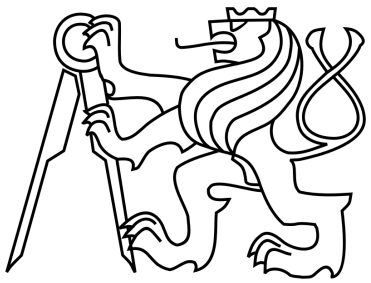
Kombinování klasifikátorů

Ensamble based systems



Nature Inspired
Technologies Group

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 obsazování místa nás zajímají reference na budoucího zaměstnance
- Proč nepoužít stejnou strategii i při automatizaci rozhodování?
- Tento přístup má 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 klas.:
  - (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
  - Kombinování výsledků více klasifikátorů zmenšuje 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

Proč kombinovat klasifikátory ?

- Rozděluj a panuj!
  - Rozděl data do menších lépe popsatelných skupin a nechej každý klasifikátor zvládnout jen jednu z těchto skupin

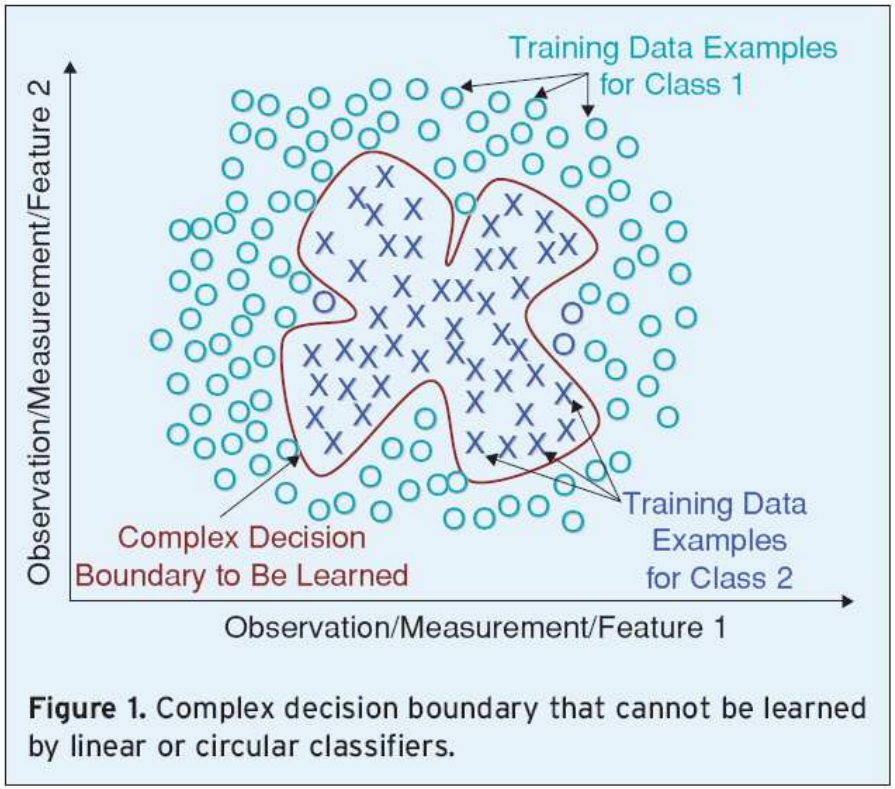


Figure 1. Complex decision boundary that cannot be learned by linear or circular classifiers.

