vzájemně si blízkých vzorů?

Volba míry vzdálenosti závisína měřítku veličin → veličiny normovat
- dělit průměrem, směrodatnou odchylkou, rozpětím (max-min)

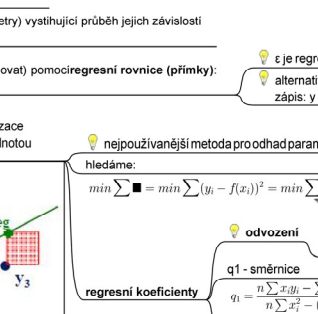
Centroid - střed shluku

Inicializace: Náhodně zvol rozklad do k shluků

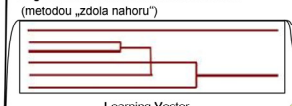
Iterace:
- **Update-step:** Urči centroidy pro všechny shluky v aktuálním rozkladu
- **Assignment-step:** Pro každý vzor **x**
  - Urči vzdálenosti d(**x**,c_k)
    - 1≤k ≤K
    - ck ~ centroid k-tého shluku
  - Nechť d(**x**,c_l) = min_k d(**x**,c_k)
    - z téhle zvlástní konstrukce získáme "index" nejbližšího shluku l
  - není-li **x** součástí shluku l, přesuň do shluku l
- nedošlo-li k žádnému přesunu -> KONEC

! *U ustni me jen nechala spocitat si nejaky centroid, resp. ukazat jak to udelam.*

Příklad:



Algoritmus hierarchického shlukování (metodou „zdola nahoru")



- na zacatku je kazdy bod cluster
- dokud je více než 1 shluk
  - Urči vzájemné vzdálenosti mezi všemi clustery
  - vzdy shluknu dva nejblizsi body do noveho clusteru

Shlukova analýza (klastrování)

LVQ - Learning Vector Quantization
- reprezentace shluků vektory
- mame klasifikator priznakoveho prostoru, chceme hranice klasifikatoru promitnout i do mapy

Inicializace:
- všech vektorů w_j **(0)**
  - váha vektoru v čase 0
- parametru učení **μ (0)**

Algoritmus:

Hlavní cyklus: ∀ trenovací vstupní vektor x_i k++
- najdi index **q** tak že w_q**(k)** je váhový vektor s nejmenší vzdáleností
  - např. euklidovská vzd. $\sqrt{\sum_i (x_i - w_q(i))^2}$
- **if $C_{wq} = C_{xi}$** $w_q(k+1) = w_q(k) + \mu(k)[x_i - w_q(k)]$
  - posuň váhový **w** k trénovacímu **x**
  - pozn.: $(x_i - w_q) = ((x_{i1} - w_{q1}), \cdots, (x_{in} - w_{qn}))$
  - C_{xi} je definovaná třída vektoru x_i
- **if $C_{wq} \neq C_{xi}$** $w_q(k+1) = w_q(k) - \mu(k)[x_i - w_q(k)]$
  - posuň **w** od **x**
- sniž parametr učení **μ (k+1)**

Ukončovací podmínka: nějaký počet iterací nebo nízná hodnota μ

příklad:

Klasifikace

k-Nearest Neigbour

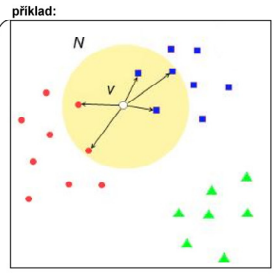
trénovací množina klasifikovaných vzoru **M**

Vstup:
- vzor **v**, který chceme klasifikovat
- a celé kladné číslo **k**

Algoritmus: v množině **M** najdeme množinu **N** obsahující **k** vzoru, ktere jsou k vzoru **v** nejbližší ze všech vzorů z **M**
- vzoru v dáme klasifikaci, která je nejčastejší v množině **N**

- lze použít i na regresi a klastrování
- probírá se jenom na cvičeni

příklad:



