Obsah

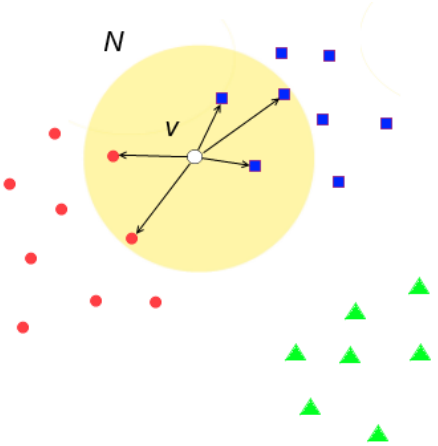
Obsah 1
1. 2. 4. Učení s učitelem - učení založené na instancích (Ch. 2) 2
- k-Nearest Neigbour 2
- Lineární regrese 3
3. Rozhodovací stromy (Ch. 9) 6
- Entropie - míra neuspořádanosti v systému 6
- Informační zisk (information gain) 6
- Prořezávání 7
5. Support vector machines (Ch. 12) 8
- Optimální separující nadrovina 8
- Support Vector Machines 9
6. Ohodnocení a volba modelů (Ch. 7) 10
- Míry chyby pro regresi 10
- Míry chyby pro klasifikaci 10
- dost dat x málo dat 10
7. Kombinace modelů (Ch. 8,10) 11
- boosting 11
8. Bayesovské učení 12
- maximálně pravděpodobná hypotéza (max. apost. probability) 12
- Bayesovsky optimální odhad 12
- maximálně věrohodná hypotéza (max. likelihood) 12
9. EM algoritmus 13
- Shlukování metodou k-středů (k-means) 13
- EM algoritmus (estimate maximize) 14
10. Učení bez učitele (Ch. 14) 15
- MBA (Asociační pravidla) Analýza nákupního koše 15
- Algoritmus APRIORI 15
11. Genetické algoritmy 16

1. 2. 4. Učení s učitelem - učení založené na instancích (Ch. 2)

☀ instance = příklady, vzory
vstupy (nezávislé proměnné)
výstupy (závislé proměnné)

❢ k-Nearest Neigbour

- Vstup:
  - trénovací množina klasifikovaných vzoru **M**
  - vzor **v**, který chceme klasifikovat
  - a celé kladné číslo **k**
- Algoritmus:
  - v množine **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 množine **N**
    - příklad:



- vlastnosti
  - se přizpůsobí libovolné (spojité..) hranici
  - predikce záleží na pozici a hodnotách pár okolních bodů, proto je **nestabilní**
  - velký rozptyl, malé vychýlení (bias)
  - trpí prokletím dimezionality
  - učíme se ukládáním dat
  - lze použít na klasifikaci, regresi a klastrování

❣ Lineární regrese

jaké parametry má lin. závislost mezi 2 veličinami

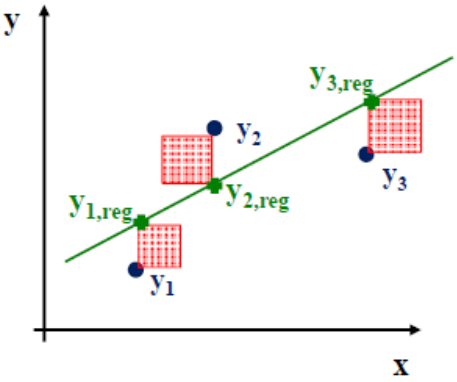
- **Regresní analýza** - určit, jaký vztah má proměnná Y k jedné anebo vícero jiným proměnným X1,…,Xn
  - typy závislostí 2 proměnných
    - funkční vztah - 2 závislé proměnné, určité x odpovídá jediná hodnota y
    - korelace - 2 náhodné (nezávislé) proměnné, síla vztahu popisovaná korelačním koef. R
    - **regrese** - vztah náhodné (nezávislé, vysvětlující) proměnné x a závislé (vysvětlované) proměnné y, má pravděpodobnostní rozdělení
- pro x a y hledáme přímku (její parametry) vystihující průběh jejich závislostí = je nejblíže všem bodům
- závislost y na x můžeme popsat (aproximovat) pomocí **regresní rovnice (přímky)**:

