Vzorové otázky na zkoušku

(http://cmp.felk.cvut.cz/cmp/courses/recognition/Exam-questions/exam-questions.pdf)

1. Definujte střední ztrátu strategie. Definujte význam všech použitých symbolů.
Pro strategii q(x): X -> D, kde X je prostor všech možných pozorování a D je prostor všech možných rozhodnutí definujeme střední ztrátu R(q) jako
$R(q) = \text{Sum}_{x \in X} \text{Sum}_{k \in K} p(x,k)*W(k, q(x))$,
kde p(x,k) je sdružená pravděpodobnost toho, že budeme pozorovat x při třídě k,
a W(k, d): K x D -> R je ztráta za rozhodnutí d při třídě k, a sčítáme přes všechny kombinace pozorování a tříd.

2. Kdy nelze rozpoznávací úlohu modelovat jako minimalizaci střední ztráty, t.j. jaká jsou omezení Bayesovské úlohy rozpoznávání?
Když neznáme pravděpodobnosti p(x,k), když nelze určit smysluplná ztrátová funkce W, nebo když hraje zásadní roli čas rozhodnutí a cena pozorování (počítá se s tím, že všechna pozorování jsou předem známa).

3. Kde se stává rozhodování podle minima střední ztráty rozhodováním dle minima aposteriorní pravděpodobnosti?
Doufám, že je tam překlep a má tam být “podle maxima aposteriorní pravděpodobnosti”, pak bych odpověděl, pokud máme 0/1 ztrátovou funkci, tj. za správně ohodnocenou třídu neplatíme žádnou ztrátu a za všechny druhy chyb platíme ztrátu 1.
Parciální riziko $R(x, q) = \text{Sum}_{k \in K} p(k|x)*W(k, q(x)) = \text{Sum}_{k:\, q(x) \neq k} p(k|x)*1 = 1 - p(k = q(x)|x)$.
Chceme min R(x, q), což dosáhneme maximalizací p(k = q(x)|x), neboli za q(x) volíme třídu k s největší apostreriorní pravděpodobností.

4. Co je to strategie rozhodování? Co je to Bayesovská strategie?
Strategie rozhodování je funkce q(x): X -> D, kde X je prostor všech možných pozorování a D je prostor všech možných rozhodnutí. Pro každé pozorování nám tedy dá určité rozhodnutí.
Bayesovská strategie je strategie q*, která minimalizuje Bayesovské riziko $R(q) = \text{Sum}_{x \in X} \text{Sum}_{k \in K} p(x,k)*W(k, q(x))$, $q^* = \text{argmin}_{q:\, X \to D} R(q)$.

5. Odvoďte tvar diskriminační funkce pro problém klasifikace do dvou tříd A, B. Rozdělení pravděpodobnosti ve třídách A i B jsou Gaussovské se stejnou kovarianční maticí C. Střední hodnoty se liší.
Máme $p(x|A) = 1 / \text{sqrt}((2*\pi)^D|C|) * \exp(- (x - x_A)^T C^{-1} (x - x_A) / 2)$, kde x_A z R^D je střední hodnota rozdělení třídy A, x je z R^D, $C = C^T$ je z R^{D*D}. Podobně pro p(x|B). Hezčeji:

$$p(\mathbf{x}|1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)}$$

(pro druhou třídu je rozdíl jen v μ_2)

Rozhodujeme podle poměru aposteriorních pravděpodobností:
x ohodnotíme jako A iff $p(A|x)/p(B|x) >= 1$
x ohodnotíme jako B iff $p(A|x)/p(B|x) < 1$
Přejdeme k log odds ratio a máme:
x ohodnotíme jako A iff $\ln(p(A|x)/p(B|x)) >= 0$

x ohodnotíme jako B iff $\ln(p(A|x)/p(B|x)) < 0$
Víme:
$p(x|A)*p(A) = p(x, A) = p(A, x) = p(A|x)*p(x)$
$p(A|x) = p(x|A)*p(A)/p(x) = p(x|A)*p(A)$, kde p(x) zanedbáme (je něco jako konstanta, respektive dále by se ve zlomku stejně zkrátilo)
$\ln(p(x|A)*p(A)/p(x|B)*p(B)) = \ln(\frac{1}{\sqrt{(2\pi)^D|C|}} * \exp(-(x - x_A)^T C^{-1}(x - x_A) / 2) * p(A) / (\frac{1}{\sqrt{(2\pi)^D|C|}} * \exp(-(x - x_B)^T C^{-1}(x - x_B) / 2) * p(B))) = -(x - x_A)^T C^{-1}(x - x_A) / 2 + (x - x_B)^T C^{-1}(x - x_B) / 2 + \ln(p(A)/p(B)) = -xC^{-1}x + 2x_A C^{-1}x - x_A C^{-1}x_A + xC^{-1}x - 2x_B C^{-1}x + x_B C^{-1}x_B + \ln(p(A)/p(B)) = 2(x_A - x_B)C^{-1}x - x_A C^{-1}x_A + x_B C^{-1}x_B + \ln(p(A)/p(B))$, kde modře je lineární člen $a^T x$ a červeně je konstantní člen b. Využili jsme $C = C^T$.
Nyní máme $f(x) = a^T x + b$ a rozhodujeme:
x ohodnotíme jako A iff $f(x) >= 0$
x ohodnotíme jako B iff $f(x) < 0$

6. Definujte rozpoznávací úlohu se třídou 'nevím' (tzv. reject option). Popište Bayesovskou strategii pro tuto úlohu.
Máme množinu tříd K. Rozhodnutí definujeme jako D = K U {"nevím"}.
Ztrátová funkce W je rozšířená 0/1 funkce:
W(k, d) = 0 iff k = d
W(k, d) = 1 iff k != d a d != "nevím"
W(k, d) = ε iff d = "nevím"
ε je ztráta při rozhodnutí nevím, v praktických problémech je $0 < \varepsilon < 1$.
Bayesovská strategie q* pro tuto úlohu je:
Pro každé x nalezneme $k^* = \text{argmin}_{k \in K} R(x, k) = \text{argmin}_{k \in K} \text{Sum}_{i != k}\, p(i|x) = \text{argmax}_{k \in K}\, p(k|x)$
$q^*(x) = k^*$ iff $R(x, k^*) = 1 - p(k^*|x) < \varepsilon$
$q^*(x) =$ "nevím" iff $R(x, k^*) = 1 - p(k^*|x) >= \varepsilon$

7. Jak lze odhadnout ztrátu strategie
a) se znalostí rozdělení pravděpodobnosti ve třídách (předpokládejme jednotkovou ztrátovou matici a shodné apriorní pravděpodobnosti):
Víme rozdělení pravděpodobnosti ve třídách, neboli p(x|k) pro všech K tříd k. p(k) = 1/K.
Máme 0/1 ztrátovou funkci (doufám, jednotková ztrátová matice vyjadřuje spíš "ziskovou" funkci, kdy W(d, k) = 1 při d = k a W(d, k) = 0 při d != k).
Ztráta strategie pak je $R(q) = \text{Sum}_{x \in X} \text{Sum}_{k \in K}\, p(x|k)*p(k)*W(k, q(x))$, zjednodušeně:
$R(q) = 1/K * \text{Sum}_{x \in X} \text{Sum}_{k:\, q(x) != k}\, p(x|k) = 1/K * \text{Sum}_{x \in X}\, (1 - p(x|q(x)))$
b) z trénovací množiny:
Máme N vzorků (x_i, k_i) v trénovací množině.
$R_{emp}(q) = 1/N * \text{Sum}_{i = 1,\ldots,N}\, W(k_i, q(x_i))$
Při 0/1 ztrátové funkci to je "počet špatně klasifikovaných" / N.

8. Formulujte úlohu Neymana-Pearsona. Definujte význam všech použitých symbolů.
V úloze Neymana-Pearsona máme dvě třídy K = {N, D}, kde N značí normální stav a D značí nebezpečný stav. Množina rozhodnutí D = K. Určíme ε_D, $0 < \varepsilon_D < 1$, což je maximální poměr chyb "ohodnocení D jako N", $\varepsilon_D(q) = \text{Sum}_{x:\, q(x) != D}\, p(x|D)$. Úloha pak vypadá:
$q^* = \text{argmin}_q \text{Sum}_{x:\, q(x) != N}\, p(x|N)$ za podmínky $\text{Sum}_{x:\, q(x) != D}\, p(x|D) <= \varepsilon_D$,
kde q: X -> D je rozhodovací funkce, X je množina všech možných pozorování.
Obecně hledáme práh μ pro $r(x) = p(x|N)/p(x|D)$, podle kterého pak rozhodujeme:
q(x) = N iff $r(x) > \mu$, q(x) = D iff $r(x) <= \mu$

U spojitých měření se místo Sum používá Integrál.

9. Formulujte minimaxní úlohu. Definujte význam všech použitých symbolů.
V úloze minmax minimalizujeme maximální chybu ϵ_k na třídách k z K = {K_1, …, K_K}. Úloha pak vypadá:
$q^* = \text{argmin}_q \max_k \epsilon_k(q) = \text{argmin}_q \max_k \text{Sum}_{x:\, q(x) != k}\, p(x|k)$,
kde q: X -> D je rozhodovací funkce, X je množina všech možných pozorování.
Minmax nezáleží na apriorních pravděpodobnostech p(k), odpovídá Bayesovskému klasifikátoru při neznámých p(k).
U spojitých měření se místo Sum používá Integrál.

10. Formulujte Waldovu úlohu. Definujte význam všech použitých symbolů.
Waldova úloha umožňuje stanovit horní mez chyb dvou tříd tím, že zavádí možnost “nevím”. Máme K = {1, 2}, D = {1, 2, “nevím”}. Zavedeme horní meze chyb na obou třídách $0 < \epsilon < 1$.
Úloha pak zní:
$q^* = \text{argmin}_q \max_k \kappa_k$ za podmínky $\epsilon_1 <= \epsilon$, $\epsilon_2 <= \epsilon$,
kde $\kappa_1 = \text{Sum}_{x:\, q(x) = \text{“nevím”}}\, p(x|1)$ a $\epsilon_1 = \text{Sum}_{x:\, q(x) = 2}\, p(x|1)$ a podobně pro κ_2 a ϵ_2.
Obecně hledáme prahy μ_u, μ_l pro r(x) = p(x|N)/p(x|D), podle kterých pak rozhodujeme:
q(x) = N iff $r(x) > \mu_u$, q(x) = D iff $r(x) < \mu_l$, q(x) = “nevím” iff $\mu_l <= r(x) <= \mu_u$
U spojitých měření se místo Sum používá Integrál.

11. Jaká je optimální strategie pro úlohu Neymana-Pearsona?
Nevím, jestli chtějí zas vzoreček:
$q^* = \text{argmin}_q (\epsilon_N = \text{Sum}_{x:\, q(x) != N}\, p(x|N))$ za podmínky $\epsilon_D = \text{Sum}_{x:\, q(x) != D}\, p(x|D) <= \epsilon'_D$
Případně slovně 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, případně jde dosáhnout randomizací výsledků. U spojitých pozorování lze dosáhnout vždy.

12. Jaká je optimální strategie pro minimaxní úlohu?
Zase buď vzoreček:
$q^* = \text{argmin}_q \max_k (\epsilon_k(q) = \text{Sum}_{x:\, q(x) != k}\, p(x|k))$
Nebo myšlenkovej pochod (https://www.youtube.com/watch?v=r37Z209M9fM :)):
Znovu obecně platí, že růst ϵ_k jedné třídy vede ke klesání (maximálně stagnaci) ϵ_k ostatních tříd. Dokud je ϵ_k nějaké třídy ostře vyšší než ostatní, snižujeme tuto chybu. Proto v optimálním řešení vždy platí (pro spojitý případ), že třídy s nejhorší chybou jsou dvě, $\epsilon_A = \epsilon_B$.
Znovu k dosažení tohoto optima u diskrétní X je někdy třeba randomizace výsledků, u spojité X lze vždy přímo.

13. Jaké je řešení Waldovy úlohy?
Zase buď vzoreček:
$q^* = \text{argmin}_q \max_k \kappa_k$ za podmínky $\epsilon_1 <= \epsilon$, $\epsilon_2 <= \epsilon$
Nebo myšlenka, že optimum nastává, když $\epsilon_1 = \epsilon = \epsilon_2$. U Walda si nejsem tolik jistý, ale přijde mi to taky tak.

14. Ukažte, Jak je závislá chyba strategie v bayesovské úloze klasifikace do dvou tříd na změně apriorních pravděpodobností?

Pro strategii q a dvě třídy K = {1, 2} máme rizika na třídách ϵ_1, ϵ_2. Pokud $\epsilon_1 = \epsilon_2$, tak se R(q) nemění se změnou p(k). Pokud $\epsilon_1 > \epsilon_2$, tak R(q) roste s rostoucí p(1) a klesá s klesající p(1).

15. Jaké znáte principy odhadu parametrů?
Odhad maximální věrohodnosti - Maximum likelihood estimate - MLE
Maximální aposteriorní pravděpodobnost - Maximum a posteriori probability - MAP
Bayesovské odvození - Bayesian inference - BI

16. Formulujte úlohu odhadu parametrů podle principu maximální věrohodnosti (maximum likelihood, ML).
Máme trénovací množinu $T = \{(x_i, k_i)\}_{1 <= i <= N}$ a známe (předpokládaný) vzorec pravděpodobnostního rozdělení s parametry Θ. Úloha zní:
$\Theta^* = \text{argmax}_\Theta\ p(T|\Theta) = \text{argmax}_\Theta\ \text{Prod}_{(x,k)\ z\ T}\ p((x, k)|\Theta)$.
Pro více tříd s různými rozděleními v praxi často platí, že parametry rozdělení jednotlivých tříd jsou nezávislé a proto lze řešit toto hledání maxima na každé třídě zvlášť. Zároveň často úlohu zjednoduší přechod k log-likelihood ln(p(T|Θ)).

17. Jak získáte odhad hustoty pravděpodobnosti metodou Parzenových oken?
Máme množinu T s N trénovacími body x z R^D. Do každého bodu x z těchto bodů vložíme jádrovou funkci $J_x(y)$: R^D -> R. Integrál před všechny J_x musí sčítat do 1, proto pro jeden platí: Integral $J_x(y)$ dy = 1/N. Hustota pravděpodobnosti v bodě y pak je $\text{Sum}_{x\ z\ T}\ J_x(y)$. Jako jádrové funkce se používají například vícedimenzionální normální rozdělení, nebo uniformní rozdělení na hyperkrychlích nebo hyperkoulích.
Kernel má parametr h = "šířka", u normálního rozdělení je to rozptyl, u hyperkrychlí délka strany a u hyperkoulí poloměr. Musí se optimálně zvolit, např. crossvalidací pomocí log-likelihood (jako v domácím úkolu).

18. Definujte strategii rozpoznávání podle k nejbližších sousedů (k nearest neighbour rule, k-NN)
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.

19. Uveďte alespoň 5 vlastností klasifikátoru dle nejbližšího souseda (1-NN)
- implementačně jednoduchý při naivním řešení se složitostí N*D, pro zrychlení je potřeba složitější řešení přes k-D stromy nebo pravděpodobnostní nejbližší sousedy
- i při asymptotickém případu N -> inf se jeho chyba stále liší od Bayesovské, asymptotická chyba lze odhadnout $\epsilon_{Bayes} < \epsilon_{1\text{-}NN} < 2*\epsilon_{Bayes} - R/(R\text{-}1) * \epsilon_{Bayes}^2$, R = pocet trid
- obecně lze použít pro libovolně velké dimenze a libovolný počet trénovacích bodů, ale problémem je konstrukce vzdálenosti a případný různý význam různých dimenzí
- velká paměťová náročnost, lze snížit redukcí trénovací množiny
- sklon overfitovat - vždy klasifikujeme pouze podle třídy nejbližšího
- není třeba předpoklad o typu distribuce, ze které pochází trénovací měření

20. Jak lze urychlit klasifikátor typu 1-NN (dle nejbližšího souseda)?
Klasifikace lze urychlit použitím k-D stromů a redukcí trénovací množiny.

K-D stromy zrychlují 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.
Dále často stačí pouhý odhad NN, protože se data většinou vyskytují ve skupinách.
Redukcí trénovací množiny zas můžeme snížit N o několik řádů, třeba z tisíců na desítky.
Voronoiovy diagramy - při klasifikaci stačí zjistit, v jaké buňce bod leží (to je rychlejší než počítat znovu vzdálenosti)

21. Jaký je asymptotický odhad chyby klasifikátoru 1-NN (dle nejbližšího souseda)? Co je to asymptotický odhad chyby?
Asymptotická chyba lze odhadnout $\epsilon_{Bayes} < \epsilon_{1\text{-}NN} < 2*\epsilon_{Bayes}$, tedy neblíží se Bayesovské chybě. Asymptotický odhad chyby znamená, že určujeme chybu pro $N \rightarrow \infty$.
Příklad pro dvě třídy $p(A) = p(B) = 0{,}5$. V určitém x máme $p(x|A) = a$, $p(x|B) = b$. Řekněme, že $a > b$. Pak $\epsilon_{Bayes} = \min(a, b) = b$. Ale $\epsilon_{1\text{-}NN} = 2*a*b$, kde $0{,}5 < a <= 1$, tedy $1 < 2*a <= 2$ a odtud máme $\epsilon_{Bayes} < \epsilon_{1\text{-}NN} < 2*\epsilon_{Bayes}$.

22. Popište perceptronový algoritmus učení. Jaké má vlastnosti?
$K = \{-1, 1\}$, $X = R^D$. Nejdříve upravíme trénovací množinu $\{(x_i; k_i)\}_1^N$ na $\{x'_i = (1, x_i)*k_i)\}_1^N$ z R^{D+1}. Perceptron je lineární klasifikátor, nalezne nadrovinu určenou vektorem w, která oddělí data tak, že $w*x'_i > 0$, pokud jsou data lineárně separovatelná.
Algoritmus:

1) init $w_0 = 0$
2) nalezni nějaké x'_i, pro které $w*x'_i <= 0$
3) pokud existuje takové x'_i, polož $w_{t+1} = w_t + x'_i$ a jdi na 2)
4) pokud neexistuje, ukonči, $w = w_t$

Vlastnosti:

- funguje pouze pro lineárně separovatelná data, pro neseparovatelná nikdy neskončí
- nenachází ničím optimální dělící nadrovinu, po nalezení nějaké se zastaví
- Novikoff theorem: pro lineárně separovatelná data máme jednotkový vektor u a skalár y z R^+ tak, že $u*x'_i >= y$, D je velikost nejdelšího vektoru z dat. Pak se Perceptron zastaví v konečném počtu kroků $t^* <= D^2/y^2$
- vychází z gradientní iterační metody při ztrátové funkci $J(w_t) = \text{Sum}_i\ -w_t*x'_i$
- pomocí dimension lifting lze oddělovat i lineárně neseparovatelná data lineární funkcí po mapování do vyšší dimenze

23. Vysvětlete, jak lze na perceptronový algoritmus učení nahlížet jako na gradientní optimalizaci.
Pokud máme ztrátovou funkci $J(w) = \text{Sum}_{x': w*x' <= 0}\ -w*x'$, tak gradient $\nabla J = \text{Sum}_{x': w*x' <= 0}\ -x'$.
Při gradientní optimalizaci se pohybujeme ve směru $-\nabla J$, tedy přičtení špatně klasifikováného x' odpovídá části kroku gradientní optimalizace.

24. Pro které rozpoznávací úlohy je lineární diskriminační funkce optimálním řešením?

- lineárně separovatelná data
- normální rozdělení se stejnou kovarianční maticí a různými středními hodnotami
- nezávislé binární vstupní vektory

- multinomiální naivní Bayes (pozorování je počet výskytů jednotlivých tříd)

25. Co je to neuronová síť? Jaké znáte základní typy.
Neuronová síť je funkce f: R^D -> R^K. Je to zřetězení lineárních funkcí (maticové násobení) a nelinearit (aktivačních funkcí). Předěly mezi lineárními funkcemi se nazývají vnitřní vrstvy. Využívá Universal approximation theorem (UAT):

- Nechť f: R^D -> R je spojitá funkce na jednotkové hyperkrychli a σ: R -> R je spojitá, omezená, nekonstantní funkce. Pak pro libovolné ε > 0 existuje přirozené číslo N tak, že |f(x) - F(x)| <= ε pro všechny x z jednotkové hyperkrychle, kde F(x) = $Sum_{1 <= i <= N}$ v_i*σ(w_i*x + b_i), kde w_i je z R^D a v_i, b_i z R.

Díky tomu, že jednotlivé části neuronové sítě jsou diferencovatelné, lze pomocí iteračních metod získat optimální řešení (lze uvíznout v lokálním optimu, hodně záleží na inicializaci).
Typy:

- konvoluční NN: hledá lokální souvislosti, např. pro rozpoznávání obrázků
- rekurentní NN: výstup NN lze znovu použít jako vstup, např. pro rozpoznávání zvuku
- automatické enkodéry: mají úzkou vnitřní vrstvu a požadují vstup = výstup, nutí tedy NN uložit komprimovanou reprezentaci dat

26. Jaké vlastnosti má aktivační funkce v neuronové síti? Uveďte příklady.
Aktivační funkce v neuronové síti je nelineární funkce σ: R -> R a v 99,9% diferencovatelná (výjimka LReLU a ReLU v 0). Zajišťuje aproximační schopnost NN (viz UAT), bez nich by NN byly pouze zřetězené Perceptrony, které by vždy šly zjednodušit do jediné vrstvy. σ je spojitá, nekonstantní a omezená funkce.
Příklady aktivačních funkcí:

- logistický sigmoid: σ(x) = 1/(1 + e^{-x})
- tanh: σ(x) = (e^x - e^{-x})/(e^x + e^{-x})
- ReLU (rectified linear unit): σ(x) = max(0, x)
- LReLU (leaky ReLU): σ(x) = max(0, x) + min(0, sx), 0 <= s <= 1

27. Popište co nejpřesněji algoritmus učení dopředné neuronové sítě metodou zpětného šíření (back-propagation). Diskutujte vlastnosti.
Protože je NN pouze jedna velká diferencovatelná matematická funkce, lze použít gradientní metodu optimalizace. Pro každý parametr v NN vypočítáme derivaci výstupu NN v závislosti na něm. Pomocí chain rule jsou derivace podle parametrů dřívějších vrstev závislé na derivacích parametrů následných vrstev. Proto postupujeme odzadu. U každého parametru (nebo obecně vrstvy), vypočítáme derivaci výstupu podle parametru samotného (pro optimalizaci daného parametru) a zároveň derivaci podle vstupu vrstvy (pro propagaci do předchozích vrstev).
Vlastnosti:

- zaručuje nalezení pouze lokálního optima, velmi záleží na prvotní inicializaci parametrů
- Rychlost konvergence velmi závisí na prvotní inicializaci parametrů

28. Porovnejte vlastnosti učení přímé neuronové sítě metodou zpětného šíření (backpropagation) a učení klasifikátoru Support Vector Machine.
Učení NN i učení SVM probíhá pomocí iteračních optimalizačních metod.

Rozdíl je v tom, že funkce u SVM je konvexní (konkávní argmax f(x) se upraví na konvexní argmin -f(x)), proto má jediné lokální minimum, neboli vždy nalezneme globálně nejlepší řešení.
Naopak funkce NN konvexní (obecně) není, proto hrozí uvíznutí v lokálním minimu nebo sedlovém bodě. O to více zde záleží na inicializaci dat, která ovlivňuje konvergenci.

29. Co je to empirická ztráta (riziko)?
Empirická ztráta je chyba na trénovací množině.

30. Co je to strukturální riziko?
Strukturální riziko je horní odhad rozdílu empirického a testovacího rizika (R_{test} <= R_{emp} + R_{strukt}). Vychází ze třídy složitosti (VC dimenze) rozhodovací funkce a počtu trénovacích dat. Je více možností, jak odhadnout horní hranici testovacího rizika (různé vzorce, někdy speciálně pro určitý typ klasifikátoru) a od toho se pak odvíjí strukturální riziko - to empirické vždy známe z naší trénovací sady.

31. Co je to Vapnik-Červoněnkisova dimenze?
VC dimenze popisuje složitost rozhodovacích funkcí. Funkce s vyšší VC dimenzí mají větší tendenci ovefitovat.
Význam VC dimenze je: maximální počet bodů (N), které můžeme v prostoru nějak rozmístit tak, aby každé jejich binární ohodnocení (celkově 2^N možností) bylo možné danou třídou rozhodovacích funkcí správně odlišit.
U lineární funkcí je VC dimenze D+1, kde D je dimenze prostoru.
Pomocí VC dimenze a chyby na trénovací sadě lze s danou pravděpodobností odhadnout horní hranici chyby na testovací sadě

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

Pro některé typy klasifikátorů (např lineární) VC dimenzi předem známe, takže se to spočítá snadno a hodí se to.

32. Jak postupujeme při návrhu klasifikátoru podle principu minimalizace strukturálního rizika?
Začínáme u tříd klasifikátorů s nižší VC dimenzí a přechází ke složitějším, pokud jednodušší nestačí. Tzn. u každé třídy nalezneme nejlepší rozhodovací funkci a zjistíme její klasifikační schopnosti. Nejsou-li dostačující, zkusíme složitější třídu.

33. Jaká je VC dimenze klasifikátoru 1-NN?
Nekonečno (i nekonečně mnoho bodů lze rozlišit, v testování daných bodů 1-NN nalezne v každém bodě nejbližšího souseda z trénovacích bodů (což jsou ty samé), což je bod sám, a ohodnotí ho na jeho třídu, která je tedy vždy správná).

34. Jaká je VC dimenze orientované nadroviny?
Orientovaná nadrovina je lineární klasifikátor, takže VC dimenze je D+1, kde D je dimenze prostoru.

35. Co je to Support Vector Machine?
SVM je lineární klasifikátor, který oddělí lineárně separovatelná data nadrovinou, která je nejvzdálenější od nejbližších bodů. Narozdíl od Perceptronu tedy hledá v určitém smyslu nejoptimálnější rozdělení.
Zároveň se dá (narozdíl od Perceptronu) zobecnit i pro lineárně neseparovatelná data přidáním ztráty za špatnou klasifikaci bodu. Další možností je dimension lifting a použití kernel tricku, díky kterému ani nemusíme znát přímo mapovací funkci.

36. Jaká minimalizační úloha se řeší při učení SVM?
V SVM získáme maximalizaci konkávní funkce:
$\text{argmax}_\alpha \text{Sum}_i \alpha_i - ½*\text{Sum}_i \text{Sum}_j \alpha_i\alpha_j y_i y_j x_i^T x_j$ za podmínky $0 <= \alpha <= C$, $\text{Sum}_i \alpha_i y_i = 0$
Tu převedeme na minimalizaci konvexní funkce a použijeme QP optimalizaci:
$\text{argmin}_\alpha ½*\text{Sum}_i \text{Sum}_j \alpha_i\alpha_j y_i y_j x_i^T x_j - \text{Sum}_i \alpha_i$ za podmínky $0 <= \alpha <= C$, $\text{Sum}_i \alpha_i y_i = 0$

37. Proč algoritmus učení SVM maximalizuje tzv. ”margin”, t.j. minimální vzdálenost bodu trénovací množiny od rozdělující nadroviny?
Proč jako z jakého důvod:
Protože větší margin znamená snížení strukturálního rizika a tedy lepší generalizaci na test data.
Proč jako jakým způsobem:
Protože je úloha tak formulovaná a QP optimalizací se najde globální optimum.

