Otázky ke zk: docs.google.com/document/d/12prKiWhRI2apqWih4CYSVgwMiAeRp7BSVi_dzfk1NMg

Cvičení

- Cv. 6: (logistická regrese, 30.10.) part 1: https://youtu.be/bnIisMd4Evw
  part 2: https://youtu.be/RP39s26-KMU
  jiný zdroj: https://youtu.be/N6QX_GP3cns
- Cv. 7 (početní, 6.11.): https://youtu.be/JeJbs80MJ_I
- Cv. 8: (Perceptron, 13.11.): https://youtu.be/80RX_cpeeLY
- Cv. 9: (SVM, 20. 11.): https://youtu.be/m0-SqopKbLA (částma nic moc zvuk)
- Cv. 10 (SVM2, 27. 11.): https://youtu.be/M16YPOiwOwM
- Cv 4.12. https://youtu.be/dWLkRBbHfR8
- Cv 11.12 (K-means): https://youtu.be/DJVlT9-63rk
- Cv 18.12. (CNN): https://youtu.be/WCssvD6RWkU
- Cv 8.1. (závěrečné): https://youtu.be/VB5-VwLpBtQ (zvuk: celkem echo)

Přednášky

6. Přednáška 12.10. Parameter estimation EN: https://youtu.be/_z96Do0QzmQ

7. Přednáška 6.11. SVM CZ: drive.google.com/file/d/1G82geHaA14QIqDBihbasnqyf9--KaKMP

8. Přednáška 13.11. Adaboost CZ:
   drive.google.com/file/d/1nao_Vp0YdyEtk21G7_vnnR90BxE_09bR - bez zvuku

9. Přednáška 20.11. Neural networks: *(jen v playlistu viz dole)*

10. Přednáška 27.11.: K-means: p1: https://youtu.be/3kdSgKxgg4s p2: https://youtu.be/rYUgmMOPNrw

11. Přednáška 4.12 EM: https://youtu.be/yRchgCy9PIk

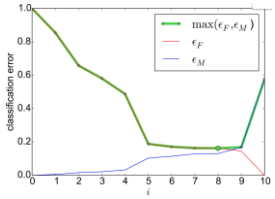
12. Přednáška 11.12. PCA & : p1: https://youtu.be/-0om_olAgII p2: https://youtu.be/Cc7yqAxLHzk

Playlist Kuba Zíka: https://www.youtube.com/playlist?list=PL7l7HfRJ__UDL9RGZ-puyWt-Ij3bI3D4d

- ***Bayesovský risk a Bayesovská strategie***

$$R(q) = \sum_{x \in X} \sum_{k \in K} p_{XK}(x,k)\, W(k, q(x)) \qquad q^* = \operatorname*{argmin}_{q \in X \to D} R(q)$$