  $y = \beta 1x + \beta 0 + \varepsilon$

  $$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^{n} x_j \hat{\beta}_j$$

  - ε je regresní odchylka
  - alternativně používaný zápis: y = bx + a

- **metodou nejmenších čtverců** - minimalizace rozdílů mezi skutečnou a očekávanou hodnotou (jednorozměrná regrese)



    - nejpoužívanější metoda pro odhad parametrů q_i do regresního modelu
    - minimalizuje střední kvadratickou chybu RSS:

$$minRSS(\beta) = min \sum \blacksquare = min \sum (y_i - f(x_i))^2 = \\ = min \sum (y_i - \beta_1 x_i - \beta_0))^2$$

    - regresní koeficienty
        - odvození
            - sumu zderivujeme podle všech qi a získáme rovnice rovné 0 (jako při hledání minima)
        - β1 – směrnice
          $\beta_1 = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2}$
            - tan úhlu s osou x
        - β0 - absolutní člen
          $\beta_0 = \frac{\sum y_i - \beta_1 \sum x_i}{n}$
    - Mnohorozměrná regrese
        - β0 se často zahrnuje do celého vektoru **β**
            - pak se mu říká bias
        - není-li X^TX singulární:
          $\overrightarrow{\beta} = (X^T X)^{-1} X^T \overrightarrow{y}$

- vlastnosti
    - je hladká, **stabilní** (cluster dat mimo přímku jí moc nerozhodí)
    - výrazně závisí na předpokladu linearity hranice
    - má **malý rozptyl**, potenciálně **velké vychýlení** (bias).

❣ **přeučení** - přílišná závislost modelu na trénovacích datech, která zvyšuje chybu na testovacích datech

- **Příklad:** růst oranžové křivky při klesající modré

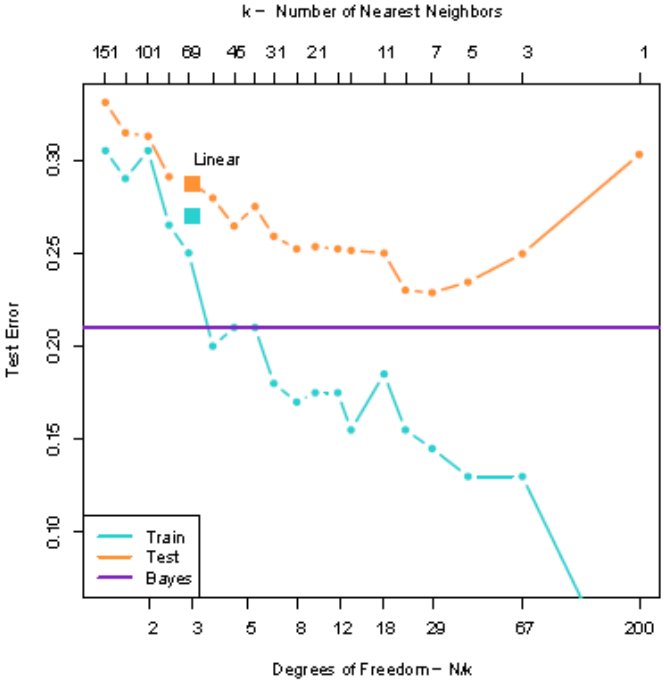


FIGURE 2.4. *Misclassification curves for the simulation example used in Figures 2.1, 2.2 and 2.3. A single training sample of size 200 was used, and a test sample of size 10,000. The orange curves are test and the blue are training error for k-nearest-neighbor classification. The results for linear regression are the bigger orange and blue squares at three degrees of freedom. The purple line is the optimal Bayes error rate.*

❣ **optimální bayesovská hranice** - Pokud víme, jakým procesem jsou data generována, můžeme v každém bodě predikovat nejčastější hodnotu (při stejné ceně chyby)