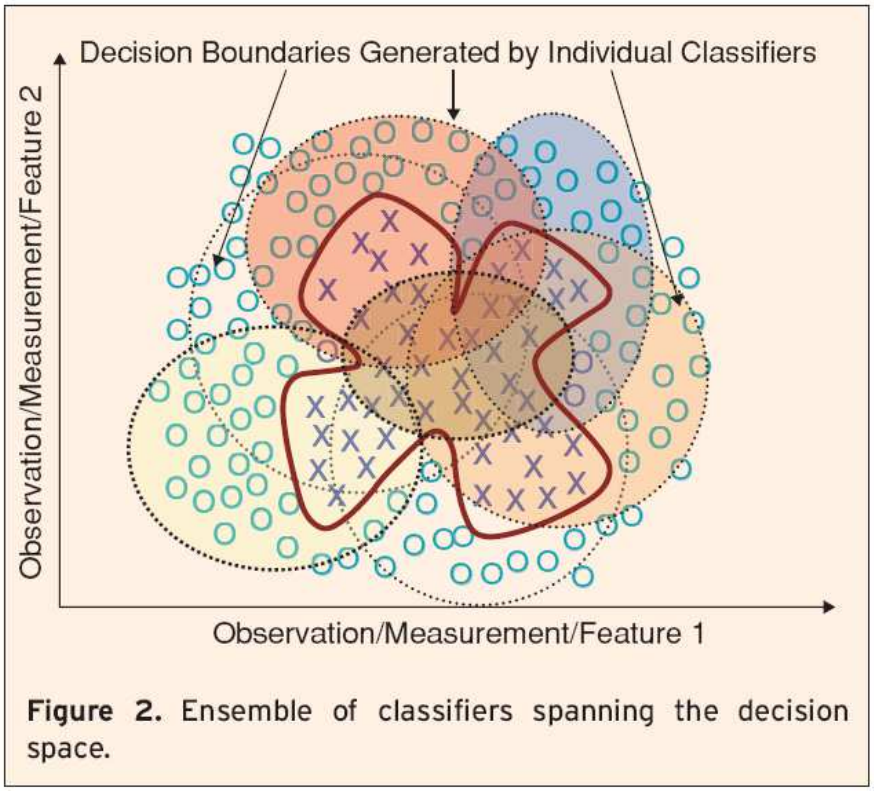


Figure 2. Ensemble of classifiers spanning the decision space.

Proč 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,..)
  - Řada aplikací úspěšně využívá kombinací klasifiktorů pro rozhodování nad různorodými daty - data fusion applications

Něco z historie

- První návrh tohoto postupu: 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 typy kombinace**:
  - **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"
  - **Fuze klasifikátorů** (*classifier fusion*)
    - Všechny klasifikátory jsou trénovány nad celým definičním oborem; fuze pak znamená sdružení jednotlivých slabších klasifikátorů tak, aby vznikl jeden výrazně lepší

Různorodost použitých klasifikátorů

- **Cíl:** vytvořte mnoho klasifikátorů, které pak bude možno kombinovat a získat lepší výsledky
- Intuice: Pokud každý klasifikátor dělá poněkud jiný typ chyb, pak jejich kombinace může být výhodná!
- Potřebujeme tedy klasifikátory takové, že
  - dávají dobré výsledky,
  - mají (*decision boundaries*) vzájemně dost různé
- 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?
    - **Vzorkovací** (*resampling*) techniky: bootstrapping nebo bagging – trénovací podskupiny jsou vybírány náhodně (s navracením) z výchozí dat

Vzorkování s navracením

- 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!

