Kombinace klasifikátorů
jako cesta pro zvýšení přesnosti

Rozhodování z více hledisek

- V běžném životě se často snažíme získat názor více expertů, než přijmeme závažné rozhodnutí:
  - Před operací se radíme s více lékaři
  - Před koupí nového vozu si přečteme recenze na několik různých produktů téže třídy
  - Při hledání místa nás zajímají reference na budoucího zaměstnvatele
- Proč nepoužít stejnou strategii i při automatizaci rozhodování?
- Tento přístup má ve strojovém učení mnoho různých jmen:
  - Multiple classifier systems,
  - committee of classifiers,
  - mixture of experts,
  - ensemble based systems

Klasifikátor
jako kombinace více klasifikátorů

Kombinace klasifikátorů (konzilium) si v řadě aplikací vede velmi úspěšně ve srovnání s jednodruhovými systémy

- **Otázky spojené s návrhem kombinovaných klasifikátorů:**

  (i) jak volit individuální základní (*base*) klasifikátory ?

  (ii) jak kombinovat jejich výsledky?

- Příklady současných postupů
  - **Bagging, boosting, AdaBoost**, postupná (stacked) generalizace a hierarchická směs expertů
  - Běžně používaná **pravidla pro kombinaci výsledků**
  - Algebraická kombinace, hlasování, *behavior knowledge space* & rozhodovací vzory (*decision templates*)

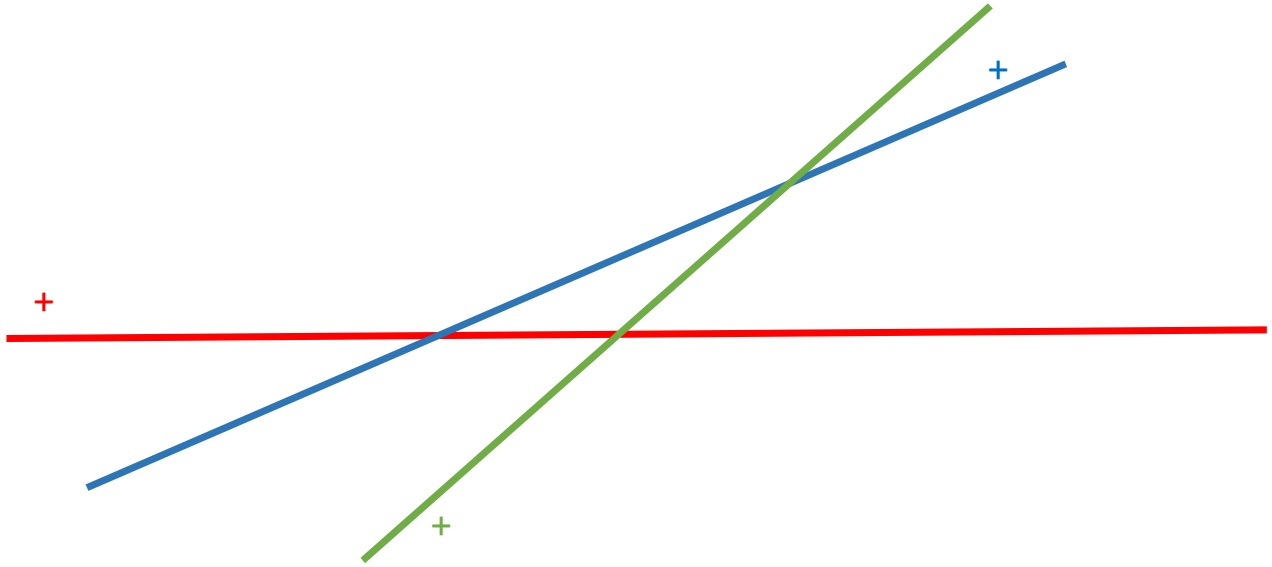
Proč kombinovat klasifikátory?

- **Statistické důvody**
  - Množina různých klasifikátorů s podobným výkonem při trénování může mít rozdílné generalizační schopnosti
  - Kombinací výsledků více klasifikátorů zmenšujeme riziko toho, že si zvolíme opravdu špatné řešení
- **Rozsáhlé objemy dat**
  - Jeden SW systém nemusí pojmout všechna zpracovávaná data. Pak je vhodné použít současně více klasifikátorů na různé výběry z dat a jejich výsledky kombinovat.
- **Malé objemy dat**
  - Kombinace je výhodná i v tomto případě: resampling techniques
- **Nejednotný formát dat –** viz data fusion

Proč kombinovat klasifikátory?

Rozděluj a panuj!

- Rozděl data do menších lépe popsatelných skupin.
- Nechej každý klasifikátor zvládnout jen jednu z těchto skupin
- Navrhni způsob kombinace navržených rozhodnutí



Jak bude vypadat hranice mezi objekty **+** a **-** , pokud je výsledná klasifikace dána hlasováním mezi 3 lineárními klasifikátory?

Výhody kombinací klasifikátorů

Přístup „***rozděluj a panuj***“:

Trénovací data jsou rozdělena na podmnožiny, ve kterých se daří dosáhnout rozumné přesnosti klasifikace jednodušším algoritmem (zde „kruhová hranice“). Výsledek je dán kombinací!