2. Předzpracování dat

3. Vizualizace

4. Rozhodovací stromy

5. Asociacni pravidla

6. Bayesovské modely

7. Neuronové sítě

8. ELM-sítě

9. Genetické algoritmy

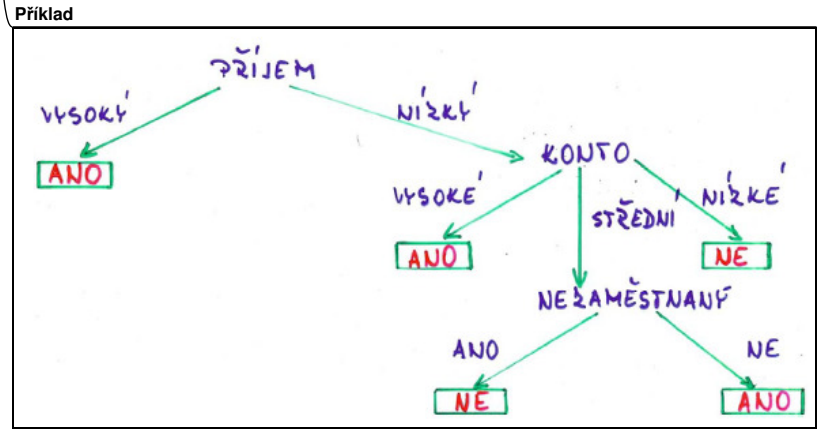
10. Pokrocile předzpracování dat

Data-Mining

- 1. Metodiky
- 2. Předzpracování dat
- 4. Rozhodovací stromy
  - Vstup: Vektory příznaků; Výstup: Třídy klasifikace
  - Vnitřní uzel se větví podle hodnoty svého atributu (s atributy rodičů již zafixovanými)
    - klasifikace mnoziny na zaklade atributu kazdeho prvku
    - posloupnost dotazu na jednotlive atributy (na rovnost nebo vetsi/mensi)
    - chceme mit co nejkratsi cesty
    - **slozitost rozhodovacich stromu:** hnlog(n) ... kde h je pocet atributu
  - **TDIDT indukce**
    - **T**op **D**own **I**nduction of **D**ecision **T**rees
    - **algoritmus**
      - Zvol jeden atribut jako kořen dílčího stromu
        - nikdy 2x stejný v 1 větvi
      - Rozděl data v tomto uzlu na podmnožiny podle hodnot zvoleného atributu a přidej uzel pro každou podmnožinu
      - existuje-li uzel, pro který nepatří všechna data do téže třídy, opakuj pro tento uzel postup od bodu 1;
      - jinak skonči
  - **Entropie** - míra neuspořádanosti v systému

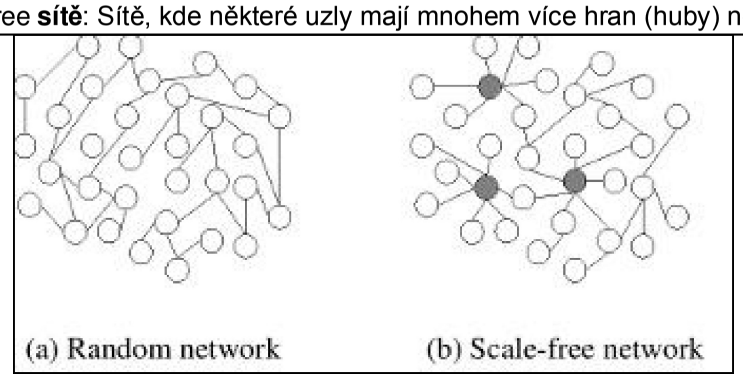
    $$H = -\sum_i p_i \log p_i$$

    - pi - relativní četnost třídy i, pro 0 doplníme výsledek limitou = 0
  - **ID3 indukce** - Jako TDIDT, ale pracuje na výstupu místo třídy jen s bool hodnotami
    - **I**terative **D**ichotomiser **3**
    - **algoritmus**
      - spocitej entropii pro každý atribut množiny S
      - rozděl S do podmnožin použitím atributu nejlépe klasifikujícím množinu
        - nejnišší součet entropii podmnožin
        - podmnožin je stejně jako hodnot atributu
      - vytvoř uzel stromu obsahující atribut
      - rekurze na podmnožiny pomocí zbývajících atributů
        - pokud mají všechny stejnou klasifikaci, nebo dojdou atributy -> list
  - **C4.5 indukce**
    - modifikace ID3
    - chybějící data
      - při konstrukci stromu se ignorují
      - při klasifikaci se odhadnou
    - spojitá data
      - kategorizace podle trénovací množiny
    - pruning
      - validace - rozdělení trénovací množiny
        - 2/3trénovací
        - 1/3validační
      - reduced-error pruning
        - náhrada podstromu listem ohodnocených nejčastěji třídou (na příslušných trénovacích vzorech)
        - nový strom zkontrolujeme na validační množině aby nebyl horší
      - roubování
        - subtree raising
        - přenesenínejpoužívanějšíhopodstromuvýše
    - rulepost-pruning
      - konverze stromu (cest do listů) na ekvivalentní sady pravidel
      - prořezání odstraněním předpokladů pokud to nezhorší očekávanout správnost klasifikace
      - usporadanipravidelpodlepresnosti
    - dolní odhad správnosti
      - vypocet spravnosti na trenovacich datech
      - vypocet smerodatne odchylky
    - kriterium pro dělení
      - použijeme atribut s nejvyšším poměrným informačním ziskem (gain ratio)
  - **C5.0/See5: boosting**
    - hlasování několika paralelních klasifikátorů
  - **CART**
    - **C**lassification **A**nd **R**egression **T**rees
    - generujebinárnírozhodovacístromy
    - prostorhodnotatributurozpivotovándělícíhodnotou
  - **SPRINT**
    - **S**calable **P**a**R**allelizable **IN**duction of decision**T**rees
    - kriterium pro dělení: Gini-index
      - GINI(D) = 1 - Σ pi^2
        - D - databáze
        - pi - frekvence Ci v D
      - $GINI_{SPLIT}(D)$ = n1/n GINI(D1) + n2/n GINI(D2)
        - n - počet prvků
    - volba atributu pro dělení: Rainforest
      - AVC tabulka pro kazdy uzel
    - indukcerozhodovacíhostromu
      - projdou se trénovací data
      - vytvoříme AVC-tabulka
      - určí se nejlepší attribut pro dělení
      - rozdělenítrénovacíchdat
      - kontrukce AVC pro následující uzel
  - **Příklad**