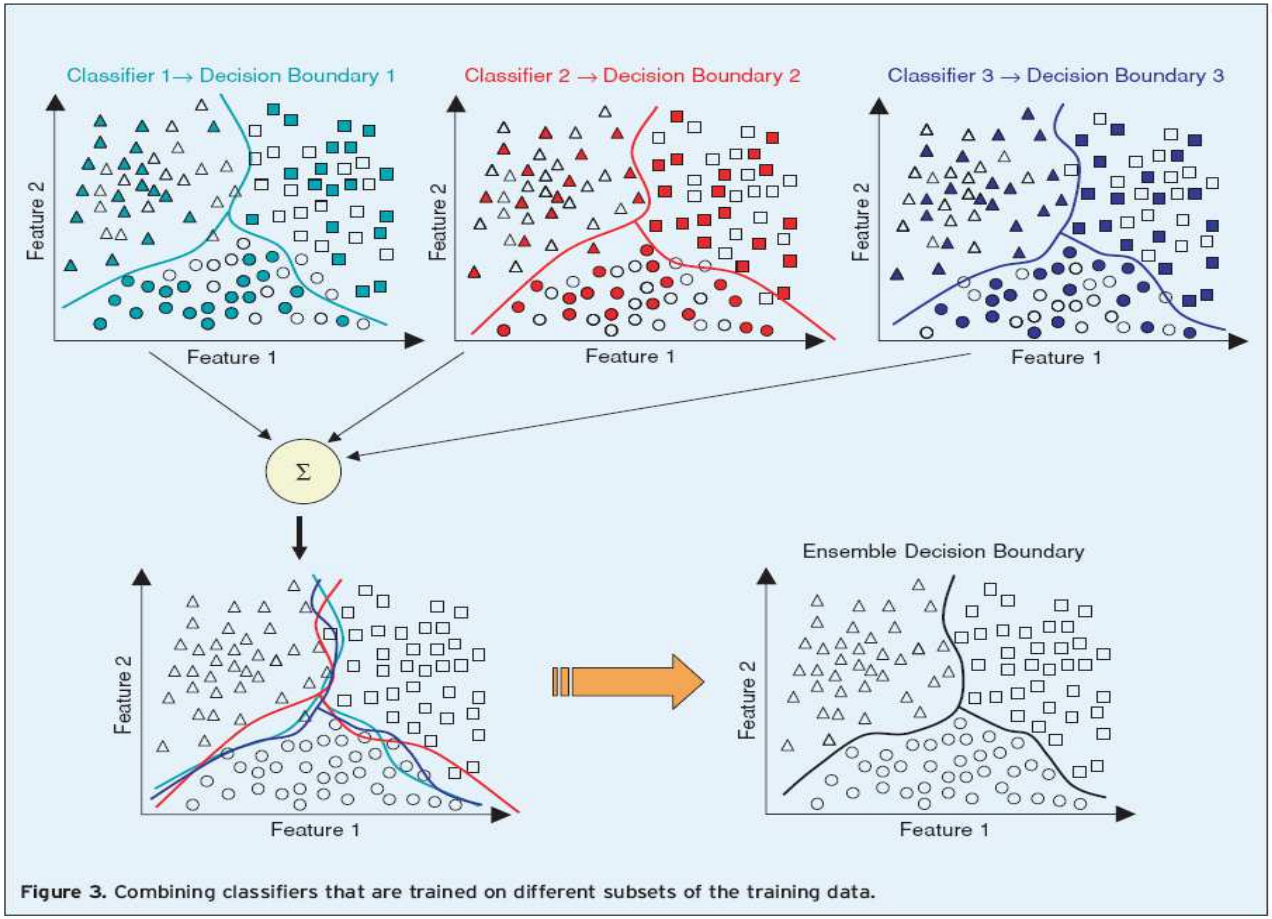


Figure 3. Combining classifiers that are trained on different subsets of the training data.

Jackknife čili k-násobné rozdělení dat

- Celý výchozí soubor dat je rozdělen do k stejně velkých bloků; každý klasifikátor je trénován na skupině dat vzniklé z původních vynecháním jednoho bloku