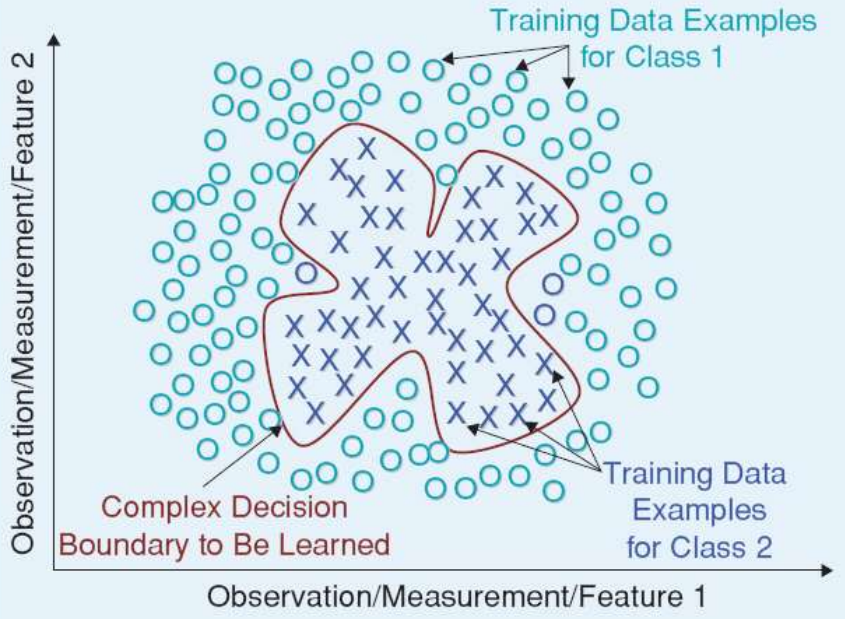


Figure 1. Complex decision boundary that cannot be learned by linear or circular classifiers.

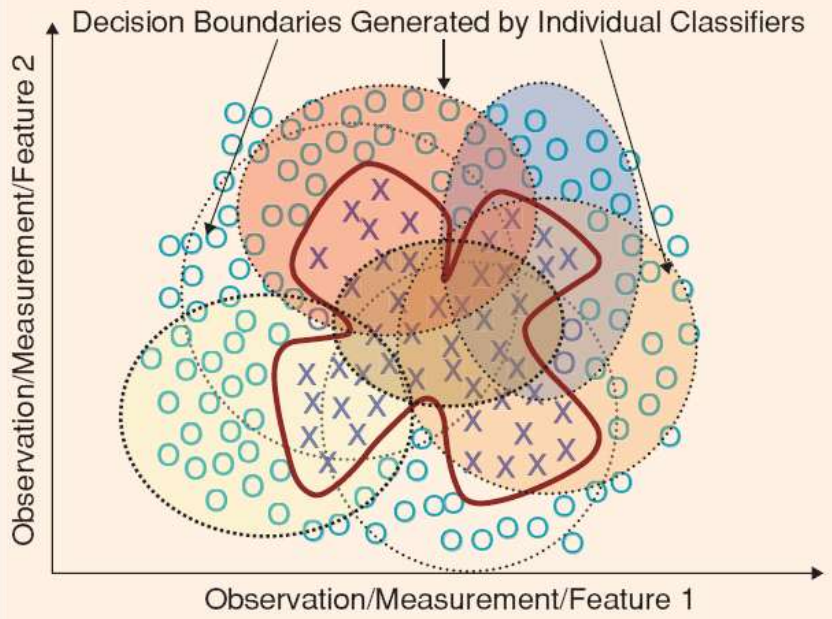


Figure 2. Ensemble of classifiers spanning the decision space.

Kdy ještě kombinovat klasifikátory?

- **Sdružování dat** (data fusion)
  - Máme-li několik souborů dat z více zdrojů, v nichž jsou použity různé typy atributů zcela odlišného charakteru (heterogeneous features), může být naučení jediného klasifikátoru velmi obtížné (např. pro pacienta máme MRI data, EEG záznam, krevní testy,..)
  - Problém s chybějícími údaji!
  - Řada aplikací úspěšně využívá kombinací klasifiktorů pro rozhodování nad různorodými daty - data fusion applications

Výlet do historie

- První návrh postupu využívajícího kombinaci klasifikátorů: Dasarathy and Sheela (1979)
- Schapire (1990) dokázal, že velmi dobrý (*strong*) klasifikátor může vzniknout kombinací více průměrných (*weak*) klasifikátorů pomocí „boosting" (předchůdce algoritmu AdaBoost)
- **2 přístupy ke kombinování**:
  - **Výběr klasifikátoru** Každý klasifikátor je natrénován tak, aby se stal expertem v nějaké části definičního oboru uvažovaných příkladů; 1 či více lokálních expertů může být vyzváno, aby navrhli řešení (rozhodnutí) pro „svou část". Jak vybírat vhodné části def.oboru?
  - **Fůze klasifikátorů** (*classifier fusion*). Všechny klasifikátory jsou trénovány nad celým definičním oborem; fůze pak znamená sdružení jednotlivých slabších klasifikátorů tak, aby vznikl jeden výrazně lepší.
  - Jak získat různorodé klasifikátory?
- Jak kombinovat výsledky?