38. Jak řeší SVM situaci, kdy trénovací množina není lineárně separabilní?
Úloha se rozšíří o ztrátu C za špatně klasifikovaná data. Ztráta se navíc násobí vzdáleností od správného poloprostoru pro danou třídu, je to hezky vidět v primární úloze:
$\text{argmin}_{w, b, \xi} ½ ||w||^2 + C*\text{Sum}_{1 <= i <= N} \xi_i$ za podmínky $y_i(w^T x_i + b) >= 1 - \xi_i$ pro $\forall i$
Častěji se ale používá duální úloha, kde se to projeví pouze omezením alfy:
$\text{argmax}_\alpha \text{Sum}_{1 <= i <= N} \alpha_i - ½*\text{Sum}_{1 <= i, j <= N} \alpha_i\alpha_j y_i y_j x_i^T x_j$ z.p. $\text{Sum}_{1 <= i <= N} \alpha_i y_i = 0$, $0 <= \alpha <= C$.
Toto by bylo pro data, která jsou “téměř” lineárně separabilní (např. kvůli šumu) ale pár bodů spadne na špatnou stranu rozhodovací nadroviny.
Pro data “hodně” neseparabilní by se využilo SVM s kernel trikem. Výhoda je, že výpočet bude pořád cca stejně náročný, protože se tam pořád objevuje pouze výpočet skalárního součinu - není třeba přímo počítat dimension lifting. Případně jde obojí dohromady - softmargin s kernelem, pak by rozhodovací hranice mohla být např. kružnice a tolerovaly by se některé body špatně klasifikované uvnitř / vně kružnice.

39. Příznakový prostor je R^2. Jakým způsobem byste hledali kvadratickou rozdělují funkci v tomto prostoru? Dokážete postup zobecnit pro R^3 či R^n?
Kvadratická funkce je má nejvýše členy řádu 2, takže bychom v R^2 hledali koeficienty pro členy $(1, x, y, xy, x^2, y^2)$. Pro R^3 by to bylo předpokládám $(1, x, y, z, xy, xz, yz, x^2, y^2, z^2)$. Pro obecný R^n pak $(1, x_i, x_ix_j)$ pro $\forall 1 <= i, j <= n$.

40. Příznakový prostor je R^2. Jakým způsobem byste hledali eliptickou rozdělují funkci v tomto prostoru? Uvažujte pouze elipsy s osami rovnoběžnými s osami souřadnicového systému.
Předpis elipsy rovnoběžné s osami je $(x - c_x)^2/a + (y - c_y)^2/b = 1$, kde c_x, c_y z R a a, b > 0.
Máme trénovací množinu $T = \{(x_i, y_i), k_i\}_{1 <= i <= N}$.
Buď bychom asi mohli mít soustavu rovnic:

$k_i((x_i - c_x)^2/a + (y_i - c_y)^2/b) > 1$ pro $\forall 1 <= i <= N$
Nebo by asi šla formulovat maximalizace margin:
$\text{argmax}_{a, b, cx, cy} \min_i k_i((x_i - c_x)^2/a + (y_i - c_y)^2/b)$ z.p. jako v soustavě výše

41. Co je to jádrová funkce? Popište učení kernel SVM. Jaké jádrové funkce se běžně používají při učení SVM.
Jádrová funkce K(x, y) popisuje skalární součin mapovaných vektorů $\varphi(x)^T\varphi(y)$ bez znalosti mapování. Umožňuje tak použití složitých mapování (potenciálně až nekonečně-dimenzionálních - např. Gaussian kernel: $K(x_i,x_j) = \exp(-||x_i-x_j||^2 / 2\sigma^2)$) v dimension lifting v SVM pro lineární oddělení lineárně neseparovatelných dat.
Učení SVM probíhá stejně jako normálně, pouze funkce přijme K(x, y):
$\text{argmax}_\alpha \text{Sum}_{1 <= i <= N} \alpha_i - ½*\text{Sum}_{1 <= i, j <= N} \alpha_i\alpha_i y_i y_j K(x_i, x_j)$ z.p. $\text{Sum}_{1 <= i <= N} \alpha_i y_i = 0$, $0 <= \alpha <= C$
Běžně se používají lineární: x^Ty, polynomiální: $(1 + x^Ty)^p$, exponenciální: $\exp(-||x - y||^2/(2*\sigma^2))$, dvouvrstvý tanh perceptron: $\tanh(ax^Ty + b)$.
Změní se klasifikace:

1) Známe φ:
   $w = \text{Sum}_{i \text{ je sv}} \alpha_i y_i \varphi(x_i)$, $b = y_{sv} - w^Tx_{sv}$ (pro nějaký support vektor x_{sv}), $q(x) = w^T\varphi(x) + b$
2) Neznáme φ:
   $b = y_{sv} - \text{Sum}_{i \text{ je sv}} \alpha_i y_i K(x_i, x_{sv})$, $q(x) = \text{Sum}_{i \text{ je sv}} \alpha_i y_i K(x_i, x) + b$

42. Jaké rozdílné vlastnosti mají perceptronové učení a učení SVM.
Perceptronové učení se zastaví po nalezení první nějaké oddělující nadroviny. SVM nalezne optimální nadroviny (z hlediska velikosti margin). Oba algoritmy jsou iterativní, u perceptronu máme horní mez počtu iterací (Novikoff theorem, pokud jsou data separabilní a při inicializaci na $w^0 = 0$) $t <= D^2/\gamma^2$, kde D je $\max_i ||x_i||$ a γ: $u*x_i >= \gamma$, $||u|| = 1$.
U SVM konvergence záleží na inicializaci α v QP.
Perceptron používá přímo body x z R^D, SVM pouze jejich skalární součiny x^Ty z R.

43. Popište algoritmus učení metodou Adaboost. Jakou minimalizační úlohu Adaboost řeší?
AdaBoost vybírá z množiny B slabých klasifikátorů T slabých klasifikátorů $h_t(x)$, ze kterých zkonstruuje silný klasifikátor $H_T(x) = \text{sign}(\text{Sum}_{1 <= t <= T} \alpha_t h_t(x))$.
Algoritmus:

1) Inicializuj t = 1, váhy trénovacích bodů na $D_{t=1}(i) = 1/N$
2) Nalezni slabý klasifikátor $h_t(x) = \text{argmin}_{h \text{ z } B} \epsilon_t(h)$, kde
   $\epsilon_t(h) = \text{Sum}_{i: h(xi) != yi} D_t(i)$, tzn. klasifikátor s nejmenší váženou chybou
3) Pokud je $\epsilon_t >= ½$, zastav (další klasifikátory už nezlepšují klasifikaci)
4) Zadej váhu nalezeného klasifikátoru $\alpha_t = ½*\ln((1 - \epsilon_t)/\epsilon_t)$
5) Updatuj váhy bodů: $D_{t+1}(i) = (D_t(i)*\exp(-\alpha_t y_i h(x_i))/Z_t)$, kde $Z_t = 2\sqrt{(\epsilon_t(1 - \epsilon_t))}$ je normalizační konstanta, $D_t(i)$ je tedy rozdělení, sčítají do 1
6) pokud t = T, ukonči, jinak jdi na 2) s t := t + 1

AdaBoost minimalizuje horní odhad chyby $1/N*\text{Sum}_i [[H_T(x_i) != y_i]] <= \text{Prod}_{1 <= t <= T} Z_t$, protože:

1) $[[H_T(x_i) != y_i]]$ iff $[[y_i*H_T(x_i)< 0]]$ iff $[[y_i*\text{Sum}_{1 <= t <= T} \alpha_t h_t(x) < 0]]$
2) $[[y_i*\text{Sum}_{1 <= t <= T} \alpha_t h_t(x) < 0]] <= \exp(-y_i*\text{Sum}_{1 <= t <= T} \alpha_t h_t(x))$
3) $D_{T+1}(i) = D_T(i)/(N * \text{Prod}_{1 <= t <= T} Z_t) = \exp(-y_i*\text{Sum}_{1 <= t <= T} \alpha_t h_t(x))/(N * \text{Prod}_{1 <= t <= T} Z_t)$
   $D_{T+1}(i) * \text{Prod}_{1 <= t <= T} Z_t = \exp(-y_i*\text{Sum}_{1 <= t <= T} \alpha_t h_t(x))/N$
   $\text{Sum}_i D_{T+1}(i) * \text{Prod}_{1 <= t <= T} Z_t = \text{Sum}_i \exp(-y_i*\text{Sum}_{1 <= t <= T} \alpha_t h_t(x))/N$
   $1 * \text{Prod}_{1 <= t <= T} Z_t = \text{Sum}_i \exp(-y_i*\text{Sum}_{1 <= t <= T} \alpha_t h_t(x))/N$ (...protože $D_{T+1}(i)$ je rozdělení)

4) $1/N * Sum_i [[H_T(x_i) != y_i]] <= Sum_i \exp(-y_i * Sum_{1 <= t <= T} \alpha_t h_t(x))/N = Prod_{1 <= t <= T} Z_t$

44. Popište algoritmus k průměrů (k-means).

Máme trénovací data bez jejich tříd $T = \{x_i\}_{i=1}^N$. Chceme je rozdělit do K shluků tak, aby suma vzdáleností bodů od center shluků byla minimální:

$(c_1, \ldots, c_K)^* = \text{argmin}_{(c1, \ldots, cK)} Sum_{i=1}^N \min_k ||x_i - c_k||^2$

Algoritmus:

1) init centra c_k (náhodně nebo pomocí K++ means)
2) každý bod přiřaď k nejbližšímu centru: $T_k = \{x: ||x - c_k||^2 <= ||x - c_j||^2 \ \forall j\}$
3) update center do těžiště shluků:
   $c_k = 1/|T_k| * Sum_{x z Tk} x$ iff $|T_k| > 0$
   $c_k = \text{reinit}$ iff $|T_k| = 0$
4) pokud se žádný shluk T_k nezměnil, ukonči, jinak jdi na 2)

Algoritmus lze zobecnit pro jakoukoliv funkci vzdálenosti d(x, y), změní se pouze 2. a 3. krok. K++ means inicializace zvyšuje pravděpodobnost nalezení globálního minima. Nicméně protože jde o iterační algoritmus, stále můžeme uvíznout v lokálním extrému (lze řešit více běhy a výběrem nejlepšího řešení).

45. EM algoritmus.

EM algoritmus je iterační metoda pro odhad parametrů, když odhad nelze vyřešit analyticky (například pomocí MLE).

Máme trénovací data $T = \{x_i\}_{i=1}^N$, kde měření nemusí být kompletní. Chceme nalézt maximálně věrohodný odhad parametrů θ, které popisují pravděpodobnostní rozdělení.

Algoritmus:

1) init $\theta^{t=0}$ (náhodně)
2) vypočítat příslušnost dat x_i k jednotlivým třídám q_i:
   $q^{t+1} = \text{argmax}_q L(q \mid \theta^t)$
   Poznámka: Příslušnost ke třídě lze vyjádřit jako pravděpodobnost. K-means dá každému x natvrdo label, kdežto EM mu přiřadí jakoby čísla <0,1> tzn. s jakou pravděpodobností náleží do dané třídy. Váhy jsou tedy vlastně ty pravděpodobnosti. x, které má vyšší pravděpodobnost, že do dané třídy patří, pak více ovlivní, jak se v současné iteraci změní odhad parametrů pro tuhle třídu.
3) přepočíst parametry podle vah q_i:
   $\theta^{t+1} = \text{argmax}_\theta L(\theta \mid q^{t+1})$
4) jít na krok 2)

Zkouška 2020

1. (5 points) Describe the K-means algorithm (1pt). Prove that K-means converges to a local minimum (2pts). Describe the K-means++ algorithm (1pt). Give an example when employing K-means++ leads, on average, to better solution than K-means (1pt).

2. (5 points) Formulate the logistic regression problem (1 point). Derive the gradient descent learning method for this task (2.5 points).
Given the training set $T = \{(-1,-1),(-2,-1),(0,1),(2,1)\}$, starting point $w = (0,0)$, and an initial step size of 1, do the first iteration of gradient descent. (1.5 points).

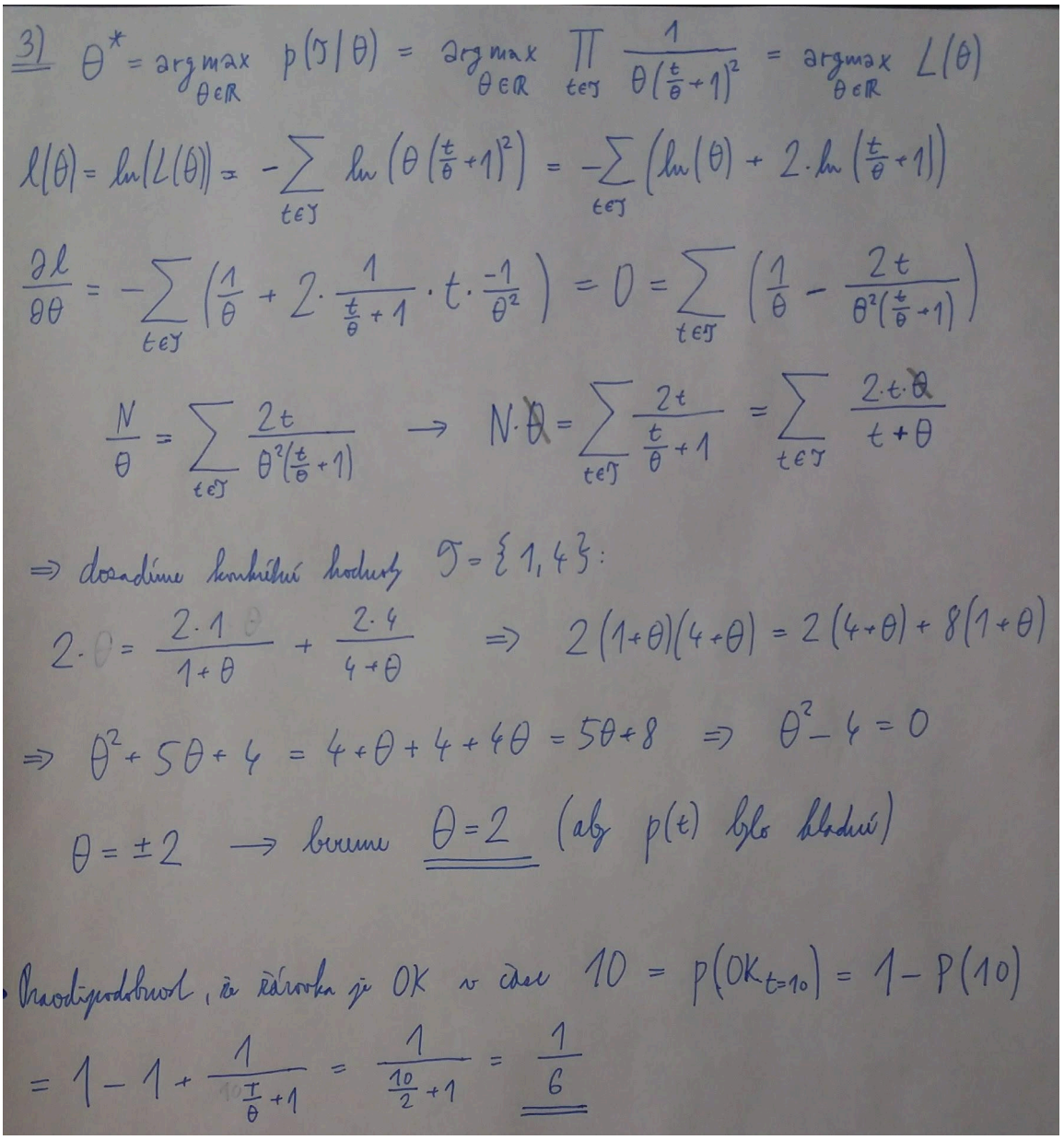
3. (5 points) Parameter Estimation. The reliability of light bulbs is defined by the probability density function
$$p(t) = \frac{1}{\theta\left(\frac{t}{\theta}+1\right)^2}, \quad t \in [0,\infty],$$
where $p(t)\delta$ is the probability that a bulb fails in a short time interval $(t, t+\delta)$. That is, the probability $P(T)$ that the bulb fails in interval $(0,T)$, is
$P(T) = \int_0^T p(t)dt = 1 - \frac{1}{(\frac{T}{\theta}+1)}$. What is the maximum likelihood estimate of the reliability parameter θ if in an experiment with two bulbs the following lifetimes have been observed:
$t_1 = 1.0, t_2 = 4.0$? (4pts)
For this setting, what is the probability that a light bulb is still OK at time 10? (1pt)

4. (5 points) Decision tree classifiers.
   (a) (2 points) Describe the simplest top-down tree-building algorithm that selects the decision rule at a node by maximization of information gain.
   (b) (3 points) Discuss advantages and disadvantages of decision trees.

1)
K-means: je to tu někde
Konvergence: taky tu je (v kroku 2 si nemůže žádný bod pohoršit, v kroku 3 se nemůže zhoršit globální součet, musíme zajistit konzistenci při rovnosti vzdáleností, celkově je N^K možných stavů = konečný počet)
K-means++: taky tu už je
Příklad: ten typický obdélník

2)
Viz **Zkouška 2016 1)**

3)



4)

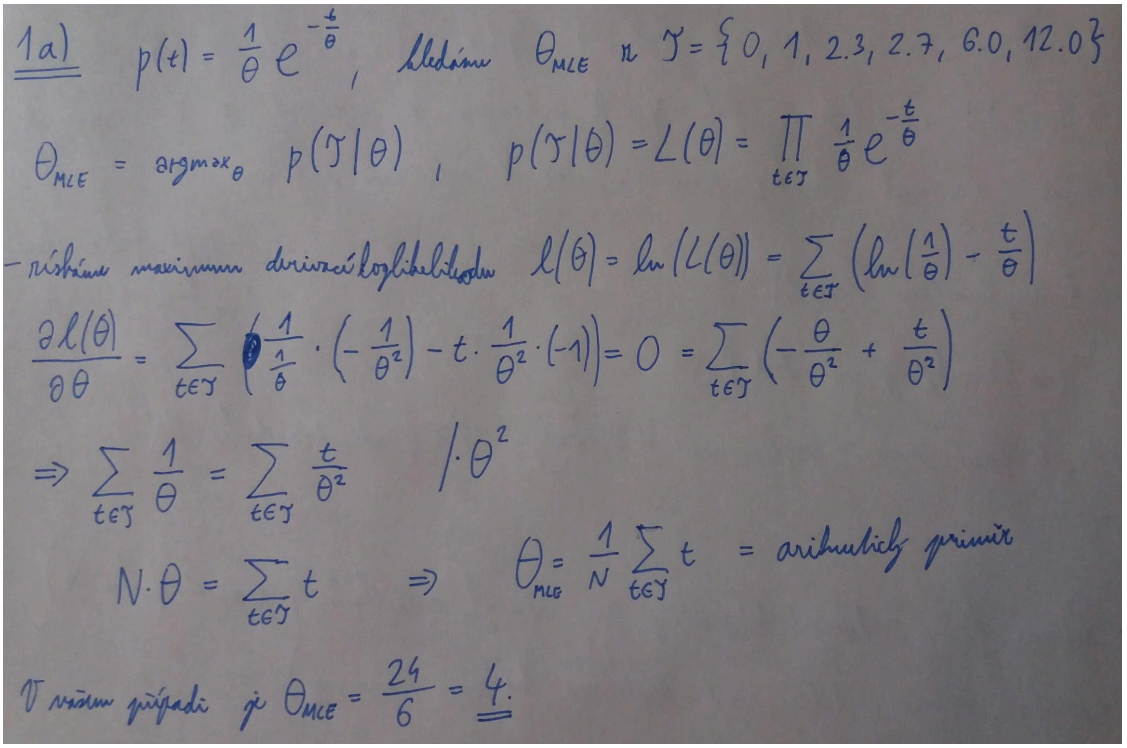
Viz **Zkouška 2017 1)**

Zkouška 2019

(http://cmp.felk.cvut.cz/cmp/courses/recognition/Exam-questions/test_2019-eng.pdf)

1) ?????

b) a c) nevím



b)

Druhá část b) pomocí EM: **?????**

1) init θ
2) $q_i = 1/\theta * e^{-ti/\theta}$ pro i = 1, 2, 3, 4, $q_j = e^{-tmax/\theta}$ pro j = 5, 6
3) $\theta = (q_1*t_1 + q_2*t_2 + q_3*t_3 + q_4*t_4 + q_5*t_{max} + q_6*t_{max})/(Sum_{i = 1..6}\ q_i)$

c)

$p(fail) = Int_0^{tmax}\ 1/\theta * e^{-t/\theta}\ dt = 1 - e^{-tmax/\theta}$

$p(good) = Int_{tmax}^{inf}\ 1/\theta * e^{-t/\theta}\ dt = e^{-tmax/\theta}$

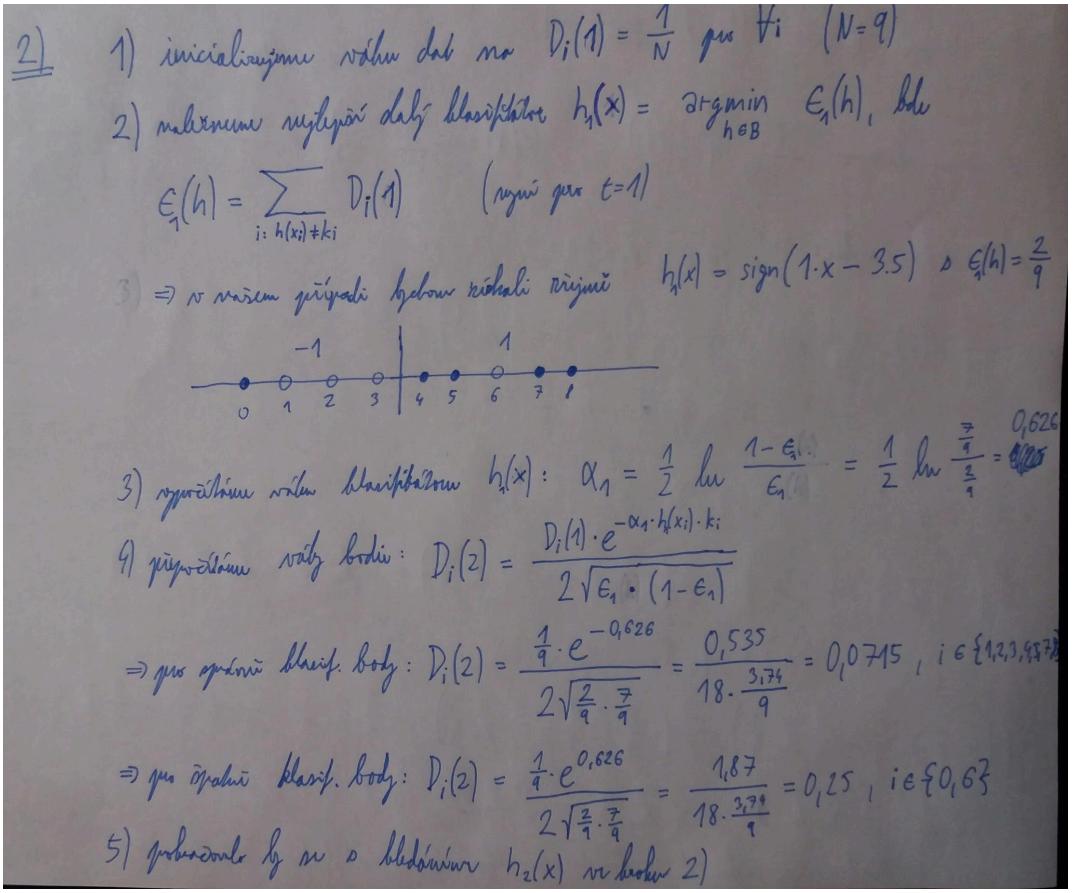
$\theta^* = argmax_\theta\ (e^{-t/\theta})^G * (1 - e^{-t/\theta})^F$ - substituujeme $\pi = e^{-t/\theta}$

$\pi^* = argmax_\theta\ \pi^G * (1 - \pi)^F = argmax_\theta\ G*ln(\pi) + F*ln(1 - \pi)$

$\partial l/\partial \pi = G/\pi - F/(1 - \pi)$, odtud $\pi = G/(F+G) = e^{-t/\theta}$

$\theta^* = t_{max}/ln((F + G)/G)$

2)



3)

3) K-NN: Máme množinu testovacích dat $\mathcal{T} = \{(x_i, k_i)\}_{i=1}^{N}$, kde $x_i \in \mathbb{R}^D$, $k_i \in \{1, \dots, R\}$. Pro testovaný bod $y \in \mathbb{R}^D$ nalezneme K nejbližších x_i z $\mathcal{T}$ podle d a ohodnotíme ho jako to k_i, které má mezi K nejbližšími většinu.

d ... funkce vzdálenosti
$d: \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$

⊕ jednoduchá implementace (vaření)

⊕ funguje pro libovolný počet tříd R

⊕ možnosti rozšíření naivní implementace: k-D stromy, výběr pravděpodobnostního NN

⊖ potřeba definice funkce vzdálenosti d (problémy při různých "měřítkách" / významech prvků x)

⊖ vysoká paměťová náročnost (lze řešit redukcí trénovací množiny)

4)

4) Maximalizujeme $IG(a) = H(P) - \sum_{i \in \{0,1\}} \frac{|P_i|}{|P|} H(P_i)$, kde a je atribut podle kterého dělíme na množiny, kdy a je $i \in \{0,1\}$

$\Rightarrow$ minimalizujeme $\sum_{i \in \{0,1\}} \frac{|P_i|}{|P|} H(P_i)$, $H(X) = -\sum_{k \in \{0,1\}} \frac{|X^k|}{|X|} \ln \frac{|X^k|}{|X|}$ (X^k je podmnožina bodů třídy k v X)

- a = mokré kopyto: $P_0 = \{4,6\}$, $P_1 = \{1,2,3,5,7,8\}$:

$$\frac{|P_0|}{|P|} H(P_0) + \frac{|P_1|}{|P|} H(P_1) = \frac{2}{8}\left(-\frac{1}{2}\ln\frac{1}{2} - \frac{1}{2}\ln\frac{1}{2}\right) + \frac{6}{8}\left(-\frac{3}{6}\ln\frac{3}{6} - \frac{3}{6}\ln\frac{3}{6}\right) = \frac{1}{4}\ln\frac{1}{2} - \frac{3}{4}\ln\frac{1}{2}$$
$$= -\ln\frac{1}{2} = \ln 2 \doteq 0{,}69$$

- a = comby: $P_0 = \{2,5,6,8\}$, $P_1 = \{1,3,4,7\}$:

$$\frac{4}{8}\left(-\frac{2}{4}\ln\frac{2}{4} - \frac{2}{4}\ln\frac{2}{4}\right) + \frac{4}{8}\left(-\frac{2}{4}\ln\frac{2}{4} - \frac{2}{4}\ln\frac{2}{4}\right) = -\ln\frac{1}{2} = \ln 2 \doteq 0{,}69$$

- a = vousatka: $P_0 = \{1,2,4,6\}$, $P_1 = \{3,5,7,8\}$:

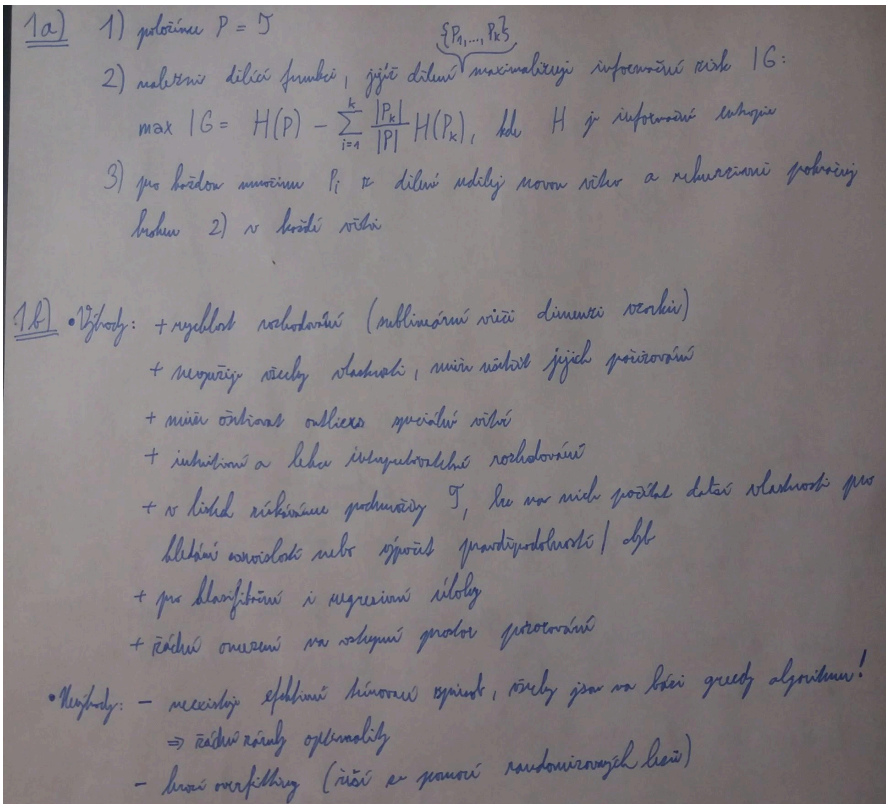
$$\frac{4}{8}\left(-\frac{3}{4}\ln\frac{3}{4} - \frac{1}{4}\ln\frac{1}{4}\right) + \frac{4}{8}\left(-\frac{1}{4}\ln\frac{1}{4} - \frac{3}{4}\ln\frac{3}{4}\right) = -\frac{1}{4}\ln\frac{1}{4} - \frac{3}{4}\ln\frac{3}{4} = \frac{1}{4}\ln 4 + \frac{3}{4}\ln\frac{4}{3}$$
$$\frac{1}{4}\ln 4 + \frac{3}{4}\ln 4 - \frac{3}{4}\ln 3 = 2\ln 2 - \frac{3}{4}\ln 3 \doteq 0{,}56$$

Informační zisk je maximalizován dělením podle atributu "vousatka" na $\{1,2,4,6\}$ a $\{3,5,7,8\}$.