- ***"Pravděpodobnosti"***
  - p(x,k) … sdružená (joint) probability; p(k) … prior probability
  - p(xlk) nebo p(xlθ)… likelihood → likelihood ratio používané v Neyman-Pearson, Minimax, Wald, Logistické regresi,
  - p(klx) … aposterior probability - Bayesovská strategie rozhoduje pro třídu s nejvyšší posteriorní pravděpodobností (given some měření x) ("odds" v log. regresi)
  - p(x,k) = p(xlk) * p(k) = p(klx) * p(x) … dle Bayesovy věty
- **MLE:** Známe 'formu' rozdělení s neznámými parametry θ (jejich počet je nízký)
  - $\hat{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \operatorname*{argmax}_{\boldsymbol{\theta}} p(\mathcal{T}|\boldsymbol{\theta})$ $\hat{\theta} = \operatorname*{argmax}_{\theta} \prod_{i=1}^{N} p(x_i|\theta)$
  - dělám odhad parametrů pro každou třídu zvlášť, postupně:
  - pomocí log-likelihood: produkt přejde na sumu → potenciální zjednodušení derivace
  - L(θ) = Prod(p(tᵢ)) → log (součin přejde na součet) → zderivuji a položím rovno nule
  - asymptoticky přejde ke správným parametrům
- **Neyman-Pearson**
- **:** formulace problému
  - K = {D, N} … dangerous state, normal state
  - známé conditionals: p(x|D), p(x|N); neznámé/neexistující prior probabilities p(D), p(N)
  - hledáme optimální strategii q* pro kterou platí:
    - chyba klasifikace dangerous třídy je nižší než threshold $\bar{\epsilon}_D \in (0,1)$, zatímco:
    - chyba klasifikace normal třídy je co nejnižší
  - tedy: $q^* = \operatorname*{argmin}_{q: X \to K} \sum_{x: q(x) \neq N} p(x \mid N)$ subject to: $\sum_{x: q(x) \neq D} p(x \mid D) \leq \bar{\epsilon}_D$
  - strategie je pak charakterizována pomocí: ϵ_N (false alarm) a ϵ_D (overlooked danger)
  - optimální strategie q*: existuje co nejmenší μ které zajistí:
    - $\frac{p(x \mid N)}{p(x \mid D)} > \mu \quad \Rightarrow \quad q(x) = N$ a $\frac{p(x \mid N)}{p(x \mid D)} \leq \mu \quad \Rightarrow \quad q(x) = D$ ("likelihood ratio")
    - $\epsilon_D \leq \bar{\epsilon}_D$ (overlooked danger < threshold)
  - pro spojitý případ je obecně nejlepší vzít takovou rozhodovací funkci, která má $\epsilon_D = \epsilon'_D$, protože s rostoucí ϵ_D klesá (či maximálně stagnuje) ϵ_N. Tato rovnost nelze vždy přímo dosáhnout u diskrétních pozorování
- **Minimax:** známé conditionals: p(x|D), p(x|N); neznámé/neexistující prior probabilities p(D), p(N)
  - X … set of observations, q: X → K … strategie
  - hledá strategii, která minimalizuje chybu na nejhůře klasifikované třídě (error nejhůře klasifikované třídy je nejmenší)
  - error je suma pravděpodobností p(x|k) na množině, kde q(x) rozhodne, že x nepatří do k
  - $q^* = \operatorname*{argmin}_{q: X \to K} \max_{k \in K} \epsilon(k)$, kde $\epsilon(k) = \sum_{x:\, q(x) \neq k} p(x \mid k)$ (popř. integrál pro spojité x)



  - pro 2 třídy použijeme likelihood ratio
  - ve spojitém případě se dostaneme do rovnosti errorů na třídách; místo sumy integrál
  - naleznu práh t (integruj: 0>t, t>inf), pro který se rovnají a pak: q(x)=1 když x<t, a naopak
  - nezáleží na apriorních pravděpodobnostech p(k)

- odpovídá Bayesovskému klasifikátoru při neznámých p(k) (??)

- ***Wald Task:*** 2 třídy: K = {1, 2}, nastavíme threshold ε ∈(0,1), chyba na žádné třídě ho nesmí překročit; D = {1, 2, "nevím"}
  - optimální strategie $q^* = \underset{q: X \to D}{\operatorname{argmin}} \max_{i=\{1,2\}} \kappa_i$ subject to
    (classif error pro class1: $\epsilon_1 = \sum_{x:\, q(x)=2} p(x \mid 1)$, undecided rate for kl1: $\kappa_1 = \sum_{x:\, q(x)=?} p(x \mid 1)$)
  - strategie je charakterizována $\epsilon_1 \leq \epsilon,\ \epsilon_2 \leq \epsilon$ pomocí ϵ_1, ϵ_2, κ_1, κ_2 (classification error; undecided rate)
  - pro spojité měření: integrály místo sum
  - pro 2 třídy použitím likelihood ratia r(x) - hledám 2 thresholdy: $q(x) = \begin{cases} 2 & \text{for } r(x) < \mu_l \\ 1 & \text{for } r(x) > \mu_h \\ ? & \text{for } \mu_l \leq r(x) \leq \mu_h \end{cases}$