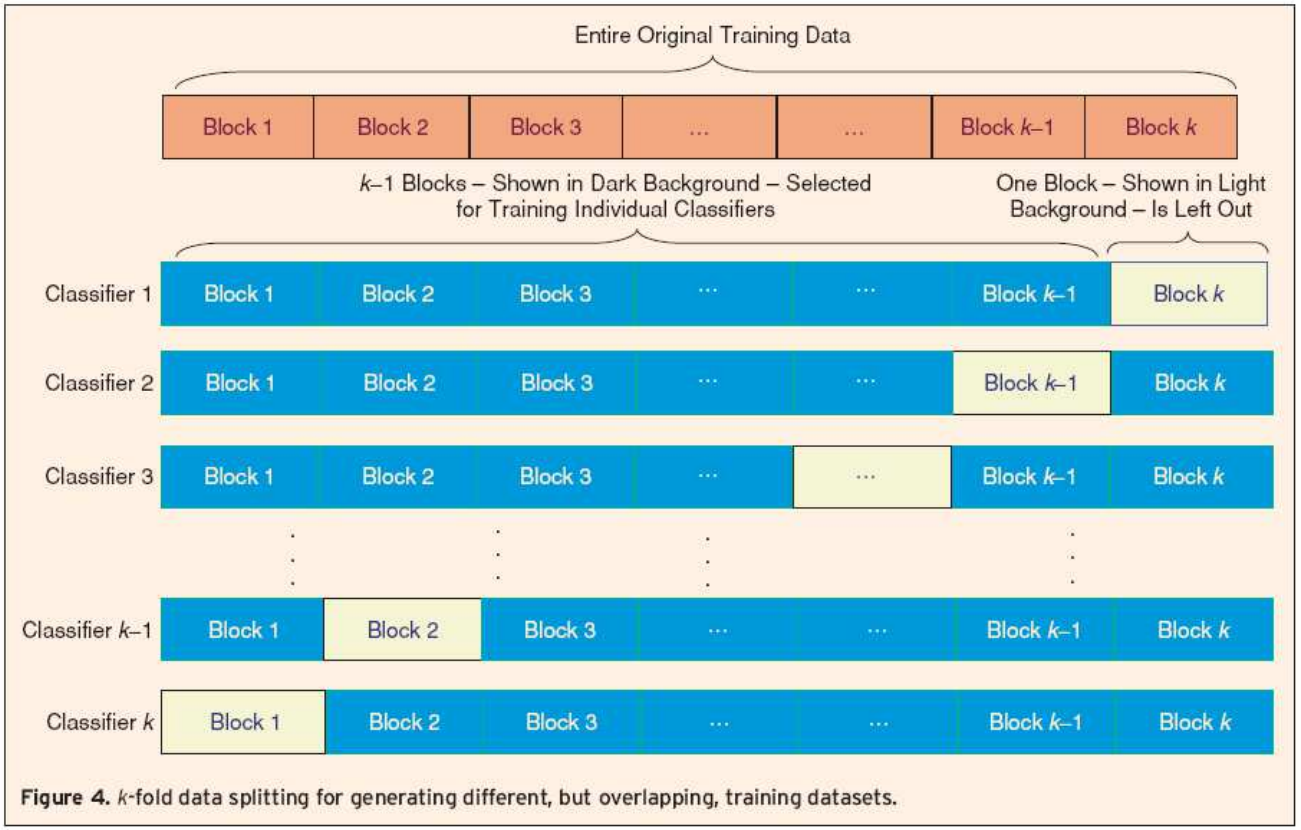


Figure 4. *k*-fold data splitting for generating different, but overlapping, training datasets.

Jiné postupy, jak dosáhnout různorodosti

- Lze kombinovat rozhodnutí různých typů klasifikátorů (vícevrstvý perceptron - MLP, rozhodovací strom, neuronová síť NN, SVM)
- Tentýž typ klasifikátoru lze použit s různými výchozími hodnotami parametrů
  - U MLP nebo NN je rozdíl dán výchozími váhami 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, zkratka za bootstrap aggregating

- 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: $\boldsymbol{E} := \emptyset$

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

Alg.stroj.učení **ASU**. Celé číslo $\boldsymbol{T}$ rovné počtu iterací a $\boldsymbol{F}$ díl pro bootstrap. *

Dělej $t = 1,\ldots,\boldsymbol{T}$

1. Vytvoř pomocí bootstrapping repliku $\boldsymbol{S_t}$ trén. dat $\boldsymbol{S}$ odpovídající dílu $\boldsymbol{F}$
2. Aplikací **ASU** na $\boldsymbol{S_t}$ vytvoř hypotézu (klasifikátor) $\boldsymbol{h_t}$
3. Přidej klasifikátor $\boldsymbol{h_t}$ do vytvářeného souboru klasifikátorů) $\boldsymbol{E}$

Použití souboru $\boldsymbol{E}$ pro hlasování o zařazení neklasifikované instance $\boldsymbol{x}$

1. Zařaď $\boldsymbol{x}$ pomocí všech hypotéz v $\boldsymbol{E}$
2. Nechť $\boldsymbol{v_{t,j}} = 1$ právě tehdy, když hypotéza $\boldsymbol{h_t}$ zařadí $\boldsymbol{x}$ do třídy $\boldsymbol{j}$ (jinak je 0)
3. Vypočti celkový počet hlasů $\boldsymbol{V_j} = \sum_{t=1} \boldsymbol{v_{t,j}}$ pro jednotlivé třídy $j = 1, \ldots, \boldsymbol{C}$
4. Zařaď $\boldsymbol{x}$ do třídy, jejíž počet hlasů je nejvyšší

$\boldsymbol{F}$ určuje, kolik prvků budeme brát z původní množiny mohutnosti **N** v %:
při metodě **bootstrapping** vybíráme **N***$\boldsymbol{F}$/100 prvků s navracením.

Variace na téma „bagging“

Náhodný les vzniká kombinací klasifikátorů, které všechny mají formu rozhodovacích stromů, které se liší volbou trénovacích parametrů