- obě hodnoty stejně časté

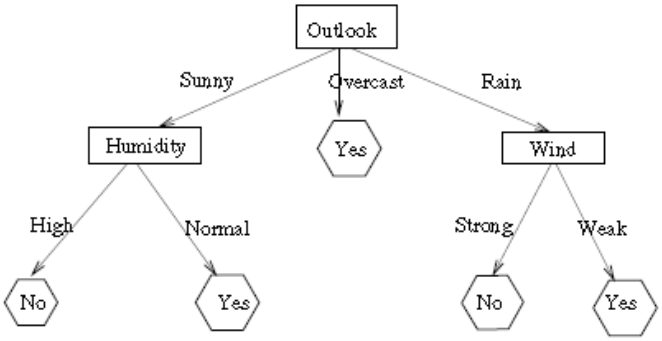
❣ **prokletí dimenzionality** - nejbližší sousedé jsou při velké dimenzi hodně daleko

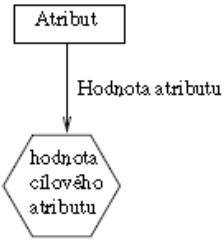
- řeší se například pomocí PCA

3. Rozhodovací stromy (Ch. 9)

Vstup: Vektory příznaků; Výstup: Třídy klasifikace
Vnitřní uzel se větví podle hodnoty svého atributu (s atributy rodičů již zafixovanými)

- klasifikace množiny na zaklade atributu každého prvku
- posloupnost dotazu na jednotlivé atributy (na rovnost nebo vetší/menší)
- chceme mít co nejkratší cesty
- složitost rozhodovacích stromu: *hnlog(n)*
    - kde *h* je pocet atributu





❢ Entropie (disorder) - míra neuspořádanosti v systému

$$E_G(data) = -\sum_g \underbrace{\frac{|data_{G=g}|}{|data|}}_{\text{pomer dat ve tride a vsech}} \log_2 \frac{|data_{G=g}|}{|data|} = -\sum_i p_i \log_2 p_i$$

- p_i - relativní četnost třídy i, pro 0 doplníme výsledek limitou = 0
- p_i - počet dat v *data* patřící do třídy *g* dělený celkovým počtem dat v *data*

❢ Informační zisk (information gain)

$$Gain(data, X_j) = \underbrace{E_G(data)}_{\text{celkova entropie}} - \underbrace{\sum_{x_j} \frac{|data_{X_j=x_j}|}{|data|} E_G(data_{X_j=x_j})}_{\text{quality of test}}$$

- entropie mínus vážená entropie

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 entropií podmnožin
    - podmnožin je stejně jako hodnot atributu
    - nejvyšší gain
  - vytvoř uzel stromu obsahující atribut
  - rekurze na podmnožiny pomocí zbývajících atributů
    - pokud mají všechny stejnou klasifikaci, nebo dojdou atributy -> list

❣ Prořezávání

- Mám validační data, nepoužitá k učení.
- Na těchto datech spočtu chybu predikce stromu.
- Rekurzivně zvažuji sloučit listy a z rodiče udělat list – provedu, pokud to sníží chybu.
- Occamova břitva – jednodušší modely jsou lepší
- Nebo můžeme i bez validačních dat:
  - ❣ **CART** - krosvalidací určím velikost stromu (bez potřeby validačních dat)
    - generuje stromy postupným slučováním listů
    - **C**lassification **A**nd **R**egression **T**rees
    - prostor hodnot atributu rozpivotován dělící hodnotou
    - Minimalizace přes pivoty:
      $2P_L P_R \sum_i |P(C_i|t_L) - P(C_i|T_R)|$
  - statistikou motivovaný odhad chyby z trénovacích dat.

5. Support vector machines (Ch. 12)

❢ Optimální separující nadrovina

nadrovina okolo které je takový maximálně široký pás, že na každé straně jsou vzorky jedné třídy

- $max_{\beta,\beta 0,||\beta||=1} M$
  kde $y_i(x_i^T\beta + \beta_0) \geq M$
  - $= min_{\beta,\beta 0}||\beta||$
    kde $y_i(x_i^T\beta + \beta 0) \geq 1$
    - $M = 1/||\beta||$
- Cílovou třídu (= y_i) kódujeme -1,1
- Data pošleme do optimalizační procedury, dostaneme pro každý datový vzor αi
- Vybereme vzorek s $0 < \alpha_i < \gamma$, označíme s.
  - parametr γ ("váhu" $\sum \xi_i$) musíme určit laděním
- **Klasifikační pravidlo** (nový vzorek *x* klasifikujeme)

$$\hat{G}(x) = \sum_{i=1}^{N} \alpha_i y_i x^T x_i + (y_s - x_s^T \beta).$$

  - což je
    $\hat{G}(x) = sgn(x^T\beta + \beta_0)$
- **slack ξ**$_i$ (odchylky, relaxační proměnné) - vzdálenost vzorku i od hranice jeho třídy
  - je-li příklad na správné straně $\xi_i = 0$