Různorodost použitých klasifikátorů

- **Cíl:** vytvořte mnoho klasifikátorů vhodných pro kombinování. Intuice: Pokud každý klasifikátor dělá poněkud jiný typ chyb, pak jejich kombinace může být výhodná!
- Potřebujeme tedy klasifikátory takové, že
  - dávají relativně dobré výsledky,
  - mají vzájemně dost různé rozhodovací hranice (*dec. boundaries*)

  Taková množina klasifikátorů se nazývá různorodá (*diverse*)
- **Jak zajistit vznik různorodé množiny klasifikátorů?**
  - Individuální klasifikátory jsou trénovány nad různými trénovacími množinami dat
  - Jak získat vhodné různé trénovací množiny?
    - **k-násobné dělení** (*jackknife*): ***k*** stejně velkých disjunktních bloků; každý klasifikátor je trénován na skupině dat vzniklé z původních vynecháním jednoho bloku
    - **Vzorkovací** (*resampling*) techniky: bootstrapping nebo bagging – trénovací podskupiny jsou vybírány náhodně (s navracením) z výchozí dat

Vzorkování bez navracení - k-násobné dělení (jackknife)

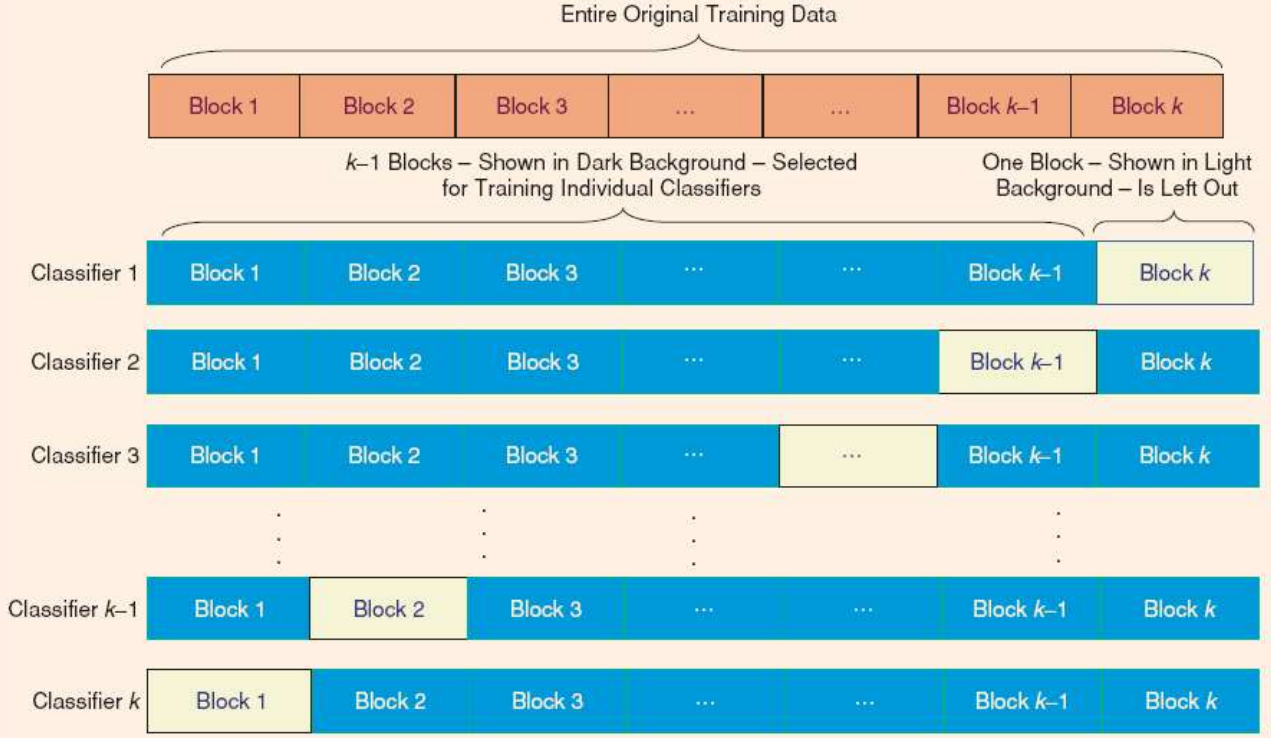


Figure 4. *k*-fold data splitting for generating different, but overlapping, training datasets.

Vzorkování s navracením

3 náhodně vybrané & překrývající se trénovací skupiny jsou použity pro vznik 3 klasifikátorů, jejichž rozhodnutí se pak kombinuje v jediném, které je přesnější než každé z nich samostatně!