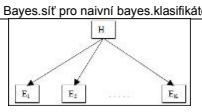


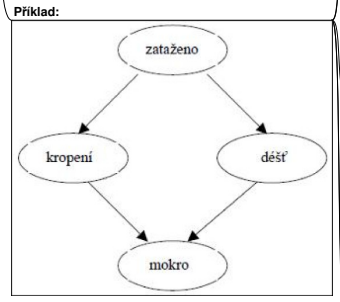
- 5. Asociacni pravidla
- 6. Bayesovské modely
- 7. Neuronové sítě
- 8. ELM-sítě
- 9. Genetické algoritmy
- 10. Pokrocile předzpracování dat

- Data-Mining
  - ✓ 1. Metodiky
  - ✓ 2. Předzpracování dat
  - ✓ 4. Rozhodovací stromy
  - ✓ 5. Asociacni pravidla
    - !! MBA
      - Množina produktů v transakci, které se spolu vyskytují nejčastěji? Virtuální položky pro "metadata".
      - 3 míry [0; 1]
        - $podpora(pravidlo) = \frac{\#i\&j}{\#vsech}$
        - $spolehlivost(pravidlo) = \frac{\#i\&j}{\#i}$
      - $zlepseni(pravidlo) = \frac{p(i\&j)}{p(i)p(j)} = \frac{\#i\&j\#všech}{\#i\#j}$
        - 💡 pokud je <1 je pravidlo horší než náhodná volba
      - hlavní kroky
        - zvolte položky na adekvátní úrovni
        - vytvořte if-then pravidla z tabulky četností
        - určete nejlepší pravidla (zlepšení by mělo být >1)
      - Prořezávání, zobecňování (značky, výrobci), disociační pravidla (NOT; pochybná užitečnost), časové řady (rozšíření do okénka)
    - !! ✓ **Algoritmus APRIORI** - generování asociačních pravidel (frequent itemsets)
      - 💡 Generování kombinací do šířky - kombinace délky k z kombinací délky k-1
        - Každou kombinaci délky k-1 porovnáme s každou jinou, shodují-li se v k-2 itemech, spojíme
        - Spojené zpětně profiltrujeme, chybí-li nám nějaká jejich podkombinace délky k-1, zahodíme je
      - Do L1 přiřaď všechny kategorie, které dosahují alespoň požadované četnosti
      - for(k=2; L_{k-1}!={}; k++)
        - vygeneruj na základě Lk-1 kandidáty Ck
          - 💡 kartézský součin Lk-1 x Lk-1 a odstraň pokud její podkombinace délky k-1 není obsažena v Lk-1
        - Do Lk zařaď ty kombinace z Ck, které dosáhly alespoň požadovanou četnost
      - return ∪ Lk;
    - ! **S**cale-**F**ree **sítě**: Sítě, kde některé uzly mají mnohem více hran (huby) než ostatní



      - 💡 nalezení vztahů mezi údaji, vizualizace linků a vztahů
      - **využití:**
        - počítačové sítě
          - odolné proti náhodným poruchám, zranitelné při koordinovaném útoku
        - sociální sítě
        - monitorování vztahů mezi společnostmi
        - marketing
  - ✓ 6. Bayesovské modely
  - ✓ 7. Neuronové sítě
  - ✓ 8. ELM-sítě
  - ✓ 9. Genetické algoritmy
  - ✗ 10. Pokrocile předzpracování dat

- Data-Mining
  - ✓ 1. Metodiky
  - ✓ 2. Předzpracování dat
  - ✓ 4. Rozhodovací stromy
  - ✓ 5. Asociacni pravidla
  - ✓ 6. Bayesovské modely
    - ! ! ✓ **Naivní Bayesovský klasifikátor**; mám jev a klasifikaci, při tréninku počítám četnosti a P(jev|klasifikace)
      - Při použití podle Bayesova pravidla počítám podmíněnou P(klasifikace|jev)
        $$P(H|E) = \frac{P(H \cap E)}{P(E)} = \frac{P(E|H)P(H)}{P(E)}$$
        - podmíněná=aposteriorní
      - ! **klasifikace pomocí naivního bayesovského klasifikátoru** budeme hledat hypotézu s největší aposteriorní pravděpodobností
        $$H_t = P(H_t) \times \prod P(E_k|H_t) = \frac{\text{Pocet vzoru z t}}{\text{Pocet vsech vzoru}} \times \prod \frac{\text{Pocet vzoru z t splnujicich } E_k}{\text{Pocet vzoru z t}}$$
        - **Př.:** Bayes.síť pro naivní bayes.klasifikátor
          