- ***Parzen:*** do každého bodu vkládáme jádrovou funkci
  - je třeba normovat tak, aby integrál přes všechny jádrové funkce sčítal do 1
    - každý jádro přenásobit 1/N (N = počet bodů x)
  - kernel může být např. hyperkrychle, hyperkoule nebo "hypergauss"

- **K-NN:** algoritmus

Algorithm:

1. Given x, find K points $S = \{(x'_1, k'_1), (x'_2, k'_2), ..., (x'_K, k'_K)\}$ from the training set $\mathcal{T}$ which are closest to x in the metric d:

$$S = \{(x'_1, k'_1), (x'_2, k'_2), ..., (x'_K, k'_K)\} \equiv \{(x_{r_1}, k_{r_1}), (x_{r_2}, k_{r_2}), ..., (x_{r_K}, k_{r_K})\} \quad (18)$$

$$r_i\text{: the rank of } (x_i, k_i) \in \mathcal{T} \text{ as given by the ordering } d(x, x_i) \quad (19)$$

2. Classify x to the class k which has majority in S:

$$k = \underset{l \in R}{\operatorname{argmax}} \sum_{i=1}^{K} [\![k'_i = l]\!] \qquad (x'_i, k'_i) \in S \quad (20)$$

-
  - Máme trénovací množinu T = $\{(x_i, k_i)\}_{1 <= i <= N}$. Pro testovací bod y nalezneme k nejbližších trénovacích bodů x_i a y ohodnotíme jako té třídy, která má mezi k nejbližšími body většinu.

  - výhody/nevýhody
    (+) jednoduchá implementace (naivní řešení - složitost: D*N)
    (+) pro 1-NN: lze najít upper-bound ($\epsilon_{NN} \leq 2\epsilon_B$) a lower-bound: ϵ_B (Bayes error)
    (-) při naivní implementaci pomalé, vysoké nároky na paměť počítače (příp. nutná úprava dat)
    → lze urychlit pomocí k-D trees nebo Voronoi nebo redukcí trn množiny
    (-) není jasné jak zkonstruovat metriku d
    (potenciální problém: různý význam různých dimenzí)
    (-) vysoké nároky na paměť, (-) může overfitovat
    (+) není potřeba předpoklad na distribuci, ze které data pocházejí
    (+) pro libovolný počet tříd
  - *urychlení klasifikátoru 1-NN:*
    - redukcí trénovací množiny: snížení N např. z tisíců na desítky
    - pomocí k-D stromů: ty urychlují vyhledávání podle vzdálenosti v geometrickém prostoru pomocí ukládání dat ve stromové struktuře. Při procházení stromu se pro každou větev určí minimální možná vzdálenost nejbližšího bodu ve větvi od posuzovaného bodu. Po prvním projití do listu pak můžeme ořezat větve, které jistě nemají bližší bod. Je to sublineární vyhledávání, ale složitost je funkcí dat, nejde tedy garantovat.

    - pomocí Voronoiových diagramů: při klasifikaci stačí zjistit v jaké buňce bod leží (rychlejší než počítat vzdálenosti)