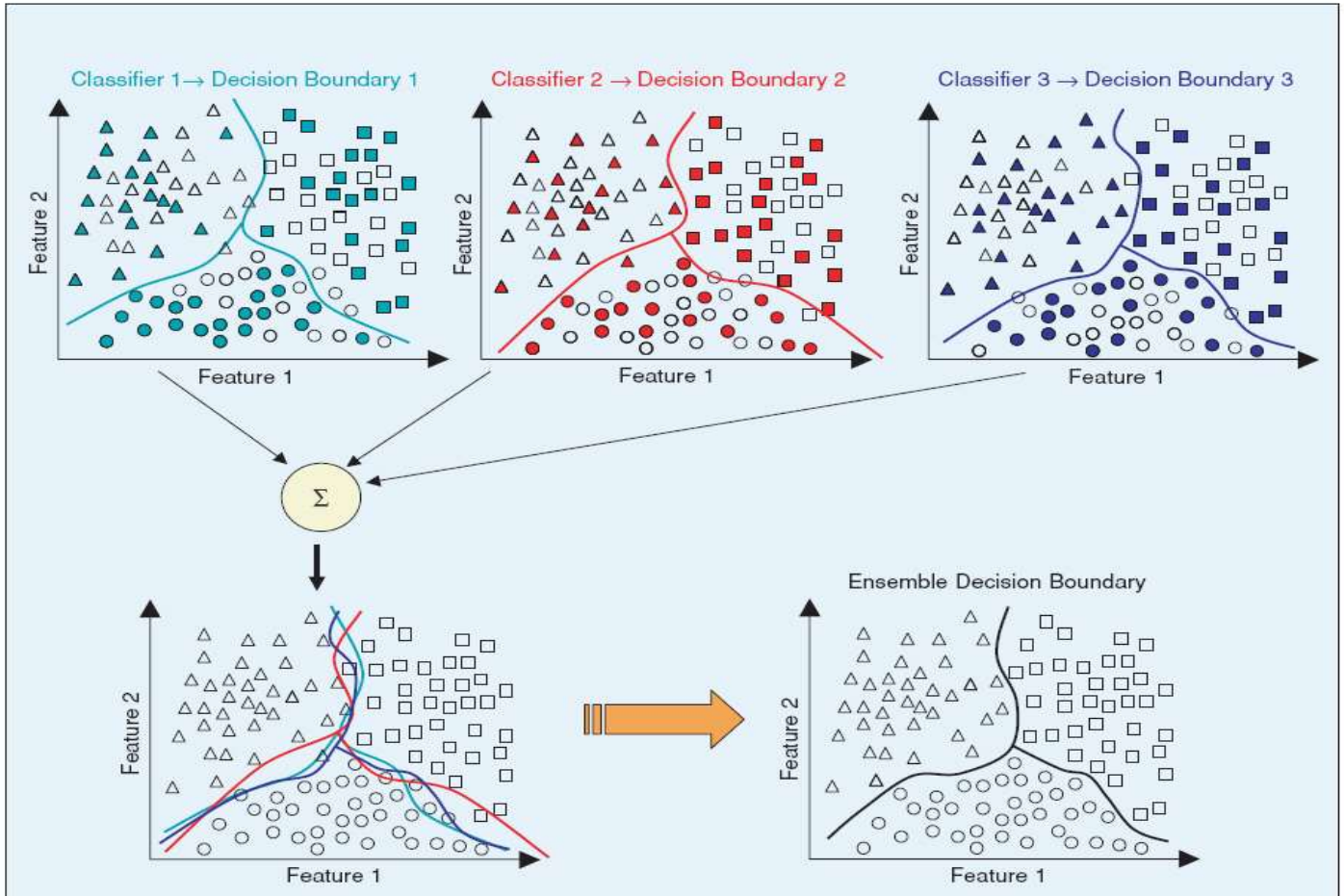


Figure 3. Combining classifiers that are trained on different subsets of the training data.

Další možnosti, jak dosáhnout různorodosti

- Kombinace rozhodnutí různých typů klasifikátorů (např. vícevrstvý perceptron - MLP, rozhodovací strom, neuronová síť NN, SVM)
- Tentýž typ klasifikátoru použitý s různými výchozími hodnotami parametrů
  - U MLP nebo NN je rozdíl dán výchozími vahami různých hran, volbou architektury, atd.
  - U rozhodovacích stromů můžeme měnit volbu diskretizace atributů, kriterium větvení, způsob prořezávání stromu, maximální hloubku stromu …
  - Změny výchozích parametrů ovlivňují (ne)stabilitu výsledného klasifikátoru (lokální minimum)
- Různorodosti lze dosáhnout i tím, že pro učení nepoužijeme popis objektu všemi atributy, ale jen jejich náhodně vybranou částí: metoda náhodného podprostoru (*random subspace*)

Bagging

Jedna z nejstarších metod pro tvorbu souboru klasifikátorů

- Velmi intuitivní metoda jednoduchá na implementaci, s překvapivě dobrými výsledky:
  - Velké množství (řekněme 200) trénovacích podmnožin je náhodně vybráno – s náhradou – z výchozích trénovacích dat
  - Každá z těchto trénovacích podmnožin je použita pro vygenerování klasifikátoru (tentýž typ pro všechny podmnožiny)
  - Výsledné jednotlivé klasifikátory se kombinují hlasováním

- Bagging je zvlášť atraktivní pro malé soubory trénovacích dat, protože relativně velká část dat se dostane do každé trénovací podmnožiny

Pseudokód pro bagging

Vstup: $E := \emptyset$

Správně klasifik.trénovací data S zařazená do tříd z množiny $\Omega = \{\omega_1,\ldots,\omega_C\}$.