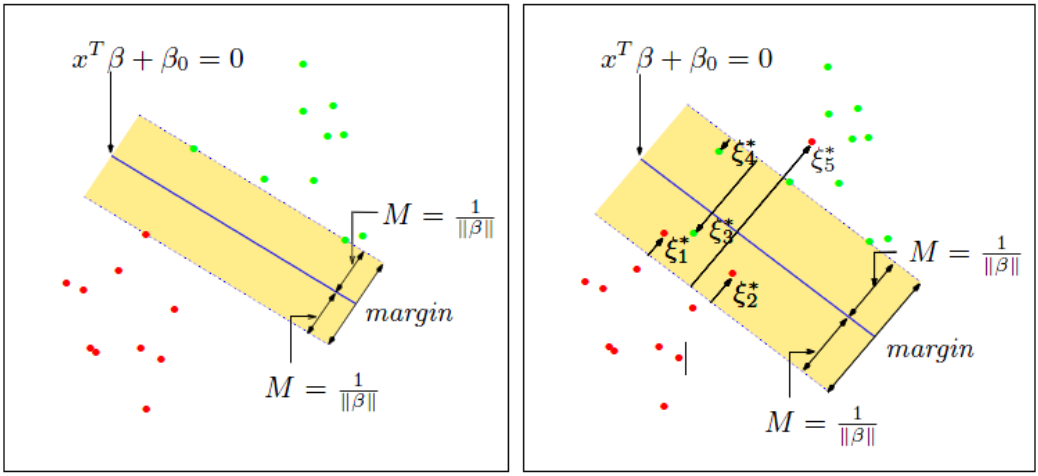


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width $2M = 2/\|\beta\|$. The right panel shows the nonseparable (overlap) case. The points labeled ξ_j^* are on the wrong side of their margin by an amount $\xi_j^* = M\xi_j$; points on the correct side have $\xi_j^* = 0$. The margin is maximized subject to a total budget $\sum \xi_i \leq$ constant. Hence $\sum \xi_j^*$ is the total distance of points on the wrong side of their margin.*

❣ Support Vector Machines

hledají optimální separující narovinu v nelineární transformaci prostoru

- separujeme pomocí nelineární funkce:
  $f(x) = h(x)^T\beta + \beta_0 = \sum \alpha_i \gamma_i \langle h(x), h(x_i)\rangle + \beta_0$
  - skalární součin nového a trénovacího vzoru

v transformovaném prostoru

- volíme transformační prostor tak aby se dobře počítal skalární součin $\Rightarrow$ málo dimenzí
- **jádrové (kernal) funkce** - počítaní skalární součin v transformačním prostoru
  - Příklady:

$$d\text{th-Degree polynomial: } K(x, x') = (1 + \langle x, x'\rangle)^d,$$

$$\text{Radial basis: } K(x, x') = \exp(-\gamma\|x - x'\|^2),$$

$$\text{Neural network: } K(x, x') = \tanh(\kappa_1\langle x, x'\rangle + \kappa_2).$$

- Klasifikační pravidlo

$$\hat{G}(x) = \sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + (y_s - x_s^T\beta).$$

6. Ohodnocení a volba modelů (Ch. 7)

❢ Míry chyby pro regresi

- jak se chybí predikce $\hat{f}(x)$ od správné hodnoty Y
- Regrese - spojitá cílová veličina (nejčastěji)
  - kvadratická chyba: $L(Y, \hat{f}(x)) = (Y - \hat{f}(x))^2$
  - absolutní chyba: $L(Y, \hat{f}(x)) = |Y - \hat{f}(x)|$