      - Naivní, protože předpokládám, že jevy jsou nezávislé; v praxi obvykle nejsou, přesto funguje překvapivě dobře
      - na půl cesty mezi pravidly a neuronkami
      - **Podmíněná nezávislost**
        - A je podm. nezavisla na B zname-li C, pokud P(A,B|C)=P(A|C)P(B|C)
          - alt. značení: C: A⊥B | C
        - v Bayesovske siti je A podm. nezavisla na sve potomky
      - **definice**
        - Bayesovska sit je DAG graf
        - každému uzlu u (~náhodné veličině) je přiřazeno pst rozdělení tvaru P(u|rodiče (u))
          - rodiče (u) jsou uzly, ze kterých vycházejí hrany do uzlu u
        - ∀ uzly jsou očíslovány tak, že rodiče jsou před dětmi
      - **Sdružená pst distribuce celé sítě -** P(u1,...,un) = Π P(ui|rodice(ui))
        - tzv. "faktorizace sdružené distribuce"
      - **Inference** (pravděpodobnostní odvozování)
        - **diagnostická (bottom-up)** která příčina má max. aposteriorní pst
        - **kauzální (top-bottom)** pst jevu za předpokladu nějaké příčiny
      - **Učení**
        - **Známá struktura**
          - **veličiny plně pozorovatelné** (vše vidíme)
            - **metoda max.věrohodného odhadu**
              - spočítat z dat odhady podmíněných pravděpodobnostních distribucí pro jednotlivé uzly sítě
              - **Cíl:** nalézt maximálně věrohodný odhad vzhledem k tréninovacím datům (spočítat na základě četností)
                $$P(H_1|E_1, E_2) = \frac{n(H_1 \wedge E_1 \wedge E_2)}{n(E_1 \wedge E_2)}$$
              - vlastně naivní bayes
          - **veličiny částečně pozorovatelné** (některé veličiny=uzly sítě nelze pozorovat nebo šum)
            - **gradientní metoda**
              - maximalizujeme pst **P(D|h)**, že pozorujeme daná trénovací data **D**
              - pokud platí podmíněné psti
              - výpočet gradientu fce ***ln P(D|h)***
                $$\frac{\partial \ln P(D|h)}{\partial p_{ijk}} = \sum_{d \in D} \frac{P(Y_i = y_{ij}, U_i = u_{ik}|d)}{p_{ijk}}$$
                - p_ijk - podmíněná pst že veličina Yi nabývá y_ij, pokud rodiče Ui této veličiny nabývají u_ik
              - analogie NS učící skryté vrstvy
            - **EM algoritmus** (expectation maximalisation)
              - **Inicializace:** náhodně zvol parametry distribuce
              - **Hlavní cyklus:**
                - **Expektace:** podle parametrů distribuce spočítej hodnotu skrytých veličin
                - **Maximalizace:** z nich urči max. věrohodný odhad parametrů distribuce
                - pokud došlo ke změně parametrů opakuj
                  - iteruj až k pevnému bodu
        - **Neznámá struktura**
          - **veličiny plně pozorovatelné** (prohledávání prostoru modelů)
            - výpočetně náročné – různých množin rodičů může být 2^k
          - **veličiny částečně pozorovatelné** průšvih
      - **Příklad:**
        

        - **Sdružená distribuce:** P(Z,K,D,M) = P(Z) P(K|Z) P(D|Z) P(M|K, D)
        - tabulka podm. pstí uzlů:

Z	P(K=0)	P(K=1)
0	0.5	0.5
1	0.9	0.1

P(K|Z)

P(Z=0)	P(Z=1)
0.5	0.5

P(Z)

K	D	P(M=0)	P(M=1)
0	0	1.0	0.0
1	0	0.1	0.9
0	1	0.1	0.9
1	1	0.01	0.99

P(M|K,D)

Z	P(D=0)	P(D=1)
0	0.8	0.2
1	0.2	0.8

P(D|Z)

        - diagnostická(bottom-up)inference:

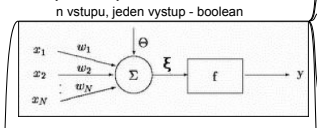
$$P(K=1|M=1) = \frac{P(K=1,M=1)}{P(M=1)} = \frac{\sum_{z,d} P(Z=z,K=1,D=d,M=1)}{P(M=1)} =$$
$$= \frac{\sum_{z,d} (P(Z=z)P(K=1|Z=z)P(D=d|Z=z)P(M=1|K=1,D=d))}{P(M=1)} =$$
$$= \frac{0.2781}{P(M=1)}$$

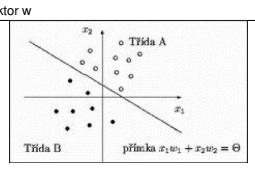
$$P(D=1|M=1) = \frac{P(D=1,M=1)}{P(M=1)} = \frac{\Sigma_{z,k} P(Z=z,K=k,D=1,M=1)}{P(M=1)} =$$
$$= \frac{\Sigma_{z,k} (P(Z=z)P(K=k|Z=z)P(D=1|Z=z)P(M=1|K=k,D=1))}{P(M=1)} =$$
$$= \frac{0.4581}{P(M=1)}$$