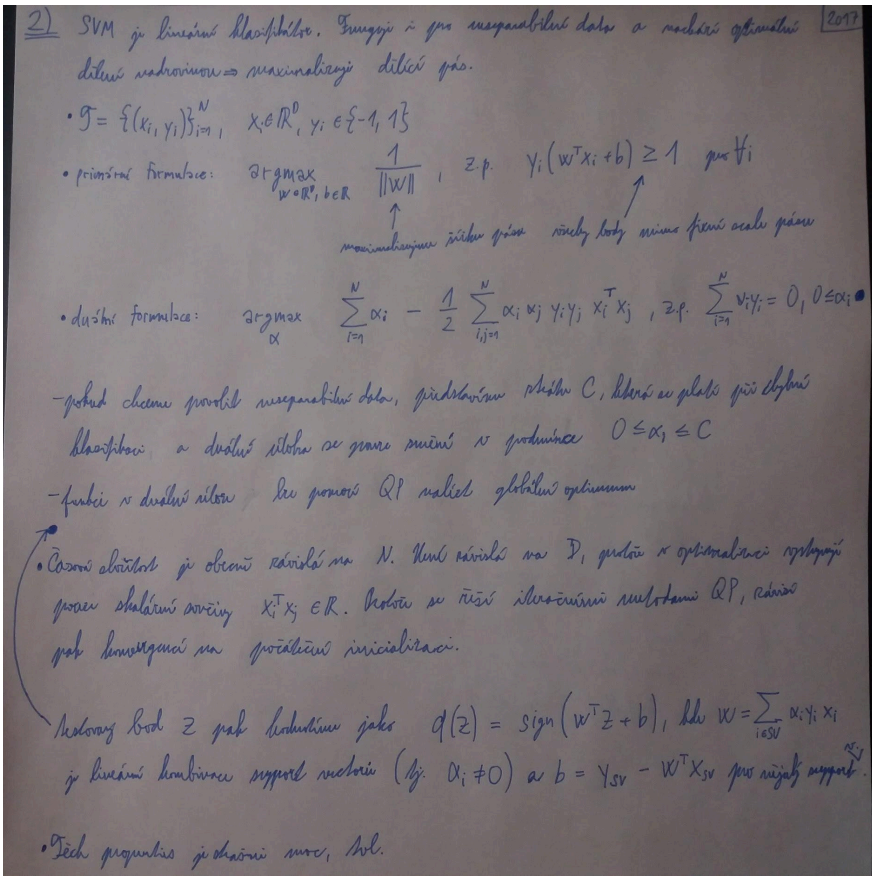
Zkouška 2017

(http://cmp.felk.cvut.cz/cmp/courses/recognition/Exam-questions/test_2017.01.16.pdf)

1)



2)



3) a), b)

3) a)

Máme data $\{x_i\}_{i=1}^N$, $x_i \in \mathbb{R}^D$, která chceme rozdělit do K shluků.

Hledáme K center shluků tak, aby $\{c_k\}^* = \underset{\{c_k\}}{\operatorname{argmin}} \sum_{i=1}^{N} \min_{c_k} \|c_k - x_i\|^2$

Algoritmus: 1) init $\{c_k\}$

2) pro každý bod nalézt nejbližší centrum
⇒ vytvořit shluky $\mathcal{T}_k = \{x_i : \|x_i - c_k\|^2 \leq \|x_i - c_j\|^2 \text{ pro } \forall j\}$

3) posunout centra shluků do těžišť: $c_k = \begin{cases} \frac{1}{|\mathcal{T}_k|} \sum_{x \in \mathcal{T}_k} x & \text{iff } |\mathcal{T}_k| \neq 0 \\ \text{nenech} & \text{iff } |\mathcal{T}_k| = 0 \end{cases}$

4) pokud se shluky nepohnuly, ukonči, jinak jdi na 2)

b) mění se pouze 2. a 3. bod: 2) $\mathcal{T}_k = \{x_i : \|x_1 - c_k\|_1 \leq \|x_i - c_j\|_1 \text{ pro } \forall j\}$

3) posunout centra shluků do mediánů

3) c), d), e)

c) Také se jim změní 2 a 3. bod. 3. bod bude záviset o sobě optimalizační úlohu, použije se úlohou Weiszfeldův algoritmus.

d) $\cdot I \times N \times K \times M$ ← každý případ je jiná nálože v závislosti na použité funkci rozdílnosti

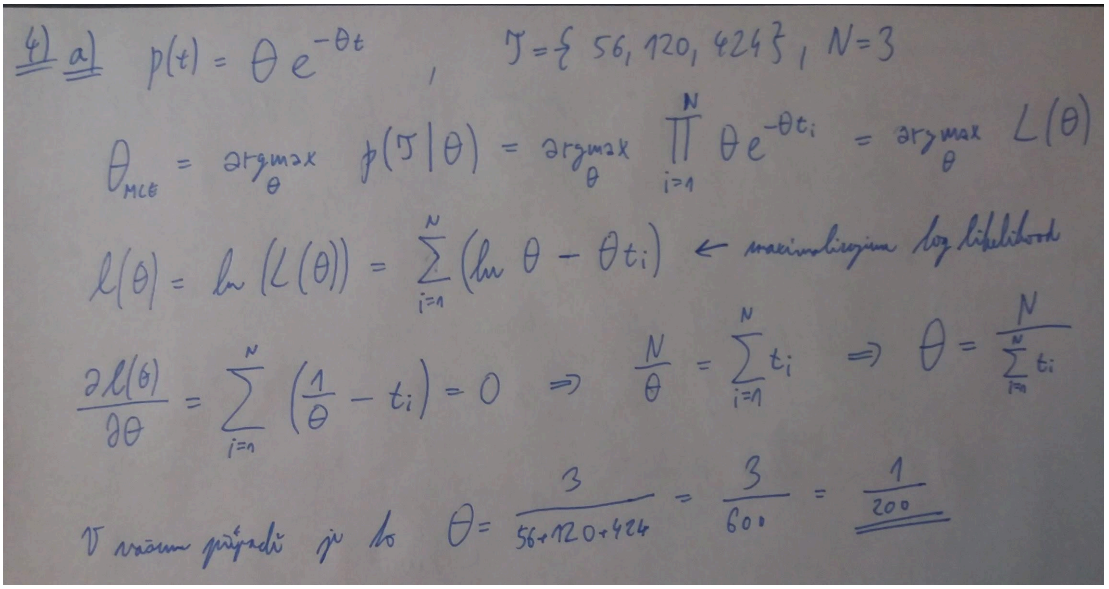
I počet úloh; N v každé úloze provádíme; K každý hledáme shluk s každým centrem

• algoritmus výsledek musí uložit v konečném čase (I je konečné číslo)

e) • init bod 1) se rozšíří: a) vybereme náhodně c_1 (z rovnoměrného pravděpodobnostního)
b) pro každý bod x_i vypočítej $d_i = \min_{c_k} \|c_k - x_i\|$
c) vybereme c_t náhodně z x_i podle $p(x_i) = \frac{d_i^2}{\sum_{i=1}^{N} d_i^2}$
d) pokud $t = K$ ukončí, jinak pokračuj [illegible] bodem b s $t+1$

4) ?????

V b) zase můj odvěký nepřítel: žárovky co přežily test...



4) a) $p(t) = \theta e^{-\theta t}$, $T = \{56, 120, 424\}$, $N = 3$

$$\theta_{MLE} = \operatorname{argmax}_\theta p(T|\theta) = \operatorname{argmax}_\theta \prod_{i=1}^{N} \theta e^{-\theta t_i} = \operatorname{argmax}_\theta L(\theta)$$

$$l(\theta) = \ln(L(\theta)) = \sum_{i=1}^{N} (\ln\theta - \theta t_i) \leftarrow \text{maximalizujem log likelihood}$$

$$\frac{\partial l(\theta)}{\partial \theta} = \sum_{i=1}^{N} \left(\frac{1}{\theta} - t_i\right) = 0 \Rightarrow \frac{N}{\theta} = \sum_{i=1}^{N} t_i \Rightarrow \theta = \frac{N}{\sum_{i=1}^{N} t_i}$$

V našem případě je to $\theta = \frac{3}{56+120+424} = \frac{3}{600} = \frac{1}{200}$

Zkouška 2016

(http://cmp.felk.cvut.cz/cmp/courses/recognition/Exam-questions/test_2016.01.21-eng.pdf)

1)

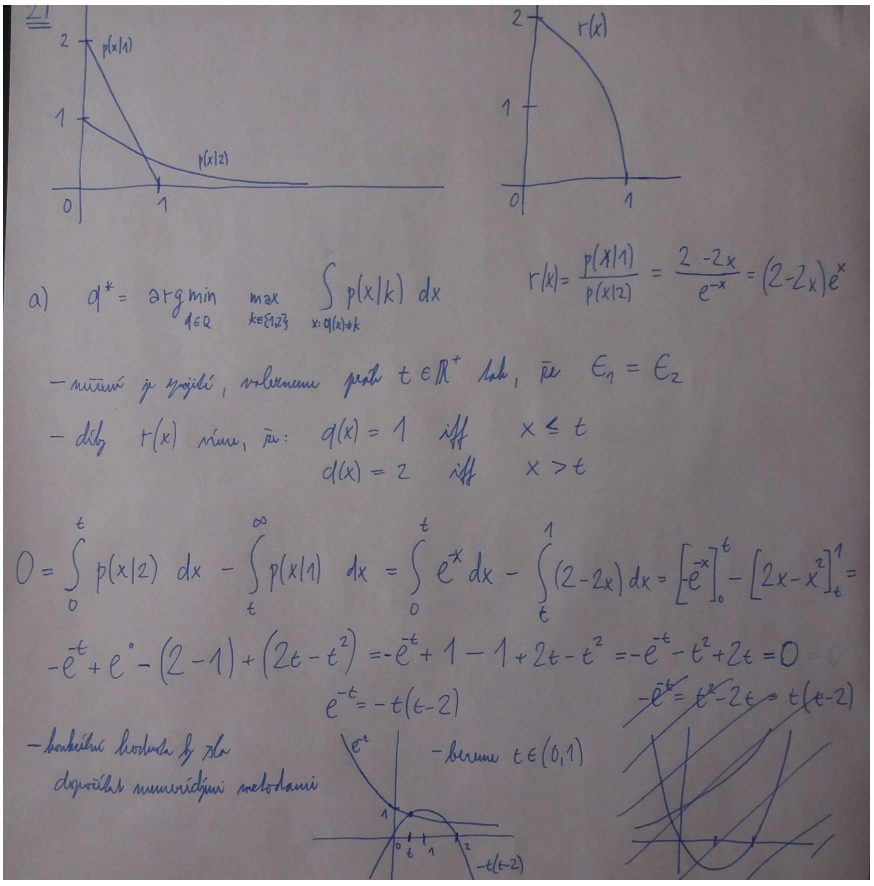
Máme pravděpodobnosti p(x|1), p(x|2), p(1) a p(2). x ohodnotíme jako třídy 1 iff p(x|1)*p(1) > p(x|2)*p(2) iff p(x|1)*p(1)/(p(x|2)*p(2)) > 1 iff ln(p(x|1)*p(1)/(p(x|2)*p(2))) > 0.
Pokud je funkce ln(p(x|1)*p(1)/(p(x|2)*p(2))) lineární, problém se zjednoduší na hledání optimálních lineárně dělících parametrů w a b. Pak:
x ohodnotíme jako 1 iff $w^Tx + b > 0$ (iff $w^Tx > 0$ po úpravě trénovací množiny)
Algoritmus:

1) upravíme trénovací množinu $\{(x_i, y_i)\}$, x z R^D, y = +-1, na $\{y_i*(x_i,1)\}$. Init w z R^{D+1}
2) 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 $w := w - \mu*\nabla_w E$ (jdeme proti směru gradientu, protože minimalizujeme E)
3) Opakujeme krok 2)

S konkrétními hodnotami:

1) Získáme T = {(1, -1), (2, -1), (0, 1), (2, 1)}, w = (0, 0)
2) $\nabla_w E = - Sum_i x*e^{-wx}/(1 + e^{-wx}) = -½((1, -1) + (2, -1) + (0, 1) + (2, 1))$ (e^{-wx} je teď vždy 0)
   $\nabla_w E = -½(5, 0) = (-2.5, 0)$
   w = (0, 0) - 1*(-2.5, 0) = (2.5, 0)
3) Podle přednášky: Pokud $E(w_{nové}) < E(w_{staré})$, tak step = 2*step, jinak: step = step/2

2)



b) $K = \{2=D, 1=N\}$, $p(x|D) = e^{-x}$, $p(x|N) = 2-2x$ pro $x \in \langle 0,1 \rangle$

$\Rightarrow$ nová hodnota x blíže k 0 klasifikovat jako $1=N$

$\Rightarrow$ $\epsilon_{D=2} \leq 0.1$ $\Rightarrow$ máme zvýšit pravděpodobnost, že dostaneme rozsudek

$$0,1 = \int_0^t e^{-x}\,dx = \left[-e^{-x}\right]_0^t = -e^{-t} + e^0 = 1 - e^{-t} = 0,1$$

$$e^{-t} = 0,9$$

$$-t = \ln(0,9)$$

$$t = +0,105$$

Získáme: $q(x) = 1 = N$ iff $x < 0,105$

$q(x) = 2 = D$ iff $x \geq 0,105$

c) $D = \{1,2,?\}$, musíme nalézt t_2 mez pro odhad stavu 2

$\Rightarrow$ $\epsilon_1 \leq 0.1$ $\Rightarrow$ bude "snížen od 1"

$$0,1 = \int_{t_2}^{\infty} p(x|1)\,dx = \int_{t_2}^{1} (2-2x)\,dx + \int_1^{\infty} 0\,dx = \left[2x - x^2\right]_{t_2}^1 =$$

$$0,1 = (2-1) - (2t_2 - t_2^2) = 1 - 2t_2 + t_2^2 \Rightarrow t_2^2 - 2t_2 + 0,9 = 0$$

$$t_2 = \frac{+2 \pm \sqrt{4 - 4\cdot 1\cdot 0,9}}{2} = 1 \pm \frac{\sqrt{0,4}}{2} = 1 \pm 0,316 \Rightarrow \text{bereme } t_2 \in (0,1)$$

Získáme: $q(x) = 1$ iff $x < t = 0,105$

$q(x) = 2$ iff $x > t_2 = 0,684$

$q(x) = ?$ iff $t \leq x \leq t_2$

Lze jistě zvolit hodnotu nerozhodnosti: K_1, K_2:

$$K_1 = \int_{x:\,q(x)=?} p(x|1)\,dx = \int_{0,105}^{0,684} (2-2x)\,dx = \left[2x - x^2\right]_{0,105}^{0,684} \doteq 0,7$$

$$K_2 = \int_{x:\,q(x)=?} p(x|2)\,dx = \int_{0,105}^{0,684} e^{-x}\,dx = \left[-e^{-x}\right]_{0,105}^{0,684} = \left(-e^{-0,684} + e^{-0,105}\right) \doteq 0,316$$

3)

SVM:

- jen pro data z R^D
- trénovací množina může být libovolně velká, při klasifikaci se nevyužívá, pouze se z její části (tzv. support vektorů) vypočítá lineární klasifikátor $q(x) = sign(w^T x)$
- dobře generalizuje, jelikož minimalizuje VC dimenzi maximalizací margin mezi třídami dat, díky konvexnosti rozhodovací funkce a QP optimalizaci nalezne globální optimum
- při učení stačí uchovávat skalární součiny všech x_i, pro klasifikaci stačí uchovávat jediný vektor w z R^D
- učení je závislé na velikosti trénovací množiny a rychlosti konvergence QP optimalizace, která kriticky závisí na počáteční inicializaci
- 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)"

DT:

- pro jakékoliv prostory, i nenumerické, s různými hodnotami/měřítky v různých dimenzích, apod.
- trénovací množina může být libovolně velká, při klasifikaci se nevyužívá (klasifikace je sublineární vůči dimenzi dat)
- žádné záruky na generalizaci/optimalitu řešení
- při učení je potřeba uchovávat veškerá data, při klasifikaci pak již DT samotný (posloupnost otázek/rozhodnutí)
- 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)
- odchýlené hodnoty lze jednoduše odchytávat jako speciální větve stromu

4)

Máme množinu B slabých klasifikátorů h(x). Vybíráme T slabých klasifikátorů z B do silného klasifikátoru $H(x) = sign(Sum_{t=1..T}\ \alpha_t h_t(x))$, kde α_t jsou nezáporné váhy.

Algoritmus:

1) K trénovacím datům (x_i, y_i) přidáme váhu $D_1(i) = 1/N$, x_i z R^D, $y_i = +-1$, N = počet dat, t = 1
2) Najdeme nejlepší slabý klasifikátor z B minimalizací vážené ztráty: $h_t = argmin_{h\ z\ B}\ \epsilon_t(h)$, kde $\epsilon_t(h) = Sum_{i:\ h(i)\ !=\ yi}\ D_t(i)$
3) Pokud $\epsilon_t >= ½$, ukonči (další klasifikátory už nepřináší zlepšení)
4) Váha klasifikátoru h_t je $\alpha_t = ½\ ln((1 - \epsilon_t)/\epsilon_t)$ ($\alpha_t > 0$ protože $\epsilon_t < ½$)
5) Přepočítáme váhy bodů: $D_{t+1}(i) = D_t(i)*exp(-\alpha_t h_t(x) y_i) / Z_t$, kde Z_t je normalizační konstanta, aby $D_t(i)$ bylo rozdělení, $Z_t = Sum_i\ D_t(i)*exp(-\alpha_t h_t(x) y_i) = 2*\sqrt{(\epsilon_t(1 - \epsilon_t))}$ Váhy špatně klasifikovaných se zvětší, správně klasifikovaných se sníží.
6) Pokud t < T, jdi na 2)

Odvození α_t: chceme minimalizovat Z_t:

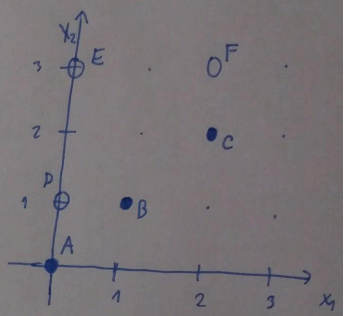
$Z_t = Sum_i\ D_t(i)*exp(-\alpha_t h_t(x) y_i) = Sum_{i:\ ht(x)\ !=\ yi}\ D_t(i)*e^{\alpha t} + Sum_{i:\ ht(x)\ =\ yi}\ D_t(i)*e^{-\alpha t}$

$\partial Z_t/\partial \alpha_t = Sum_{i:\ ht(x)\ !=\ yi}\ D_t(i)*e^{\alpha t} + Sum_{i:\ ht(x)\ =\ yi}\ D_t(i)*e^{-\alpha t}*(-1) = 0$

$\partial Z_t/\partial \alpha_t = \epsilon_t*e^{\alpha t} - (1 - \epsilon_t)*e^{-\alpha t} = 0$, odtud: $\epsilon_t*e^{\alpha t} = (1 - \epsilon_t)*e^{-\alpha t}$

$e^{2*\alpha t} = (1 - \epsilon_t)/\epsilon_t$, odtud: $\alpha_t = ½\ ln((1 - \epsilon_t)/\epsilon_t)$

4) $T = \{((0,0),1,\frac{1}{6}),((1,1),1,\frac{1}{6}),((2,2),1,\frac{1}{6}),((0,1),-1,\frac{1}{6}),((0,3),-1,\frac{1}{6}),((2,3),-1,\frac{1}{6})\}$



$t=1$: 1) chyby: $\epsilon_{h1m} = \frac{2}{6}$ pro $m=1$

$\epsilon_{h2m} = \frac{3}{6}$ pro $m=-1$

$\epsilon_{h3m} = \frac{3}{6}$ pro $m=-1$

$\epsilon_{h4m} = \frac{1}{6}$ pro $m=5$ (správně klasifikuje D)

2) $\epsilon_{h45} < \frac{1}{2}$, $h_1(x) = \delta(x_2 \le 5/2)$

3) $\alpha_1 = \frac{1}{2}\ln\left(\frac{\frac{5}{6}}{\frac{1}{6}}\right) = \frac{1}{2}\ln(5) = 0{,}805$

4) $D_2(D) = \frac{1}{6}\cdot e^{\alpha_1} / Z_1 = \frac{2{,}24\cdot 3}{6\cdot\sqrt{5}} = \frac{1{,}12}{\sqrt{5}} \doteq 0{,}5$

$D_2(A,B,C,E,F) = \frac{1}{6}\cdot e^{-\alpha_1} / Z_1 = \frac{0{,}44\cdot 3}{6\cdot\sqrt{5}} = \frac{0{,}22}{\sqrt{5}} \doteq 0{,}1$

$Z_1 = 2\sqrt{\epsilon_1(1-\epsilon_1)} = 2\sqrt{\frac{1}{6}\frac{5}{6}} = 2\frac{\sqrt{5}}{6} = \frac{\sqrt{5}}{3}$

$t=2$: 1) chyby (duplicitě je klasifikován dobře bod D):

$\epsilon_{h1m} = 0{,}2$ pro $m=1$ $\quad \epsilon_{h2m} = 0{,}3$ pro $m=-1$

$\epsilon_{h3m} = 0{,}3$ pro $m=7$ $\quad \epsilon_{h4m} = 0{,}2$ pro $m=1$

2) $\epsilon_{h11} < \frac{1}{2}$, $h_2(x) = \delta(x_1 > 1/2)$ (správně klasifikuje A a F)

3) $\alpha_1 = \frac{1}{2}\ln\left(\frac{4/5}{1/5}\right) = \frac{1}{2}\ln(4) = 0{,}693$

4) $D_2(A,F) = 0{,}1\cdot e^{\alpha_1} / Z_2 \doteq \frac{2.5}{10.4} = \frac{1}{4}$

$D_2(D) = 0{,}5\cdot e^{-\alpha_1} / Z_2 = \frac{0{,}5\cdot 5}{2.4} = 0{,}3125$

$D_2(B,C,E) = 0{,}1\cdot e^{-\alpha_1} / Z_2 = \frac{0{,}5\cdot 5}{10.4} = \frac{1}{16}$

$Z_2 = 2\sqrt{\frac{1}{5}\frac{4}{5}} = 2\frac{2}{5} = \frac{4}{5}$

$H_2(x) = \text{sign}(0{,}8\cdot h_{45}(x) + 0{,}7\cdot h_{11}(x))$

Zkouška 2015

(http://cmp.felk.cvut.cz/cmp/courses/recognition/Exam-questions/test_2015.01.09-eng.pdf)

1)

a)

Proběhlo celkově n hodů. Každý z nich bereme jako nezávislý, proto v každém byla pravděpodobnost q, že padne šestka, a 1-q že nepadne šestka. Víme, že nejdříve n-1-krát nepadla, pak jednou padla, proto daný vzoreček.

b)

$q_{MLE} = \text{argmax}_q\ p(n|q) = \text{argmax}_q\ (1-q)^{n-1}q = \text{argmax}_q\ L(q)$

$l(q) = \ln(L(q)) = (n-1)\ln(1-q) + \ln(q)$

$l'(q) = -(n-1)/(1-q) + 1/q = 0$

$(n-1)/(1-q) = 1/q$

$qn-q = 1-q$, odtud $qn = 1$ a pak $q = 1/n$

c)

$q < 1/n_{max}$

d)

$q_{MLE} = \text{argmax}_q\ p(n|q) = \text{argmax}_q\ (1-q)^{nmax} = \text{argmax}_q\ L(q)$

$l(q) = \ln(L(q)) = n_{max}*\ln(1-q)$

$l'(q) = -n_{max}/(1-q) = 0$… nemá stacionární bod, musí nabývat extrému v krajním bodě:

$q \to 1 \Rightarrow l(q) = n_{max}*\ln(0) = -\inf$

$q \to 0 \Rightarrow l(q) = n_{max}*\ln(1) = 0$

Což říká i intuice, že $q_{MLE} = 0$, protože čím vyšší q, tím vyšší šance, že by se objevilo, jenže ono se neobjevilo vůbec

e)

Buď to bude q_{MLE} co nám normálně vyjde, nebo krajní bod <0,1; 0,2>, který je blíž.