Alg.stroj.učení **ASU**. Celé číslo T rovné počtu iterací a F díl pro bootstrap. *

Dělej $t = 1,\ldots,T$

- Vytvoř pomocí bootstrapping repliku S_t trén. dat S odpovídající dílu F
- Aplikací **ASU** na S_t vytvoř hypotézu (klasifikátor) h_t
- Přidej klasifikátor h_t do vytvářeného souboru klasifikátorů) E

Použití souboru E pro hlasování o zařazení neklasifikované instance x :

- Nechť $v_{t,j} = 1$ právě tehdy, když hypotéza h_t zařadí x do třídy j (jinak je **0**)
- Vypočti celkový počet hlasů $V_j = \sum_{t=1..T} v_{t,j}$ pro jednotlivé třídy $j = 1, .. , C$
- Zařaď x do třídy, jejíž počet hlasů je nejvyšší

* F určuje, kolik prvků budeme brát z původní množiny mohutnosti **N** v ⍰:
při metodě **bootstrapping** vybíráme **N***F/100 prvků s navracením.

Variace na téma „bagging“

Náhodný les vzniká kombinací klasifikátorů ve tvaru rozhodovacích stromů s odlišnou volbou trénovacích parametrů

- Tyto parametry mohou odpovídat změně trénovacích dat jako je tomu u bagging (*bootstrapped replicas of the training data*),
- Nebo také mohou být důsledkem volby *jiné podmnožiny sledovaných atributů* jako v případě metod s náhodným podprostorem (*random subspace methods*)
- ...

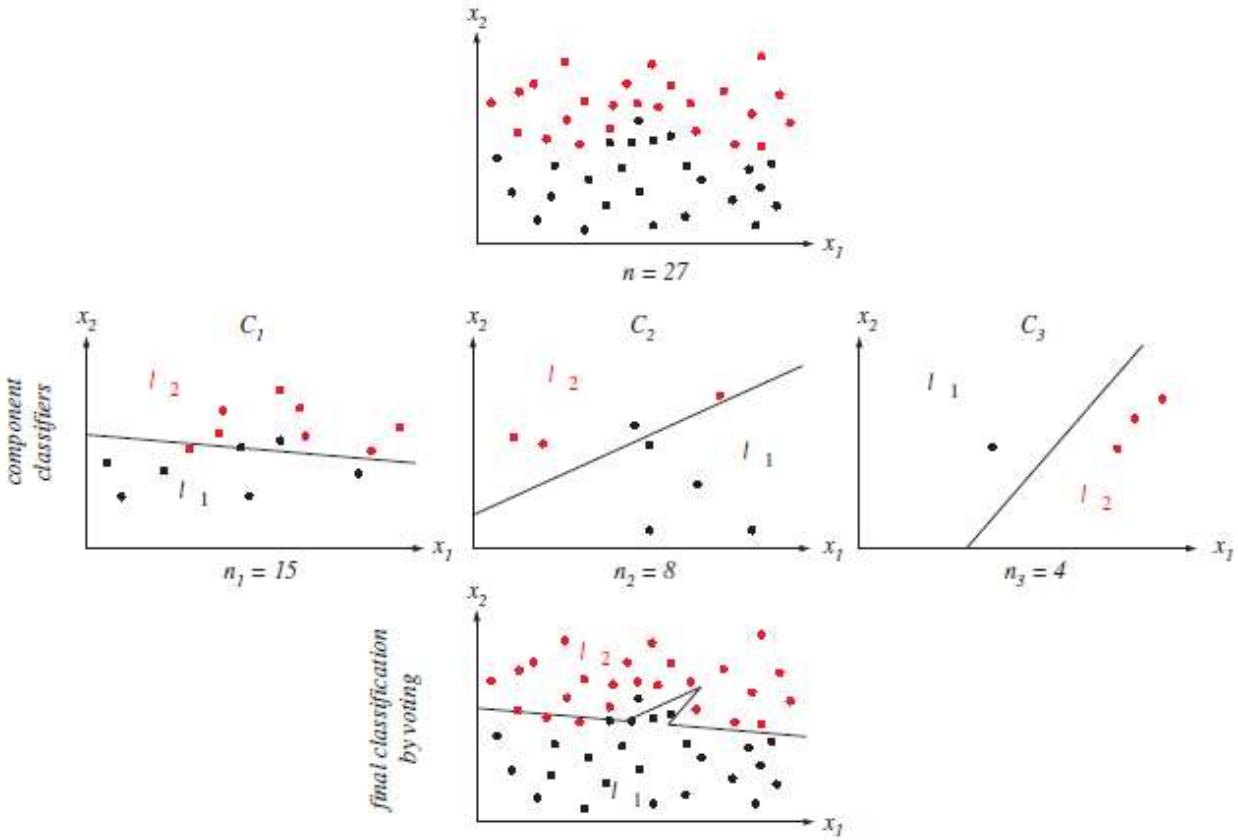
Boosting

- Snaží se vylepšit výkon slabého klasifikátoru (*weak learner*) a tak udělat z něj udělat silný (*strong*) klasifikátor
- **Boosting** se opírá o soubor klasifikátorů, které vznikají cíleným vzorkováním dat a jsou kombinovány pomocí hlasování
  - Postupné **vzorkování je strategicky zaměřováno** tak, aby každý nový vzorek obsahoval nejpotřebnější data pro další rozhodování
- Boosting vytváří **3 slabé** (dobré, ale ne skvělé) **klasifikátory:**