❢ Míry chyby pro klasifikaci

- **klasifikace** - diskrétní *G* nabývající jednu z *K* možných hodnot *1...K*, modelujeme pravděpodobnosti $p_k(X) = P_r(G = k|X)$, predikujeme to s největší $p_k(X)$
- **chyba predikce** (0-1 chyba) $L(G, \hat{G}(X)) = I(G \neq \hat{G}(X))$
- **log-likelihood** = - logaritmus věrohodnosti (cross-entropy, deviance) $L(G, \hat{p}(x)) = -2\sum_k (G = k) log\hat{p}_k(X) k = 1 = -2log\hat{p}_G(X)$

❢ dost dat x málo dat

- Ohodnocení modelu, dost dat



  - Doporučuje se poměr: ½,1/4,1/4
- **Ohodnocení klasifikátoru, málo dat**
  - Tuto úlohu převedu na ohodnocení algoritmu při málo datech.
- Průměrem přes několik až mnoho vzorků získám odhad chyby algoritmu, který použiji i na můj ohodnocovaný klasifikátor (přímo nebo přepočtený).
- **Ohodnocení algoritmu, málo dat**
  - **krosvalidace** (10 dílů) provést více testů, výsledky zprůměrovat
  - **one-leave-out** (spec.případ krosvalidace) vždy testuji jen na 1 vzorku
  - **bootstrap** do testovací/trénovací množiny vybíráme s opakováním
    - odhad chyby: součet prům.chyb příkladů
      - *korekce pro počet dat (0,632) + chyba na trén.datech
      - *korekce pro přeučení

7. Kombinace modelů (Ch. 8,10)

bagging - bootstrap aggregating

- pomocí bootstrap vygenerujeme více datových sad,
- naučíme modely, necháme hlasovat či průměrujeme.

stacking - učíme model (váhy) který kombinuje nižší modely

❢ boosting

učíme na vážených datech, váhu špatně klasifikovaných dat zvyšujeme

❢ AdaBoost.M1

1. přiřaď stejnou váhu $w_i = \frac{1}{N}$ všem trénovacím instancím
2. ∀*m* ∈ *M* iterací
   (a) spočti chybu err_m modelu $G_m(x)$ na vážených trénovacích datech
   $$err_m = \underbrace{\frac{\sum_i w_i I(y_i \neq G_m(x_i))}{\sum_i w_i}}_{\text{pomer spatnych vah ke vsem}}$$
   ☀ *I* je char.fce (vrací 1 pokud je splněna podmínka, jinak 0)
   (b) $\alpha_m \leftarrow log\frac{1-err_m}{err_m}$
   (c) pokud $err_m = 0$ nebo $err_m \geq 0.5$ končíme tvorbu modelu
   (d) upravíme váhy $w_i \leftarrow w_i exp(\alpha_m I(y_i \neq G_m(x_i)))$
   (e) normalizuj váhy všech instancí: $\sum_i w_i = 1$ (vynucuje distribuci)
3. vracíme prediktor $G(x) = sign[\sum \alpha_m G_m(x)]$

8. Bayesovské učení

❢ maximálně pravděpodobná hypotéza (max. apost. probability)

☀ podmíněná = aposteriorní

- $h_{MAP} = argmax_i P(h_i|data) = argmax_i P(data|h_i).P(hi)$
- spočtu přes tabulku

❢ Bayesovsky optimální odhad

- váží předpovědi jednotlivých hypotéz pravděpodobností těchto hypotéz
  $P(N = c|data) = \sum P(N = c|h_j, data).P(h_j, data)$
- $= \sum P(N = c|h_j).P(h_j, data)$
- $P(h|data) = \frac{P(h \cap data)}{P(data)} = \frac{P(data|h)P(h)}{P(data)}$