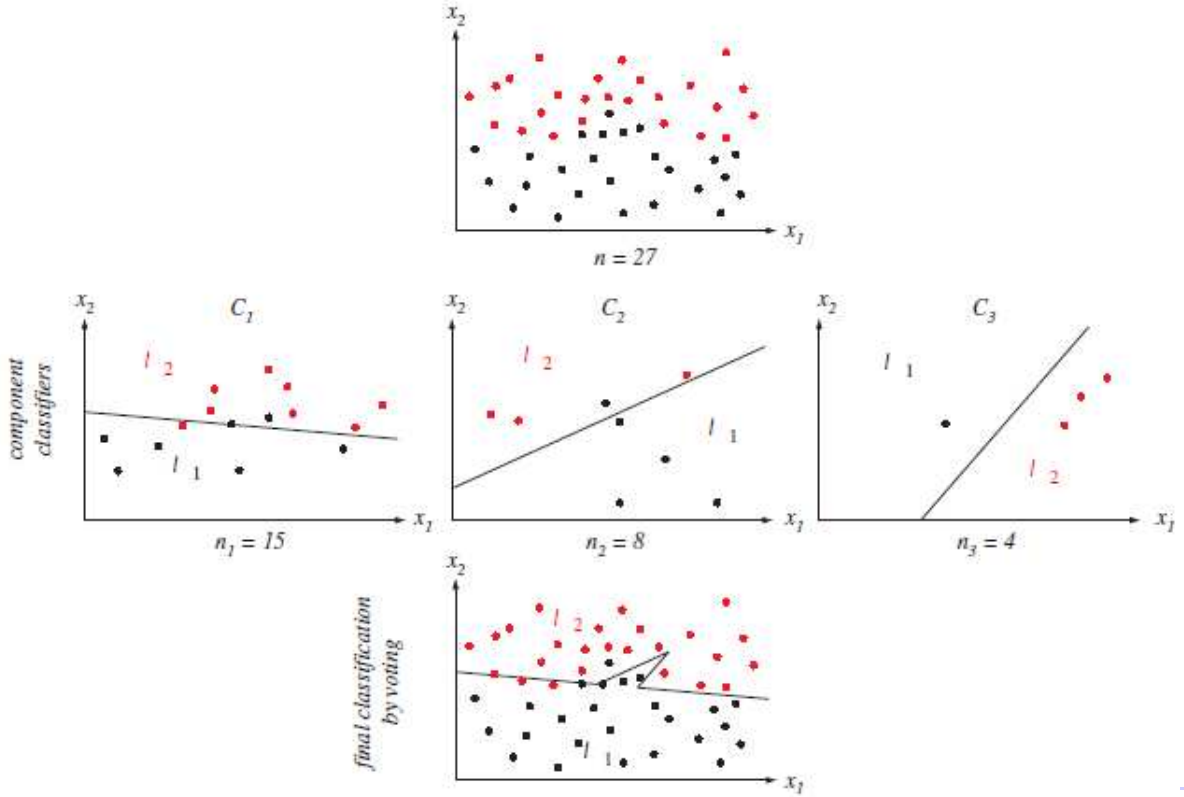
- Tyto parametry mohou odpovídat změně trénovacích dat jako je tomu u bagging (*bootstrapped replicas of the training data*),
- Nebo také mohou být důsledkem volby *jiné podmnožiny sledovaných atributů* jako v případě metod s náhodným podprostorem (*random subspace methods*)

Boosting (vylepšování)

- Snaží se vylepšit výkon slabého klasifikátoru (*weak learner*) a tak udělat z něj udělat silný (*strong*) klas.
- **Boosting** se opírá o soubor klasifikátorů, které vznikají 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 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
- Hlasující trojice má vždy lepší klasifikační přesnost než každý z klasifikátorů samostatně



Nature Inspired Technologies Group

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.M1**

Algoritmus AdaBoost.M1

Vstup : Mnozina N prikladu $S = [(x_i, y_i)], i = 1, \cdots, N$ klas. do trid $y_i \in \Omega, \Omega = \{\omega_1, \cdots \omega_C\}$;

Slaby algoritmus strojoveho uceni **ASU**;

Parametr T specifikujici planovany pocet iteraci.

Inicializace $D_1(i) = \frac{1}{N}, i = 1, \cdots, N$

Delej pro $t = 1,2, \cdots T$:

1. Vyber trenovaci podmnozinu S_t , vybranou podle distribuce D_t .
2. Pouzij **ASU** na S_t a vytvor tak hypotezu (klasifikator) h_t .
3. Vypocti chybu vysledne hypotezy na S

$$h_t : \varepsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$$

Je - li $\varepsilon_t > 1/2$, **KONEC**.

4. Poloz $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$.
5. Uprav distribucni funkci tak, ze se snizi vaha spravne klasifikovanych prikladu

$$D_t : D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t \text{ , pro } h_t(x_i) = y_i \\ 1 \text{ , jinak} \end{cases}$$

kde Z_t je normalizacni konstanta zvolena tak, aby $\sum_i D_{t+1}(i) = 1$, t.j. aby slo o pravedpodobnostni rozlozeni.

Zaverecne rozhodnuti - vazene hlasovani : pro neklasifikovanou instanci x.

