Otázky ke zk: [docs.google.com/document/d/12prKiWhRI2apqWih4CYSVgwMiAeRp7BSVi\_dzfk1NMg](https://docs.google.com/document/d/12prKiWhRI2apqWih4CYSVgwMiAeRp7BSVi_dzfk1NMg)

## Cvičení

* Cv. 6: (logistická regrese, 30.10.) part 1: <https://youtu.be/bnIisMd4Evw>

part 2: <https://youtu.be/RP39s26-KMU>

jiný zdroj: <https://youtu.be/N6QX_GP3cns>

* Cv. 7 (početní, 6.11.): <https://youtu.be/JeJbs80MJ_I>
* Cv. 8: (Perceptron, 13.11.): <https://youtu.be/80RX_cpeeLY>
* Cv. 9: (SVM, 20. 11.): <https://youtu.be/m0-SqopKbLA> (částma nic moc zvuk)
* Cv. 10 (SVM2, 27. 11.): <https://youtu.be/M16YPOiwOwM>
* Cv 4.12. <https://youtu.be/dWLkRBbHfR8>
* Cv 11.12 (K-means): <https://youtu.be/DJVlT9-63rk>
* Cv 18.12. (CNN): <https://youtu.be/WCssvD6RWkU>
* Cv 8.1. (závěrečné): <https://youtu.be/VB5-VwLpBtQ> (zvuk: celkem echo)

## Přednášky

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

7. Přednáška 6.11. SVM CZ: [drive.google.com/file/d/1G82geHaA14QIqDBihbasnqyf9--KaKMP](https://drive.google.com/file/d/1G82geHaA14QIqDBihbasnqyf9--KaKMP/view)

8. Přednáška 13.11. Adaboost CZ:  
 [drive.google.com/file/d/1nao\_Vp0YdyEtk21G7\_vnnR90BxE\_09bR](https://drive.google.com/file/d/1nao_Vp0YdyEtk21G7_vnnR90BxE_09bR/view?usp=sharing) - bez zvuku