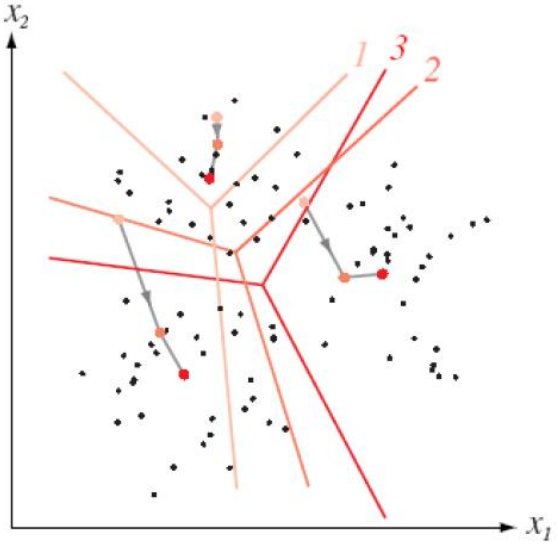
❢ maximálně věrohodná hypotéza (max. likelihood)

- všechny hypotézy považuji za stejně pravděpodobné
- hledám nejpravděpodobnější odhad
- $h_{ML} = argmax_i P(data|h_i)$
- např. pro alt. rozdělení: $P(data|h_p) = p^c(1-p)^l$
  - p najdu zlogaritmováním($\Rightarrow$argmin) a zderivováním (+položením =0)

9. EM algoritmus

❢ Shlukování metodou k-středů (k-means)

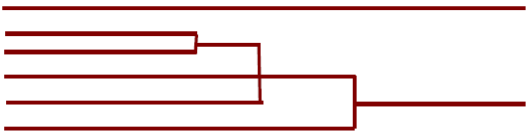
- Inicializace:
  - Náhodně zvol rozklad do k shluků
- Iterace:
  - **Update-step:** Urči centroidy pro všechny shluky v aktuálním rozkladu
    - např: $\mathbf{C} = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_k}{k}$
  - **Assignment-step:** Pro každý vzor **x** přesuň do nejbližšího shluku
  - nedošlo-li k žádnému přesunu -> KONEC
    - Příklad:



- Volba reprezentace důležitější než algoritmus
- např. pro vektorovou kvantizaci zpracování obrazu
- k-medoids pro vzdálenost bez metriky
  - vzdálenost musíme počítat pokaždé = kvadr. složitost

Algoritmus hierarchického shlukování (metodou „zdola nahoru“)

- na začátku je každý bod cluster
- dokud je více než k shluků
  - Urči vzájemné vzdálenosti mezi všemi clustery
  - vždy shluknu dva nejbližší body do nového clusteru



❣ EM algoritmus (estimate maximize)

- používá se pro odhad nepozorovaných veličin
- Inicializace:
  - náhodně zvol parametry distribuce
- Hlavní cyklus:
  - **Estimate:** podle parametrů distribuce spočítej hodnotu skrytých veličin
  - (odhad hodnot nepozorovaných dat)
  - **Maximalizace:** z nich urči max. věrohodný odhad parametrů distribuce
  - (max.věrohodnost vzhledem k datům přes uvažované modely)
  - pokud došlo ke změně parametrů, opakuj
    - iteruj až k pevnému bodu

❣ Směs jednorozměrných gausovských distribucí

- 2 distribuce mají parametry: $\pi, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2$
- na začátku μ náhodně, $\pi = 0.5$, $\sigma_1 = \sigma_2 =$ výběrový rozptyl
- **Estimate** - krok:
  - $$\gamma_i = \frac{\pi \Phi_{N(\mu_1,\sigma_1^2)}(x_i)}{((1-\pi)\Phi_{N(\mu_2,\sigma_2^2)}(x_i) + \pi \Phi_{N(\mu_1,\sigma_1^2)}(x_i))}$$
    - ϕ je distribuční funkce
- **Maximize** krok: odhadnout střední hodnoty a rozptyly
  - $\mu_1 = \sum_i \gamma_i x_i / \sum_i \gamma_i$ (vážené průměry)
  - $\sigma_2^2 = \sum_i (1-\gamma_i)(x_i - \mu_2)^2 / \sum_i (1-\gamma_i)$
  - $\pi = \sum_i \gamma_i / N$ (pravděpodobnost 1. klastru)