$q_{MAP} = \text{argmax}_q\ p(q|n) = \text{argmax}_q\ p(n|q)p(q) = \text{argmax}_{q\ z\ 0,1...0,2}\ p(n|q) = \text{argmax}_{q\ z\ 0,1...0,2}\ (1-q)^{n-1}q = \text{argmax}_{q\ z\ 0,1...0,2}\ L(q)$

$l(q) = \ln(L(q)) = (n-1)\ln(1-q) + \ln(q)$

…

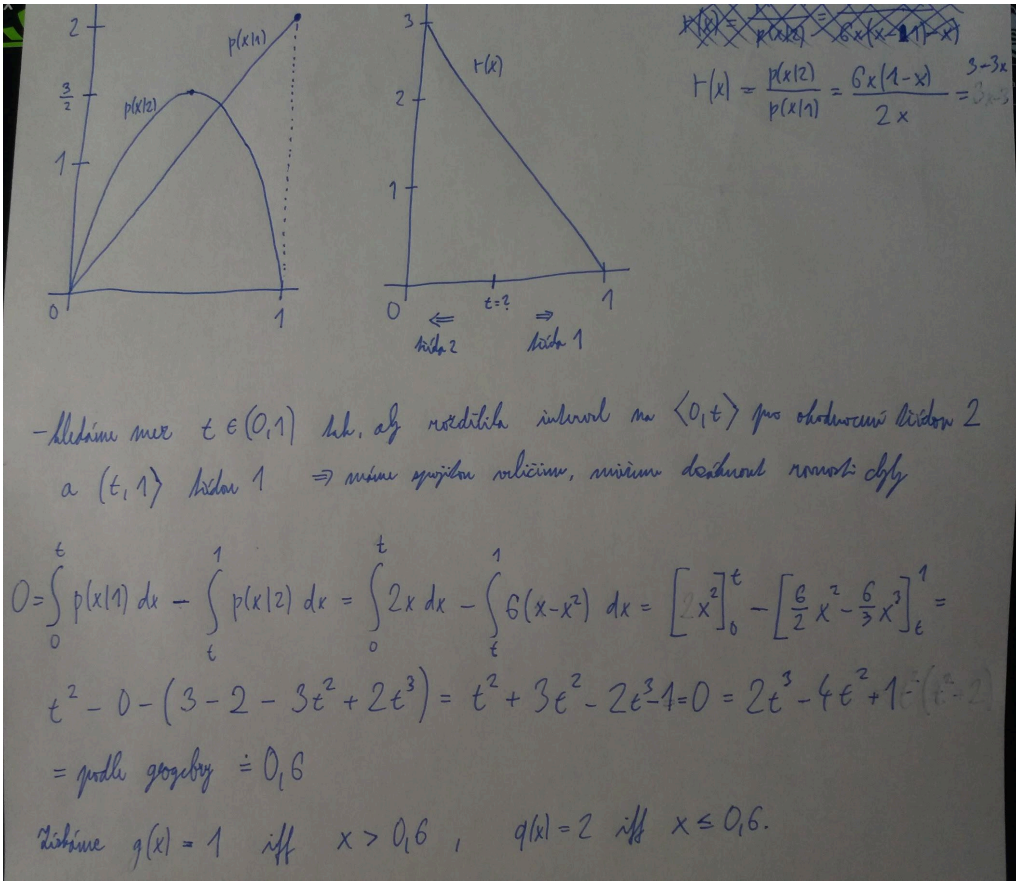
$q_{MLE} = 1/n = 1/1 \Rightarrow q_{MAP} = 0,2$

2)

Máme třídy k z {1, …, K}. Známe p(x|k) pro všechny třídy. Chceme minimalizovat největší riziko na třídě (uvažujeme 0/1 ztrátovou funkci):

$q^* = \text{argmin}_q \max_k \text{Sum}_{x:\, q(x) != k}\ p(x|k)$ (případně integrál místo sumy u spojitých veličin)

V příkladu je předpokládám chyba a má to být: p(x|1) = 2*x (aby to integrovalo do 1).



$r(x) = \frac{p(x|2)}{p(x|1)} = \frac{6x(1-x)}{2x} = 3-3x$

– hledáme míru $t \in (0,1)$ tak, aby rozdělila interval na $\langle 0, t\rangle$ pro ohodnocení třídou 2 a $(t, 1\rangle$ třídou 1 ⇒ máme spojitou veličinu, musíme dosáhnout rovnosti obsahů

$$0 = \int_0^t p(x|1)\,dx - \int_t^1 p(x|2)\,dx = \int_0^t 2x\,dx - \int_t^1 6(x-x^2)\,dx = \left[x^2\right]_0^t - \left[3x^2 - 2x^3\right]_t^1 =$$

$$t^2 - 0 - (3 - 2 - 3t^2 + 2t^3) = t^2 + 3t^2 - 2t^3 - 1 = 0 = 2t^3 - 4t^2 + 1 = (t-2)$$

= podle grogeby $\doteq 0{,}6$

Dostáváme $g(x) = 1$ iff $x > 0{,}6$, $g(x) = 2$ iff $x \leq 0{,}6$.

3)

Máme $T = \{x_i\}_{i=1..N}$, x_i z R^D. Chceme je rozdělit do K shluků (tříd) tak, abychom minimalizovali sumu vzdáleností bodů od center svých tříd. Tzn. hledáme K center c_k:

$(c_1, \ldots, c_K)^* = \text{argmin}_{(c1, \ldots, cK)}\ \text{Sum}_{i=1..N}\ ||x_i - c_k||$

Algoritmus:

1) init $(c_1, \ldots, c_K)$ (například náhodně výběrem z T nebo pomocí KM++)
2) každý bod zařadíme do třídy, jejíž centrum je mu nejblíž:
   $T_k = \{x_i : ||x_i - c_k|| <= ||x_i - c_j||$ pro všechna j$\}$
3) updatujeme centra do těžiště jejich shluků:
   $c_k = 1/|T_k| * \text{Sum}_{x\ z\ Tk}\ x$ (nebo reinicializujeme, když je T_k prázdné)
4) pokud se žádné T_k nezměnilo, ukonči, jinak jdi na 2)

a)

$C = K*C_w + (\min_{((X1, Y1), \ldots, (XK, YK))}\ \text{Sum}_{i=1..n}\ \min_{k\ z\ 1..K} ||(x_i, y_i) - (X_k, Y_k)||) * C_p$

b) + c)

Budeme iterativně postupovat pro K = 1, 2, …

Pro každé K vypočítáme C:

$C = K*C_w + (\min_{((X1, Y1), \ldots, (XK, YK))} \mathrm{Sum}_{i=1..n} \min_{k\ z\ 1..K} ||(x_i, y_i) - (X_k, Y_k)||) * C_p$

$\min_{((X1, Y1), \ldots, (XK, YK))} \mathrm{Sum}_{i=1..n} \min_{k\ z\ 1..K} ||(x_i, y_i) - (X_k, Y_k)||$ nalezneme upravený K-Means, který bude používat L2 euklidovskou normu bez kvadrátu:

1) init (c_1, …, c_K) (například náhodně výběrem z T nebo pomocí KM++)
2) každý bod zařadíme do třídy, jejíž centrum je mu nejblíž:
   $T_k = \{x_i : ||x_i - c_k|| <= ||x_i - c_j||$ pro všechna j}
3) updatujeme centra pomocí Weiszfeldova algoritmu (iterativní algoritmus), tj:
   $c_k = \mathrm{argmin}_c \mathrm{Sum}_{x\ z\ Tk} ||x - c||$
4) pokud se žádné T_k nezměnilo, ukonči, jinak jdi na 2)

Vztah K a ceny za studně je lineární a rostoucí, vztah K a ceny za trubky je nerostoucí. Přijde mi, že optimální K je poslední K, u kterého se celková cena C nezvýší oproti předchozí (viz. Problem 10.3 b) ze sbírky)

d)

Jako v předchozím bodě, ale upravíme KM algoritmus pro L1 Manhattanskou normu:

1) init (c_1, …, c_K) (například náhodně výběrem z T nebo pomocí KM++)
2) každý bod zařadíme do třídy, jejíž centrum je mu nejblíž:
   $T_k = \{x_i : ||x_i - c_k||_1 <= ||x_i - c_j||_1$ pro všechna j}
3) updatujeme centra do mediánu jejich shluku (pomocí upraveného quick sortu):
   $c_k = \mathrm{argmin}_c \mathrm{Sum}_{x\ z\ Tk} ||x - c||_1$
4) pokud se žádné T_k nezměnilo, ukonči, jinak jdi na 2)

4)

SVM je lineární klasifikátor, nachází oddělující nadrovinu s největším možným marginem.
$T = \{(x_i, y_i)\}_{i=1..N}$, $y_i = +-1$
Jeho úkol je zřejmý v původním zápisu úlohy:
$(w, b)^* = \text{argmax}_{w,b} \min_i d(x_i)$ za podmínky $y_i(w^T x_i + b) > 0$, kde $d(x_i)$ je vzdálenost x_i od nadroviny
Primární úloha:
$(w, b)^* = \text{argmin}_{w,b} ½||w||^2$ za podmínky $y_i(w^T x_i + b) >= 1$
Používá se ale duální forma úlohy:
$\alpha^* = \text{argmax}_\alpha \text{Sum}_i \alpha_i - ½ \text{Sum}_i \text{Sum}_j \alpha_i\alpha_j y_i y_i x_i^T x_j$ za podmínky $\alpha >= 0$, $\text{Sum}_i y_i\alpha_i = 0$
Nenulová α_i pak určují support vectory SV, které ovlivňují řešení:
$w = \text{Sum}_{i\ z\ SV} \alpha_i y_i x_i$, $b = y_{sv} - w^T x_{sv}$
Rozhodujeme pak $q(z) = \text{sign}(w^T z + b)$.
Funkce v argmax je konvexní, maximalizaci tedy řešíme QP, které nám dá globální maximum. Rychlost učení velmi závisí na inicializaci.
SVM dobře generalizují, 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.

Neseparovatelná data:
Upravíme úlohy přidáním ztráty za špatnou klasifikaci bodu C:
Primární úloha:
$(w, b)^* = \text{argmin}_{w,b} ½||w||^2 + C*\text{Sum}_i \xi_i$ za podmínky $y_i(w^T x_i + b) >= 1 - \xi_i$, $\xi_i >= 0$
Duální úloha:
$\alpha^* = \text{argmax}_\alpha \text{Sum}_i \alpha_i - ½ \text{Sum}_i \text{Sum}_j \alpha_i\alpha_j y_i y_i x_i^T x_j$ za podmínky $0 <= \alpha <= C$, $\text{Sum}_i y_i\alpha_i = 0$

Při použití dimension lifting $\varphi(x)$: $R^D -> R^M$ 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:
$\alpha^* = \text{argmax}_\alpha \text{Sum}_i \alpha_i - ½ \text{Sum}_i \text{Sum}_j \alpha_i\alpha_j y_i y_i K(x_i, x_j)$ za podmínky $0 <= \alpha <= C$, $\text{Sum}_i y_i\alpha_i = 0$
Klasifikace má dva způsoby:

1) použijeme mapování $\varphi(x)$:
   Získáme $w = \text{Sum}_{i\ z\ SV} \alpha_i y_i \varphi(x_i)$, $b = y_{sv} - w^T\varphi(x_{sv})$ a klasifikujeme
   $q(z) = \text{sign}(w^T\varphi(z) + b)$
   To znamená, že musíme znát φ a klasifikace je zpomalena oproti lineární variantě výpočtem $\varphi(z)$
2) použijeme kernel $K(x, y)$:
   $b = y_{sv} - \text{Sum}_{i\ z\ SV} \alpha_i y_i K(x_i, x_{sv})$
   $q(z) = \text{sign}(b + \text{Sum}_{i\ z\ SV} \alpha_i y_i K(x_i, z))$
   To znamená, že si musíme pro klasifikaci pamatovat α_i, y_i, x_i pro support vektory a klasifikace je zpomalena výpočtem $\text{Sum}_{i\ z\ SV} \alpha_i y_i K(x_i, z)$

Sbírka příkladů

(https://cw.fel.cvut.cz/b201/_media/courses/be5b33rpz/labs/rpz_exercise_book.pdf)

1.1

Protože p(A|B)*p(B) = p(A ∩ B) = p(B ∩ A) = p(B|A)*p(A)

1.2

a)

p(rain) = 0,27
p(no rain) = 0,73
p(1) = 0,4
p(2) = 0,4
p(3) = 0,15
p(5) = 0,05

b)

p(1 nebo 2 | rain) = (p(1 | rain) + p(2 | rain)) / p(rain) = 0,14 / 0,27

1.3

Cancer = {yes, no}, Test = {positive, negative}. Víme:
p(+, y) = 0,98, p(-, y) = 0,02, p(-, n) = 0,97, p(+, n) = 0,03, p(y) = 0,001

a)

p(y, +) = p(+, y) * p(y) / (p(+, y) * p(y) + p(+, n) * p(n)) = 0,98*0,001/(0,98*0,001 + 0,03*0,999) = 0,0316

b)

p(n, -) = p(-, n) * p(n) / (p(-, n) * p(n) + p(-, y) * p(y)) = 0,97*0,999/(0,97*0,999 + 0,02*0,001) = 0,99998

1.4

D = (y, n), pro každé x můžeme spočítat parciální riziko $R(x, q) = Sum_k\ p(k|x)*W(k, q(x))$

a)

$R(x, q_{yes}) = 0,4*0 + 0,2*C + 0,2*C + 0,2*C = 0,6*C$
$R(x, q_{no}) = 0,4*C + 0,2*0 + 0,2*0 + 0,2*0 = 0,4*C$
Nižší Bayesovské riziko je pro volbu "no"

b)

$R(x, q_{yes}) = 0,4*0 + 0,2*C/2 + 0,2*C/2 + 0,2*C/2 = 0,6*C/2 = 0,3*C$
$R(x, q_{no}) = 0,4*C + 0,2*0 + 0,2*0 + 0,2*0 = 0,4*C$
Nižší Bayesovské riziko je pro volbu "yes"

1.5

a)

Počítáme parciální riziko $q^*(x) = argmin_q\ R(q, x)$
$R(q, x) = Sum_{k\ z\ K}\ p(k|x)*W(k, q(x))$
$= R(d, x) = Sum_{k\ z\ K}\ p(k|x)*W(k, d) = Sum_{k\ !=\ d}\ p(k|x)*1 = 1 - p(d, x)$
$argmin_q\ R(q, x) = argmin_d\ (1 - p(d, x)) = argmax_d\ p(d, x)$

b)
D = K = (0,1). Bayesovské rozhodování q(x) lze vyvodit ze zlomku aposteriorních pravděpodobností.
$q_0(x) = 0$, parciální riziko $R(x, q_0) = Sum_{k != d}\ p(k|x)*1 = p(1|x)$
$q_1(x) = 1$, parciální riziko $R(x, q_1) = Sum_{k != d}\ p(k|x)*1 = p(0|x)$
Bereme menší riziko, což lze vyčíst už předem ze zlomku p(0|x)/p(1|x).
Pokud p(0|x)/p(1|x) >= 1, volíme 0, jinak volíme 1.
Po dosazení p(0|x) = p(x|0)p(0)/p(x) a p(1|x) = p(x|1)p(1)/p(x) získáme:
p(x|0)p(0) / p(x|1)p(1) >= 1 a odtud p(x|0)/p(x|1) >= p(1)/p(0) = const = Θ
c)
$p(x|0) = 1/(sqrt(2*pi)*\sigma_0) * exp(-(x - \mu_0)^2/(2*\sigma_0^2))$, obdobně pro p(x|1).
p(0|x)/p(1|x) = p(x|0)*p(0)/(p(x|1)*p(1)) =
$= \sigma_1*p(0)/(\sigma_0*p(1)) * exp(- (x - \mu_0)^2/(2*\sigma_0^2) + (x - \mu_1)^2/(2*\sigma_1^2))$
p(0|x)/p(1|x) <> 1 je ekvivalentní s ln(p(0|x)/p(1|x)) <> 0,
neboli q(x) = 0 iff p(0|x)/p(1|x) >= 1 iff ln(p(0|x)/p(1|x)) >= 0
$\ln(p(0|x)/p(1|x)) = 0 = - (x - \mu_0)^2/(2*\sigma_0^2) + (x - \mu_1)^2/(2*\sigma_1^2) + \ln(\sigma_1*p(0)/(\sigma_0*p(1))$
$0 = - \sigma_1^2*(x - \mu_0)^2 + \sigma_0^2*(x - \mu_1)^2 + 2*\sigma_0^2*\sigma_1^2*\ln(\sigma_1*p(0)/(\sigma_0*p(1))$
$0 = - \sigma_1^2*(x^2 - 2*\mu_0*x + \mu_0^2) + \sigma_0^2*(x^2 - 2*\mu_1*x + \mu_1^2) + 2*\sigma_0^2*\sigma_1^2*\ln(\sigma_1*p(0)/(\sigma_0*p(1))$
$0 = (\sigma_0^2 - \sigma_1^2)*x^2 + 2*(\sigma_0^2*\mu_1 - \sigma_1^2*\mu_0)*x + \sigma_0^2*\mu_1^2 - \sigma_1^2*\mu_0^2 + 2*\sigma_0^2*\sigma_1^2*\ln(\sigma_1*p(0)/(\sigma_0*p(1)) =$
ax^2+bx+c

1.6
Oblacnost O = {1,2,3,4}
Vypočítal jsem:
p(rain|1) = 0,05, p(rain|2) = 0,3, p(rain|3) = 0,6, p(rain|4) = 0,8
Pro rozhodnuti mi vysla rizika (p = pravdepodobnost deste)
R(umbrella) = 0 + 5(1 - p) = 5 - 5p
R(no umbrella) = 10p - 2(1 - p) = 12p - 2
R(100) = 5p - 0 = 5p
Vysla mi optimalni reseni:
O = 1: no umbrella (s ocekavanou ztratou -1,4)
O = 2: 100 (s ocekavanou ztratou 1,5)
O = 3: umbrella (s ocekavanou ztratou 2)
O = 4: umbrella (s ocekavanou ztratou 1)

1.7
a)
3 bity musely být 000, 011, 101 nebo 110. Pro dane pravdepodobnosti mame:
000 = 0,7*0,6*0,3 = 0,126
011 = 0,7*0,4*0,7 = **0,196** s pravděpodobností 0,196/0,484 = 0,405
101 = 0,3*0,6*0,7 = 0,126
110 = 0,3*0,4*0,3 = 0,036
b)
R(request again) = 0,405 * “zbytečný request” = 0,405 * C
R(skip) = 0,595 * “přehlédnout chybu” = 0,595 * 100 * C
Určitě se vyplatí request again

1.8

Máme p(k) = ⅓, pravděpodobnosti p(k|x) = p(x|k)*p(k) = p(x|k)/3:

p(1|0) = 0,399/3 = 0,133, p(2|0) = 0,199/3 = 0,066, p(3|0) = 0,065/3 = 0,022

p(1|1) = 0,242/3 = 0,081, p(2|1) = 0,176/3 = 0,059, p(3|1) = 0,121/3 = 0,040

(přepsal jsem se a mám třídy 0,1,2 místo 1,2,3)

a) W = 0/1, x = 0

R(x = 0, q(x) = 0) = 0,133*0 + 0,066*1 + 0,022*1 = 0,088

R(x = 0, q(x) = 1) = 0,133*1 + 0,066*0 + 0,022*1 = 0,155

R(x = 0, q(x) = 2) = 0,133*1 + 0,066*1 + 0,022*0 = 0,199

Bereme nejmenší parciální riziko, takže q(x) = 0

b) W = 0/1, x = 1

R(x = 1, q(x) = 0) = 0,081*0 + 0,059*1 + 0,040*1 = 0,099

R(x = 1, q(x) = 1) = 0,081*1 + 0,059*0 + 0,040*1 = 0,121

R(x = 1, q(x) = 2) = 0,081*1 + 0,059*1 + 0,040*0 = 0,140

Bereme nejmenší parciální riziko, takže q(x) = 0

c) W = 0/1/2, x = 0

R(x = 0, q(x) = 0) = 0,133*0 + 0,066***2** + 0,022*1 = 0,154

R(x = 0, q(x) = 1) = 0,133***2** + 0,066*0 + 0,022*1 = 0,288

R(x = 0, q(x) = 2) = 0,133*1 + 0,066*1 + 0,022*0 = 0,199

Bereme nejmenší parciální riziko, takže q(x) = 0

b) W = 0/1/2, x = 1

R(x = 1, q(x) = 0) = 0,081*0 + 0,059***2** + 0,040*1 = 0,158

R(x = 1, q(x) = 1) = 0,081***2** + 0,059*0 + 0,040*1 = 0,202

R(x = 1, q(x) = 2) = 0,081*1 + 0,059*1 + 0,040*0 = 0,140

Bereme nejmenší parciální riziko, takže q(x) = 2

Pravděpodobnost špatného rozhodnutí při W = 0/1, x = 1:

Rozhodli jsme pro d = 1. p(k != d) = (p(2|x) + p(3|x))/(p(1|x) + p(2|x) + p(3|x)) = 0,4375.

1.9

a)

K = {0, 1, ..., K}, X = {1, 2, ..., K}, D = {yes, no}

b)

Nejsem si jistý, ale tak snad, dává mi to smysl.

$p(k >= K/2) = q^{K/2}$

$p(k >= K/2 \mid x) = q^{K/2 - x}$ iff x < K/2

= 1 iff x >= K/2

c)

q(x) = n iff x < K/2 && $q^{K/2 - x} < ½$

= y iff x < K/2 && $q^{K/2 - x} >= ½$

= y iff x >= K/2

2.1
Uděláme si poměr p(x|M)/p(x|F):

0,056 (11)	0,034 (12)	0,117 (9)	0,647 (7)
0,065 (10)	0,237 (8)	2,814 (3)	2,235 (4)
2 (5)	1,75 (6)	inf (1)	inf (2)

Nejdříve vycházíme z toho, že vše klasifikujeme jako ženy. Postupně klasifikujeme jako muže x od nejvyšších poměrů, při tom kontrolujeme abychom nepřesáhli 0,2 u součtu p(x|D) těchto x.
Získáme:
$\epsilon_D = 0 + 0 + 0{,}145 + 0{,}017 + 0{,}001 + 0{,}008 + 0{,}017 = 0{,}188$
$\epsilon_N = 0{,}011 + 0{,}005 + 0{,}011 + 0{,}005 + 0{,}071 = 0{,}103$
Pak lze určit mez z (0,237; 0,647), např. $\Theta = 0{,}4$ a rozhodovat:
x ohodnotit jako muže iff $p(x|M)/p(x|F) > \Theta$
x ohodnotit jako ženu iff $p(x|M)/p(x|F) < \Theta$
Buď skončíme takto, nebo lze optimum ještě vylepšit randomizací. Máme možnost ještě ohodnotit 0,2 - 0,188 = 0,012 žen špatně => další x na řadě (8) má u žen p(x|D) = 0,299. Proto u x (8) můžeme ohodnotit 0,012/0,299 = 0,04 = 4% vzorků jako muže, čímž získáme:
$\epsilon_D = 0 + 0 + 0{,}145 + 0{,}017 + 0{,}001 + 0{,}008 + 0{,}017 + 0{,}04*0{,}299 = 0{,}2$
$\epsilon_N = 0{,}011 + 0{,}005 + 0{,}011 + 0{,}005 + 0{,}96*0{,}071 = 0{,}100$

2.2
Nejhorší případ úspěšnost strategie q je, když třída k, která má vyšší $\epsilon_k(q)$ (tzn. při trénování bylo p(k) < p(druhá k)), bude mít při testování p(k) = 1.
a)
$R(q) = \mathrm{Sum}_{k\ z\ K}\ \mathrm{Int}_{x\ z\ X}\ p(x,k)*W(k, q(x)) = \mathrm{Sum}_{k\ z\ K}\ p(k)\ \mathrm{Int}_{x\ z\ X}\ p(x|k)*W(k, q(x))$
$R(q, \pi) = \pi * \mathrm{Int}_{x\ z\ X}\ p(x|1)*W(1, q(x)) + (1 - \pi) * \mathrm{Int}_{x\ z\ X}\ p(x|2)*W(2, q(x)) = \pi*\epsilon_1 + (1 - \pi)*\epsilon_2$
$R(q, \pi) = \pi*(\epsilon_1 - \epsilon_2) + \epsilon_2$, kde ϵ_k je ztráta na třídě k
R(q) roste s π iff $\epsilon_1 > \epsilon_2$, R(q) je konst. iff $\epsilon_1 = \epsilon_2$, R(q) klesá s π iff $\epsilon_1 < \epsilon_2$
b)
R(q, π) je lineární funkce π, extrému nabývá v krajním bodě definičního oboru <0, 1>
(v případě konstantní funkce je nezávislá na změny π a extrém je kdekoliv v <0, 1>)
c)
$\max_{\pi\ z\ \{0,1\}}\ (\epsilon_2 + (\epsilon_1 - \epsilon_2)*\pi) = \max\{\epsilon_1, \epsilon_2\}$
Po úpravě na 0/1 loss mame $\epsilon_1 = \mathrm{Int}_{x:\ g(x)\ !=\ 1}\ p(x|1)$. Chceme minimalni bayesovske riziko, neboli minimalizujeme max z techto integralu, coz je minimax.
d)
R(q) je konstantní kdyz $\epsilon_1 = \epsilon_2$. To je presne kdyz se chyba na obou tridach rovna, neboli jsme dosáhli minimálního maxima (dalsi snizeni chyby na jedne ze tříd by znamenalo zvyseni druhe).