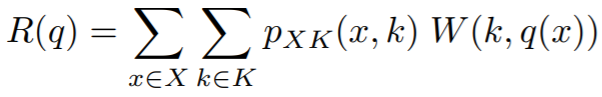
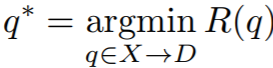
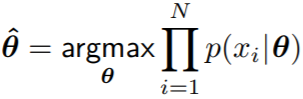
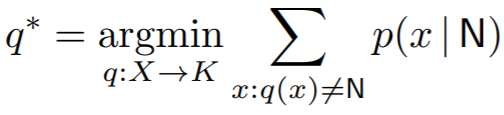
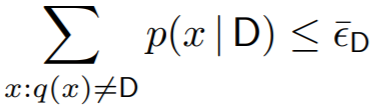
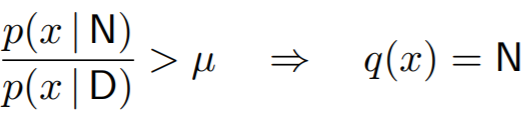
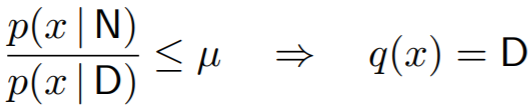
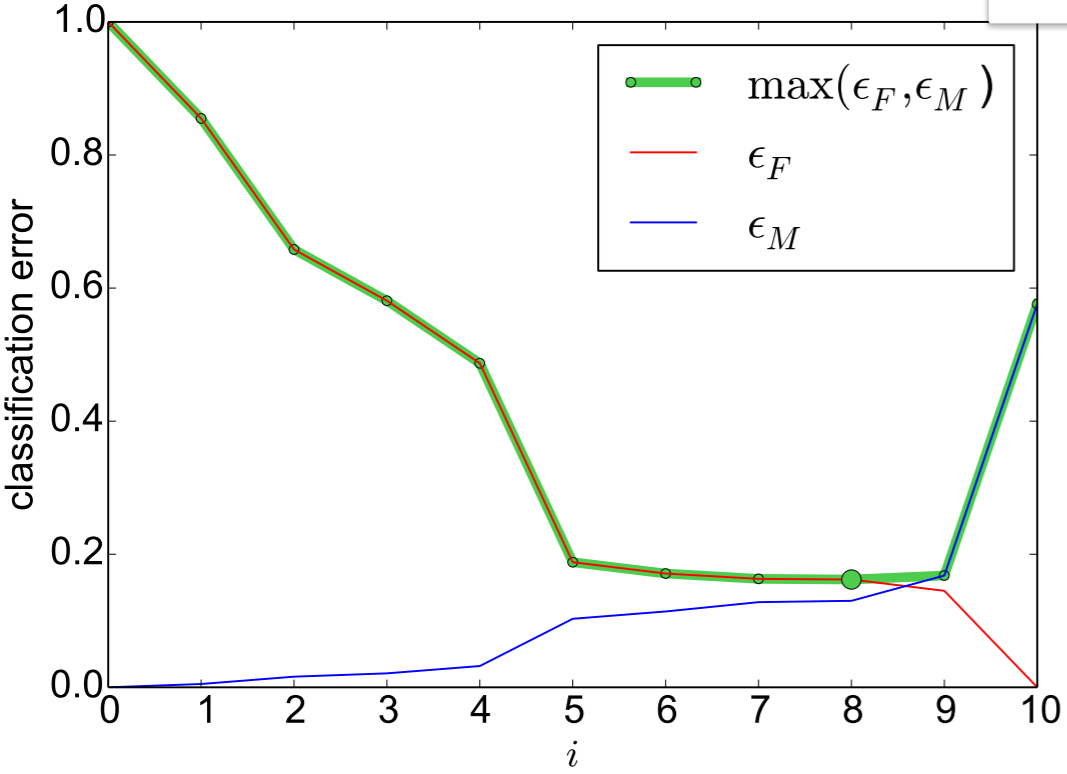
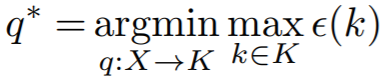
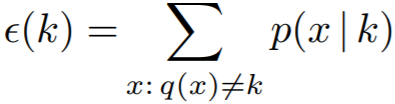
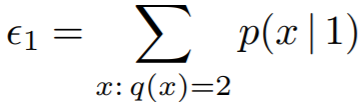
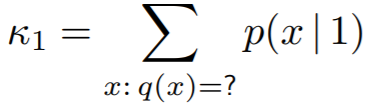
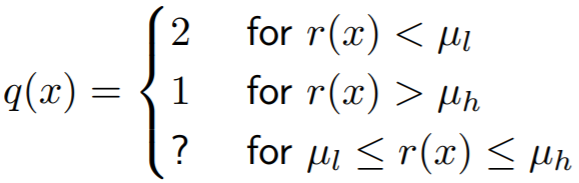
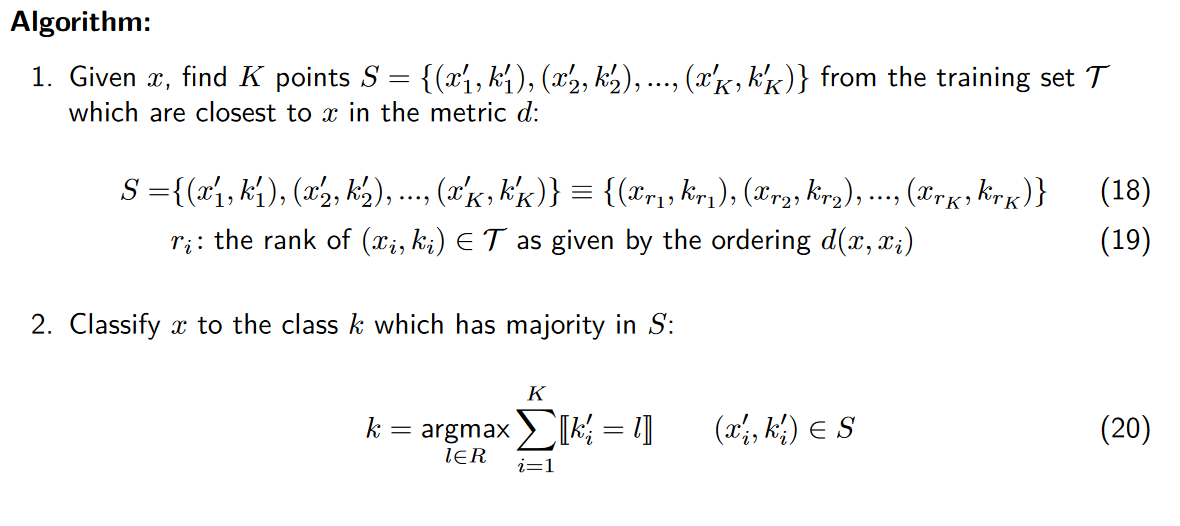
9. Přednáška 20.11. Neural networks: *(jen v playlistu viz dole)*