1. První klasifikátor **C1** je natrénován na náhodně vybrané podmnožině **T1** výchozích trénovacích dat
2. Trénovací množina **T2** pro tvorbu druhého klasifikátoru **C2** je vybrána jako **nejinformativnější množina** vzhledem výsledkům klasifikátoru **C1:** polovinu dat v **T2** klasifikuje **C1** správně, druhou polovinu tvoří ty příklady, na kterých **C1** dělá chybu
3. Třetí klasifikátor **C3** je natrénován na těch instancích trénovacích dat, na kterých se klasifikace **C1** & **C2** liší

Boosting

- Uvažujme případ, kdy **ASU** (alg.stroj.učení) hledá **lineární hranici** postupem minimalizace čtverců odchylek (metoda nejmenších čtverců)
- Hlasující trojice má vždy lepší klasifikační přesnost než každý z klasifikátorů samostatně



AdaBoost

- **AdaBoost** (1997) zobecňuje boosting. **AdaBoost.M1** pracuje dokonce s problémy, kde se vyskytuje více tříd
- AdaBoost generuje množinu hypotéz (klasifikátorů), které kombinuje váženým hlasováním o zařazení do tříd tak, jak je navrhují jednotlivé klasifikátory
- Hypotézy vznikají natrénováním slabších klasifikátorů nad vzorky dat, které iterativně mění distribuci trénovacích dat tak, že postupně **zvýhodňují dosud špatně klasifikované instance:** Tedy postupně vznikající klasifikátory jsou trénovány na stále obtížněji klasifikovatelných instancích dat

Pseudokod pro AdaBoost. *Algoritmus M*1

Vstupy: Množina **N** příkladů **S** = $\{ x_i , y_i \}_{i \le N}$ klas. do **c** tříd $y_i \in \Omega = \{\omega_1 ,.. , \omega_c \}$

Slabý algoritmus strojového učení **ASU**

Parametr **T** specifikující maximální plánovaný počet iterací

Iniciace: Distribuce $D_1(i) = 1/ N$ pro $i = 1, \dots , N$

Pracovní cyklus pro $t = 1, \dots , T$:

1.Vyber z **S** trénovací množinu S_t podle distribuce D_t

2.Použij algoritmus **ASU** na S_t a vytvoř tak hypotézu (klasifikátor) h_t

3.Odhadni pravděpodobnost chyby ε_t hypotézy h_t na množině **S** z *počtu a závažnosti chyb*, kterých se h_t dopustí: $\varepsilon_t = \sum_{\{i \le N \,\&\, h_t(x_i) \ne y_i\}} D_t(i)$

4. Pokud $\varepsilon_t > ½$ pak KONEC CYKLU

jinak polož $\beta_t = \varepsilon_t /(1- \varepsilon_t)$ a definuj novou distribuci $D_{t+1}(i) = F_{t+1}(i)/ Z_{t+1}$ snižující váhu správně klasifikovaných příkladů takto:

Definice pomocných funkcí $F_{t+1}(i)$: IF $h_t(x_I) = y_I$ THAN $F_{t+1}(i) = \beta_t$ ELSE $F_{t+1}(i) = 1$

$Z_{t+1} = \sum_{i=1}^{N} F_{t+1}(i)$

Závěrečné rozhodnutí – vážené hlasování pro neklasifikovanou instanci x:

i. Získej pro instanci x a pro každou ze tříd ω_k celkový počet hlasů $V_k (x)$, které jim celkem přiřadí všechny klasifikátory vytvořené během celého pracovního cyklu vážený jejich „kvalitou“

$V_k (x) = \sum_{\{i \le t \,\&\, h_t(x) = \omega_k\}} \log (1/ \beta_t)$

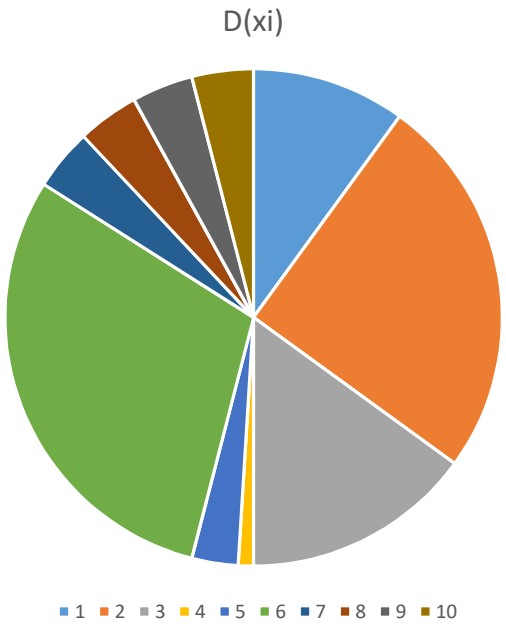
ii. Instanci x přiřaď třídu s nejvyšší vahou!

Výběr podle distribuční fukce: princip ruletového kola

Každému prvku x_i výchozí množiny je přidělena vlastní kruhová výseč o ploše úměrné hodnotě distribuční fukce pro tento prvek $D(x_i)$. Výsledek je podobný koláčovému grafu, viz obr. Dole.