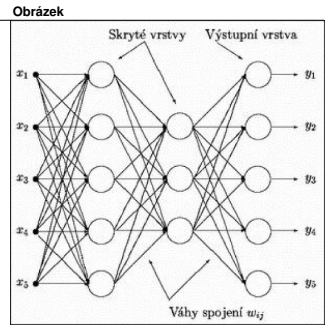
        - Maximální aposteriorní pravděpodobnost má tedy příčina déšť; P(D=1|M=1) > P(K=1|M=1).
        - kauzální(top-down)inference:

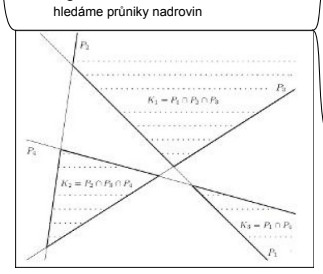
$$P(M=1|Z=1) = \frac{P(M=1,Z=1)}{P(Z=1)} = \frac{\Sigma_{k,d} P(Z=1,K=k,D=d,M=1)}{P(Z=1)} =$$
$$= \frac{P(Z=1)\Sigma_{k,d}(P(K=k|Z=1)P(D=d|Z=1)P(M=1|K=k,D=d))}{P(Z=1)} =$$
$$= 0.7452$$

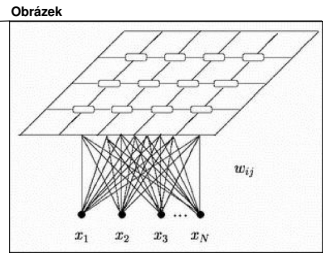
    - ! ✓ **Bayesovské sítě** - DAG podmíněné závislosti veličin, uzel na výstupu dává pravděpodobnost generovanou statickým rozdělením přes vstupní pravděpodobnosti rodičů
  - ✓ 7. Neuronové sítě
  - ✓ 8. ELM-sítě
  - ✓ 9. Genetické algoritmy
  - ✗ 10. Pokrocile předzpracování dat

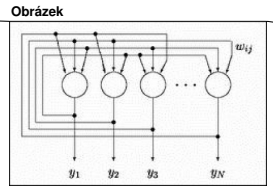
- 1. Metodiky
- 2. Předzpracování dat
- 4. Rozhodovací stromy
- 5. Asociacni pravidla
- 6. Bayesovské modely
- Data-Mining
  - 7. Neuronové sítě
    - Učení s učitelem
      - Perceptrony
        - ! Jednoduchý perceptron je výpočetní jednotka s prahem ϑ , která pro vstupy x1,…,xn∈R a váhy w1,…,wn dává výstup**1**, jestliže platí Σwi xi ≥ϑ (tzn. w x ≥ϑ) a jinak**0**.
        - ! jednoduchý neuron
          n vstupu, jeden vystup - boolean
          

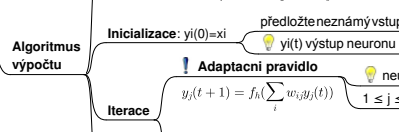
          - **práh neuronu:** ϑ ∈ R
            - zpravidla se v literatuře označuje jako theta (ϑ, θ, Θ)
          - **potenciál neuronu:** ξ = Σwi xi + ϑ
            - označuje se jako ξ (xí) nebo jako u
          - **přenosová (aktivační) fukce:** f (ξ)
            - typy výstupů
              - f = 1 neuron je aktivní
              - f = 1/2 tichý
                - tzn leží v dělící nadrovině
              - f = 0 pasivní
            - příklady
              - sgn
              - jednotkový skok
              - sigmoida: $\frac{1}{1+e^{-\xi}}$
          - **výstup neuronu:** y = f (ξ = Σwi xi + ϑ)
        - **(absolutně) lineárně separabilní**mn. A a B - v n-rozměrném prostoru, pokud ∃w1,…,wn,ϑ∈R že ∀ (x1,...,xn)∈A splňuje Σwi xi (>) ≥ϑ a ∀(x1,...,xn)∈B splňuje Σwi xi<ϑ
          - **Příklad:** množina co není lin. separabilní
            
        - ! **Algoritmus učení (Hebb)**
          hledáme dělící nadrovinu a na ní kolmý váhový vektor w
          

          - **Inicializace:** náhodnězvolímevstupníváhywi(0)(1≤i≤n+1)
          - **Učení:**
            - předložíme požadovaný výstup**d** pro trénovací vzor **x**
            - spočteme skut.výstup y(t)=sgnΣwi(t)xi(t)
            - **adaptace vah**: wi(t+1) = wi(t) + η * xi(t) * (d(t)-y(t))
              - 0≤η≤1 je koeficient učení (η je éta)
              - Chybně klasifikovaný vektor uvnitř (ve směru váhového vektoru) odčítáme (otáčíme w od něj)
              - Chybně klasifikovaný vektor vně přičítáme (otáčíme w k němu)
            - pokud t nedosáhl požadované hodnoty, opakuj (pro další trénovací dvojice x a d(t))
          - zkráceně: postupne beru prvky z trenovaci mnoziny, v pripade nespravne klasifikace otacim vektor (pricitam k nemu prvek) k/od prvku (vynasobeny ucicim parametrem η<1)
        - ✗ **Konvergence učení** - P a N jsou konečné a lin.separabilní množiny => perc.alg.učení provede konečný počet aktualizací vektoru w_t
        - **Přihrádkový (ratchet) algoritmus** - vždy si pamatuji váhový vektor, který zatím prošel nejvíce úspěšnými klasifikacemi za sebou, až zastavím učení, vydám ten
          - vylepseni: kapsa, kam davam nejlepsi reseni, vracim reseni z kapsy
      - Back-propagation
        - **Vícevrstvá neuronová síť**, potřebuji propagovat updaty váhových vektorů
          - neexistují zpětné vazby ani vazby v rámci 1 vrstvy
          - vnitřní(skryté)neurony
          - výstupní neurony
          - **Obrázek**
            