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

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

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

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

* ***Bayesovský risk a Bayesovská strategie*  
   **
* ***“Pravděpodobnosti”***
  + p(x,k) … sdružená (joint) probability; p(k) … prior probability
  + p(xIk) nebo p(xIθ)… likelihood → likelihood ratio používané v Neyman-Pearson, Minimax, Wald, Logistické regresi,
  + p(kIx) … aposterior probability - Bayesovská strategie rozhoduje pro třídu s nejvyšší posteriorní pravděpodobností (given some měření x) (“odds” v log. regresi)
  + p(x,k) = p(xIk) \* p(k) = p(kIx) \* p(x) … dle Bayesovy věty
* **MLE:** Známe ‘formu’ rozdělení s neznámými parametry θ (jejich počet je nízký)
  +  
  + dělám odhad parametrů pro každou třídu zvlášť, postupně:
  + pomocí log-likelihood: produkt přejde na sumu → potenciální zjednodušení derivace
  + L(θ) = Prod(p(ti)) → log (součin přejde na součet) → zderivuji a položím rovno nule
  + asymptoticky přejde ke správným parametrům
* **Neyman-Pearson**
* **:** formulace problému
  + K = {D, N} … dangerous state, normal state
  + známé conditionals: p(x|D), p(x|N); neznámé/neexistující prior probabilities p(D), p(N)
  + hledáme optimální strategii q\* pro kterou platí:
    - chyba klasifikace dangerous třídy je nižší než threshold ∈ (0,1), zatímco:
    - chyba klasifikace normal třídy je co nejnižší
  + tedy: subject to: 
  + strategie je pak charakterizována pomocí: єN (false alarm) a єD (overlooked danger)
  + optimální strategie q\*: existuje co nejmenší µ které zajistí:
    -  a  (“likelihood ratio”)
    - (overlooked danger < threshold)
  + pro spojitý případ je obecně nejlepší vzít takovou rozhodovací funkci, která má єD **=** є’D, protože s rostoucí єD klesá (či maximálně stagnuje) єN. Tato rovnost nelze vždy přímo dosáhnout u diskrétních pozorování
* **Minimax:** známé conditionals: p(x|D), p(x|N); neznámé/neexistující prior probabilities p(D), p(N)
  + X … set of observations, q: X → K … strategie
  + hledá strategii, která minimalizuje chybu na nejhůře klasifikované třídě (error nejhůře klasifikované třídy je nejmenší)
  + error je suma pravděpodobností p(x|k) na množině, kde q(x) rozhodne, že x nepatří do k
  + , kde (popř. integrál pro spojité x)
  + pro 2 třídy použijeme likelihood ratio
  + ve spojitém případě se dostaneme do rovnosti errorů na třídách; místo sumy integrál
  + naleznu práh t (integruj: 0>t, t>inf), pro který se rovnají a pak: q(x)=1 když x<t, a naopak
  + nezáleží na apriorních pravděpodobnostech p(k)
  + odpovídá Bayesovskému klasifikátoru při neznámých p(k) (??)
* ***Wald Task:*** 2 třídy: K = {1, 2}, nastavíme threshold є ∈(0,1), chyba na žádné třídě ho nesmí překročit; D = {1, 2, “nevím”}
  + optimální strategie subject to   
    (classif error pro class1: , undecided rate for kl1: )
  + strategie je charakterizována pomocí є1, є2, κ1, κ2 (classification error; undecided rate)
  + pro spojité měření: integrály místo sum
  + pro 2 třídy použitím likelihood ratia r(x) - hledám 2 thresholdy: 
* ***Parzen:*** do každého bodu vkládáme jádrovou funkci
  + je třeba normovat tak, aby integrál přes všechny jádrové funkce sčítal do 1
    - každý jádro přenásobit 1/N (N = počet bodů x)
  + kernel může být např. hyperkrychle, hyperkoule nebo “hypergauss”
* **K-NN:** algoritmus
* 
  + Máme trénovací množinu T = {(xi, ki)}1 <= i <= N. Pro testovací bod y nalezneme k nejbližších trénovacích bodů xi a y ohodnotíme jako té třídy, která má mezi k nejbližšími body většinu.
  + výhody/nevýhody