2.3
a)
$q^* = \mathrm{argmin}_q\ \epsilon_q(N)$ za podmínky $\epsilon_q(D) <= \epsilon$,
kde $\epsilon_q(N) = \mathrm{Sum}_{x:\ q(x)\ =\ D}\ p(N|x)$, $\epsilon_q(D) = \mathrm{Sum}_{x:\ q(x)\ =\ N}\ p(D|x)$ a ϵ je určená mez.

b)
Hledáme mez θ poměru p(x|N)/p(x|D) = r(x) tak, abychom rozhodovali
x ohodnotit jako N iff r(x) >= θ
x ohodnotit jako D iff r(x) < θ
r(x) = 1/(x + 0,5) je klesající funkce na <0, 1>, pozorování blíže k 0 budeme označovat za N, blíže k 1 za D. Chceme abychom ohodnotili špatně 10% D stavů, neboli
$0,1 = Int_{0 < t < H}(t + 0,5)\ dt = [0,5*t^2 + 0,5t]_0^H = 0,5*H^2 + 0,5*H - 0$
$0 = H^2 + H - 0,2$, $H_{1,2} = - 0,5 \pm 0,67$, ale chceme H z <0, 1>, takže H = 0,17
θ pak získáme jako 1/(0,17 + 0,5) = 1,5, takže hodnotíme:
x ohodnotit jako N iff r(x) >= θ = 1,5
x ohodnotit jako D iff r(x) < θ = 1,5

2.4
a)
$q^* = argmin_q\ max_k\ \epsilon_q(k)$, kde $\epsilon_q(k) = Int_{x:\ q(x)\ !=\ k}\ p(x|k)$ je chyba na k-te tride
b)
Máme jednu mez θ pro r(x) = p(x|1)/p(x|2). Po proložení grafu r(x) konstantním x = θ máme ale dva průsečíky, které dělí x na 3 intervaly <-1, t_1), <t_1, t_2>, (t_2, 1>, navíc díky symetrii víme, že $t_1 = - t_2$.
c)
U minimaxu u dvou tříd se spojitými pozorování víme, že optimum je v situaci kdy $\epsilon_q(1) = \epsilon_q(2)$. Díky symetrii se navíc může omezit na interval <0, 1>, kde nalezneme mez t, kterou poté promítneme jako -t do <-1, 0>.
Na <0, 1> máme p(x|1) = x, p(x|2) = 1 - x. r(x) = p(x|1)/p(x|2) = x/(1 - x) je rostoucí, proto x blíže k 0 budeme hodnotit jako 2, blíže k 1 (a na druhé straně blíže k -1) jako 1.
Hledáme t tak, aby:
$Int_{0 < x < t}\ x\ dx = Int_{t < x < 1}\ (1 - x)\ dx$
$0 = [0,5*x^2]_0^t - [x - 0,5*x^2]_t^1$
$0 = (0,5*t^2 - 0) - (1 - 0,5 - t + 0,5*t^2) = 0,5*t^2 - 0,5 + t - 0,5*t^2 = t - 0,5 = 0$, odkud t = 0,5
Meze pro x jsou $t_1 = -0,5$ a $t_2 = 0,5$
Mez pro r(x) je θ = p(t|1)/p(t|2) = 0,5/(1 - 0,5) = 1
Získáváme klasifikátor:
q(x) = 1 iff r(x) > 1 iff x je z <-1, -0.5) U (0.5, 1>
q(x) = 2 iff r(x) <=1 iff x je z <-0.5, 0.5>
Chyba na třídách $\epsilon_q(1) = \epsilon_q(2) = 2*Int_{0 < x < 0,5}\ x\ dx = 2*[0,5*x^2]_0^{0,5} = 2*(⅛ - 0) = ¼$

3.1

a)

Model bude mít jeden parametr p z <0, 1>, p(H) = p, p(T) = 1 - p, K = {H, T}, **T** = (H,H,H,T,T)

$L(p) = p(\mathbf{T}|p) = Prod_{i\ od\ 1\ do\ N}\ p(x_i) = Prod_{k\ z\ K}\ p(k)^{nk} = p^{nH}*(1 - p)^{nT}$

$l(p) = n_H*ln(p) + n_T*ln(1 - p)$

$l'(pi) = n_H / p + n_T * (-1) / (1 - p) = 0$

Získáme: $n_H / p = n_T / (1 - p)$, odkud $1 - p = p*n_T/n_H$ a $p = n_H / (n_H + n_T)$,

coz pro $n_H = 3$, $n_T = 2$ dava p = 3/5

b)

Tady nu = η:

$p(H) = 1 / (1 + e^{-nu})$, odkud $p(T) = 1 / (1 + e^{nu})$

$L(nu) = Prod_{k\ z\ K}\ p(k)^{nk} = 1 / ((1 + e^{-nu})^{nH} * (1 + e^{nu})^{nT})$

$l(nu) = - ln((1 + e^{-nu})^{nH} * (1 + e^{nu})^{nT}) = - nH*ln(1 + e^{-nu}) - nT*ln(1 + e^{nu})$

$l'(nu) = -n_H*e^{-nu}*(-1)/(1 + e^{-nu}) - n_T*e^{-nu}/(1 + e^{nu}) = 0$

$n_H*e^{-nu} / (1 + e^{-nu}) = n_H*e^{nu} / (1 + e^{nu})$

$n_H/(1 + e^{nu}) = n_H/(1 + e^{-nu})$

Pak přes roznasobeni a kvadratickou rovnici $2*e^{2nu} - e^{nu} - 3 = 0$ k $e^{nu} = 3/2$ a p(T) = 2/5

Ta sigmoid funkce je logistická regrese, rozhoduje pomocí lineární funkce poměru p(1|x)/p(2|x). Lze použít pouze u některých problémů a tento (multinomiální naivní Bayes) je jedním z nich, dále normální rozdělení se stejnou kovarianční maticí a nezávislé binární vstupní vektory.

Problém by asi teoreticky nastal při p(H) = 0 nebo 1, protože obor hodnot sigmoidy je (0, 1).

3.2

a)

To znamena, ze ponožku vratime po vytazeni zpatky

b)

p*p

c)

$l(N = 10, R = 2) = (10\ nad\ 2)*p^2*(1 - p)^8$

$l(N, R) = (N\ nad\ R)*p^R*(1 - p)^{N-R}$

d)

$L(p) = p^R*(1 - p)^{N-R}$

$l(p) = R*ln(p) + (N-R)*ln(1-p)$

$l'(p) = R/p - (N - R)/(1 - p) = 0$, což vede na p = R/N

3.3

a)

$L(\mu, \Sigma) = Prod_{x\ z\ T}\ p(x)$

$L(\mu, \Sigma) = Prod_{x\ z\ T}\ (2*\pi)^{-d/2}*(det(\Sigma))^{-1/2}*e^{(-1/2*(x-\mu)^T*\Sigma^{-1/2}*(x-\mu))}$

$l(\mu, \Sigma) = Sum_{x\ z\ T}\ (- ½*ln((2*\pi)^d * det(\Sigma)) - ½*(x-\mu)^T*\Sigma^{-1/2}*(x-\mu))$

$l(\mu, \Sigma) = Sum_{x\ z\ T}\ (- ½*ln((2*\pi)^d * det(\Sigma)) - ½*x^T*\Sigma^{-1/2}*x + x^T*\Sigma^{-1/2}*\mu - ½*\mu^T*\Sigma^{-½}*\mu)$

$l'(\mu, \Sigma)_\mu = Sum_{x\ z\ T}\ (x^T*\Sigma^{-1/2} - \mu^T*\Sigma^{-1/2}) = 0$

$Sum_{x\ z\ T}\ x^T*\Sigma^{-1/2} = N*\mu^T*\Sigma^{-½}$ ($\Sigma^{-½}$ musí mít inverzi)

$1/N * Sum_{x\ z\ T}\ x = \mu$

b) ?????

$l(\mu, \Sigma) = Sum_{x\ z\ T}\ (- ½*ln((2*\pi)^d*det(\Sigma)) - ½*(x - \mu)^T*\Sigma^{-1/2}*(x - \mu))$

$l'(\mu, \Sigma)_\Sigma = Sum_{x\ z\ T}\ (- ½ * \Sigma^{-1} - ½*(x - \mu)*(x - \mu)^T) = 0$

$Sum_{x\ z\ T}\ \Sigma^{-1} = Sum_{x\ z\ T}\ (x - \mu)*(x - \mu)^T$… což není správně ale, překáží tam ta inverze u Σ

3.4

a)

$L(h) = p(T|h) = Prod_{1 <= i <= N}\ h*e^{-h*ti}$

$l(h) = Sum_{1 <= i <= N}\ ln(h*e^{-h*ti}) = n*ln(h) - Sum_{1 <= i <= N}\ h*t_i$

$l'(h) = n/h - Sum_{1 <= i <= N}\ t_i = 0$, odtud $h = n/(Sum_{1 <= i <= N}\ t_i) = (ar.\ průměr)^{-1}$

b)

Expected lifetime je stredni hodnota $t = E(t) = Integral_{-inf,\ inf}\ t*p(t)\ dt$

$E(t) = Integral_{-inf,\ inf}\ t*1/\theta*e^{-t/\theta}\ dt$ = | per partes: $u = t \rightarrow u' = 1$, $v' = e^{-t/\theta} \rightarrow v = e^{-t/\theta}*-\theta$ | =

$[-\theta*t*e^{-t/\theta}]_0^{inf} - 1/\theta*Integral_{-inf,\ inf}\ -\theta*e^{-t/\theta}\ dt = 0 + Integral_{-inf,\ inf}\ e^{-t/\theta}\ dt = [-\theta*e^{-t/\theta}]_0^{inf} = \theta$

θ rika za jak dlouho se rozbije prumerne žárovka, h rika kolik žárovek se rozbije za jednotku casu.

c)

$h^* = argmax_h\ p(h|T)$

$p(h|T) = p(T|h)*p(h) = e^{-h} * Prod_i\ h*e^{-h*ti}$

$ln(p(T|h)*p(h)) = -h + Sum_i\ ln(h*e^{-h*ti}) = -h + n*ln(h) - h*Sum_i\ t_i$

derivace $= -1 + n/h - Sum_i\ t_i = 0$, odtud $h_{MAP} = n/(1 + Sum_i\ t_i)$

Ocekavana zivotnost O = ar. prumer + 1/n

3.5

a)

Z predchoziho prikladu mame $h = n/(Sum_i\ t_i) = 3 / 600 = 1 / 200 = 0.005$

b) ?????

Nevim jak na to vubec teda

3.6

a)

$L(\theta) = 1 / \theta^n$, max $L(\theta)$ pro $\theta \rightarrow 0$, ale $\theta >= max_i\ x_i$, proto $\theta = max_i\ x_i$

b)

Ano, protoze jsme pravděpodobně neviděli maximální číslo.

c)

$\theta <= M$, $p(\theta) = 1/M$ (uniformní rozdělení)

$p(\theta|T) = p(T|\theta)*p(\theta) = 1 / (\theta^n * M)$, coz zas vede na $\theta = max_i\ x_i$

d) ?????

Aposteriorní rozdělení je na $<max_i\ x_i; inf)$, s tim ze nizsi hodnoty jsou pravdepodobnejsi

Nevím, jestli chtějí přímo vzorec, pak $p(\theta|T) = p(T|\theta)*p(\theta)$, kde ale $p(\theta)$ nevím jak popsat, možná se dá zanedbat, protože pokud je θ náhodná veličina, každá hodnota má stejnou pravděpodobnost

$\theta^* = argmin_\theta\ R(\theta) = argmin_\theta\ Integral_{max\ xi,\ inf}\ p(t|T)*(t - \theta)^2\ dt$

Podle wikipedie (https://en.wikipedia.org/wiki/Bayes_estimator#Minimum_mean_square_error_estimation) by to mělo být $\theta^* = Integral_{max\ xi,\ inf}\ \theta*p(\theta|T)\ dt = Integral_{max\ xi,\ inf}\ \theta/\theta^n\ dt = Integral_{max\ xi,\ inf}\ \theta^{-n+1}\ dt =$

$[1/(-n+2)*\theta^{-n+2}]_{max\ xi}^{inf} = inf$, což je asi kravina (bral jsem zde ten zanedbaný $p(\theta)$, viz výše)

4.1

$L(d) = p(T|d) = Prod_{1 <= i <= K} d_i^{ni}$

$l(d) = Sum_{1 <= i <= K} n_i * ln(d_i)$, $Sum_{1 <= i <= K} di/K = 1$

$Lagrange = Sum_{1 <= i <= K} n_i * ln(d_i) + \lambda' * ((Sum_{1 <= i <= K} d_i) - K)$

Derivace Lagrange podle $d_i = n_i/d_i + \lambda' = 0$, odtud $n_i/d_i = \lambda = konst$, $d_i = n_i/\lambda$

Víme $Sum_{1 <= i <= K} d_i = K$, odtud $Sum_{1 <= i <= K} n_i / \lambda = K$, což je $N / \lambda = K$

Máme $\lambda = N / K$, takže $d_i = n_i/\lambda = K * n_i/N$

4.2

Máme trénovací data $T = \{(x_i, k_i)\}$. Pro testovanou hodnotu x nalezneme K nejblizsich x_j z T. Pro x volíme třídu k takovou, ktera ma mezi x_j vetsinu.

Plusy:

- jednoduchost (při naivní implementaci)

Mínusy:

- přefitovává se
- časová náročnost (při naivní implementaci prochází trénovací body lineárně, lze zlepšit použitím k-d stromů a/nebo pravděpodobnostními zárukami nejbližšího souseda)
- paměťová náročnost (při naivní implementaci potřebuje všechny trénovací body, jde zlepšit redukcí trénovací množiny)

1-NN: nejblizsi je (4.5, A), volíme A

3-NN: nejblizsi jsou (4.5, A), (6, B) a (3, B), volíme B

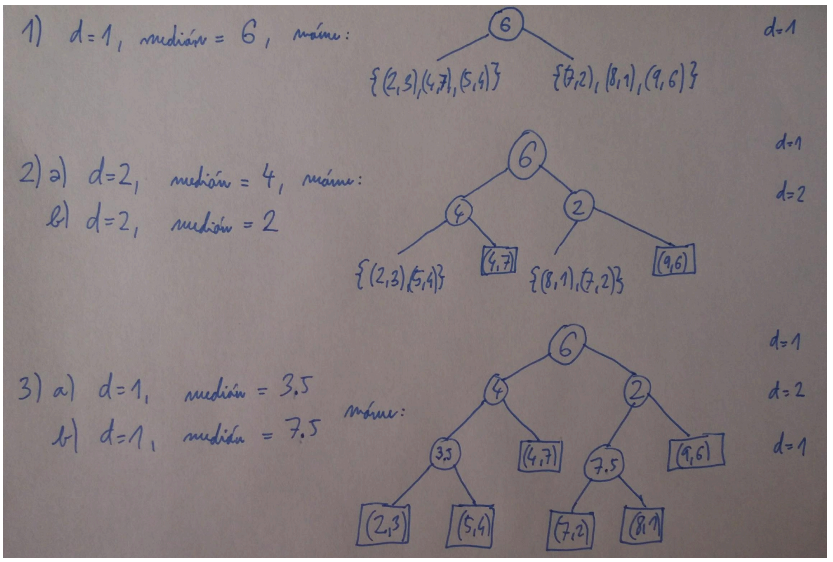
5-NN: nejblizsi jsou (4.5, A), (6, B) a (3, B), (2, A) a (1.5, A), volíme A

4.3

Algoritmus:

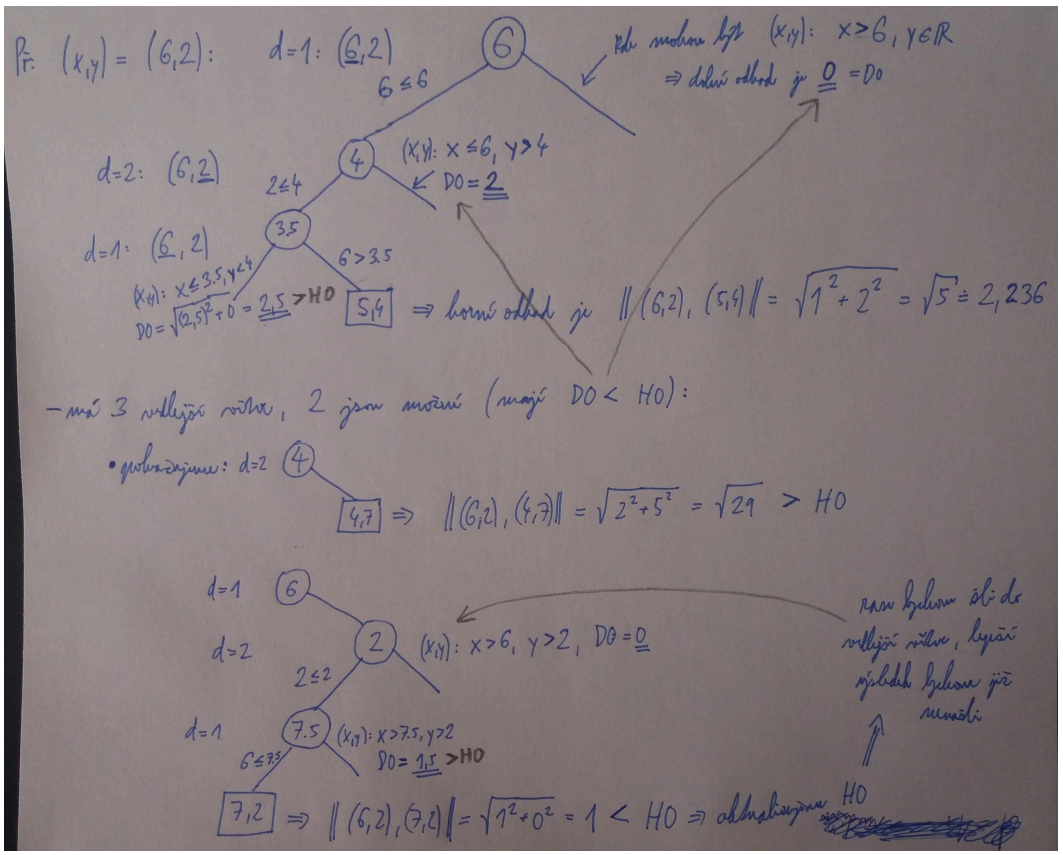
1) Začínáme na indexu vstupních vektorů d = 1 (beru v matice one-indexed), aktuální data = trénovací množina
2) Pokud počet aktuálních dat je 1, ukonči větev rekurze.
3) Najdeme medián aktuální části dat podle d-tého indexu a rozdělíme data na dvě části a ty dáme do větví aktuálního uzlu
4) Rekurzivně pokračujeme do větví krokem 2) a s indexem d+1 (modulo D)

Vytvoření k-D stromu:

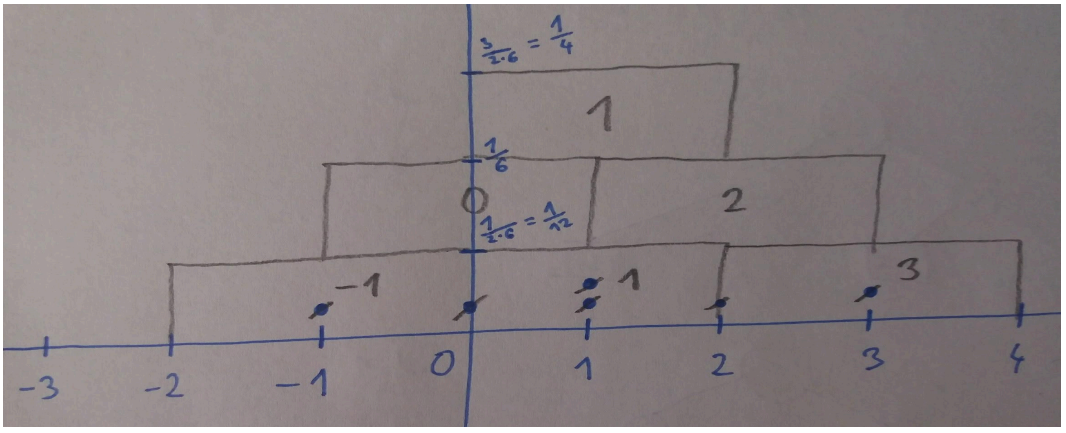


Hledání NN:
Máme bod (x, y). Procházíme od kořene dolů, vždy se vydáváme do té větve, kam nás zavede hodnota x nebo y podle d a větvící nerovnosti v dané hloubce. Při každém zanoření se nám u druhé (nenavštívené) větve určí nejbližší možný soused v ní. Po doražení do listu máme horní odhad nejbližší souseda. Navštívíme ty větve, kde mohou být bližší sousedi.

Příklad (k-D ušetřilo porovnávání s (2, 3) a (8, 1), což není moc teď zrovna, ale při větších množinách to samozřejmě je v průměru více znát):



4.4
Integrál pod kombinací všech jádrových funkcí musí být 1, proto do každého bodu nevložíme obdélník s výškou ½, ale s výškou 1/(2*N) = 1/12:



4.5 ?????

Mame Lagrange = $\text{Sum}_i \text{Sum}_j K_{ij}/K_i * \log(\pi_j * K_i) + h' * (\text{Sum}_i \pi_i - 1)$

“Derivace Lagrange podle π_j” = $\text{Sum}_i K_{ij}/(K_i * \pi_j) + h' = 0$

Odtud $\pi_j = 1/h * \text{Sum}_i K_{ij}/K_i$ ($h = -h'$)

Po dosazení do $\text{Sum}_i \pi_i = 1$ mame $h = \text{Sum}_i \text{Sum}_j K_{ij}/K_i$

5.1

a)

Pres $p(1|x)/p(-1|x) = e^{w^*x}$ získáme $p(1|x) = 1 / (1 + e^{-w^*x})$, $p(-1|x) = 1 / (1 + e^{w^*x})$

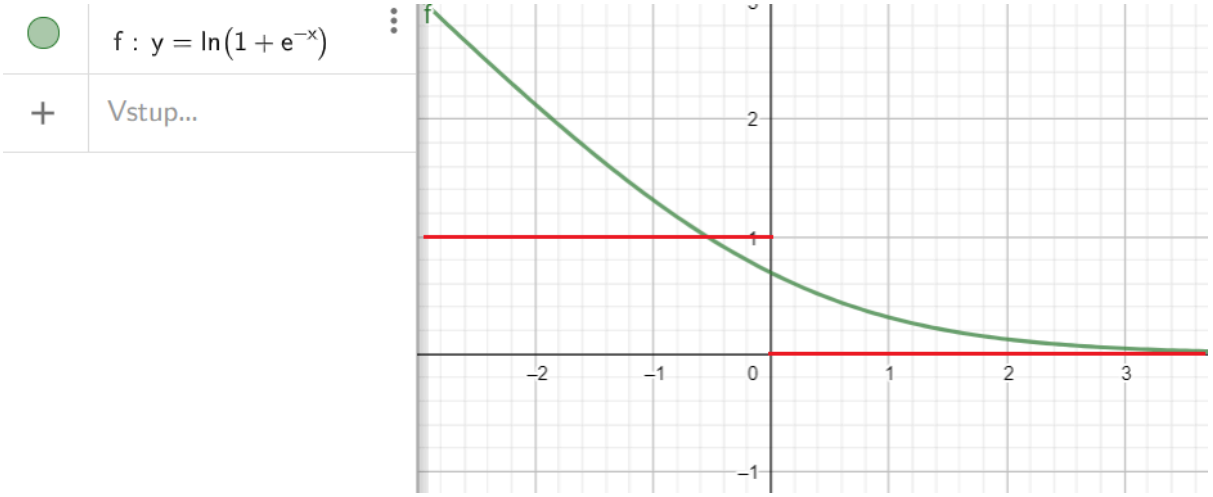
b)

$p(x, k) = p(k|x)*p(x)$ -> $\ln(p(x, k)) = \ln(p(k|x)) + \ln(p(x))$, kde $\ln(p(x))$ je const

$p(k|x) = 1/(1 + e^{-k^*w^*k})$, tedy negace součtu přes všechna trénovací data E(w):

$E(w) = - l(w) = - \text{Sum}_i \ln(1/(1 + e^{-ki^*w^*ki})) = \text{Sum}_i \ln(1 + e^{-ki^*w^*xi})$

c)



E(w) je soucet konvexnich funkci => je konvexní

5.2

a)

$1 - 1/(1 + e^{-z}) = (1 + e^{-z} - 1)/(1 + e^{-z}) = e^{-z}/(1 + e^{-z}) = 1/(1 + e^{z})$

b)

Derivace $1/(1 + e^{-z}) = -1*e^{-z}*(-1)/(1 + e^{-z})^2 = e^{-z}/(1 + e^{-z}) * 1/(1 + e^{-z}) = 1/(1 + e^{z}) * 1/(1 + e^{-z})$

c)

Derivace $\ln(\sigma(z)) = 1/\sigma(z) *$ (Derivace $\sigma(z)) = 1/\sigma(z) * \sigma(z)*(1 - \sigma(z)) = 1 - \sigma(z)$

d)

Derivace $\ln(\sigma(w^Tx))$ podle x = (Derivace $\ln(\sigma(w^Tx))$ podle w^Tx) * (Derivace w^Tx podle x) =

$= (1 - \sigma(w^Tx)) * w^T$

e)

Derivace $- \ln(\sigma(z)) = -\sigma(-z) = \sigma(z) - 1$, což je rostoucí funkce

Druhá derivace $- \ln(\sigma(z))$ = Derivace $- \sigma(-z)$ podle $z = -1 *$ (Derivace $\sigma(-z)$ podle $-z) *$ (Derivace $-z$ podle $z) = -1 *$ (Derivace $\sigma(z)$ podle $z) * -1 = \sigma(z)(1 - \sigma(z))$, což je soucin kladných cisel, který je kladný

Problem 6.1

a)

Lagrangeova funkce:

$L = ||\mathbf{x}-\mathbf{y}||^2 + \lambda(\mathbf{a}^T\mathbf{y} + b)$, x, a, b jsou konst, y proměnná a λ multiplikátor

Podmínka: $a^Ty + b = 0$

Derivace L podle y = $-2(x-y) + \lambda a = 0$, odtud:

$2(x-y) = \lambda a$

[pokračování od Káji Balagazové]

Dosadíme $y = x - \lambda a/2$ do $a^Ty+b=0$ a dostaneme lambdu:

$\lambda = 2\ (a^Tx+b)/(a^Ta)$

a vysledna vzdalenost je:

$||x-y|| = ||x - x + \lambda a/2|| = |\lambda/2|*||a|| = (|a^Tx+b| / ||a||^2) * ||a|| = |a^Tx+b| / ||a||$

[Od Kačky Mackové:]

Jde nad tím přemýšlet i z úhlu pohledu lineární algebry. Rozhodovací nadrovina je dána normálovým vektorem a a posunem b. Hledaná vzdálenost je délka projekce bodu x na tento normálový vektor posunutý o b. Pak není třeba počítat lagrange. Vektor a vydělíme ||a||, aby byl jednotkový. Promítneme na něj bod x:

$proj(x) = a^* (a^Tx+b) / ||a||^2$ a máme projekci.

Projekci znormujeme a máme:

$||proj(x)|| = ||a||*|a^Tx+b| / ||a||^2 = |a^Tx+b| / ||a|| = ||x-y||$

b)

Máme $x \in R^D$ a $k \in \{1, -1\}$. Definujeme $x' = k_i*(x_i, 1) \in R^{D+1}$ a $w = (a, b) \in R^{D+1}$

Pak $w^Tx' = k_i(a^Tx_i + b) >= 0$ iff $sign(a^Tx_i + b) = k_i$ tj. iff $q(x) = k_i$ neboli q klasifikuje správně

Problem 6.2

a)

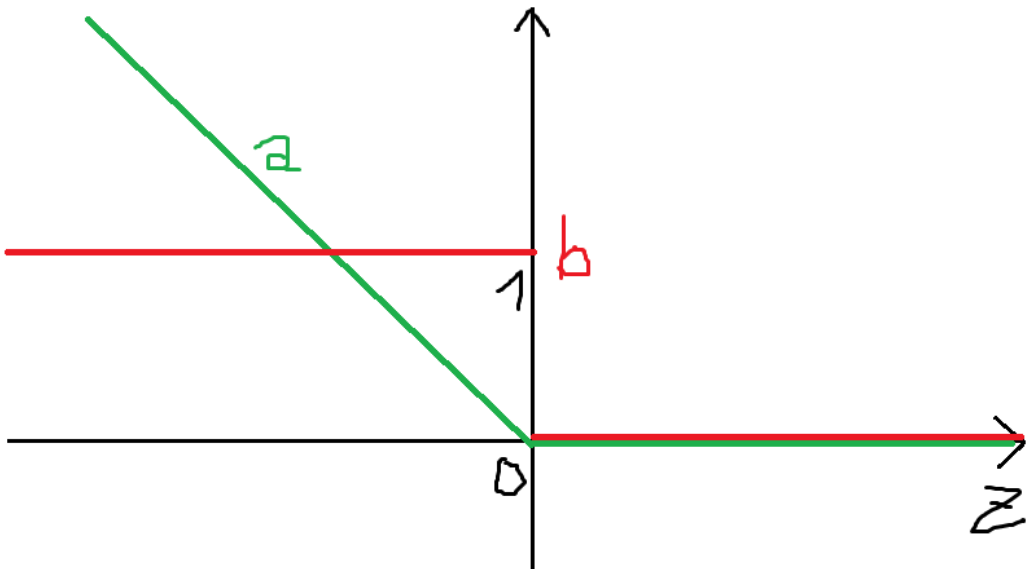
max(-z, 0) (zeleně)

b)

$R_{emp}(q(x)) = W(q(x), k) = 1$ iff $q(x) != k$ (tj. iff $z < 0$), 0 else (červeně)

c)

logaritmus ze cvičení 5.1c)



Problem 6.3

a)

Přepíšeme si:

$l_i(w) =$ $-w^T x_i$ iff x_i je špatně ohodnocený

0 iff x_i je správně ohodnocený

$\nabla_w l_i(w) =$ $-x_i$ iff x_i je špatně ohodnocený

0 iff x_i je správně ohodnocený

Takže správně ohodnocený bod $w^{t+1} = w^t + 0$ nezmění a špatně ohodnocený updatuje $w^{t+1} = w^t + \varepsilon * x_i$

b)

Intuice říká, že na velikosti výsledného vektoru w nezáleží, jde pouze o jeho směr. Výsledný vektor w = ε*(lineární kombinace vektorů x), po vydělení ε je směr stále stejný. Násobením ε (ε > 0) se sice změní velikost $w^T x_i$, ale znaménko zůstane stejné (a znaménko je to, co nás zajímá).