- **Logistická regrese**
  - lineární klasifikátor, log odds
  - $q(\mathbf{x}) = 1 \quad \text{if} \quad p(1|\mathbf{x}) > p(2|\mathbf{x}) \quad \Leftrightarrow \quad \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} > 1 \quad \Leftrightarrow \quad \ln \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} > 0$
  - a obrácené znaménko pro q(x) = 2
  - pokud je poměr lineární, přecházíme na problém hledání w=[w_0,w]; $w^Tx > 0 \rightarrow q(x)=1$
  - priors způsobí jen posun (jsou obsaženy v členu w_0)
  - log odds: $\ln \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} = \mathbf{w} \cdot \mathbf{x}$ → p(1|x) = p(2|x) exp(w·x), p(2|x) = p(1|x) exp(−w·x)
    → 1 = p(1|x) + p(2|x) = p(1|x) (1 + exp(−w·x)) = p(2|x) (1 + exp(w·x))
  - k : {-1,1} → $p(k|\mathbf{x}) = \frac{1}{1+e^{-k\mathbf{w}\cdot\mathbf{x}}}$ (sigmoida)
  - algoritmus:
    0. upravit trn mn: {(x_i, y_i)}, x z R^D, y = +-1, na {y_i*(x_i,1)}. Init w z R^{D+1}
    1. minimalizujeme funkci E(w) = -l(w) = Sum$_i$ ln(1 + e^{-wx})
    Vypočítáme gradient $\nabla_w E$ = - Sum$_i$ x*e^{-wx}/(1 + e^{-wx})
    Updatujeme w:= w - μ*$\nabla_w E$ (jdeme proti směru grad, protože minimalizujeme E)
    Pokud E($w_{nové}$) < E($w_{staré}$): step = 2*step, jinak: step = step/2 a neupdatuj w,g,E
    2. goto 1
- *Diskriminační funkce pro 2 třídy se stejnou* ***kovarianční maticí***
  - $\ln \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} = \ln \frac{p(\mathbf{x}|1)}{p(\mathbf{x}|2)} \frac{p(1)}{p(2)} = \ln \frac{p(\mathbf{x}|1)}{p(\mathbf{x}|2)} + \ln \frac{p(1)}{p(2)}$
  - $\ln \frac{p(1|\mathbf{x})}{p(2|\mathbf{x})} = \ln \frac{\cancel{(2\pi)^{-\frac{D}{2}}}|\boldsymbol{\Sigma}|^{-\frac{1}{2}}}{\cancel{(2\pi)^{-\frac{D}{2}}}|\boldsymbol{\Sigma}|^{-\frac{1}{2}}} e^{-\frac{1}{2}\{(\mathbf{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) - (\mathbf{x}-\boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_2)\}} + \ln \frac{p(1)}{p(2)}$
  - $\ln(e^x)=x$
  - maticově roznásobit
  - $x^T\Sigma^{-1}x$ se požerou
  - $x^T\Sigma^{-1}\mu = \mu^T\Sigma^{-1}x$ (protože Σ je symetrická: $\Sigma = \Sigma^T$)
  - $\underbrace{[(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^\top \boldsymbol{\Sigma}^{-1}]}_{\mathbf{w}} \mathbf{x} + \underbrace{\frac{1}{2}(\boldsymbol{\mu}_1^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2) + \ln \frac{p(1)}{p(2)}}_{w_0}$
  - → **w**·**x** + w_0 , kde **w** ∈ R^D, w_0 ∈ R
  - Závěr: Pro 2 třídy s norm. rozdělením se stejnými kov. maticemi, log odds je lin. fce
- Kov. mat: → vl. čísla (a vektory): vidím osy elipsy Gausse, velké vl. č. = velká var v daném směru
- **Perceptron**
  - algoritmus:
    - příprava dat: $\mathbf{x_i} = k_i * [y_i \mathbf{x_i}]$, **w** = [$w_0$ **w**] → $\mathbf{x_i}$, **w** ∈ R^{D+1}
    - t=0, $w^{(t)} = [0]$
    - najdi x_j: $x_j \cdot w \leq 0 \rightarrow w^{(t+1)} = w^{(t)} + x_j$
  - pro neseparabilní data:
    - dimension lifting
    - s použitím batch version, pamatuju si nejlepší w*, kdy počet špatně klasif byl nejnižší

$$J_p(\mathbf{w}) = \sum_{\substack{\mathbf{x}\in X \\ \mathbf{w}\cdot\mathbf{x}\leq 0}} (-\mathbf{w}\cdot\mathbf{x}) \qquad \nabla J_p(\mathbf{w}) = \frac{\partial J}{\partial \mathbf{w}} = \sum_{\substack{\mathbf{x}\in X \\ \mathbf{w}\cdot\mathbf{x}\leq 0}} (-\mathbf{x})$$

- cost funkci lze redefinovat: →
- update buď klasicky špatně klasifikovaným x ;anebo batch version: všechny špatně klasifikované a update jejich součtem (možné ještě škálovat aby krok nebyl tak velký)