(+) jednoduchá implementace (naivní řešení - složitost: D\*N)

(+) pro 1-NN: lze najít upper-bound () a lower-bound: єB (Bayes error)

(-) při naivní implementaci pomalé, vysoké nároky na paměť počítače (příp. nutná úprava dat) → lze urychlit pomocí k-D trees nebo Voronoi nebo redukcí trn množiny

(-) není jasné jak zkonstruovat metriku d  
 (potenciální problém: různý význam různých dimenzí)

(-) vysoké nároky na paměť, (-) může overfitovat

(+) není potřeba předpoklad na distribuci, ze které data pocházejí

(+) pro libovolný počet tříd

* + *urychlení klasifikátoru 1-NN:*
    - redukcí trénovací množiny: snížení N např. z tisíců na desítky
    - pomocí k-D stromů: ty urychlují vyhledávání podle vzdálenosti v geometrickém prostoru pomocí ukládání dat ve stromové struktuře. Při procházení stromu se pro každou větev určí minimální možná vzdálenost nejbližšího bodu ve větvi od posuzovaného bodu. Po prvním projití do listu pak můžeme ořezat větve, které jistě nemají bližší bod. Je to sublineární vyhledávání, ale složitost je funkcí dat, nejde tedy garantovat.
    - pomocí Voronoiových diagramů: při klasifikaci stačí zjistit v jaké buňce bod leží  
      (rychlejší než počítat vzdálenosti)

* **Logistická regrese**
  + lineární klasifikátor, log odds
  +   
    a obrácené znaménko pro q(x) = 2
  + pokud je poměr lineární, přecházíme na problém hledání w=[w0,w]; wTx > 0 → q(x)=1
  + priors způsobí jen posun (jsou obsaženy v členu w0)
  + log odds:  → p(1|x) = p(2|x) exp(w·x), p(2|x) = p(1|x) exp(−w·x)

→ 1 = p(1|x) + p(2|x) = p(1|x) (1 + exp(−w·x)) = p(2|x) (1 + exp(w·x))

* + k : {-1,1} →  (sigmoida)
  + algoritmus:

0. upravit trn mn: {(xi, yi)}, x z RD, y = +-1, na {yi\*(xi,1)}. Init w z RD+1

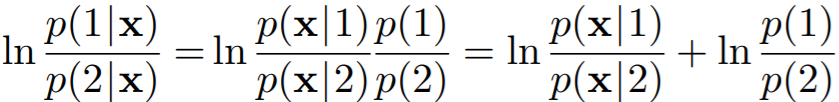
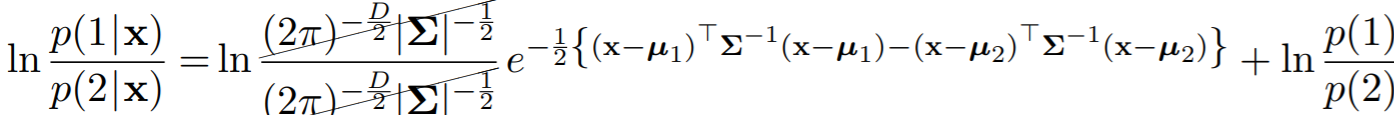
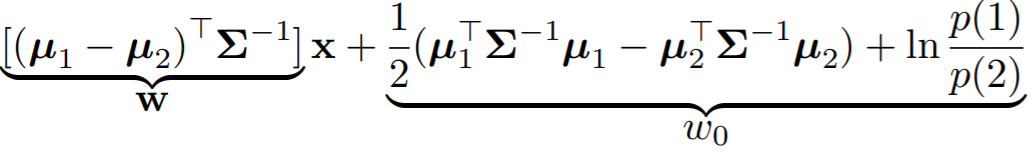
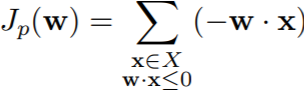
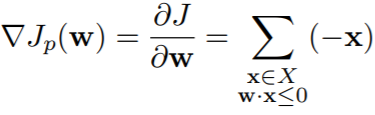
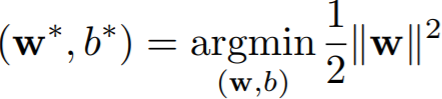
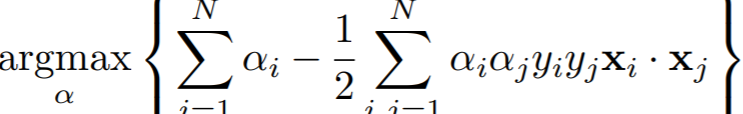
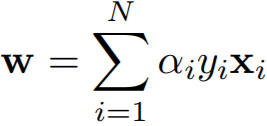
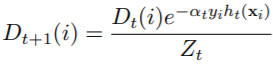
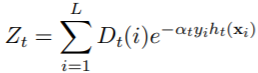
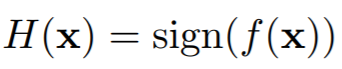
1. minimalizujeme funkci E(w) = -l(w) = Sumi ln(1 + e-wx)

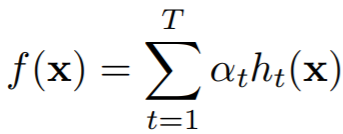
Vypočítáme gradient ∇wE = - Sumi x\*e-wx/(1 + e-wx)

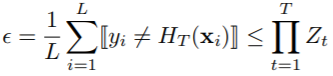
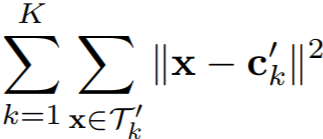
Updatujeme w:= w - μ\*∇wE (jdeme proti směru grad, protože minimalizujeme E)

Pokud E(wnové) < E(wstaré): step = 2\*step, jinak: step = step/2 a neupdatuj w,g,E