1. Ziskej celkovy pocet hlasu pro jednotlive tridy vazeny "kvalitou" klasifikatoru

$$V_j = \sum_{t:h_t(x) = \omega_j} \log \frac{1}{\beta_t}, \; j = 1, \cdots, C.$$

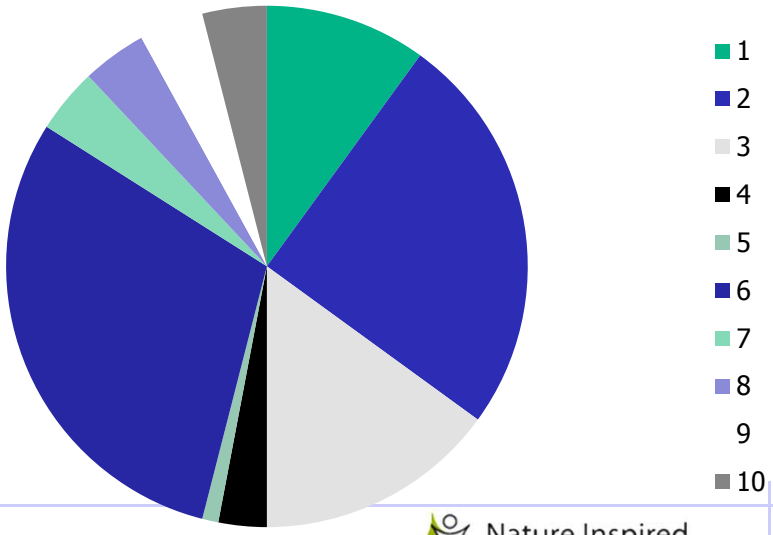
2. Instanci x prirad tridu s nejvyssi vahou!

Výběr podle distribuční fukce:
Jak se provádí ?

Lze použít například **princip ruletového kola**, kdy

1. 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
2. Kolo se roztáčí Nx 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(xi)
1	0,1
2	0,25
3	0,15
4	0,03
5	0,01
6	0,3
7	0,04
8	0,04
9	0,04
10	0,04



AdaBoost.M1

- AdaBoost algoritmus **pracuje sériově**: klasifikátor (C_{K-1}) vzniká dřív než klasifikátor C_K