- **SVM**
  - lineární klasifikátor; nachází optimální řešení (dělící nadrovina maximalizuje šířku dělícího pásu); lze upravit i pro lin. neseparabilní data; (w, b)* = $\text{argmax}_{w,b}$ $\min_i$ d(x_i) za podmínky $y_i(w^T x_i + b) > 0$, kde d(x_i) je vzdálenost x_i od nadroviny
  - T = {x_i, y_i} (i=1...N), $x_i \in R^D$, $y_i \in \{-1,1\}$
  - primární formulace: $(\mathbf{w}^*, b^*) = \underset{(\mathbf{w},b)}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{w}\|^2$, tj. argmax 1/IIwII, w ∈ R^D, b ∈ R
  - duální f: α*= $\underset{\alpha}{\operatorname{argmax}} \left\{ \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i}^{} \sum_{i=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \right\}$ (podm: α >= 0, Sum_i $y_i\alpha_i$ = 0)
    → pomocí QP najdeme glob. optimum
  - strategie pak: $q(\mathbf{x}) = \text{sign}(\mathbf{w}\cdot\mathbf{x} + b)$, $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$ (pouze α>0, tj. SV (ostatní mají α=0))
  - učení pomocí iteračních metod, funkce je konvexní → dospěje do globálního minima
  - Soft margin SVM: pro nesep. data představíme C - ztráta za špatně klasifikovaný bod; $0 \leq \alpha_i \leq C$; Primární úloha: (w, b)* = $\text{argmin}_{w,b}$ ½||w||² + C*Sum_i ξ_i za podmínky $y_i(w^T x_i + b) >= 1 - \xi_i$, $\xi_i >= 0$; Duální úloha: α* = argmax_α Sum_i α_i - ½ Sum_i Sum_j $\alpha_i\alpha_j y_i y_i x_i^T x_j$ za podmínky 0 <= α <= C, Sum_i $y_i\alpha_i$ = 0
  - Při použití dimension lifting φ(x): R^D -> R^M nemusíme používat/znát přímo předpis mapování. Stačí znát kernel funkci K(x, y) = $\varphi(x)^T\varphi(y)$. Tou lze nahradit například i mapování do teoreticky nekonečně dimenzionálních prostorů apod. Učení SVM probíhá stejně, pouze duální úloha přijme kernel funkci: α* = argmax_α Sum_i α_i - ½ Sum_i Sum_j $\alpha_i\alpha_j y_i y_i K(x_i, x_j)$ za podmínky 0 <= α <= C, Sum_i $y_i\alpha_i$ = 0
  - Klasifikace má dva způsoby:
    - použijeme mapování φ(x): Získáme w = $\text{Sum}_{i\ z\ SV}$ $\alpha_i y_i \varphi(x_i)$, b = y_{sv} - $w^T\varphi(x_{sv})$ a klasifikujeme: q(z) = sign($w^T\varphi(z)$ + b); To znamená, že musíme znát φ a klasifikace je zpomalena oproti lineární variantě výpočtem φ(z)
    - použijeme kernel K(x, y): b = y_{sv} - $\text{Sum}_{i\ z\ SV}$ $\alpha_i y_i K(x_i, x_{sv})$; q(z) = sign(b + $\text{Sum}_{i\ z\ SV}$ $\alpha_i y_i K(x_i, z)$); To znamená, že si musíme pro klasifikaci pamatovat α_i, y_i, x_i pro support vektory a klasifikace je zpomalena výpočtem $\text{Sum}_{i\ z\ SV}$ $\alpha_i y_i K(x_i, z)$
  - časová složitost závislá na N (ne na D; při optimalizaci jsou počítány skalár. součiny x_i^T,x_j)
  - rychlost učení závisí na inicializaci
  - dobře generalizuje, používají pouze skalární součiny dat, tzn. že jsou nezávislé na vstupní dimenzi, umí pracovat i s neseparabilními daty, při klasifikaci stačí znát pouze w a b, lze zobecnit pro oddělení nelineárně separovatelných dat pomocí dimension lifting a kernel triku.
  - SVM se složitě zobecňují pro oddělení více tříd a nehodí se do inkrementálního učení.
- *Perceptron vs SVM*
  - Perceptron nalezne nějaké řešení, SVM “dobré” (maximalizující margin)
  - u perceptronu pro separabilní data zaručena konvergence (Novikoff teorém)