2. goto 1

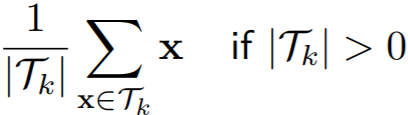
* *Diskriminační funkce pro 2 třídy se stejnou* ***kovarianční maticí***
  + 
  + 
  + ln(ex)=x
  + maticově roznásobit
  + xTΣ-1x se požerou
  + xTΣ-1μ = μTΣ-1x (protože Σ je symetrická: Σ = ΣT)
  + 
  + → **w**·**x** + w0 , kde **w** ∈ RD, w0 ∈ R
  + Závěr: Pro 2 třídy s norm. rozdělením se stejnými kov. maticemi, log odds je lin. fce
* Kov. mat: → vl. čísla (a vektory): vidím osy elipsy Gausse, velké vl. č. = velká var v daném směru
* **Perceptron**
  + algoritmus:
    - příprava dat: **xi** = ki \* [yi **xi**], **w** = [w0 **w**] → **xi, w** ∈ RD+1
    - t=0, w(t) = [0]
    - najdi xj: xj·w ≤ 0 → w(t+1) = w(t) + xj
  + pro neseparabilní data:
    - dimension lifting
    - s použitím batch version, pamatuju si nejlepší w\*, kdy počet špatně klasif byl nejnižší
  + cost funkci lze redefinovat:  → 
  + update buď klasicky špatně klasifikovaným x ;anebo batch version: všechny špatně klasifikované a update jejich součtem (možné ještě škálovat aby krok nebyl tak velký)
* **SVM**
  + lineární klasifikátor; nachází optimální řešení (dělící nadrovina maximalizuje šířku dělícího pásu); lze upravit i pro lin. neseparabilní data; (w, b)\* = argmaxw,b mini d(xi) za podmínky yi(wTxi + b) > 0, kde d(xi) je vzdálenost xi od nadroviny
  + T = {xi, yi} (i=1...N), xi ∈ RD, yi ∈ {-1,1}
  + primární formulace: , tj. argmax 1/IIwII, w ∈ RD, b ∈ R
  + duální f: α\*= (podm: α >= 0, Sumi yiαi = 0)  
    → pomocí QP najdeme glob. optimum
  + strategie pak: , (pouze α>0, tj. SV (ostatní mají α=0))
  + učení pomocí iteračních metod, funkce je konvexní → dospěje do globálního minima
  + Soft margin SVM: pro nesep. data představíme C - ztráta za špatně klasifikovaný bod; ; Primární úloha: (w, b)\* = argminw,b ½||w||2 + C\*Sumi ξi za podmínky yi(wTxi + b) >= 1 - ξi, ξi >= 0; Duální úloha: α\* = argmaxα Sumi αi - ½ Sumi Sumj αiαjyiyixiTxj za podmínky 0 <= α <= C, Sumi yiαi = 0
  + Při použití dimension lifting φ(x): RD -> RM nemusíme používat/znát přímo předpis mapování. Stačí znát kernel funkci K(x, y) = φ(x)Tφ(y). Tou lze nahradit například i mapování do teoreticky nekonečně dimenzionálních prostorů apod. Učení SVM probíhá stejně, pouze duální úloha přijme kernel funkci: α\* = argmaxα Sumi αi - ½ Sumi Sumj αiαjyiyiK(xi, xj) za podmínky 0 <= α <= C, Sumi yiαi = 0
  + Klasifikace má dva způsoby:
    - použijeme mapování φ(x): Získáme w = Sumi z SV αiyiφ(xi), b = ysv - wTφ(xsv) a klasifikujeme: q(z) = sign(wTφ(z) + b); To znamená, že musíme znát φ a klasifikace je zpomalena oproti lineární variantě výpočtem φ(z)
    - použijeme kernel K(x, y): b = ysv - Sumi z SV αiyiK(xi, xsv) ; q(z) = sign(b + Sumi z SV αiyiK(xi, z)); To znamená, že si musíme pro klasifikaci pamatovat αi, yi, xi pro support vektory a klasifikace je zpomalena výpočtem Sumi z SV αiyiK(xi, z)
  + časová složitost závislá na N (ne na D; při optimalizaci jsou počítány skalár. součiny xiT,xj)
  + rychlost učení závisí na inicializaci
  + dobře generalizuje, používají pouze skalární součiny dat, tzn. že jsou nezávislé na vstupní dimenzi, umí pracovat i s neseparabilními daty, při klasifikaci stačí znát pouze w a b, lze zobecnit pro oddělení nelineárně separovatelných dat pomocí dimension lifting a kernel triku.
  + SVM se složitě zobecňují pro oddělení více tříd a nehodí se do inkrementálního učení.
* *Perceptron vs SVM*
  + Perceptron nalezne nějaké řešení, SVM “dobré” (maximalizující margin)
  + u perceptronu pro separabilní data zaručena konvergence (Novikoff teorém)
* **Adaboost:** množina B sl. klasifikátorů h(x)algoritmus
  + Init: t=1, Dt=1(i)=1/N
  + vyberu ht(x) s nejmenší chybou єt: 
    - divná závorka dá jedničku pokud výraz uvnitř platí, jinak nulu (tj. při chybě → 1)
  + pro vybraný slabý klasifikátor ht mám chybu єt
    - pokud єt větší než 0,5 pak co?? nebo jen když se rovná 0,5 tak končím?
  + Váha daného slabého klasifikátoru: 
  + Update vah dat: , normalizační faktor Zt: 
    - lze vypočítat také takto: 
  + Nakonec (po ukončení iterování) – výsledný (silný) klasifikátor: ,