        - **Použití**:
          - klasifikace obrazů
          - aproximace funkcí
          - řízení
          - predikce časových řad
        - ! ✓ **Adaptační pravidla (odvození)** aktualizace synaptických vah pro:
          - **Chybová fce:** Minimalizuji na trénovací množině odchylku přes výstupní neurony
            $$E = 1/2 \sum_p \sum_j (y_{jp} - d_{jp})^2$$
            - chceme minimalizovat chybu E = y - d
            - p - vzory
            - j - výstupní neurony
            - yjp - skutečná odezva
            - djp - požadovaná odezva
          - **Váhový update**: wij (t+1)=wij (t)+$\Delta_E$wij (t)
            - proti směru gradientu chybové funkce
            - od výstupní vrstvy směrem ke vstupní
            - **proto se metoda jmenuje back-prop**
            - $\Delta_E$wij (t) - přírůstek váhy wij k minimalizaci E
          - Výstupníneuron(odvození):
            $$\Delta_E w_{ij} = -\frac{\partial E}{\partial w_{ij}} = -\frac{\partial E}{\partial \xi_j}\frac{\partial \xi_j}{\partial w_{ij}} = -\frac{\partial E}{\partial \xi_j} y_i = -\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial \xi_j} y_i = -(y_j - d_j) f'(\xi_j) y_i = \delta_j y_i$$
            - parc.derivace - ∂E/∂wij určuje směr největšího úbytku fce
              - spec.případem je hledání minima fce pomocí nulových bodů její derivace
          - Vnitřníneuron(odvození):
            $$\Delta_E w_{ij} = -\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial \xi_j} y_i = -(\sum_k \frac{\partial E}{\partial \xi_k}\frac{\partial \xi_k}{\partial y_j})\frac{\partial y_j}{\partial \xi_j} y_i = -(\sum_k \frac{\partial E}{\partial \psi_k} w_{jk})\frac{\partial y_j}{\partial \xi_j} y_i = -(\sum_k \delta_k w_{jk}) f'(\xi_j) y_i = \delta_j y_i$$
        - ! ✓ **Algoritmus trénování** hledáme průniky nadrovin
          

          - **Inicializace:** náhodnězvolímeváhywi(0)(1≤i≤n+1)
          - **Učení:**
            - předložíme požadovaný výstup**d** pro trénovací vzor **x**
            - spočteme skut.výstup $y_j = f(\xi_j) = \frac{1}{1+e^{-\lambda \xi_j}}$
              - kde ξj = Σwij yi
              - výstupy pak tvoří vstup na další vrstvě
            - **adaptace vah**: $w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j y_i + \alpha_m(w_{ij}(t) - w_{ij}(t-1))$
              - wij(t) váha synapse z neuronu i do j v čase t
              - 0≤η≤1 je koeficient učení (η je éta)
              - αm≤1 - moment učení
              - δj je chyba učení
                $$\delta_j = \begin{cases} \lambda y_j(1-y_j)(d_j - y_j) & \text{pro výstupní uzly} \\ \lambda y_j(1-y_j)(\sum_k \delta_k w_{jk}) & \text{pro skryté uzly} \end{cases}$$
                - k - index pro neurony**následující** za neuronem j
                - λ - je strmost přenosové funkce
              - mame opet trenovaci mnozinu, pro x spocteme p, f(p)=y, porovname s d
            - opakuj dokud chyba není menší než stanovená hodnota
      - Učení bez učitele
        - ! ! ✓ **Kohonenovy mapy** Self-organizing maps (SOM)
          - síť ze dvou vrstev
            - horní **kompetiční** výstupní vrstva má **m** neuronů
              - výstupní hodnota je definována vzdáleností mezi vstupním a váhovým vektorem $y_i(t) = \sum_j (x_j(t) - w_{ij}(t))^2$
            - spodní (příznaková) vstupní vrstva má **n** neuronů
            - **Obrázek**
              