- **Adaboost:** množina B sl. klasifikátorů h(x)
  algoritmus
  - Init: t=1, $D_{t=1}(i)=1/N$
  - vyberu $h_t(x)$ s nejmenší chybou ϵ_t: $h_t = \arg\min_{h\in\mathcal{B}} \epsilon_t; \quad \epsilon_t = \sum_{i=1}^{L} D_t(i)\,[\![y_i \neq h(\mathbf{x}_i)]\!]$
    - divná závorka dá jedničku pokud výraz uvnitř platí, jinak nulu (tj. při chybě → 1)
  - pro vybraný slabý klasifikátor h_t mám chybu ϵ_t
    - pokud ϵ_t větší než 0,5 pak co?? nebo jen když se rovná 0,5 tak končím?
  - Váha daného slabého klasifikátoru: $\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
  - Update vah dat: $D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t}$, normalizační faktor Z_t: $Z_t = \sum_{i=1}^{L} D_t(i)e^{-\alpha_t y_i h_t(\mathbf{x}_i)}$
    - lze vypočítat také takto: $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$
  - Nakonec (po ukončení iterování) – výsledný (silný) klasifikátor: $H(\mathbf{x}) = \mathrm{sign}(f(\mathbf{x}))$, $f(\mathbf{x}) = \sum_{t=1}^{T}\alpha_t h_t(\mathbf{x})$
- *Adaboost: Upper bound theorem:* $\epsilon = \frac{1}{L}\sum_{i=1}^{L}[\![y_i \neq H_T(\mathbf{x}_i)]\!] \leq \prod_{t=1}^{T} Z_t$
- *Adaboost: Vlastnosti:*
  - (+) jednoduchá implementace
  - (+) dobrá generalizace
    (= chová se dobře k testovacím datům po natrénování, neoverfittuje moc)
  - (-) overfitting kvůli noise
- **K-means**
  - data x_i, chceme roztřídit do k tříd; hledáme K center shluků:
    {ck*} = $\mathrm{argmin}_{(ck)} \sum_{k=1}^{K}\sum_{\mathbf{x}\in\mathcal{T}'_k} \|\mathbf{x}-\mathbf{c}'_k\|^2$ … abychom minimalizovali vzdálenosti od center
  - algoritmus:
    0. inicializace k centroidů (náhodně/k-means++)
    1. přiřazení bodů k nejbližšímu centroidu: $\mathcal{T}_k = \{\mathbf{x}\in\mathcal{T} : \forall j, \|\mathbf{x}-\mathbf{c}_k\|^2 \leq \|\mathbf{x}-\mathbf{c}_j\|^2\}$
    2. spočítání nového těžiště shluku: $c_k = \frac{1}{|\mathcal{T}_k|}\sum_{\mathbf{x}\in\mathcal{T}_k}\mathbf{x} \quad \text{if } |\mathcal{T}_k|>0$, pokud je $T_k = \emptyset$: reinit c_k
    3. pokud $T_k = T_{k+1}$ (stejné clustery jako předchozí iterace), tak konec, jinak goto 1
  - lze zobecnit pro jakoukoliv metriku d
  - Inicializace pomocí K++ means zvyšuje pravděpodobnost nalezení globálního minima, alg: c_1 vybrán úplně náhodně, $d_l = \min_{\mathbf{c}\in\mathcal{C}}\|\mathbf{x}_l-\mathbf{c}\|$, $p(l) = \frac{d_l^2}{\sum_{l=1}^{L} d_l^2}$, dokud nemám K center
  - Můžeme uvíznout v lokálním extrému (lze řešit více běhy a výběrem nejlepšího řešení)
  - alg končí v konečném počtu iter; konečný počet stavů: N^K,

Okomentoval(a): [1]: Když = 0,5 --> konec. Protože při erroru 0,5 je úspěch stejný jako při náhodném vybírání (a tedy tento slabý klasifikátor už nezlepšuje klasifikaci.) Platí pro klasifikaci do 2 tříd.