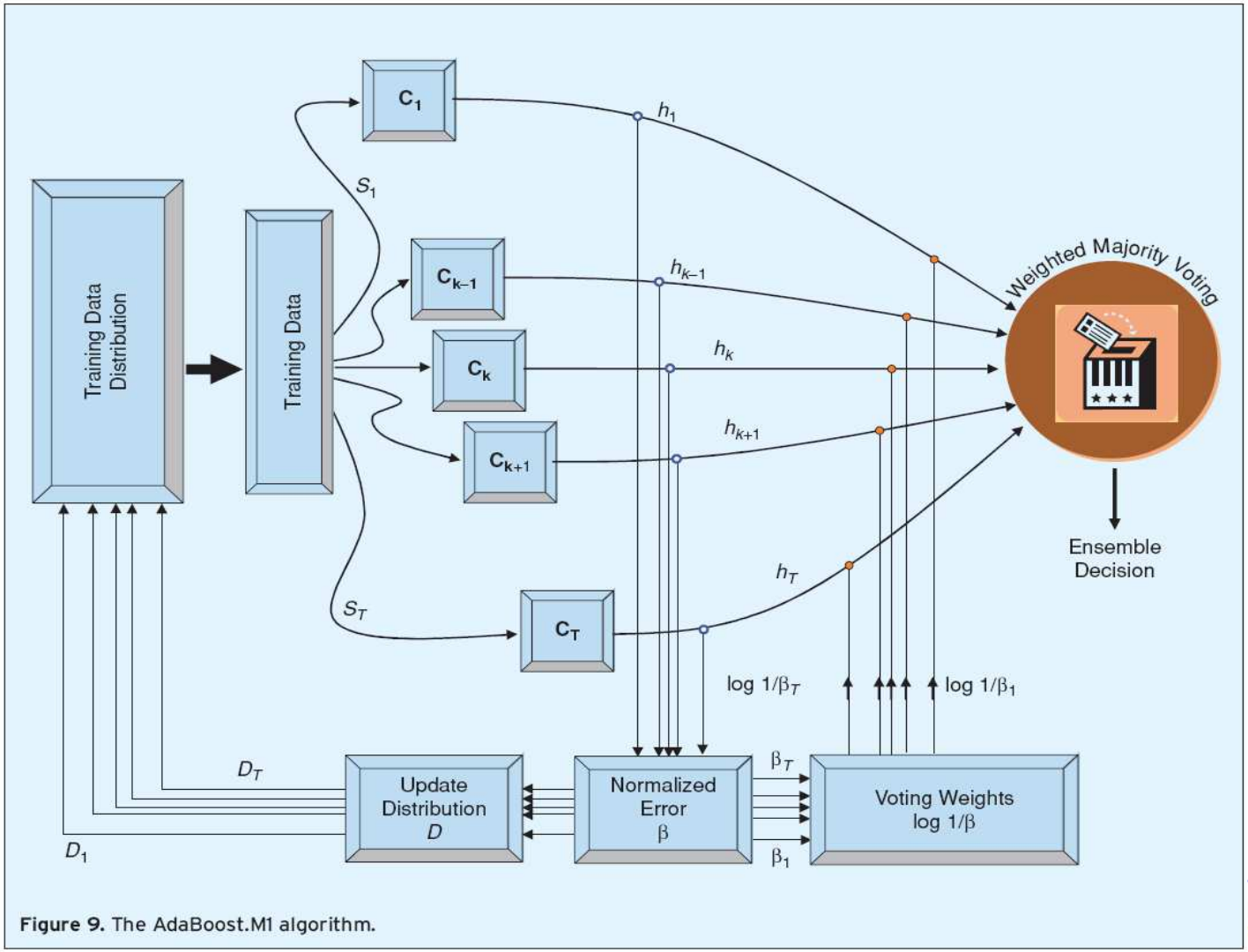
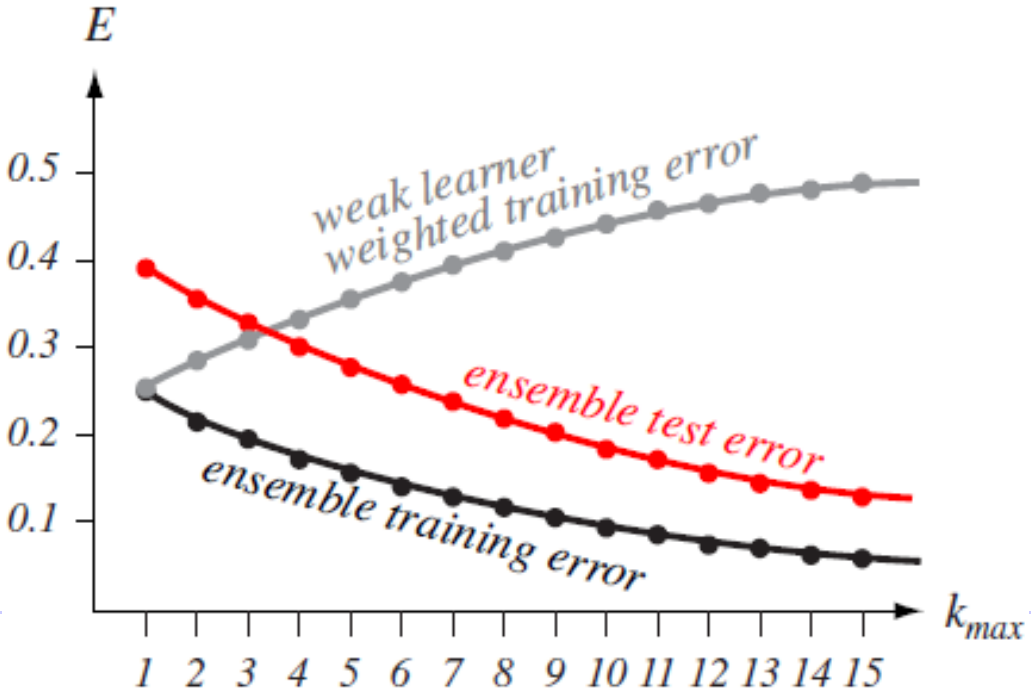


Figure 9. The AdaBoost.M1 algorithm.