Kolo se roztáčí ***N*** krát a při každém zastavení vybere 1 prvek (jedná se o výběr s navracením) – větší pravděpodobnost výběru mají prvky s vyšší hodnotou distribuční funkce.

i	D(x_i)
1	0,1
2	0,25
3	0,15
4	0,01
5	0,03
6	0,3
7	0,04
8	0,04
9	0,04
10	0,04
suma	**1**



Vzorkování s navracením

AdaBoost algoritmus **pracuje sériově**: klasifikátor (C_K-1) vzniká dřív než klasifikátor C_K

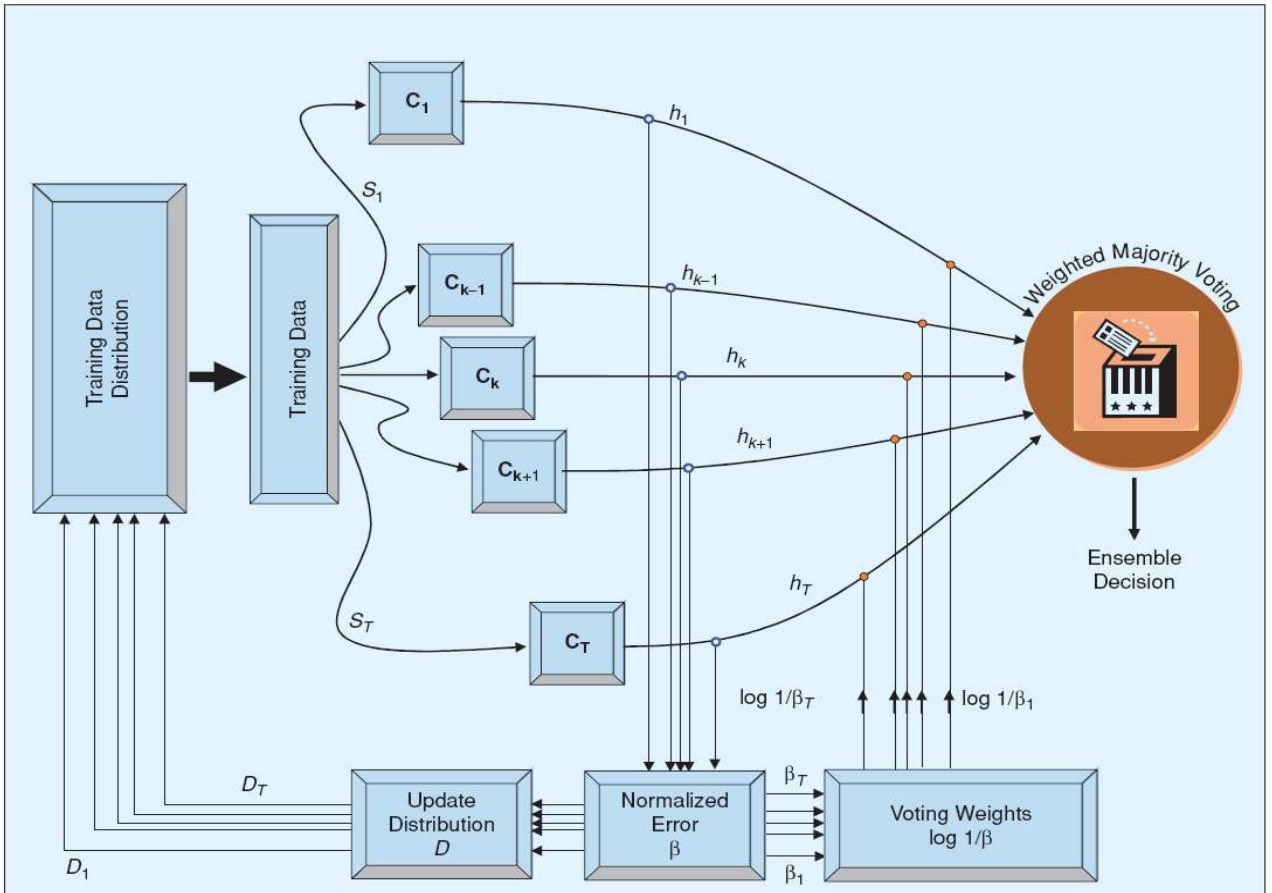
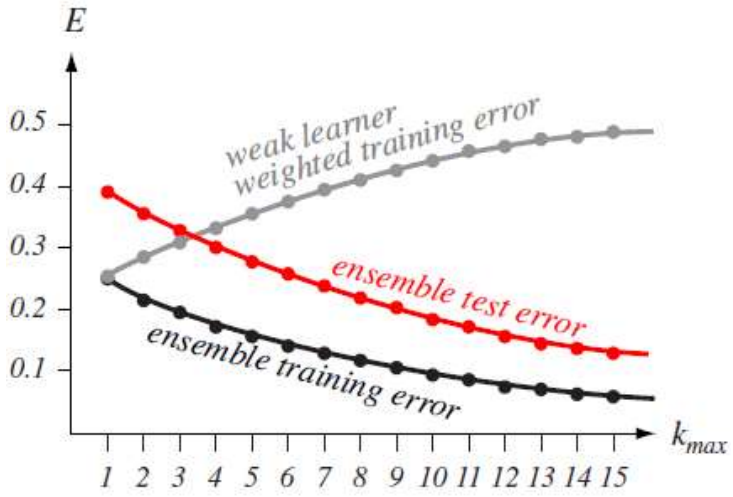


Figure 9. The AdaBoost.M1 algorithm.