- složitost: I(počet iter)·N·K·M; N·K: v každé iter porovnávám každý bod s každým shlukem

- **Decision trees**
  - IG (information gain) umět spočítat
    - Entropie
      - $\log_2$; informační entropie [bit]: vysoká - nic nevím, nízká - vím co se stane
    - $$IG(A) = H(P) - \sum_{i=1}^{k} \frac{|P_i|}{|P|} H(P_i)$$
    - spočítám IG pro každý atribut → tam kde vyjde nejvyšší, ten použiju
    - $$H(P) = -\sum_{j} \frac{|P^{(j)}|}{|P|} \log \frac{|P^{(j)}|}{|P|}$$ kde $P^{(j)}$ je počet bodů které spadnou do j-té dané třídy
    - entropy of P_i: $$H(P_i) = -\sum_{j} \frac{|P_i^{(j)}|}{|P_i|} \log \frac{|P_i^{(j)}|}{|P_i|} \quad \text{(entropy of } P_i\text{)}$$
  - výhody/nevýhody
    - (+) interpretovatelnost; rychlost rozhodování; nepotřeba použít všechny měření - jen ty relevantní; outliers ošetřeny speciální větví; pro klasifikační i regresní úlohy
    - (-) neexistuje efektivní způsob trénování (všechny na bázi greedy algoritmu → není záruka optimality); hrozí overfitting (→ randomizované stromy)
  - top-down: na začátku nastavím P=T; spočítám IG pro každý atribut, zvolím pro dělení ten který mi dá nejvyšší IG; rekurze pro větve
  - DT vs. SVM:
    - pro jakékoliv prostory, i nenumerické, s různými hodnotami/měřítky v různých dimenzích, apod. X jen pro data z R^D
    - trénovací množina může být libovolně velká, při klasifikaci se nevyužívá (klasifikace je sublineární vůči dimenzi dat) X trénovací množina může být libovolně velká, při klasifikaci se nevyužívá, pouze se z její části (tzv. support vektorů) vypočítá lineární klasifikátor q(x) = sign($w^T x$)
    - žádné záruky na generalizaci/optimalitu řešení X dobře generalizuje, jelikož minimalizuje VC dimenzi maximalizací margin mezi třídami dat, díky konvexnosti rozhodovací funkce a QP optimalizaci nalezne globální optimum
    - při učení je potřeba uchovávat veškerá data, při klasifikaci pak již DT samotný (posloupnost otázek/rozhodnutí) X při učení stačí uchovávat skalární součiny všech x_i, pro klasifikaci stačí uchovávat jediný vektor w z R^D
    - učení je závislé na složitosti stromu, respektive nic jako učení neprobíhá, strom je pouze sestaven, při stavbě stromu lze dodržovat určitá pravidla pro jeho obecně lepší klasifikaci (TDIDT) X učení je závislé na velikosti trénovací množiny a rychlosti konvergence QP optimalizace, která kriticky závisí na počáteční inicializaci
    - odchýlené hodnoty lze jednoduše odchytávat jako speciální větve stromu X vliv odchýlených hodnot lze regulovat velikostí ztráty za špatně klasifikovaný bod C, ta určuje směr “úzký margin bez špatně klasifikovaných bodů (vyšší C)” vs “širší margin se špatně klasifikovanými body (nižší C)”

- **VC dimenze**
  - měřítko kapacity
  - definována jako mohutnost (cardinality) největší sady bodů, které může algoritmus rozbít
  - pro lineární klasifikátory: VC = $\dim_{\text{prostoru}}$ + 1 (2D: přímka oddělí max 3 body vždy → VC=3)
  - Vapnik and Chervonenkis inequality:

$$\Pr\left(R(q) \leqslant R_{emp}(q) + \sqrt{\frac{1}{N}\left[h\left(\log\left(\frac{2N}{h}\right)+1\right) - \log\left(\frac{\eta}{4}\right)\right]}\right) = 1-\eta$$