Máme $w^0 = 0$, hledáme x_i: $w^{0T} x_i <= 0$. To je hned pro první x_1.

Updatujeme $w^1 = 0 + \varepsilon * x_1 = \varepsilon * x_1$. Hledáme x_i: $w^{1T} x_i = \varepsilon * x_1^T x_i <= 0$. ε je ale kladná konstanta, proto lze nerovnost vydělit bez změny znaménka, neboli na ε nezáleží.

Problem 6.4

Upravíme si T na T' = {(2, -1, −1),(0, 0, −1),(0, 2, 1),(0, 3, −1),(2, 2, 1)}

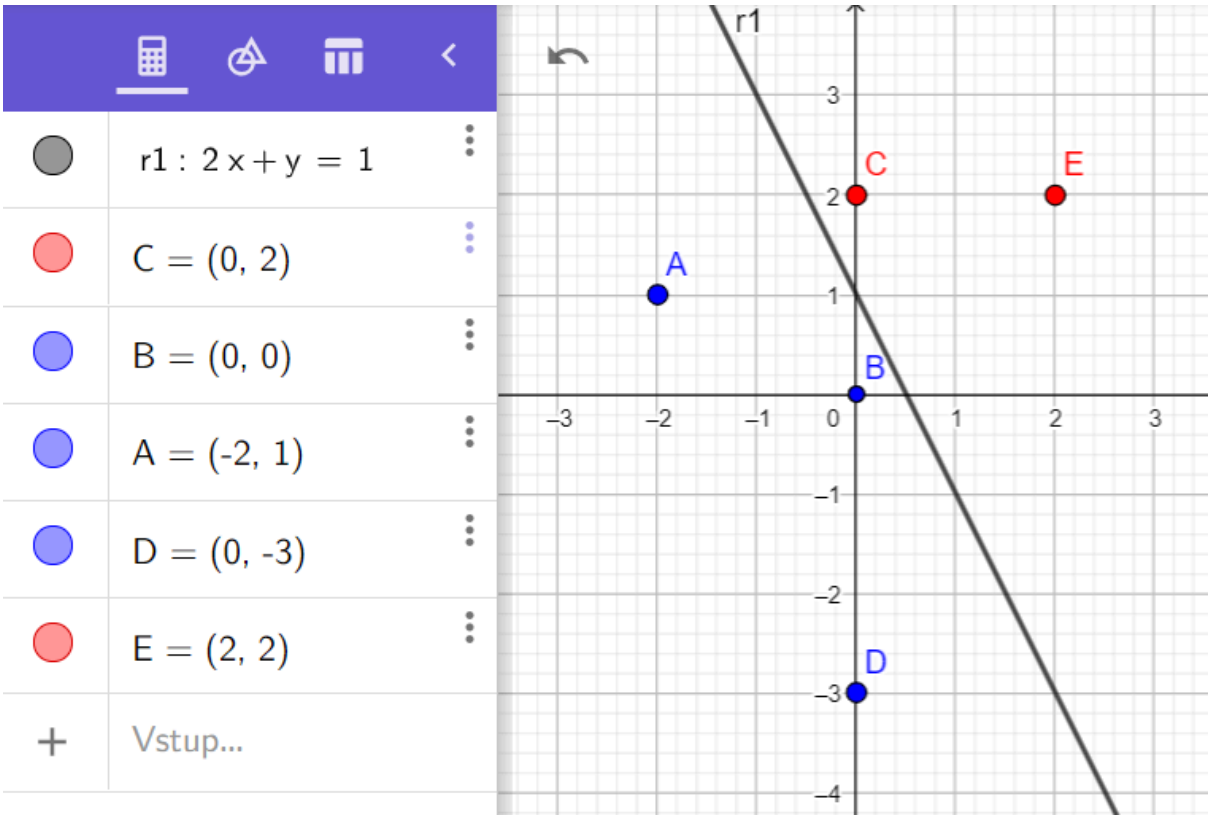
Pracujeme pouze se skalárním součinem $w^T x$ (bez b), chceme $w^T x >= 0$ pro všechny body, inicializujeme $w_0 = (0, 0, 0)$ a sekvenčně procházíme množinu T.

$w_0 = (0, 0, 0)$ - chyba pro bod A

$w_1 = (2, -1, -1)$ - chyba pro bod C

$w_2 = (2, 1, 0)$ - chyba pro bod B

$w_3 = (2, 1, -1)$ - všechny body jsou správně ohodnoceny => w = (2, 1), b = -1



Problem 6.5

Základní krok (t = 0):

$||w^0||^2 = 0 <= 0*\max ||x_i||^2 = 0$

Indukční krok:

Indukční předpoklad: $||w^t||^2 <= t*\max ||x_i||^2$ (chceme dokázat platnost pro t+1)

$||w^{t+1}||^2 = ||w^t + x^t||^2 = (w^t + x^t)^T(w^t + x^t) = w^t w^t + 2*w^t x^t + x^t x^t = ...$

[pokračování od Káji Balagazové]

$\ldots = ||w^t||^2 + 2w^t x^t + ||x^t||^2 \leq_1 ||w^t||^2 + ||x^t||^2 \leq_2 ||w^t||^2 + \max||x||^2 \leq_3 t*\max||x||^2 + \max||x||^2 =_4 = (t + 1)*\max||x||^2$

$\leq_1$ .. x^t je spatne klasifikovany vektor, tzn $w^t x^t <= 0$, takze se muzeme zbavit $2*w^t x^t$

$\leq_2$ .. protoze $||x_i||^2 <= \max_i ||x_i||^2$

$\leq_3$ .. podle indukčního předpokladu $||w^t||^2 <= t*\max||x_i||^2$

$=_4$ .. a staci jenom vytknout $\max||x||^2$ a mame dokazano, ze $||w^{t+1}||^2 \leq (t + 1)*\max||x||^2$

Problem 7.1

a)

Řešení je jednoduché zamyšlení:

Minimalizujeme výraz, kde je ξ v kladném násobku, neboli ho chceme pro minimalizaci co nejmenší. Máme ale omezení ξ >= 0 a $(w^Tx + b)y >= 1 - ξ$. Zvolíme tedy nejmenší takové ξ >= 0, aby $(w^Tx + b)y >= 1 - ξ$ platilo.

Formálněji:

$\min_{w,b,e}$ ½ $||w||^2$ + C*Sum_i $ξ_i$ = $\min_{w,b,e}$ F(w,b,e) za podmínky $(w^Tx_i + b)y_i >= 1 - ξ_i$, $ξ_i >= 0$

Derivace F podle $ξ_i$ = C, kde C je kladná konstanta určující ztrátu za špatně ohodnocené x_i

Pro minimalizaci volíme nejmenší $ξ_i$, které splňuje podmínky.

$ξ_i = 0$ iff $(w^Tx_i + b)y_i >= 1$ (tj. iff x_i je správně ohodnoceno)

$ξ_i = 1 - (w^Tx_i + b)y_i$ iff $(w^Tx_i + b)y_i < 1$ (tj. iff x_i je špatně ohodnoceno)

Lze to případně řešit i opravdu graficky jako průnik polorovin $ξ_i >= 0$ a $ξ_i >= 1 - (w^Tx_i + b)y_i$. Na těchto polorovinách hledáme nejmenší hodnotu funkce K + C*$ξ_i$, kde K je konstantní součet zbytku původní funkce K = ½ $||w||^2$ + C*$\text{Sum}_{j \text{ != } i}$ $ξ_j$.

b)

- Přechod od vzorce a) do b):

Máme $\min_{w,b,e}$ ½ $||w||^2$ + C*Sum_i $ξ_i$. C je konstanta, hledáme argmin, hodnota min nás nezajímá, proto může vydělit. Získáme:

$\min_{w,b,e}$ 1/(2*C) $||w||^2$ + Sum_i $ξ_i$

Za $ξ_i$ dosadíme nejmenší hodnotu tak, aby platily podmínky $(w^Tx_i + b)y_i >= 1 - ξ_i$, $ξ_i >= 0$ (viz a)). Tahle hodnota se dá kompaktněji napsat jako max(1 - $(w^Tx_i + b)y_i$, 0). Tím se zbavíme proměnné $ξ_i$. Získáme:

$\min_{w,b,e}$ 1/(2*C) $||w||^2$ + Sum_i max(1 - $(w^Tx_i + b)y_i$, 0)

- Přechod od vzorce a) do b) [od Vitka Lupinka]:

Taky se dá hezky vyřešit úvahou/graficky jako a) máme tam ξ >= 0 a $(w^Tx + b)y >= 1 - ξ$, což když vyplotíme, tak je vlastně obrázek c) a z něj je hezky vidět, že $ξ_i$ = max(1 − $(w^Tx_i + b)y_i$, 0) tudíž to můžu udělat pro všechny a dostanu: $\min_{w,b}$ 1/2C*$||w||^2$ + Sum_i(max(1 − $(w^Tx_i$ + b)*y_i, 0)).

- Hledání minima:

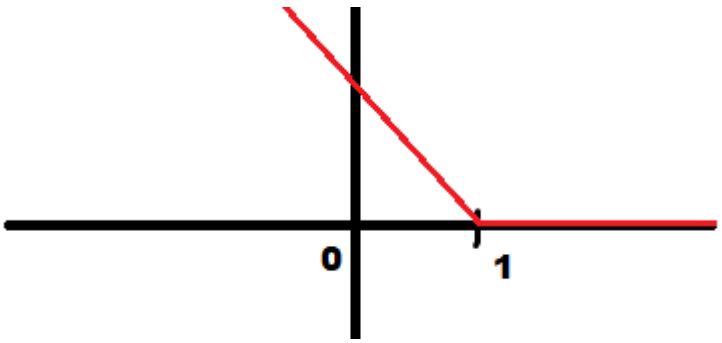
Řekl bych, že si pro každé w a b můžeme vypočítat gradient a použít tedy iterační metody optimalizace:

F = min $||w||^2$/(2*C) + Sum(přes špatně ohodnocené i) (1 - $(w^Tx_i + b)y_i$)

F derivace podle w = w/C - Sum(přes špatně ohodnocené i) y_ix_i = 0

F derivace podle b = Sum(přes špatně ohodnocené i) $-y_i$ = 0

c)



Z tohohle plyne, že i body na správné straně od rozdělující nadroviny, ale blíže než 1, jsou považovány za špatně ohodnocené. U Perceptronu jsou za špatně ohodnocené považovány opravdu pouze z = $(w^Tx_i + b)y_i$ < 0. U logistické regrese je nenulová ztráta pro všechny z.

d) ?????

Podle mě ne, když C → ∞, tak vypadne první člen a máme

min Sum$_i$ max(1 - ($w^T x_i$ + b)y_i, 0),

což má smysl u neseparabilních dat, chceme najít oddělující přímku, tak aby byly všichni správně ohodnocení nebo když už špatně, tak co nejblíž.

Perceptron v takové situaci nic neříká, nefunguje.

e)

IF Vybraný bod je správně ohodnocený:

$F = ||w||^2/(2nC) + 0$

$F'_w = w/(nC)$

$F'_b = 0$

IF Vybraný bod je špatně ohodnocený:

$F = ||w||^2/(2nC) + 1 - (w^T x_i + b)y_i$

$F'_w = w/(nC) - x_i y_i$

$F'_b = -y_i$

Funkce roste nejstrměji ve směru gradientu, jdeme v opačném směru s velikostí kroku ε:

$w_{t+1} = w_t - \varepsilon * F'_w$

$b_{t+1} = b_t - \varepsilon * F'_b$

Problem 7.2

a)

[řešení od Davida Krause]

Víme $n = w/||w||$, $w^T x_0 + b = 0$ neboli $w^T x_0 = -b$

$n^T(x - x_0) = w^T(x - x_0)/||w|| = (w^T x - w^T x_0)/||w|| = (w^T x + b)/||w||$

To je vzdálenost bodu od přímky (na jednu stranu plus, na druhou mínus). Když to celé vynásobím y (+/- 1) a bod bude správně klasifikován, výsledek bude kladný.

b)

$w^*, b^* = \text{argmax}_{w,b} \min_{x,y} (w^T x + b)y / ||w||$ za podmínky $(w^T x + b)y / ||w|| >= 0$ (dělení ||w|| v podmínce není nutné, protože ||w|| nezmění znaménko.

c)

- To je tak jasný, až nevím jak to dokazovat. Asi takhle:

1) $\min_i d_i <= d_{max}$: kdyby $\min_i d_i > d_{max}$, tak lze d_{max} zvětšit, tedy d_{max} není max(d: d <= d_i)
2) $\min_i d_i >= d_{max}$: kdyby $\min_i d_i < d_{max}$, tak neplatí podmínka v max(d: d <= d_i)

Proto $\min_i d_i = d_{max}$.

- Přechod k nové formulaci:

$w^*, b^* = \text{argmax}_{w,b} \min_{x,y} (w^T x + b)y / ||w||$ za podmínky $(w^T x + b)y / ||w|| >= 0$ pro $\forall (x, y)$

Přidáme novou proměnnou d:

$\text{argmax}_{w,b,d}$ d za podmínky $(w^T x + b)y / ||w|| >= 0$ pro $\forall (x, y)$ a $d <= \min_{x,y} (w^T x + b)y / ||w||$

$\text{argmax}_{w,b,d}$ d za podmínky $(w^T x + b)y / ||w|| >= 0$ a $d <= (w^T x + b)y / ||w||$ pro $\forall (x, y)$

$\text{argmax}_{w,b,d}$ d za podmínky $0 <= d <= (w^T x + b)y / ||w||$ pro $\forall (x, y)$

d)

No jasný, zajímá nás znaménko výrazu $w^T x + b$, které se nezmění vynásobením kladnou konstantou. Přesněji:

$(w^T x + b)y / ||w|| >= d >= 0$, substituujeme:

$(s * w^T x + s * b)y / \sqrt{(sw^T sw)} >= d >= 0$

$s*(w^T x + b)y / |s|*||w|| >= d >= 0$ (s > 0)

$(w^T x + b)y / ||w|| >= d >= 0$

e)
$\max_{w,b}$ 1 / ||w|| za podmínky $(w^T x + b)y >= 1$ pro $\forall (x, y)$
f)
Protože ||w|| >= 0, tak **argmax 1/||w||** je pro **argmin ||w||** a **argmin ||w||²** je pro **argmin ||w||**

Problem 7.3
a)
$\min_{w,b,\xi} \max_{\alpha}$ ½$w^T w$ + C1$^T\xi$ −$(w^T X + by^T - 1^T + \xi^T)\alpha$ za podmínky ξ >= 0, α >= 0
Význam:
Pokud jsou všechny body správně ohodnoceny (tj. $w^T X + by^T - 1^T + \xi^T >= 0$), tak tak $-(w^T X + by^T - 1^T + \xi^T) = -G$ je záporné, my chceme **max -G*α**, kde jsou v tu chvíli α násobeny zápornými koeficienty, proto je max pro minimální α, neboli pro α = 0, a tedy -G*α = 0
Pokud je nějaký bod x_i špatně klasifikování, $-G_i*\alpha > 0$ a tedy lze maximalizovat maximalizací $\alpha_i = \infty$, proto -G*α = ∞
b) ?????
Obecne plati: $\max_a \min_b <= \min_b \max_a$
Mame: $\max_{\alpha} \min_{w,b,\xi}$ ½$w^T w$ + C1$^T\xi$ − $(w^T X + by^T - 1^T + \xi^T)\alpha$ za podmínky ξ >= 0, α >= 0
Řešíme vnitřní $\min_{w,b,\xi} f(w,b,\xi) = \min_{w,b,\xi}$ ½$w^T w$ + C1$^T\xi$ − $w^T X\alpha$ − $by^T\alpha$ + $1^T\alpha$ − $\xi^T\alpha$
w:
Transponujeme a pak derivace f podle $w = w^T - \alpha^T X^T = 0$ a odtud w = Xα
ξ:
Dáme k sobě členy C1$^T\xi$ − $\xi^T\alpha$ = C1$^T\xi$ − $\alpha^T\xi$ = $(C1 - \alpha)^T\xi$
$\min_{\xi >= 0} (C1 - \alpha)^T\xi$, kde $(C1 - \alpha)$ jsou vlastně koeficienty pro ξ
Pokud $(C1 - \alpha) >= 0$, minimalizujeme výraz minimální ξ, neboli ξ = 0, tak $(C1 - \alpha)^T\xi = 0$
Pokud existuje $(C - \alpha_i) < 0$, lze dosáhnout -∞ tím, že ξ_i -> ∞
Celkově chceme ale max min f, proto nechceme -∞, neboli nám to dává omezení α <= C
b:
Protože b není omezené, pokud jeho koeficient $y^T\alpha$ není nula, lze člen $y^T\alpha b$ minimalizovat volbou b = + nebo -∞ k hodnotě -∞. Zase ale chceme celkově max min, proto získáváme podmínku $y^T\alpha = 0$
Teď v $\max_{\alpha} \min_{w,b,\xi}$ ½$w^T w$ + C1$^T\xi$ − $w^T X\alpha$ − $by^T\alpha$ + $1^T\alpha$ − $\xi^T\alpha$ za podmínky ξ >= 0, α >= 0 nahradíme w = Xα, místo C1$^T\xi$ − $\xi^T\alpha$ přidáme podmínku α <= C a místo $by^T\alpha$ přidáme podmínku $y^T\alpha = 0$. Získáme:
$\max_{\alpha}$ ½$\alpha^T X^T X\alpha$ − $\alpha^T X^T X\alpha$ + $1^T\alpha$ za podmínky α >= 0, α <= C, $y^T\alpha = 0$, neboli
$\max_{\alpha}$ -½$\alpha^T X^T X\alpha$ + $1^T\alpha$ za podmínky 0 <= α <= C, $y^T\alpha = 0$

Problem 7.4
a)
Vliv mají ty X_i, kde $\alpha_i > 0$
b)
Když je někde ostrá nerovnost, existuje dostatečně malé delta, aby ostrá nerovnost platila i po změně, to je jasné.
Vztah s α asi:
V primární úloze bod (x, y) ostře splňuje $y(w^T x + b) > 1$. Duální úloha je
$\text{argmax}_a \min_{w,b} (½ ||w||^2 - \text{Sum}_i a_i(y_i(w^T x_i + b) - 1))$, kde pro náš bod je $y(w^T x + b) - 1 = D > 0$.
Příslušná alfa a tedy je odčítána jako -D*a. Chceme argmax_a, proto je tato alfa a = 0.

Problem 7.5

a)

$\max_a \text{Sum}_i (a) - ½ \text{Sum}_{i,j} (a_i a_j y_i y_j \varphi(x_i)\varphi(x_j))$ za podmínky $\text{Sum}_i a_i y_i = 0$, $0 <= a <= C$

b)

To asi ne, pokud třeba uvažujeme nějaké crazy $K(x, y) = \sin(x_1^5 y_1^5 + \text{sqrt}(x_1 y_1))^{x2y2}$, tak asi nějaký předpis $\varphi(x)\varphi(y)$ nenajdeme. Může nás spasit možná Taylorův rozvoj, ale ten má odchylku nebo to je nekonečná posloupnost. Kernel funkce může být prostě každá pozitivně semidefinitní funkce $R^D x R^D \rightarrow R$.

c)

Pro bod z máme $q(z) = \varphi(w)^T \varphi(z) + b = \text{Sum}_{i\ sv} a_i y_i \varphi(x_i)^T \varphi(z) + b = \text{Sum}_{i\ sv} a_i y_i K(x_i, z) + b$, kde b předpočítáme (sv = support vector), protože víme:

$y_{sv}(\varphi(w)^T \varphi(x_{sv}) + b) = y_{sv}(\text{Sum}_{i\ sv} a_i y_i K(x_i, x_{sv}) + b) = 1$ pro nějaký support vector x_{sv}, a odtud máme:

$\text{Sum}_{i\ sv} a_i y_i K(x_i, x_{sv}) + b = y_{sv}$ (protože $y_{sv} = +-1$)

$b = y_{sv} - \text{Sum}_{i\ sv} a_i y_i K(x_i, x_{sv})$

Problem 7.6

a)

$K(x, y) = 1 + x_1 y_1 + x_2 y_2 + x_1^2 y_1^2 + x_2^2 y_2^2 + x_1 x_2 y_1 y_2$

b)

$K(x, y) = (1 + x_1 y_1 + x_2 y_2)^2 = 1 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 x_2 y_1 y_2 + x_1^2 y_1^2 + x_2^2 y_2^2$

Odtud máme $\varphi(x) = (1, \sqrt{(2)}x_1, \sqrt{(2)}x_2, \sqrt{(2)}x_1 x_2, x_1^2, x_2^2)$

c)

ano, šlo by to, získáme polynom 2D proměnných řádu 2d, mapování bude do dimenze k, kde k = počet členů polynomu, každá dimenze mapování bude obsahovat odmocninu konstanty * součin všech prvků jednoho vektoru v daném členu

Výpočet kernel function (získání hodnoty) je cca D násobení ($x_1 * y_1$, …, $x_D * y_D$) + D sčítání ($1 + x_1 * y_1 + … + x_D * y_D$) + d násobení (mocnění na d).

Pro získání mapování potřebujeme získat předpis kernel function:

D násobení a získáme ($1 + x_1 * y_1 + … + x_D * y_D$), což má D+1 členů. Při umocnění na druhou uděláme (D+1)(D+1) násobení a vznikne člen s D(D+1) členy (myslím), takže další násobení původním ($1 + x_1 * y_1 + … + x_D * y_D$) (tj., celkově umocnění na třetí) je D(D+1)(D+1) násobení a vznikne výraz s D*D*(D+1) členy atd. Takže poslední krok mocnění máme asi $D^{d-1}(D+1)(D+1)$ násobení plus (nepočítal jsem sčítání).

Výpočet mapování pak je odmocnění konstant u členů ve výsledném výrazu s $D^d(D+1)$ členy a rozdělení částí x a y v každém členu.

Nechce se mi nad počty operací přemýšlet nějak do hloubky, prostě kvůli tomu že pro získání mapování potřebujeme předpis a tak nemůžeme po každém krok členy sečíst do jednoho skaláru, nám přibude strašně moc operací. Asi :D

Problem 8.1

a)

chyba je když sign(f(x)) != y, což je iff

f(x) < 0 a y = 1

f(x) > 0 a y = -1

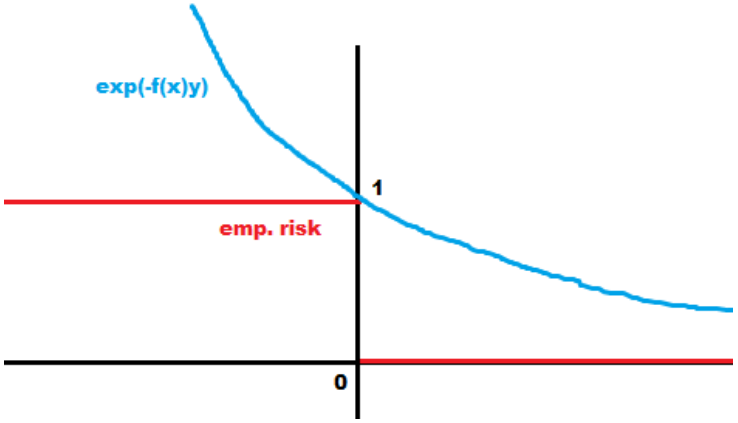
=> f(x)*y < 0

Ty divný závorky dávají 1 iff výraz uvnitř je true, jinak 0, takže to sečte všechny chyby. Přijde mi jen divný, že to není dělený velikostí trénovací množiny, v přednášce to bylo násobené $D_t(i)$, což bylo inicializované na 1/N.

b)

Když f(x)y <= 0 tak emp. risk je 1, ale -f(x)y >= 0, takže $e^{-f(x)y} >= 1$

Když f(x)y > 0 tak emp. risk je 0, ale $e^{-f(x)y} > 0$ platí vždy



Výhodou může být, že i správně ohodnocené body mají přidělenou nenulovou chybu, takže se je algoritmus stále snaží ohodnotit “lépe”, neboli dostat dál od oddělující hranice. Zároveň je to spojitá, diferencovatelná funkce.

c)

Máme mapování h(x): R^D -> R^T a chceme najít lineární klasifikátor takto mapovaných bodů.

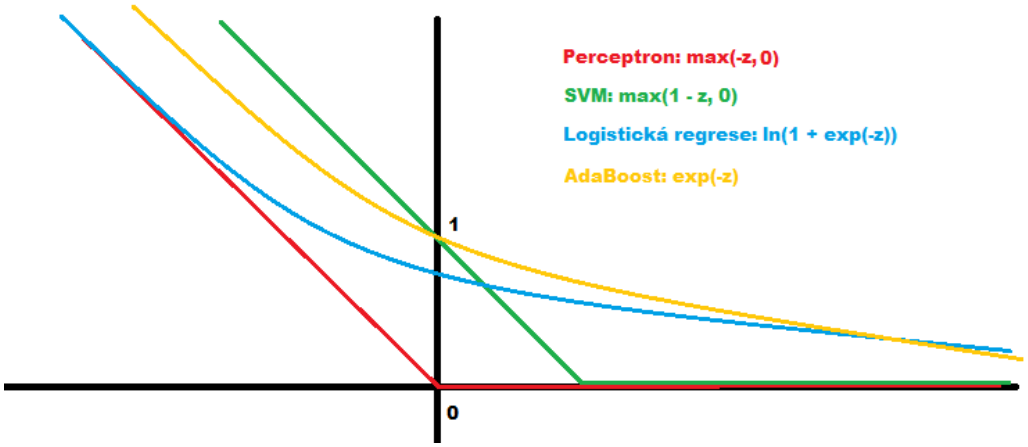
Tzn. hledáme koeficienty a z R^T, tak aby chyba na trénovacích datech $\{(x_i, y_i)\}_{1 <= i <= N}$

$R_{emp} = Sum_{1 <= i <= N}\ e^{-a*h(x)*y}$ byla minimální => $argmin_a\ R_{emp}$

SVM: $R_{emp} = ||w||^2/(2C) + Sum_{1 <= i <= N}\ max(1 - (w^T x_i + b)y_i, 0)$ => $argmin_{w,b}\ R_{emp}$

Perceptron: $R_{emp} = 1/N\ Sum_{1 <= i <= N}\ max(-w^T x_i, 0)$

LogReg: $R_{emp} = ln(1 + e^{-ki*w*xi})$



Problem 8.2

a) ?????