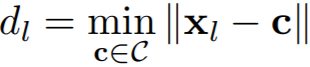
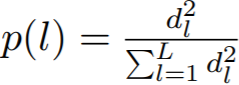
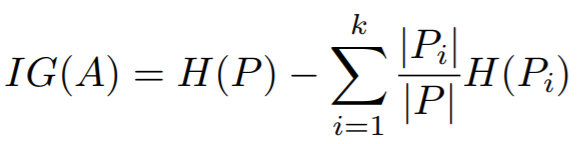
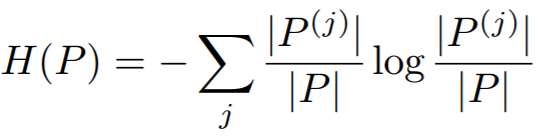
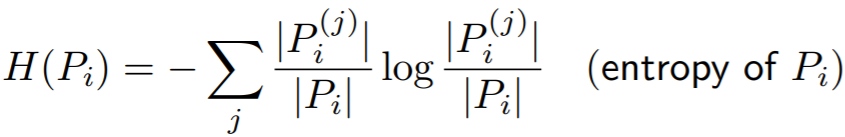
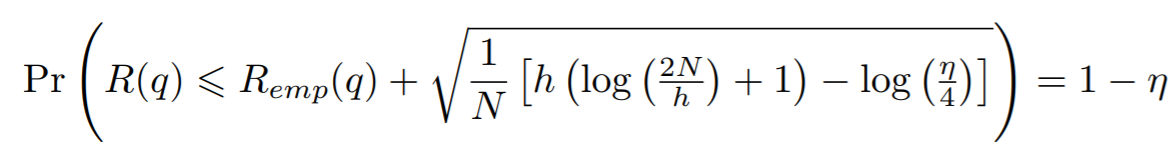
* *Adaboost: Upper bound theorem:* 
* *Adaboost: Vlastnosti:*
  + (+) jednoduchá implementace
  + (+) dobrá generalizace  
    (= chová se dobře k testovacím datům po natrénování, neoverfittuje moc)
  + (-) overfitting kvůli noise
* **K-means**
  + data xi, chceme roztřídit do k tříd; hledáme K center shluků:  
    {ck\*} = argmin(ck)  … abychom minimalizovali vzdálenosti od center
  + algoritmus:

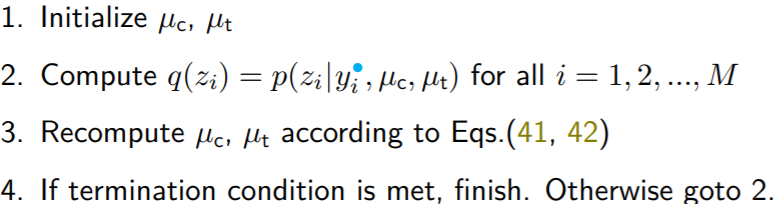
0. inicializace k centroidů (náhodně/k-means++)

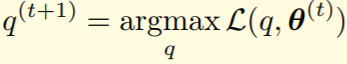
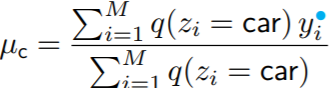
1. přiřazení bodů k nejbližšímu centroidu: 

2. spočítání nového těžiště shluku: ck = , pokud je Tk = ∅: reinit ck

3. pokud Tk = Tk+1 (stejné clustery jako předchozí iterace), tak konec, jinak goto 1