    - člen s odmocninou se nazývá strukturální risk - ten je horní odhad rozdílu R a R_{emp}
  - 1-NN: VC = inf
- **NN vs SVM, Adaboost a K-NN**: na obrázky
- Porovnávání
  - na jaký data, učení, chování vůči neseparabilním datům, chování vůči noise/outliers, konvergence, paměťové nároky
- **EM (Expectation maximization)**
  - použití:
    - missing data - měření jsou nekompletní (pro některé x chybí nějaká složka)
    - latent variables - měření kompletní, ale bylo by mnohem lepší kdyby byla ještě nějaká informace (?), např. mix více rozdělení typicky
  - algoritmus
    1. Initialize μ_c, μ_t
    2. Compute $q(z_i) = p(z_i|y_i^\bullet, \mu_c, \mu_t)$ for all $i = 1, 2, ..., M$
    3. Recompute μ_c, μ_t according to Eqs.(41, 42)
    4. If termination condition is met, finish. Otherwise goto 2. (S2: “E-step”, S3: “M-step”)
  - $q^{(t+1)} = \underset{q}{\operatorname{argmax}}\,\mathcal{L}(q, \boldsymbol{\theta}^{(t)})$, $\boldsymbol{\theta}^{(t+1)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\,\mathcal{L}(q^{(t+1)}, \boldsymbol{\theta})$
  - jako k-means (nemáme informaci o třídě) (příklad car/truck):
    - 0. init μc, μt
    - 1. $q(z_i) = p(z_i|y_i^\bullet, \mu_c, \mu_t)$
    - 2. μc, μt: $\mu_c = \dfrac{\sum_{i=1}^{M} q(z_i = \text{car})\, y_i^\bullet}{\sum_{i=1}^{M} q(z_i = \text{car})}$
    - ukončit pokud je termination condition splněna
    - k-means se liší pouze v bodě 1 - tam se přiřazuje pouze 1 nebo 0

      $q(z_i = \text{car}) = [\![\,|y_i^\bullet - \mu_c| < |y_i^\bullet - \mu_t|\,]\!]$
  - častá aplikace: odhad mixu Gaussů
  - M-step počítá pravděpodobnost, že daný bod je generovaný danými komponenty
  - E-step počítá neznámé parametry analyticky
  - je užitečný a efektivní dále pro exponenciální rozdělení
  - chytře maximalizuje likelihood tím, že tlačí lower bound nahoru
  - iterativní metoda, nemusí skončit v globálním maximu

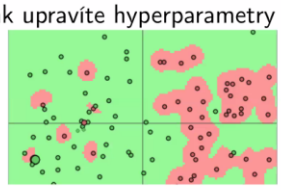
- opatrně inicializovat (stejně jako u K-means, NNs, …)

Test3

Úloha 1. (2b.) Máme trénovací sadu s měřeními $x_i \in R^2$ a labely $y_i \in \{1, 2\}$, $T = \{((3, 1); 2), ((2, 2); 2), ((2, 1); 1), ((1, 3); 1)\}$. Proveďte 3 iterace učení Perceptronu (vypisujte důležité hodnoty).

Úloha 2. (1b.) Trénujete soft-margin SVM s Gauss kernelem $K(x, y) = \exp(-\|x - y\|^2/2\sigma^2)$. Klasifikátor na trénovacích datech je na obrázku, na validačních datech ale nefunguje dobře. Jak upravíte hyperparametry C a σ? Proč?

1. zmenšit C, zmenšit σ
2. zmenšit C, zvětšit σ
3. zvětšit C, zmenšit σ
4. zvětšit C, zvětšit σ



Úloha 3. (1b.) Jaká vlastnost platí pro vztah mezi optimálními hodnotami kriteriální funkce primální a duální úlohy SVM?

Úloha 4. (1b.) Mějme funkci $K_\alpha(x, y) = \alpha(x + y)$ pro $x, y \in \mathcal{R}^3$ a $\alpha > 0$. Je tato funkce validní jádrová funkce (kernel)? Pokud ne, dokažte. Pokud ano, najděte odpovídající feature mapu $\phi(x)$.