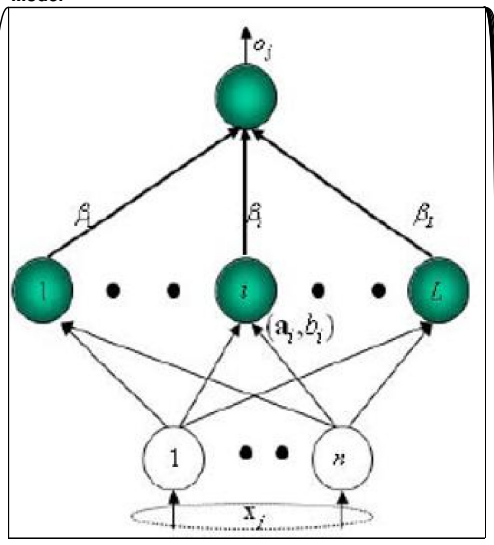
          - **Lateralni funkce** Φ(i,j) vraci koeficient učení v zavislosti na vzdalenosti neuronu i j na vystupu
            - Φ je velké fí :)
          - **Algoritmus učení**
            - **Inicializace**: náhodně zvolím váhy
              - wij je váha mezi vstupním neuronem i a výstupním j
            - **Učení:**
              - předložím trénovací vzor
              - vypočtu vzdálenost mezi vst. a váhovým vektorem každého neuronu a předám kompetiční vrstvě $d_i = \sum_j (x_j(t) - w_{ij}(t))^2$
              - vyberu výstupní neuron c s minimální di a označím ho jako vítěze
              - **Adaptace vah** v okolí c $\Delta w_{ij} = \alpha(t)\Phi(c,j)(x_i(t) - w_{ij}(t)))$
                - vigilační koef. α se s časem snižuje
                - vigilance ~ bdělost
              - opakuji pro všechny vstupní vektory dokud se váhy mění
                - Kompetice neuronů o vzorek, vítěz inhibuje ostatní
                - Neurony topologicky rozprostřeny, soutěží o váhový vektor nejbližší vstupu
                - chceme aby neurony s vahovym vektorem blizko v priznakovem prostoru byly vedle sebe i ve vystupnim prostoru
          - **Použití:**
            - shlukování
            - analýza dat
            - vytváření sémantických map
          - **Pro učení s učitelem**
            - ✗ **LVQ** (Vektorová kvantizace)
              - learningvectorquantization
              - mame klasifikator priznakoveho prostoru, chceme hranice klasifikatoru promitnout i do mapy
              - **LVQ1**: adaptuji pouze nejblizsi neuron, ale v pripade spatne tridy w neuronu opacnym smerem!
              - LVQ2.1: adaptuji, jsou-li dva nejblizsi neurony v ruznych tridach (jeden v moji, druy v cizi) a jsem v ramci okenka delici nadplochy
              - LVQ3: trosku adaptuji i kdyz jsou oba sousede v me tride
    - Učení s učitelem
      - Hopfieldovy site
        - jednovrstvá **rekuretní** síť s pevnými vahami
          - Bipolární vstupy i výstupy {+ 1 ; -1 } - mohou být i binární
          - **Obrázek**
            

            - každé 2 neurony jsou spojené symetrickými vahami wij = wji, každý neuron je vstupní i výstupní
          - **Použití**:
            - asociativní paměť
            - optimalizační úlohy
            - rekonstrukce neúplných a šumem poškozených obrazů
          - **Skoková přenosová funkce** $f_h(x)$
            

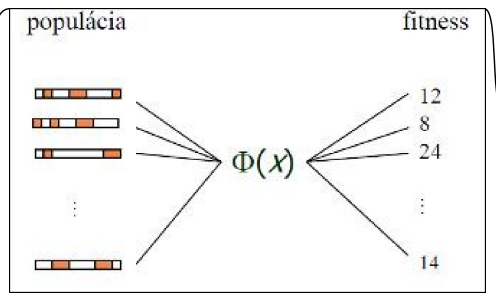
            - formálně $f_h(x) = \begin{cases} 1 & \text{pro } x > 0 \\ -1 & \text{pro } x \le 0 \end{cases}$
          - **Učení** (Hebb): "what fires together, wires together"
            $$w_{ij} = \begin{cases} \sum_s x_i^s x_j^s & \text{pro } i \neq j \\ 0 & \text{pro } i = j \end{cases}$$
            - wij je váha synapse mezi neurony i a j
            - ! x_i^s je i-tá složka s-tého vektoru
          - **Algoritmus výpočtu**
            - **Inicializace**: yi(0)=xi
              - předložte neznámý vstupní vzor
              - yi(t) výstup neuronu i v čase t
            - **Iterace**
              - ! **Adaptacni pravidlo** $y_j(t+1) = f_h(\sum_i w_{ij} y_j(t))$
                - neurony se iterativně updatují
                - 1 ≤ j ≤ n
              - opakujeme dokud se výstupy neuronů neustálí
                - výstupy pak reprezentují trén.vzor nejlépe odpovídající předloženému vzoru
                - taky může zůstat oscilující mezi dvěma stavy
            - přejdi ke kroku Inicializace
          - Konvergence při vybavení - energetická funkce neustále klesá
          - **Energetická fukce** vyjadřuje energii sítě ve stavu**x** $E(x) = -\frac{1}{2}\sum_j \sum_i w_{ij} x_i x_j$
            - **Věta:** síť dosáhne stabilního stavu v lokálním minimum energetické funkce
  - 8. ELM-sítě
  - 9. Genetické algoritmy
  - ✗ 10. Pokrocile předzpracování dat