$R = Sum_{1 <= i <= N} e^{-Sum(1 <= j < t) aj*hj(xi)*yi - at*ht(xi)*yi} = Sum_i D_t(i) * e^{- at*ht(xi)*yi}$

$e^{-Sum(1 <= j < t) aj*hj(xi)*yi} = D_t(i) > 0$, ale nevím jak dokázat to sčítání do 1

Problem 8.3

a)

$min_{at, ht} Sum_{1 <= i <= N} D_t(i)*e^{-at*ht(xi)*yi} = min_{at, ht} Sum_{i: ht(x) != yi} D_t(i)*e^{at} + Sum_{i: ht(x) = yi} D_t(i)*e^{-at}$

b)

$Sum_{špatně\ i} D_t(i)*e^{at} + Sum_{dobře\ i} D_t(i)*e^{-at} =$

$= Sum_{špatně\ i} D_t(i)*e^{at} + Sum_{všechny\ i} D_t(i)*e^{-at} - Sum_{špatně\ i} D_t(i)*e^{-at} =$

$= e^{-at} * Sum_{všechny\ i} D_t(i) + (e^{at} - e^{-at}) * Sum_{špatně\ i} D_t(i)$

c)

Pro $a > 0$ je $e^{at} - e^{-at} > 0$. Předchozí vyjádření závisí na h_t pouze ve členu $(e^{at} - e^{-at}) * Sum_{špatně\ i} D_t(i)$, proto pokud chceme výraz minimalizovat přes h_t, musíme minimalizovat $Sum_{špatně\ i} D_t(i)$ (protože to má kladný koeficient $e^{at} - e^{-at}$).

d)

Derivace podle $a = e^{-at} * (-1) * Sum_{všechny\ i} D_t(i) + (e^{at} * 1 - e^{-at} * (-1)) * Sum_{špatně\ i} D_t(i) =$

$-e^{-at} * Sum_{dobře\ i} D_t(i) + e^{at} * Sum_{špatně\ i} D_t(i) = -e^{-at} * (1 - \varepsilon_t) + e^{at} * \varepsilon_t = 0$

$e^{-at} * (1 - \varepsilon_t) = e^{at} * \varepsilon_t$

$(1 - \varepsilon_t) / \varepsilon_t = e^{2at}$

$2a_t = ln((1 - \varepsilon_t) / \varepsilon_t)$

$a_t = ½ * ln((1 - \varepsilon_t) / \varepsilon_t)$

e)

$a_t = ½ * ln((1 - \varepsilon_t) / \varepsilon_t) > 0$ iff $ln((1 - \varepsilon_t) / \varepsilon_t) > 0$ iff $(1 - \varepsilon_t) / \varepsilon_t > 1$ iff $\varepsilon_t < ½$

Problem 8.4

1) Inicializujeme váhy všech n = 9 bodů na $D_1(i) = 1/n = 1/9$ pro všechny body
2) nalezneme klasifikátor s nejmenší chybou $\varepsilon_1 = Sum_{špatně\ i} D_1(i)$
   Tipuju, že to bude třeba $h_1 = sign(1*x - 3.5)$, neboli oddělí body mezi 3 a 4, pak $\varepsilon_1 = 2/9$
3) Váha toho klasifikátoru a_1 je $½ * ln((1 - \varepsilon_1)/\varepsilon_1) = ½ * ln(7/2) = 0{,}6264 = ln(sqrt(7/2))$
4) Přepočítáme váhy $D_2(i) = D_1(i) * e^{-a1*h1(xi)*yi} / Z_1$, kde $Z_1 = Sum_{1 <= j <= 9} D_1(j) * e^{-a1*h1(xj)*yj}$
   Tj. pro špatně ohodnocené body 0 a 6:
   $D_1(j) * e^{-a1*h1(xj)*yj} = 1/9 * e^{0{,}6264} = 1/9 * sqrt(7/2) = 0{,}208$
   A pro správně ohodnocené body:
   $D_1(j) * e^{-a1*h1(xj)*yj} = 1/9 * e^{-0{,}6264} = 1/9 * sqrt(2/7) = 0{,}06$
   => $Z_1 = 2 * 0{,}208 + 7 * 0{,}06 = 0{,}836$
   A pak pro špatně ohodnocené body 0 a 6: $D_2(j) = 0{,}208 / 0{,}836 = 0{,}2488$
   A pro správně ohodnocené body: $D_2(j) = 0{,}06 / 0{,}836 = 0{,}0718$
   Lze si výpočet zjednodušit rovností $Z_t = 2*\sqrt{((1 - \varepsilon_1)*\varepsilon_1)}$
5) Pokračovalo by se znovu krokem 2)

Problem 9.1

a) Backpropagation - 5 - Computationally efficient automatic differentiation for scalar-valued composite functions.
b) Gradient - 3 - Is necessary to find a step direction for gradient descent.
c) Chain rule - 4 - A rule to compute gradient of composite functions.
d) Training loss minimization - 1 - A way to learn neural networks.
e) SGD (stochastic gradient descent) - 2 - Method to optimize training loss.

Problem 9.2

a) Gradient of f - 3 - Column vector of partial (or total) derivatives in case f is scalar-valued, i.e. m = 1.
b) Derivative of f - 1 - A linear mapping approximating f locally around a point.
c) Jacobian of f - 2 - Expression of the derivative in coordinates as a matrix.

Problem 9.3

Jakože po roce od MA2 si představovat, jak vlastně vypadají derivace ve vyšších dimenzích, je fakt záhul, fuj...

a)

$y(x) = Wx$: $R^D \to R^M$, $L(y)$: $R^M \to R^1$, $L(y(x))$: $R^D \to R^1$, takže matice $\partial L/\partial x$ bude z R^{1*D}

$\partial L/\partial x_i = \text{Sum}_j\ \partial L/\partial y_j * \partial y_j/\partial x_i = \text{Sum}_j\ \partial L/\partial y_j * W_{j,i}$

$\partial L/\partial x = \text{Sum}_j\ \partial L/\partial y_j * \partial y_j/\partial x = \text{Sum}_j\ \partial L/\partial y_j * W_{j,:} = \partial L/\partial y * W$, kde $\partial L/\partial y$ je z R^{1*M} (řádkový vektor) a W je z R^{M*D}

b)

Gradient je D^T. $D = \partial L/\partial y * W \to D^T = (\partial L/\partial y * W)^T = W^T * (\partial L/\partial y)^T = W^T * \text{Grad}_y L$

c) ?????

Bereme teď $L(W)$: $R^{M*D} \to R$

$\partial L/\partial W_{i,j} = \partial L/\partial y_i * \partial y_i/\partial W_{i,j} = \partial L/\partial y_i * x_j$

Protože je obecně derivace funkce $R^a \to R^b$ matice z R^{a*b} (ne R^{b*a}), tak v matici $D = \partial L/\partial W$ bude na pozici $D_{i,j}$ hodnota $\partial L/\partial W_{j,i}$. Získáme:

$D = \partial L/\partial W = x * \partial L/\partial y$, kde x je z R^{D*1} a $\partial L/\partial y$ je z R^{1*M}, tedy D je z R^{D*M}.

$\text{Grad}_W L = (\partial L/\partial W)^T = \text{Grad}_y L * x^T$ z R^{M*D}. Snad.

Tady je hezky vysvětlené: http://cs231n.stanford.edu/handouts/linear-backprop.pdf

Problem 10.1

a)

máme fixní $\{c_k\}_{k=1,...,K}$ => sumu minimalizujeme minimalizací jejích sčítanců => pro všechny body x_i nalezneme nejbližší c_k: $r(i) = \min_{k=1,...,K} ||x_i - c_k||^2$

b)

máme fixní dělení na clustery => sumu vzdáleností od center ve všech clusterech minimalizujeme minimalizací vzdáleností od centra v jednotlivých clusterech => v každém clusteru k uděláme: $c_k = \text{argmin}_{c \text{ z } R\text{^}D} \text{Sum}_{i:\ r(i) = k} ||x_i - c_k||^2$, derivací získáme
$c_k = (\text{Sum}_{i:\ r(i) = k}\ x_i) / (\text{Sum}_{i:\ r(i) = k}\ 1)$, což je těžiště clusteru
Existuje-li centrum k takové, že r(i) != k pro žádné i (neboli není mu přidělen žádný bod), pro jeho vrácení do koloběhu minimalizace ho reinicializujeme

c)

V K-means se střídají tyto dva kroky.
V prvním kroku si žádný bod nemůže pohoršit (zvětšit svou vzdálenost od nejbližšího centra vzhledem ke stavu před tímto krokem), tedy se nemůže zhoršit ani celková suma.
V druhém kroku se nemůže zhoršit součet vzdáleností v žádném clusteru, tedy se nemůže zhoršit ani celková suma (jednotlivé konkrétní body si pohoršit můžou, ale globálně ne).
Následující krok K-means tedy nemůže mít horší sumu, než předchozí. Pokud je lepší, pokračujeme, pokud je stejná, nic se nezměnilo (pokud je konzistentní rozhodování v případě rovnosti) a algoritmus končí v (lokálním) minimu => nejde se zacyklit.
Celkových možných stavů je N^K, kde N i K jsou konečné, tedy algoritmus musí skončit v konečném počtu kroků.

Problem 10.2 ?????

Intuice říká, že čím menší je suma vzdáleností mezi dvojicemi bodů, tím menší je suma vzdáleností od těžiště (body jsou prostě “blíž” u sebe). Ale jak to matematicky ukázat na tom vzorci nevím.

Problem 10.3

a)

Hledáme pozice studní $c_k = (x_k, y_k)$, $k = 1,...,K$. Jako $d(x, y)$ použijeme euklidovskou vzdálenost (L2 normu bez kvadrátu):

$\{c_k^*\} = \text{argmin}_{\{ck\} z R^{K*2}} \text{Sum}_{i = 1,...,N} \min_k ||p_i - c_k||$

K-medians by používaly Manhattanskou L1 normu:

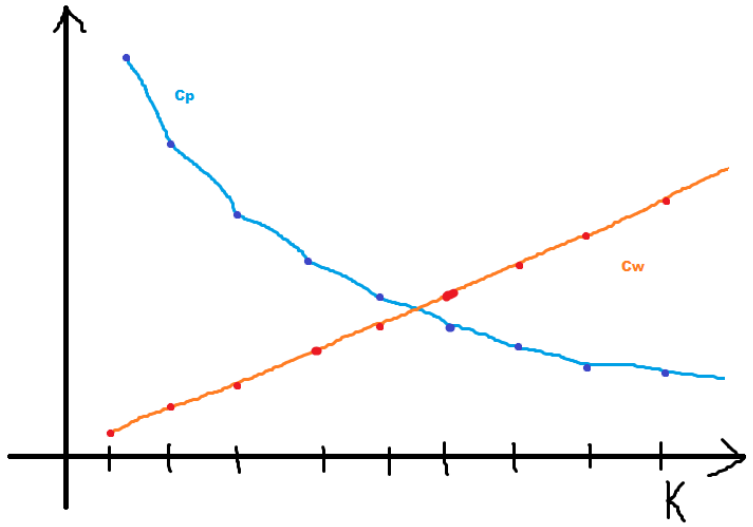
$\{c_k^*\} = \text{argmin}_{\{ck\} z R^{K*2}} \text{Sum}_{i = 1,...,N} \min_k ||p_i - c_k||_1 =$

$\text{argmin}_{\{ck\} z R^{K*2}} \text{Sum}_{i = 1,...,N} \min_k (|p_{i1} - c_{k1}| + |p_{i2} - c_{k12}|)$

b)

Nyní musíme brát v potaz kombinaci klesající ceny za potrubí a rostoucí ceny za studně.

Formálně: $K^* = \text{argmin}_{K z \{1,2,..\}.} (K*c_w + \text{argmin}_{\{c1, ..., cK\}} \text{Sum}_{i = 1..N} \min_{j z \{1, ..., K\}} d(p_i, c_j)*c_p)$



Vztah c_w vzhledem ke k je lineární. Vztah c_p není lineární, přijde mi, že platí, že každé další zlepšení hodnoty c_p s rostoucím k je menší než předchozí zlepšení (pokud by i+1. iterace našla bod X, který oproti i. iteraci udělá lepší změnu, než udělal bod Y nalezený v i. iteraci oproti i-1. iteraci, tak tento bod X mohl být nalezen již v i. iteraci místo bodu Y a udělat větší zlepšení, tedy neplatí, že Y je optimální řešení v i. iteraci, což je spor, mi přijde, ale případně mě vyveďte z omylu). Proto jakmile poprvé narazíme na iteraci m, ve které se součet $c_w + c_p$ zvětší, předchozí počet studní je ten optimální, tj. $k = m - 1$.

c)

Algoritmus pouze přijme jinou formu vzdálenosti - Manhattanskou vzdálenost (L1 normu):

$\{c_k^*\} = \text{argmin}_{\{ck\} z R^{K*2}} \text{Sum}_{i = 1,...,N} \min_k ||p_i - c_k||_1 =$

$\text{argmin}_{\{ck\} z R^{K*2}} \text{Sum}_{i = 1,...,N} \min_k (|p_{i1} - c_{k1}| + |p_{i2} - c_{k12}|)$

Problem 11.1

Máme $T = \{x_i\}_{i=1..N}$, $p(x|k) = N(\mu_k, I)$, $p(k) = 1/K$, $p(x, k) = p(x|k)p(k) = N(\mu_k, I)/K$, kde K je konstantní, proto ji můžeme v optimalizaci vynechat a brát $p(x, k) = N(\mu_k, I)$

a)

Nejdříve nalezneme pro každý bod x_i jeho odhadovanou třídu k_i:

$k_i = \text{argmax}_k\ p(x_i|k)p(k)/(\text{Sum}_{k'}\ p(x_i|k')p(k')) = \text{argmax}_k\ p(x_i|k) = \text{argmax}_k\ N(x_i \mid \mu_k, I)$

Poté pro každou třídu zvlášť upravíme její μ_k:

$\mu_k = \text{argmax}_\mu\ (\text{Prod}_{i:\ ki\ =\ k}\ 1/\sqrt{(2\pi)^D} * \exp(-½\ (x_i - \mu)^T(x_i - \mu)))$

$l_k(\mu) = \text{Sum}_{i:\ ki\ =\ k}\ (\ln(1/(2\pi)^{D/2}) - ½\ (x_i^T x_i - 2\mu^T x_i + \mu^T\mu))$

$\partial l_k(\mu)/\partial\mu = \text{Sum}_{i:\ ki\ =\ k}\ (-\ x_i^T + \mu^T) = 0$, odtud: $\mu_k = (\text{Sum}_{i:\ ki\ =\ k}\ x_i^T)/(\text{Sum}_{i:\ ki\ =\ k}\ 1)$

b)

Nyní budeme mít pro každý bod váhy $\alpha_i(k) = p(k|x_i)$ pro všechny k.

$\alpha_i(k) = p(x_i|k)p(k)/(\text{Sum}_{k'}\ p(x_i|k')p(k')) = p(x_i|k)/(\text{Sum}_{k'}\ p(x_i|k'))$

Parametry μ_k můžeme upravovat nezávisle pro každou k:

$l_k(\mu) = \text{Sum}_{i=1..N}\ (\alpha_i(k)*\ln(\exp(-½\ (x_i - \mu)^T(x_i - \mu)))) = \text{Sum}_{i=1..N}\ \alpha_i(k)*(-½\ (x_i^T x_i - 2\mu^T x_i + \mu^T\mu))$

$\partial l_k(\mu)/\partial\mu = \text{Sum}_{i=1..N}\ (\alpha_i(k)*(-\ x_i^T + \mu^T)) = 0$

$\text{Sum}_{i=1..N}\ (\alpha_i(k)*x_i^T) = \text{Sum}_{i=1..N}\ (\alpha_i(k)*\mu^T) = \mu^T * \text{Sum}_{i=1..N}\ \alpha_i(k)$

$\mu^T = \text{Sum}_{i=1..N}\ (\alpha_i(k)*\mu^T) / \text{Sum}_{i=1..N}\ \alpha_i(k)$, což je vážený průměr

Problem 12.1

a)

$\text{Sum}_{j=1..M}\ (1/N * \text{Sum}_{i=1..N}\ (u_j^T x_i)^2) = \text{Sum}_{j=1..M}\ (1/N * \text{Sum}_{i=1..N}\ u_j^T x_i u_j^T x_i) =$
$\text{Sum}_{j=1..M}\ (1/N * \text{Sum}_{i=1..N}\ u_j^T x_i x_i^T u_j)$ (protože $u_j^T x_i$ je skalár) =
$\text{Sum}_{j=1..M}\ (1/N * u_j^T * (\text{Sum}_{i=1..N}\ x_i x_i^T) * u_j) = \text{Sum}_{j=1..M}\ u_j^T S u_j$

b)

$\text{Sum}_{i=1..N}\ (x_i - \text{Sum}_{j=1..M}\ u_j^T x_i u_j)^T (x_i - \text{Sum}_{j=1..M}\ u_j^T x_i u_j) =$
$\text{Sum}_{i=1..N}\ (x_i^T x_i - 2*x_i^T * (\text{Sum}_{j=1..M}\ u_j^T x_i u_j) + (\text{Sum}_{j=1..M}\ u_j^T x_i u_j)^T (\text{Sum}_{j=1..M}\ u_j^T x_i u_j)) =$
$\text{Sum}_{i=1..N}\ (x_i^T x_i - 2*(\text{Sum}_{j=1..M}\ x_i^T u_j^T x_i u_j) + (\text{Sum}_{j=1..M}\ \text{Sum}_{h=1..M}\ u_j^T x_i^T u_j u_h^T x_i u_h)) = ...$
$u_j^T x_i$ je skalár, můžeme to vytknout před vektory, máme pak $u_j^T x_i x_i^T u_j = u_j^T x_i u_j^T x_i = (u_j^T x_i)^2$
$x_i^T u_j$ a $u_h^T x_h$ jsou skaláry, vytkneme je před vektory: $x_i^T u_j u_h^T x_i u_j^T u_h$, kde $u_j^T u_h = 1$ pro j = h, a jinak $u_j^T u_h = 0$ (u_i jsou ortonormální), takže získáme $\text{Sum}_{j=1..M}\ x_i^T u_j u_j^T x_i = \text{Sum}_{j=1..M}\ (u_j^T x_i)^2$, tak:
$\ldots = \text{Sum}_{i=1..N}\ (x_i^T x_i - 2*(\text{Sum}_{j=1..M}\ (u_j^T x_i)^2) + (\text{Sum}_{j=1..M}\ (u_j^T x_i)^2) = \text{Sum}_i\ (||x_i||^2 - (\text{Sum}_j\ (u_j^T x_i)^2)$

Problem 12.2

Z 12.1 b) víme, že chceme maximalizovat $\text{Sum}_{j=1..M}\ \text{Sum}_{i=1..N}\ (u_j^T x_i)^2$ a z 12.1 a) víme, že $\text{Sum}_{j=1..M}\ \text{Sum}_{i=1..N}\ (u_j^T x_i)^2 = N * \text{Sum}_{j=1..M}\ u_j^T S u_j$, kde N je konstanta, proto maximalizujeme $\text{Sum}_{j=1..M}\ u_j^T S u_j$, kde chceme ortonormální u_j, což je ekvivalentní té podmínce ve 12.2

a)

$\text{Lagrange}(u) = L(u) = \max_{\{u\}}\ \text{Sum}_{j=1..M}\ u_j^T S u_j - \lambda_1(u_1^T u_1 - 1) - \ldots - \lambda_M(u_M^T u_M - 1)$
$\partial L(u)/\partial u_j = 2u_j^T S - 2\lambda_j u_j^T = 0$, odkud (S = $(\text{Sum}_{i=1..N}\ x_i x_i^T)/N$ je symetrická):
$Su_j = \lambda_j u_j$

b)

Rovnost $Su_j = \lambda_j u_j$ přímo popisuje vlastní čísla a vektory z definice a navíc vlastní vektory splňují ortogonalitu, respektive i ortonormalitu po normalizaci.

c)

Víme $Su_j = \lambda_j u_j$, takže lze maximalizaci přepsat jako:
$\max_{\{uj\}}\ \text{Sum}_{j=1..M}\ u_j^T S u_j = \max_{\{uj\}}\ \text{Sum}_{j=1..M}\ u_j^T \lambda_j u_j = \max_{\{uj\}}\ \text{Sum}_{j=1..M}\ \lambda_j u_j^T u_j = \max_{\{uj\}}\ \text{Sum}_{j=1..M}\ \lambda_j$,
kde tedy λ_j je vlastní číslo vlastního vektoru u_j. Abychom to maximalizovali, bereme tedy M vlastních vektorů patřících k M největším vlastním číslům.

Problem 12.3

Proč vlastně minimalizujeme tohle? Původní úloha byla maximalizovat:
$(\mu_1 - \mu_2)^2/(s_1 + s_2)$, kde μ_k je střední hodnota projektovaných dat třídy k a s_k je rozptyl projektovaných dat třídy k.
$\mu_k = 1/N_k * \text{Sum}_{xi\ \text{třídy}\ k}\ v^T x_i = v^T(1/N_k * \text{Sum}_{xi\ \text{třídy}\ k}\ x_i) = v^T x_k$, kde x_k je těžiště bodů třídy k
$(\mu_1 - \mu_2)^2 = (v^T x_1 - v^T x_2)^2 = (v^T(x_1 - x_2))^2 = v^T(x_1 - x_2)v^T(x_1 - x_2)$, kde $v^T(x_1 - x_2)$ je skalár, lze tedy transponovat, proto:
$(\mu_1 - \mu_2)^2 = v^T(x_1 - x_2)(x_1 - x_2)^T v = v^T S_b v$
$s_k = \text{Sum}_{xi\ \text{třídy}\ k}\ (v^T x_i - v^T x_k)^2 = \text{Sum}_{xi\ \text{třídy}\ k}\ (v^T(x_i - x_k))^2 = \text{Sum}_{xi\ \text{třídy}\ k}\ v^T(x_i - x_k)v^T(x_i - x_k)$, kde $v^T(x_i - x_k)$ je skalár, lze tedy transponovat, proto:
$s_k = \text{Sum}_{xi\ \text{třídy}\ k}\ v^T(x_i - x_k)(x_i - x_k)^T v = v^T(\text{Sum}_{xi\ \text{třídy}\ k}\ (x_i - x_k)(x_i - x_k)^T)v = v^T S_k v$

Úloha se přepíše na:
$\max\ (\mu_1 - \mu_2)^2/(s_1 + s_2) = \max_v\ v^T S_b v/(v^T S_1 v + v^T S_2 v) = \max_v\ v^T S_b v/v^T(S_1 + S_2)v =$
$\max_v\ v^T S_b v/v^T S_w v$
Pokud je S_w regulární, existuje S_w^{-1}, definujeme $z = S_w^{½}v$, a získáme:
$\max\ z^T S_w^{-½} S_b S_w^{-½} z/z^T z$, kde $z^T z = v^T S_w v = 1$, takže máme:

$\max z^T S_w^{-½} S_b S_w^{-½} z$ za podmínky $z^T z = 1$
$L(z) = z^T S_w^{-½} S_b S_w^{-½} z - \lambda(z^T z - 1)$
$\partial L(z)/\partial z = 2*z^T S_w^{-½} S_b S_w^{-½} - 2*\lambda z^T = 0$
$S_w^{-½} S_b S_w^{-½} z = \lambda z$
$S_w^{-½} S_b S_w^{-½} S_w^{½} v = \lambda S_w^{½} v$
$S_w^{-1} S_b v = \lambda v$
$S_b v = \lambda S_w v$
Problém hledání vlastního čísla byl předchozí krok $S_w^{-1} S_b v = \lambda v$, kdy hledáme vlastní číslo a vektor matice $S_w^{-1} S_b$.

$\max_v v^T S_b v / v^T S_w v = \max_v \lambda v^T S_w v / v^T S_w v = \max_v \lambda$, kde λ je vlastní číslo vlastního vektoru v matice $S_w^{-1} S_b$, neboli volíme vlastní vektor matice $S_w^{-1} S_b$ příslušící jejímu největším vlastnímu číslu.

Problem 12.4
Střední hodnota:
$\mu_k = 1/N_k * \text{Sum}_{xi \text{ třídy } k} v^T x_i = v^T(1/N_k * \text{Sum}_{xi \text{ třídy } k} x_i) = v^T x_k$
Rozptyl:
$s_k = \text{Sum}_{xi \text{ třídy } k} (v^T x_i - v^T x_k)^2 = \text{Sum}_{xi \text{ třídy } k} (v^T(x_i - x_k))^2 = \text{Sum}_{xi \text{ třídy } k} v^T(x_i - x_k) v^T(x_i - x_k)$, kde $v^T(x_i - x_k)$ je skalár, lze tedy transponovat, proto:
$s_k = \text{Sum}_{xi \text{ třídy } k} v^T(x_i - x_k)(x_i - x_k)^T v = v^T(\text{Sum}_{xi \text{ třídy } k} (x_i - x_k)(x_i - x_k)^T) v = v^T \Sigma_k v$

Semestrální test 2 (2013)

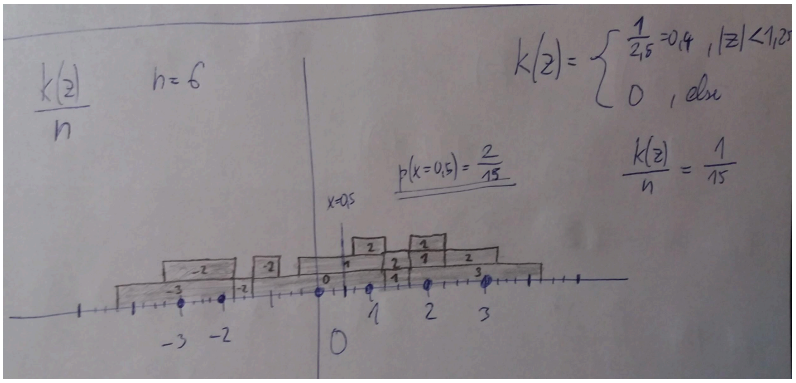
https://cw.fel.cvut.cz/b201/_media/courses/be5b33rpz/labs/rpzpractice_test2.pdf

Problem 1: (h = lambda)
Poisson(h): $p(x) = h^{x}*e^{-h}/x!$
$L(h) = p(X|h) = Prod_{i = 1..n} (h^{xi}*e^{-h}/x_i!)$
$l(h) = Sum_{i = 1..n} (x_i*ln(h) - ln(x_i!) - h)$
$l'(h) = Sum_{i = 1..n} (x_i/h - 1) = 0$, odtud $h = Sum_{i = 1..n} x_i / n$ = ar. průměr

Problem 2:
Exp(h): $p(x) = h*e^{-hx}$
Obecne h_{MLE}:
$L(h) = p(X|h) = Prod_{i = 1..n} (h*e^{-hx})$
$l(h) = Sum_{i = 1..n}(ln(h) - h*x_i)$
$l'(h) = Sum_{i = 1..n} (1/h - x_i)$, odtud $h = n / (Sum_{i = 1..n} (x_i)) = (ar. prumer)^{-1}$
Konkrétně je to $h_{MLE} = 5 / 13$
Obecne dele nez 5 let: (asi, nejsem si jistý, ale vychází to smysluplně)
$p(x>5) = Integral_{5..inf} p(x)\,dx = Integral_{5..inf} h*e^{-hx}\,dx = [h*(-1/h)*e^{-h*x}]_{5..inf} =$
$= [-1*e^{inf} + e^{-5*h}] = e^{-5*h}$
Konkretne to je $p(x>5) = e^{-5*5/13} = e^{-25/13} = 0{,}146…$

Problem 3:
Je jedno, ze jsou ty sekvence nejak deleny, proste:
T = (HHTHHTTHHHHTHHHTH), n = 17, $n_H = 12$ a $n_T = 5$
Intuice: $p = p(H) = n_H/n = 12/17$, $1 - p = p(T) = 5/17$
$p_{MLE} = argmax\ p(X|p)$
$L(p) = p(X|p) = Prod_{k\ from\ T} p(k) = p(H)^{nH}*p(T)^{nT}$
$l(p) = n_H*ln(p) + n_T*ln(1 - p)$
$l'(p) = n_H/p - n_T/(1 - p) = 0$, odtud: $n_H/p = n_T/(1 - p)$
$n_H(1 - p) = n_T*p$
$n_H - n_H*p = n_T*p$
$n_H = n_T*p + n_H*p$
$p_{MLE} = n_H / (n_H + n_T) = n_H / n = 12/17$

Problem 4:
Jadrova funkce:
$k(z) = 1/(2*1{,}25) = 2/5$ iff $|z| <= 1.25$
$k(z) = 0$ iff $|z| > 1.25$
Do každého bodu vložíme k(z)/n, získáme $p(x = 0.5) = 2/15$



Problem 5: ?????
Ani nevím, jestli tu otázku poradne chapu, je to “vyhodnotili jste kvalitu MLE odhadu tim, ze jste vykreslili jejich rozptyl jako funkci velikosti trénovací mnoziny”? Nejak nevim co odpovedet.