AdaBoost a jeho výkonnost

- AdaBoost se slabým klasifikátorem vede s rostoucím počtem kroků k_{max} k exponenciálnímu poklesu velikosti chyby výsledného klasifikátoru.
- Naopak chyba samostatných postupně vytvařených klasifikátorů roste (šedá čára), neboť tyto klasifikátory musí řešit stále těžší úlohy. Přesto, dokud chyba jednotlivých postupně vytvařených klasifikátorů je menší než 0.5, dosahuje jejich **kombinace na trénovacích datech stále menší chyby** (černá čára)**!**
- Vývoj chyby pozorovaný na testovacích datech má velmi často podobný trend jako na datech trénovacích (červená čára).



AdaBoost: empirická pozorování

- Ve většině praktických příkladů, chyba klasifikace prostřednictvím souboru klasifikátorů rychle klesá už během prvních několika iterací a konverguje k nějaké setrvalé rozumné hodnotě (ideálně k 0)
- **Empirické pozorování**: metoda AdaBoost nevede k „přeučení" (*overfitting*); pokus o vysvětlení nabízí *margin theory*

Hierarchické zobecňování (*stacked generalization*)

se snaží rozpoznat spolehlivost jednotlivých vytvořených klasifikátorů (modelů) tak, že konstruuje meta-klasifikátor: Na **1.úrovni** je vytvořen soubor klasifikátorů, jejichž vektor klasifikací (doplněný správnou klas.) tvoří trénovací data pro **2.úroveň:** návrh meta-klasifikátoru, který z výsledků klasifikací souborem klasifikátorů navrhuje výslednou klasifikaci

- Klasifikátory $\boldsymbol{C_1, \ldots, C_T}$ jsou natrénovány s využitím různých parametrů θ_1 až θ_T a tak vzniknou hypotézy (klasifikátory) $\boldsymbol{h_1}$ až $\boldsymbol{h_T}$
-
- Výstupy těchto klasifikátorů doplněné o správnou klasifikaci představují trénovací data pro vznik meta-klasifikátoru $\boldsymbol{C_{T+1}}$

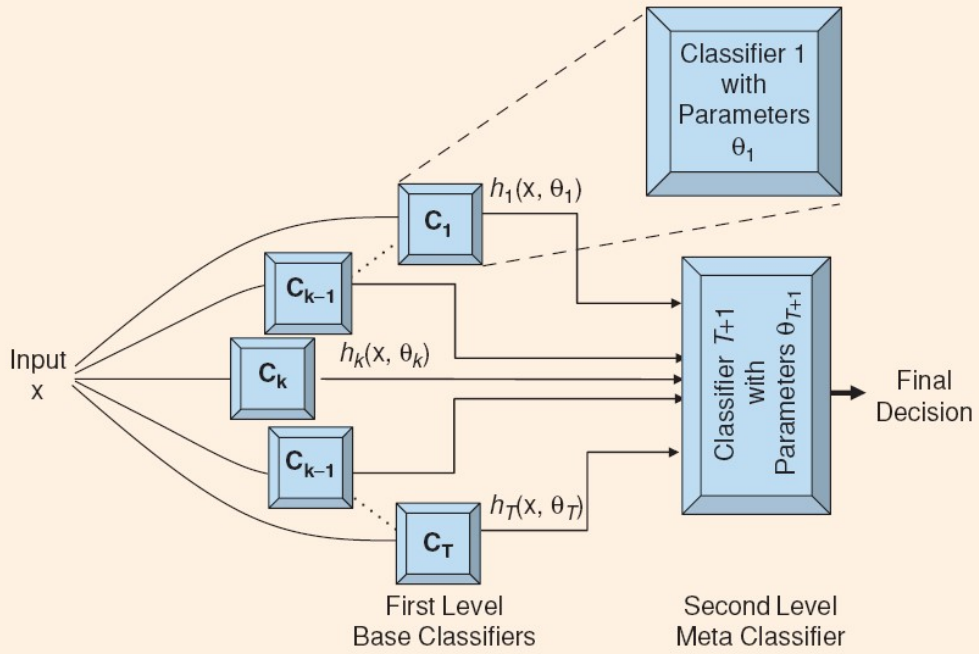


Figure 11. Stacked generalization.

Vzorkování s navracením

- Soubory klasifikátorů jsou prakticky velmi užitečné
- Pro úspěch je důležité zajistit rozdílnost použitých klasifikátorů
- Techniky pro vytvoření souboru: bagging, AdaBoost, směs expertů, podmožiny atributů, …
- Strategie pro kombinování výsledků klasifikátorů: algebraické postupy, hlasování, rozhodovací vzory.
- **Žádná technika ani strategie však není vždy zaručeně nejlepší !**

Další informace:

Polikar R., "Ensemble Learning," Scholarpedia, 2008.

Berka P.: *Dobývání znalostí z databází*, Academia 2003

Rolach Lior: Ensemble-based classifiers, *AI Review*, Vol.33 (2010), pp.1-39
(http://link.springer.com/article/10.1007%2Fs10462-009-9124-7)