Směs gausovských distribucí

- **Estimate:** každý příklad "rozdělím" mezi klastry $k = 1...K$
  - $p_k i = P(C = k|x_i) = \alpha_i . P(x_i|C = k) . P(C = k)$
  - Součtem přes všechny příklady i spočteme váhu klastrů
  - $p_k = \sum_i p_{ki}$
- **Maximize:** pro daná data spočteme maximálně věrohodný odhad
  - $\mu_k \leftarrow \sum_i \frac{p_{ki}}{p_k} x_i$
  - $\sum_k \leftarrow \sum_i (p_{ki}/p_k)(x_i - \mu_k)(x_i - \mu_k)^T$
  - $P(C = k) \leftarrow p_k / \sum_l p_l$

10. Učení bez učitele (Ch. 14)

❢ MBA (Asociační pravidla) Analýza nákupního koše

- hledá četné kombinace atributů
- 3 míry
  - support
    $podpora(pravidlo) = \frac{\#i\&j}{\#vsech}$
  - confidence (přesnost, spolehlivost, predictability)
    $spolehlivost(pravidlo) = \frac{\#i\&j}{\#i}$
    - interval [0; 1]
  - lift (zdvih, zlepšení)
    $zlepseni(pravidlo) = \frac{p(i\&j)}{p(i)p(j)} = \frac{\#i\&j\#vsech}{\#i\#j}$
    - 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)

❢ Algoritmus APRIORI

generování asociačních pravidel (frequent itemsets)

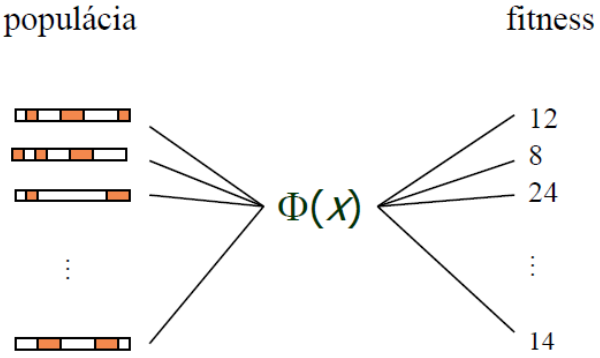
- Generování kombinací do šířky - |kombinace| = k z kombinací délky k-1
  - Každou kombinaci délky k-1 porovnáme s každou jinou, shodují-li se v k-2 itemech, spojíme
  - Spojené zpětně profiltrujeme, chybí-li nám nějaká jejich podkombinace délky k-1, zahodíme je
- Do L1 přiřaď všechny kategorie s četností ≥ t
- 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 s četností ≥ t
- return ∪ Lk;

PRIM

11. Genetické algoritmy

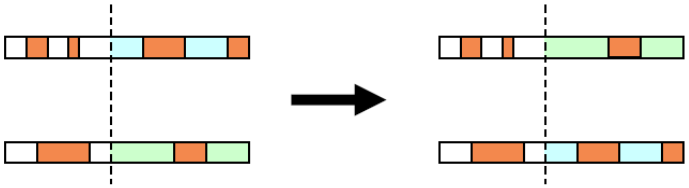
Populace P(t) umělých chromozomů

- **Chromozom (genotyp)** - řetězec symbolů kódující vlastnosti jedince (fenotyp)
- **Fitness funkce** Φ: genotyp → R
  - jak dobře řeší problém, čím vyšší tím lepší

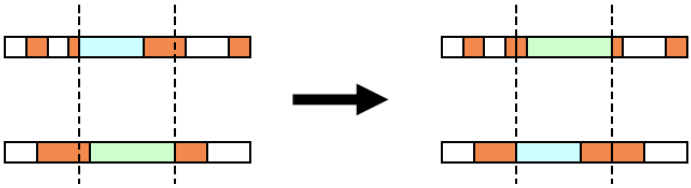


Jednoduchý genet.algoritmus

- **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
  - proveď **křížení**
    - náhodně se spárují a zkříží podle zadané pravděpodobnosti
      - **jednobodové**



      - **vícebodové**



  - **mutuj** chromozómy
    - náhodná změna bitů podle zadané pravděpodobnosti
  - skonči pokud: najdeme hledaného jedince nebo fitness nejlepšího neroste

Schema - abstraktní "šablona" genomu, ternarni retezec delky m, krome 1/0 jeste * (wildcard)

- může popisovat část chromozomu zaručující vysokou fitness
- **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.