Data-Mining

- 1. Metodiky
- 2. Předzpracování dat
- 4. Rozhodovací stromy
- 5. Asociacni pravidla
- 6. Bayesovské modely
- 7. Neuronové sítě
- 8. ELM-sítě
  - **Model**

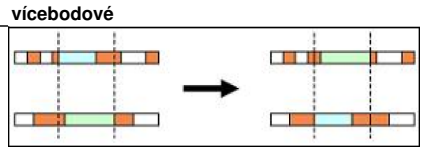


    - **1** výstupní neuron
    - **L** skrytých neuronů
    - **n** výstupních neuronů
    - a_i váhový vektor skrytého neuronu
    - b_i práh i-tého skrytého neuronu
    - dopředná síť s jednou skrytou vrstvou neuronů s libovolnou po částech spojitou přenosovou funkcí $G(a_i, b_i, x)$
    - β_i váha mezi i-tým skrytým a výstupním neuronem
    - **Výstup sítě:** $f_L(x_j) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x_j) = t_j$
      - můžeme zapsat jako **Hβ = T** $H = \begin{bmatrix} G(a_1,b_1,x_1) & \dots & G(a_L,b_L,x_1) \\ \dots & \dots & \dots \\ G(a_1,b_1,x_N) & \dots & G(a_L,b_L,x_N) \end{bmatrix}$
        - **N** je počet trénovacích vzorů
        - ti jsou požadované výstupy
  - ! **algoritmus učení**
    - Trénovací množina $\aleph = \{(x_i, t_i) \mid x_i \in \mathbb{R}^n, t_i \in \mathbb{R}^m, i=1,\dots,N\}$
    - **Vstup:**
      - výstupní funkce skrytých neuronu G(a,b,x)
      - počet skrytých neuronů L
    - **Inicializace:** náhodně zvol parametry skrytých neuronů (ai, bi)
    - Spočítej matici výstupů skryté vrstvy H
    - **Učení:**
      - Spočítej váhy pro výstupní neurony β: β = H† T
        - H† označuje Moore-Penroseovu pseudoinverzní matici pro matici výstupů skryté vrstvy neuronové sítě H
        - protoze inverzni matice ne vzdy existuje (čtvercová singulární, nebo obdélníková) tak MP matice je proste nejblizssi matice k matici inverzni kdyby existovala
        - jednoduchýneiterativnívýpočet/učení
  - **výhody**
    - jednoduchý neiterativní výpočet, který se skládá ze 3 kroků (mnohé z tradičních algoritmů učení jsou výrazně složitější)
    - velmi rychlý a přímočarý proces učení (není třeba řešit otázky lokálních minim, vhodné inicializace parametrů, přeučení a další
    - parametry skrytých neuronů ai a bi jsou nezávislé na trénovacích datechi na sobě navzájem
    - mohou generovat parametry skrytých neuronů již před předložením trénovacích dat (konvenční metody učení musí trénovací data „vidět“ předtím, než nastaví parametry skrytých neuronů)
    - lze použít s libovolnou omezenou po částech spojitou nekonstantní přenosovou funkcí (gradientní algoritmy učení pracují jen s hladkými přenosovými funkcemi)
- 9. Genetické algoritmy
  - **Chromozom (genotyp)** - řetězec symbolů kódující vlastnosti jedince (fenotyp)
  - **Fitness funkce** Φ: genotyp → R
  - **Populace P(t)** umělých chromozomů



    - jak dobře řeší problém, čím vyšší tím lepší
  - ! ! **Jednoduchý genet.algoritmus**
    - **Inicializace:** vygeneruj populaci P(0) = {**x1,...xN**} náhodných chromozomů
    - Iterace:
      - spočítej fitness všech
      - vytvoř populaci P(t+1)
        - **selekcí**
          - **reprodukce ruletou** - naškálujeme jedince na ruletu podle fitness a otočíme ruletou N-krát
          - **elitismus** - S nejlepších jedinců kopírujeme vždy
        - ! náhodně se spárují a zkříží podle zadané pravděpodobnosti proveď **křížení**
          - **jednobodové**

          - **vícebodové**



        - **mutuj** chromozómy náhodná změna bitů podle zadané pravděpodobnosti
      - skonči pokud: najdeme hledaného jedince nebo fitness nejlepšího neroste
    - může popisovat část chromozomu zaručující vysokou fitness
  - **Schema** - abstraktní "šablona" genomu, ternarni retezec delky m, krome 1/0 jeste * (wildcard)
    - **řád schématu o(H)** - počet pevných pozic
    - **délka schématu δ(H)** - vzálenost mezi první a poslední pevnou pozicí
    - **fitness schématu f(H)** - průměrná fitness odp. řetězců v populaci
  - ! ! **Věta o schematech** - v populaci během GA (exponenciálně) roste počet schémat, která jsou krátká, mají nízký řád a mají nadprůměrnou fitness
    - ! **Idea důkazu**: Díváme se postupně pro všechny genetické operátory, na čem závisí očekávaná četnost schématu v další populaci - ukazuje se, že právě na délce atd.
- 10. Pokrocile předzpracování dat