* + lze zobecnit pro jakoukoliv metriku d
  + Inicializace pomocí K++ means zvyšuje pravděpodobnost nalezení globálního minima, alg: c1 vybrán úplně náhodně, , , dokud nemám K center
  + Můžeme uvíznout v lokálním extrému (lze řešit více běhy a výběrem nejlepšího řešení)
  + alg končí v konečném počtu iter; konečný počet stavů: NK,
  + složitost: I(počet iter)·N·K·M; N·K: v každé iter porovnávám každý bod s každým shlukem
* **Decision trees**
  + IG (information gain) umět spočítat
    - Entropie
      * log2; informační entropie [bit]: vysoká - nic nevím, nízká - vím co se stane
    - 
    - spočítám IG pro každý atribut → tam kde vyjde nejvyšší, ten použiju
    - kde P(j) je počet bodů které spadnou do j-té dané třídy
    - entropy of Pi: 
  + výhody/nevýhody
    - (+) interpretovatelnost; rychlost rozhodování; nepotřeba použít všechny měření - jen ty relevantní; outliers ošetřeny speciální větví; pro klasifikační i regresní úlohy
    - (-) neexistuje efektivní způsob trénování (všechny na bázi greedy algoritmu →  
       není záruka optimality); hrozí overfitting (→ randomizované stromy)
  + top-down: na začátku nastavím P=T; spočítám IG pro každý atribut, zvolím pro dělení ten který mi dá nejvyšší IG; rekurze pro větve
  + DT vs. SVM:
    - pro jakékoliv prostory, i nenumerické, s různými hodnotami/měřítky v různých dimenzích, apod. X jen pro data z RD
    - trénovací množina může být libovolně velká, při klasifikaci se nevyužívá (klasifikace je sublineární vůči dimenzi dat) X trénovací množina může být libovolně velká, při klasifikaci se nevyužívá, pouze se z její části (tzv. support vektorů) vypočítá lineární klasifikátor q(x) = sign(wTx)
    - žádné záruky na generalizaci/optimalitu řešení X dobře generalizuje, jelikož minimalizuje VC dimenzi maximalizací margin mezi třídami dat, díky konvexnosti rozhodovací funkce a QP optimalizaci nalezne globální optimum
    - při učení je potřeba uchovávat veškerá data, při klasifikaci pak již DT samotný (posloupnost otázek/rozhodnutí) X při učení stačí uchovávat skalární součiny všech xi, pro klasifikaci stačí uchovávat jediný vektor w z RD
    - učení je závislé na složitosti stromu, respektive nic jako učení neprobíhá, strom je pouze sestaven, při stavbě stromu lze dodržovat určitá pravidla pro jeho obecně lepší klasifikaci (TDIDT) X učení je závislé na velikosti trénovací množiny a rychlosti konvergence QP optimalizace, která kriticky závisí na počáteční inicializaci
    - odchýlené hodnoty lze jednoduše odchytávat jako speciální větve stromu X vliv odchýlených hodnot lze regulovat velikostí ztráty za špatně klasifikovaný bod C, ta určuje směr “úzký margin bez špatně klasifikovaných bodů (vyšší C)” vs “širší margin se špatně klasifikovanými body (nižší C)”
* **VC dimenze**
  + měřítko kapacity
  + definována jako mohutnost (cardinality) největší sady bodů, které může algoritmus rozbít
  + pro lineární klasifikátory: VC = dimprostoru + 1 (2D: přímka oddělí max 3 body vždy → VC=3)
  + Vapnik and Chervonenkis inequality: 
    - člen s odmocninou se nazývá strukturální risk - ten je horní odhad rozdílu R a Remp
  + 1-NN: VC = inf
* **NN vs SVM, Adaboost a K-NN**: na obrázky
* Porovnávání
  + na jaký data, učení, chování vůči neseparabilním datům, chování vůči noise/outliers, konvergence, paměťové nároky
* **EM (Expectation maximization)**
  + použití:
    - missing data - měření jsou nekompletní (pro některé x chybí nějaká složka)
    - latent variables - měření kompletní, ale bylo by mnohem lepší kdyby byla ještě nějaká informace (?), např. mix více rozdělení typicky
  + algoritmus

(S2: “E-step”, S3: “M-step”)

* + , 
  + jako k-means (nemáme informaci o třídě) (příklad car/truck):
    - 0. init μc, μt
    - 1. 
    - 2. μc, μt: 
    - ukončit pokud je termination condition splněna
    - k-means se liší pouze v bodě 1 - tam se přiřazuje pouze 1 nebo 0
  + častá aplikace: odhad mixu Gaussů
  + M-step počítá pravděpodobnost, že daný bod je generovaný danými komponenty
  + E-step počítá neznámé parametry analyticky
  + je užitečný a efektivní dále pro exponenciální rozdělení
  + chytře maximalizuje likelihood tím, že tlačí lower bound nahoru
  + iterativní metoda, nemusí skončit v globálním maximu
  + opatrně inicializovat (stejně jako u K-means, NNs, …)

Test3