AdaBoost

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



Výkonnost pro AdaBoost

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

* 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

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

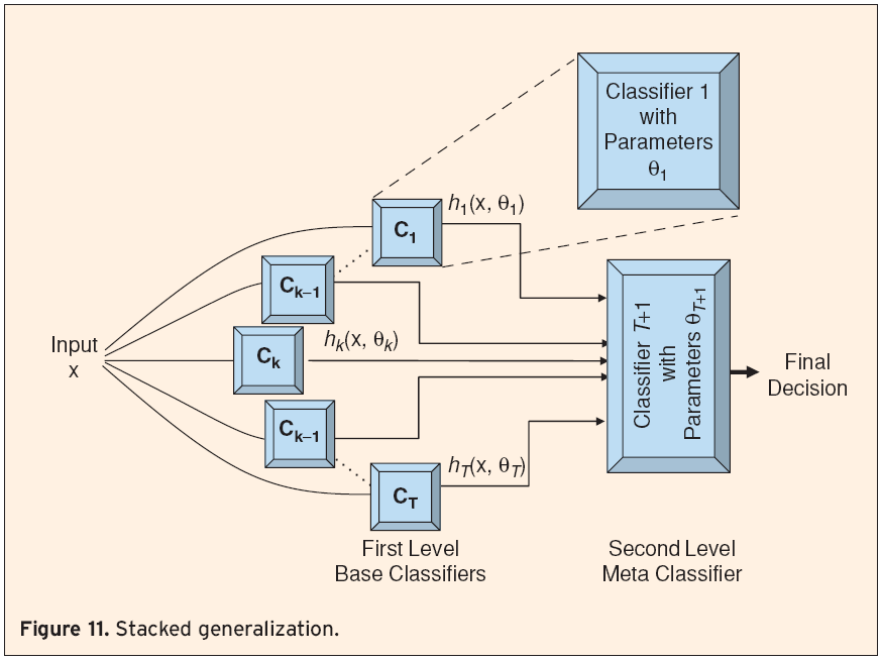


Figure 11. Stacked generalization.

* Mixture of Experts

- The pooling system may use the weights in several different ways.
  - it may choose a single classifier with the highest weight, or calculate a weighted sum of the classifier outputs for each class, and pick the class that receives the highest weighted sum.

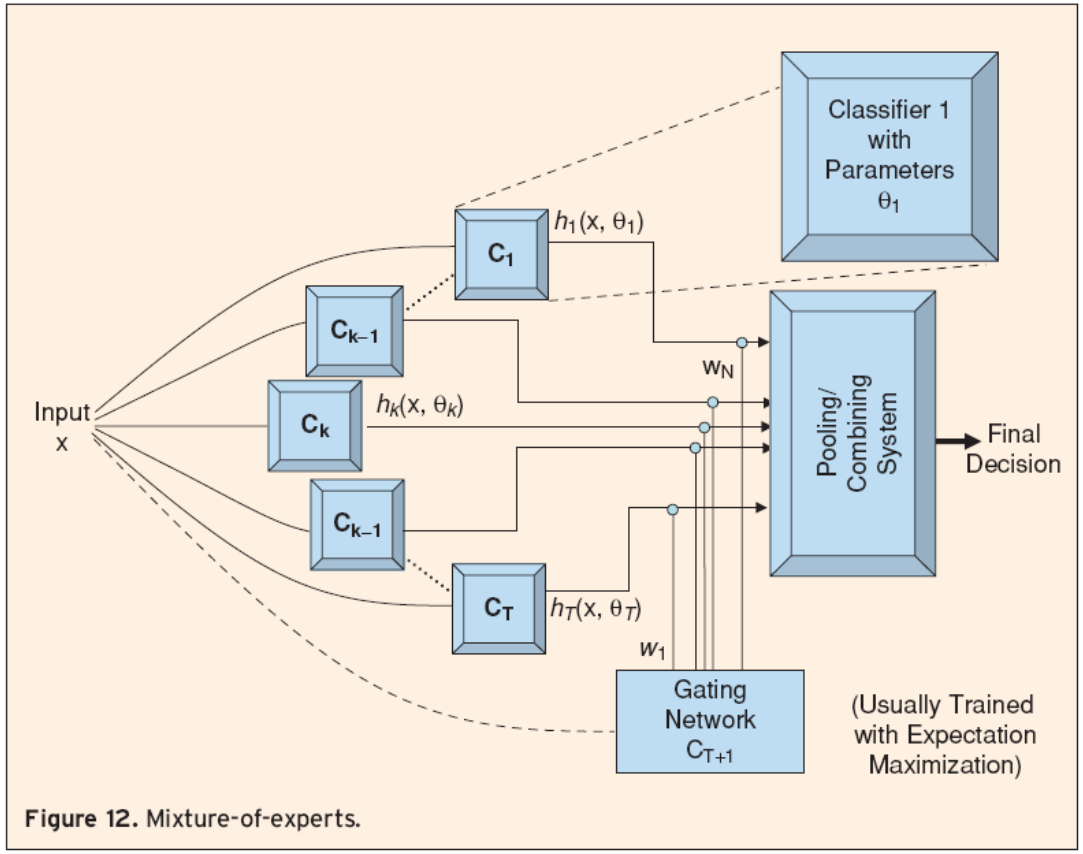


Figure 12. Mixture-of-experts.

Shrnutí

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

Polikar R., "Ensemble Learning," Scholarpedia, 2008.
Berka P.: *Dobývání znalostí z databází*, Academia 2003