Problem 6: ?????
Zas mi ta otazka prijde nejaka divna, tak zalezi co jsme odhadovali, ne? Jinak odhadnout, ze neco je z normalniho rozdeleni mi prijde jako fajn volba, hodne veci ve svete odpovida normalnimu rozdeleni. Co tu ale rict konkretne fakt nevim.

Poznámka ze cvika:
z centralni limitni vety vime, ze pro nekonecne mnoho vzorku se vlastnosti spojitých rozdělení blizi normalnimu.

Semestrální test 3 (ukázka)

https://cw.fel.cvut.cz/b191/courses/be5b33rpz/labs/exercises/test3practice?fbclid=IwAR0k0FrWOSWAauiCXaAZ8lS_O2bQM3goDj7Jk4pLHfH-SnpurQG5z_RdMJU

Task 1

Máme trénovací množinu T s body x' z R^{D+1}, které vznikly z bodů x z R^D a jejich tříd y = {-1, 1}, jako x' = (1, x)*y

1) Inicializujeme t = 0 (číslo iterace), w_t = 0 z R^{D+1}
2) Nalezneme špatně ohodnocené x z T, tj. takové x, že $w_t^T x <= 0$. Pokud takové x neexistuje, končíme a w = w_t vektor popisující oddělující nadrovinu.
3) Updatujeme $w_{t+1} = w_t + x$ a jdeme na krok 2).

Se získaným vektorem w poté klasifikujeme takto: $q(x) = sign(w^T x)$, kde x z R^{D+1} je testovaný vektor upravený stejně jako trénovací.

Task 2

Perceptron nalezne jakoukoli oddělující nadrovinu a poté končí. SVM nalezne nejlepší oddělující nadrovinu ve smysl velikosti margin - je to oddělující nadrovina nejvzdálenější od nejbližších bodů. To znamená, že SVM lépe generalizuje (má nižší VC dimenzi).

Task 3

Trénovací data jsou (x, y), C je ztráta za špatně klasifikované x (poměrná ke vzdálenosti)

Primární úloha:

$w^*, b^* = argmin_{w,b}$ ½ $||w||^2 + C*Sum_i \xi_i$ za podmínky $y_i(w^T x_i + b) >= 1 - \xi_i$, $\xi_i >= 0$

Duální úloha:

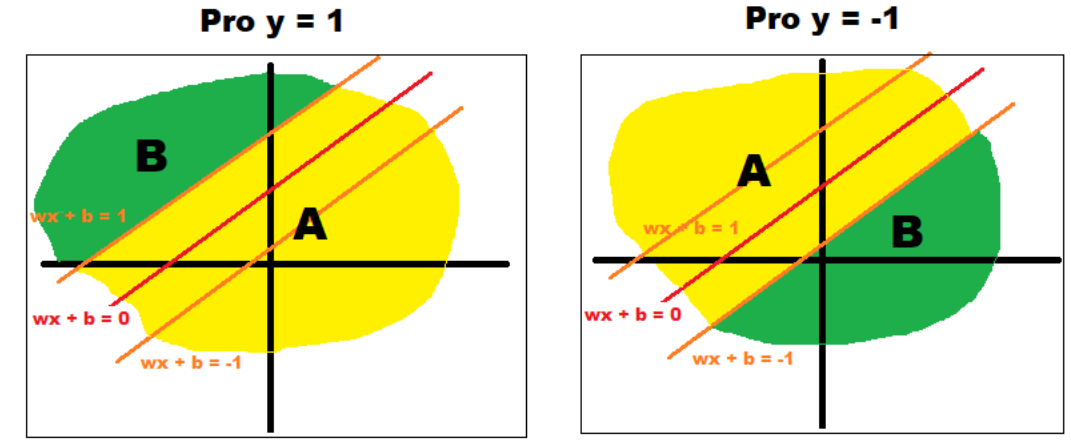
$a = argmax_a Sum_i a_i - ½ Sum_i Sum_j a_i a_j y_i y_j x_i x_j$ za podmínky $0 <= a_i <= C$, $Sum_i a_i y_i = 0$

$w = Sum_i a_i y_i x_i$

$b = y_s - w^T x_s$ pro nějaký support vector s

Klasifikujeme jako $q(y) = sign(w^T y + b)$

Task 4



Task 5 - Nejsem si stopro jistý, opravujte.

Očekávaná ztráta R(q) je průměrná ztráta na testovacích datech. Potřebujeme k ní znát sdruženou pravděpodobnost tříd a pozorování. $R(q) = Sum_{k,x} p(x, k)*W(q(x), k)$

Empirická ztráta je ztráta na trénovacích datech. Lze vypočíst bez znalosti sdružené pravděpodobnosti. $R_{emp}(q) = 1/L*Sum_{i \text{ od } 1 \text{ do } L} W(q(x_i), k_i)$

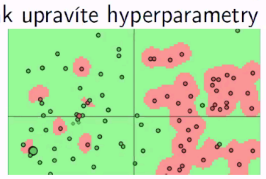
Upřednostníme očekávanou ztrátu kdykoli známe p(x, k), nebo lze dobře odhadnout. Empirická ztráta má širší využití.

Semestrální test 3

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).

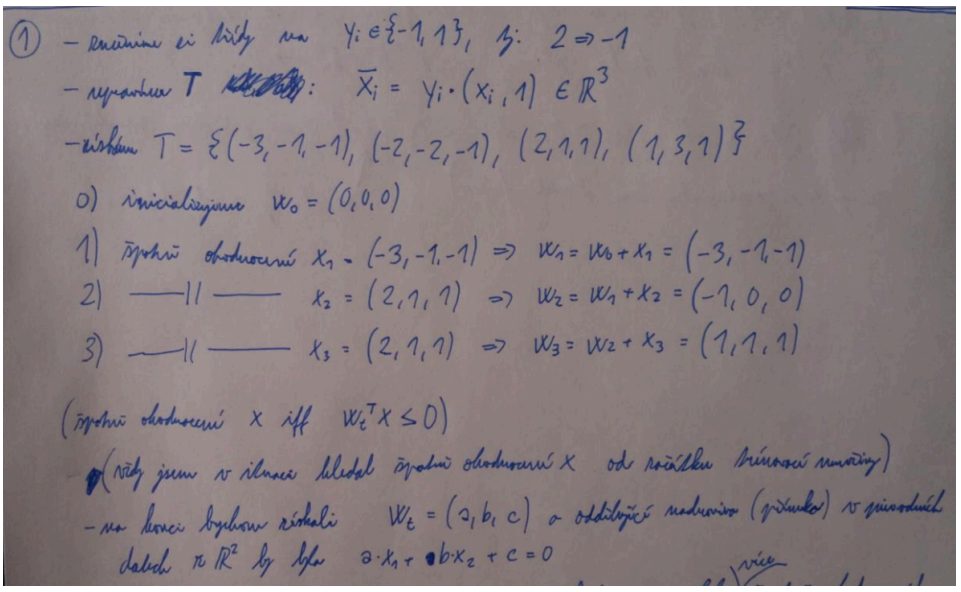
Úloha 2. (1b.) Trénujete soft-margin SVM s Gauss kernelem $K(x, y) = \exp(-\|x - y\|^2/2\sigma^2)$. Klasifikátor na trénovacích datech je na obrázku, na validačních datech ale nefunguje dobře. Jak upravíte hyperparametry C a σ? Proč?

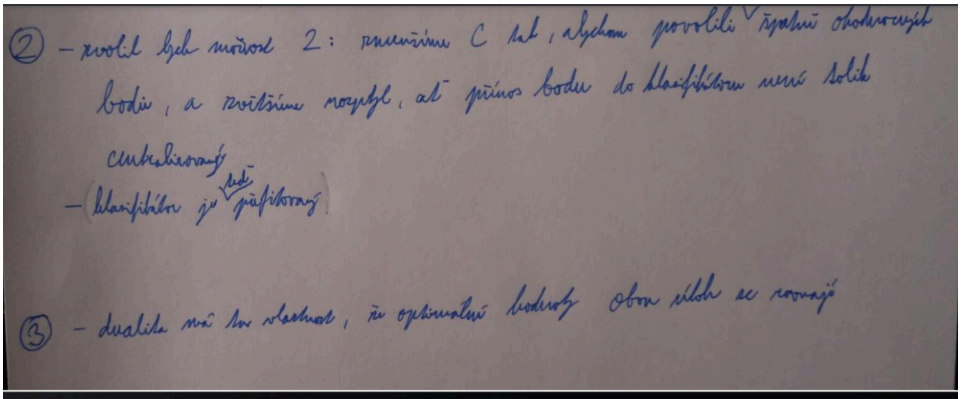
1. zmenšit C, zmenšit σ
2. zmenšit C, zvětšit σ
3. zvětšit C, zmenšit σ
4. zvětšit C, zvětšit σ

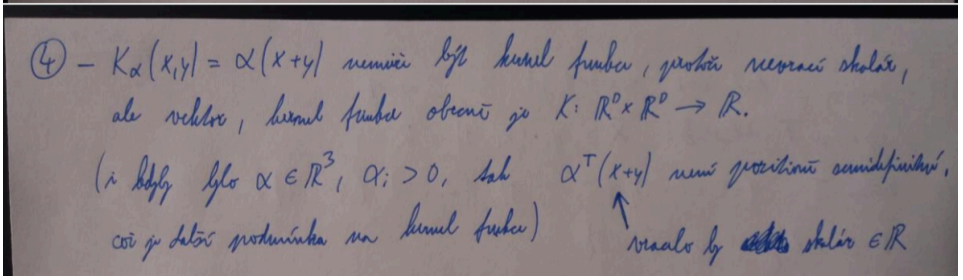


Úloha 3. (1b.) Jaká vlastnost platí pro vztah mezi optimálními hodnotami kriteriální funkce primální a duální úlohy SVM?

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 $\phi(x)$.







Semestrální test 3

1. Write down the dual task criterion function optimised by soft-margin SVM and define all symbols (only the numeric solution will by awarded by a point). How do you recognize the support vectors? *(1 point)*
2. Define a kernel function $\mathrm{K}(\mathbf{x}, \mathbf{y})$. Discuss one advantage of using the kernel Support Vector Machine compared to the linear SVM. *(1 point)*
3. Write down three advantages of the soft-margin SVM in comparison with the Perceptron algorithm. *(1 point)*
4. You are given a linear classifier $\mathbf{w} \in \mathbb{R}^{D+1}$ and a training dataset $\mathcal{T} = \{(\mathbf{x}_1; k_1), (\mathbf{x}_2; k_2), \ldots, (\mathbf{x}_L; k_L)\}$, where $\mathbf{x}_i \in \mathbb{R}^D$, $k_i \in \{-1, 1\}$ with $i = 1, 2, \ldots, L$. Formally define the linear separability. Only the numeric solution will by awarded by a point. *(1 point)*
5. You are given a training dataset $\mathcal{T} = \{(-1, 0.5; 1), (3, 3; 1), (2, -2; 1), (-2, -2; -1), (-1, -1; -1), (-1, -2.5; -1)\}$, where $\mathbf{x}_i \in \mathbb{R}^2$, $k_i \in \{-1, 1\}$ with $i = 1, 2, \ldots, 6$. Does the perceptron algorithm finish in a finite number of steps on this dataset? If so, why (give a numeric solution)? *(1 point)*

1)

$\alpha^* = \mathrm{argmax}_\alpha \mathrm{Sum}_i \alpha_i - ½ * \mathrm{Sum}_i \mathrm{Sum}_j \alpha_i\alpha_j y_i y_j x_i^T x_j$ za podmínky $\mathrm{Sum}_i \alpha_i y_i = 0$, $0 <= \alpha_i <= C$

α jsou Lagrangeovy multiplikátory, použijí se jako váhy trénovacích vektorů ve finálním klasifikátoru: $w = \mathrm{Sum}_i \alpha_i y_i x_i$, $b = y_{sv} - w^T x_{sv}$ (pro nějaký support vektor x_{sv})

(x_i, y_i) jsou trénovací body x_i z R^D a jejich třídy y_i = +-1

C je ztráta za špatně klasifikovaný bod (úměrná vzdálenosti od správně klasifikovaného poloprostoru)

Support vektory jsou takové (x_i, y_i) z trénovací množiny, které mají nenulové váhy α

2)

$K(x, y) = \varphi(x)^T \varphi(y)$, kde φ je mapování do vyšší dimenze $\varphi: R^D -> R^M$, které umožňuje použít lineární klasifikaci u lineárně neseparovatelných dat.

Výhoda: Lze tedy použít lineární klasifikátor pro oddělení lineárně neseparovatelných dat i bez explicitní znalosti φ. To umožňuje použít i potenciálně nekonečně dimenzionální mapování.

3)

a) umí oddělit lineárně neseparovatelná data
b) najde optimální řešení z hlediska velikosti marginu (v poměru ke ztrátě za špatnou klasifikaci C), narozdíl od nějakého řešení u Perceptronu
c) při učení využívá pouze skalární součiny trénovacích dat, je tedy nezávislé na dimenzi

4)

Množina T je lineárně separabilní, pokud existuje w z R^{D+1} takové, že pro všechny (x, k) z T platí: $\mathrm{sign}(w^T(x, 1)) = k$

5) ?????

Jediný, co mě napadá, jak to zjistit, je si to nakreslit, nevím, jaký chtějí numerické řešení. Novikoff theorem nám nepomůže, protože ten pouze říká "když máme vzdálenost γ a maximální velikost vektoru D, tak pokud jsou data separovatelná, tak skončí do iterace $t^* = D^2/\gamma^2$". Takže i kdybychom γ a D našli, tak to nic neříká o obecném zastavení algoritmu, když nevíme, jestli je to separovatelné.

4. RPZ Test 16.12.2020

Task 1 (AdaBoost) Consider the training set $\{(x_i, y_i) \mid i = 1 \dots n\}$ where $y_i \in \{-1, 1\}$. The AdaBoost strong classifier at step t is given by $H_t(x) = \text{sign}(f_t(x))$, where $f_t(x) = \sum_{k=1}^{t} \alpha_k h_k(x)$, $\alpha > 0$ and $h_k(x) \in \{-1, 1\}$.

a) Show that the number of training points misclassified by H_t is upper bounded by $\sum_{i=1}^{n} e^{-y_i f_t(x_i)}$ Properties of the exponent function e^z need not be proven.

b) To each data point i associate a weight $W_t(i)$ such that the upper bound in a) can be expressed as: $\sum_i W_t(i) e^{-\alpha_t y_i h_t(x_i)}$. It is not required that weights would sum to 1.

c) Show that if a training point (x_i, y_i) is misclassified by the weak classifier h_t, then its weight in the next iteration $t + 1$ will be increased and if it is correctly classified, then decreased.

(2 points)

Task 2 (K-means / EM) Let $\{x_i | i = 1 \dots n\}$ be observed points from a joint model $p(x, k)$ where $k \in \{1 \dots K\}$ is a hidden state (not observed). The model $p(x, k)$ is defined as $p(x, k) = p(x|k)p(k)$, where $p(x|k)$ is a Normal density $\mathcal{N}(\mu_k, I)$ with mean parameter μ_k for each k and $p(k)$ is uniform. Assume that initial estimates of μ_k are given.

a) Obtain K-means algorithm as follows. Given current means μ, find the most likely k_i, the assignment of mixture components to points maximizing the posterior distribution $p(k_i|x_i)$. Given current assignment k_i, find the maximum likelihood estimate of μ, i.e. maximizing the joint likelihood of x and k.

b) Obtain EM algorithm as follows. Given current means μ, find the soft assignment $\alpha_i(k) = p(k_i|x_i)$. Given weights $\alpha_i(k)$, find the parameters μ maximizing the "weighted" log-likelihood:

$$\sum_i \sum_k \alpha_i(k) \log p(x_i, k_i).$$

(3 points)

Task 1

a)

Špatně klasifikované body jsou ty, u kterých $H_t(x_i) != y_i$, neboli $f_t(x_i)*y_i < 0$.

Platí $[H_t(x_i) != y_i] <= e^{-yi*ft(xi)}$ pro všechny (x_i, y_i) z T (viz **Problem 8.1 b**), neboli platí horní mez i pro sumu:

$Sum_{i=1,\dots,N} [H_t(x_i) != y_i] <= Sum_{i=1,\dots,N} e^{-yi*ft(xi)}$

b) ????

$W_t(i)e^{-\alpha t*ht(xi)*yi} = e^{-yi*ft(xi)} = e^{-yi*Sum(j = 1..t-1)\alpha j*hj(xi) - \alpha t*ht(xi)*yi} = e^{-yi*Sum(j = 1..t-1)\alpha j*hj(xi)} e^{-\alpha t*ht(xi)*yi}$, odtud:)

$W_t(i) = e^{-yi*Sum(j = 1..t-1)\alpha j*hj(xi)}$

c)

Update váhy bodu (x_i, y_i) je definovaný $W_{t+1}(i) = W_t(i)*e^{-at*ht(xi)*yi}$, kde $a_t > 0$ je váha slabého klasifikátoru $h_t(x_i)$, $a_t = ½ \ln((1 - err_t)/err_t)$ pro $err_t < ½$.

U špatně klasifikovaný bodů je $h_t(x_i)*y_i = -1$, neboli $W_{t+1}(i) = W_t(i)*e^{at}$, $e^{at} > 1$.

U správně klasifikovaný bodů je $h_t(x_i)*y_i = 1$, neboli $W_{t+1}(i) = W_t(i)*e^{-at}$, $e^{-at} < 1$.

Task 2

Viz **Problem 11.1**

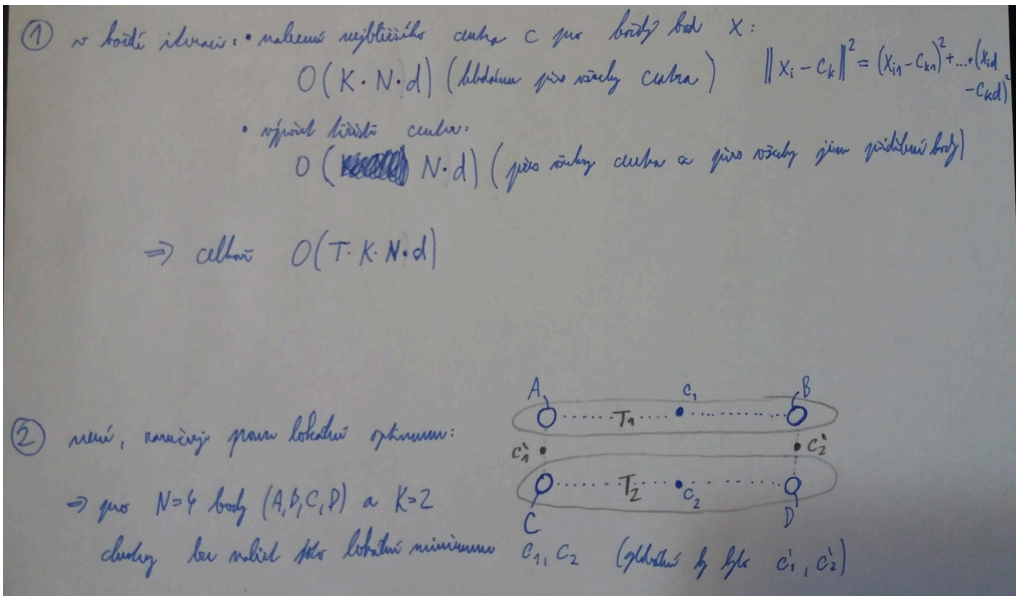
4. RPZ Test 18.12.2020

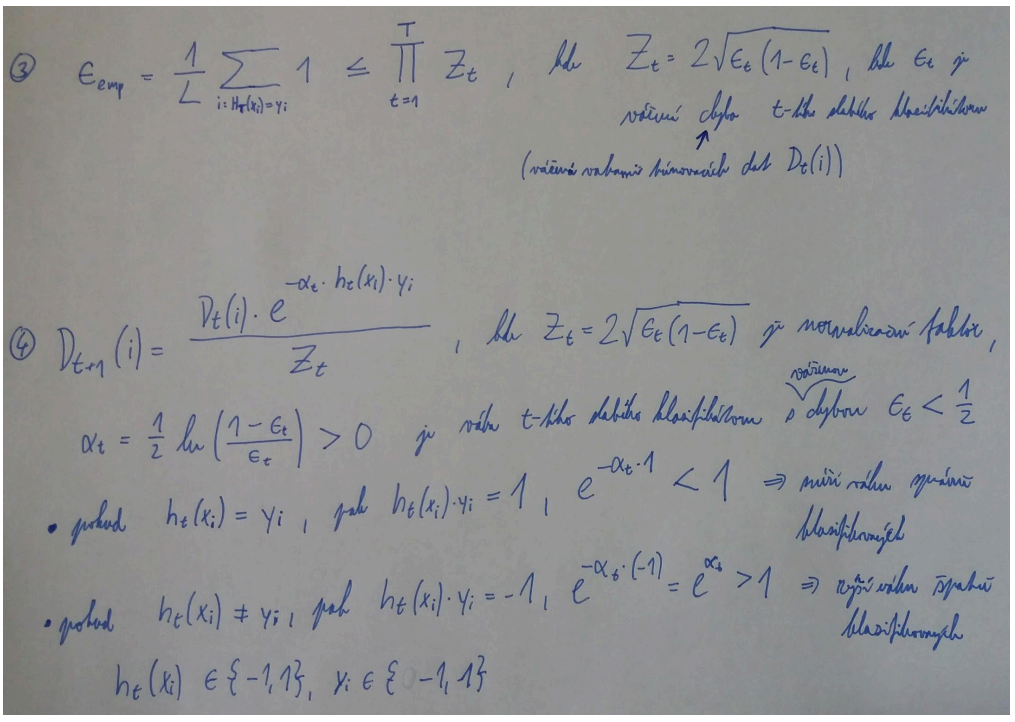
Úloha 1. (1b.) Jaká je časová složitost (počet float operací sčítání/násobení) základní verze algoritmu K-means pro K clusterů, T iterací a data $X = \{x_j \in \mathbb{R}^d | \, j = 1, .., N\}$?

Úloha 2. (1b.) Je u algoritmu K-means zaručená konvergence ke globálnímu optimu? Pokud ano, dokažte tuto vlastnost. Pokud ne, nakreslete příklad lokálního (ale ne globálního) optima.

Úloha 3. (1b.) Odvození klasifikátoru Adaboost je založeno na vztahu s empirickou chybou. Na jakém? *(Odpovězte vzorcem nebo jednou větou.)*

Úloha 4. (2b.) Uveďte vzorec z algoritmu Adaboost pro převážení trénovacích dat, tj. výpočet $D_{t+1}(i)$, a ukažte, jak zvyšuje váhu chybně klasifikovaných příkladů a snižuje váhu správně klasifikovaných příkladů.





4. RPZ Test 18.12.2020

1. The criterion of AdaBoost algorithm has an interesting connection with the empirical error. What is their relation? One keyword in one short sentence, please. *(1 point)*
2. In iteration t of the Adaboost algorithm, we select the weak classifier with the lowest value of the Adaboost criterion $Z_t = \sum_i D_t(i)e^{-\alpha_t y_i h_t(x_i)}$. We then need to assign the weight α_t to the weak classifier. How do we find that weight? Give formal solution. *(1 point)*
3. Fill in the missing words. If the series of values of clustering criterion is monotonically _____ and convergent, and there is _____ number of partitions, and no assignment is visited twice, the K-means algorithm reaches _____ minimum. In the K-means algorithm, the computational time is dominated by the complexity of _____ step. *(1 point)*
4. Formally define the **generalized** K-means algorithm. Briefly explain all symbols. *(1 point)*
5. Is the K-Means algorithm guaranteed to find a global minimum? If so, explain why. If not, give an example of some local optimum setting. *(1 point)*

1)

Keyword: horní odhad

(přesněji: $1/N * \text{Sum}_{i:\ H(xi)\ !=\ yi}\ 1 <= \text{Prod}_{t\ =\ 1..T}\ Z_t$, kde Z_t jsou normalizační konstanty rozdělení vah bodů $D_t(i)$, které minimaluzujeme optimální volbou vah slabých klasifikátorů α_t)

2)

Derivací Z_t podle α_t:

$Z_t = \text{Sum}_{i:\ H(xi)\ !=\ yi}\ D_t(i)*e^{\alpha t} + \text{Sum}_{i:\ H(xi)\ =\ yi}\ D_t(i)*e^{-\alpha t} = \epsilon_t*e^{\alpha t} + (1 - \epsilon_t)*e^{-\alpha t}$

$\partial Z_t/\partial\alpha_t = \epsilon_t*e^{\alpha t} - (1 - \epsilon_t)*e^{-\alpha t} = 0$

$\epsilon_t*e^{\alpha t} = (1 - \epsilon_t)*e^{-\alpha t}$… $\alpha_t = ½*\ln((1 - \epsilon_t)/\epsilon_t)$

3)

decreasing, finite, local, “data-to-cluster reassignment”

4)

Obecný K-means:

Máme N bodů x_i a chceme je rozdělit do K shluků.

1) init centra (c_1, …, c_K) (např. náhodně nebo podle K-Means++)
2) nalezneme každému bodu nejbližší centrum, tj. vytvoříme shluky:
   $T_k = \{x_i \mid d(x_i, c_k) <= d(x_i, c_j)$ pro všechna j$\}$ (v případě rovnosti máme pevnou strategii rozhodování)
3) updatujeme centra:
   $c_k = \text{argmin}_c\ \text{Sum}_{xi\ z\ Tk}\ d(x_i, c)$, pokud T_k není prázdná, jinak reinit c_k
4) Pokud se T_k nezměnily, končíme, jinak pokračujeme krokem 2)

c_k, x_i jsou z R^D, d: R^D x R^D -> R je funkce vzdálenosti

5)

Nemusí, viz ten příklad s obdélníkem