VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

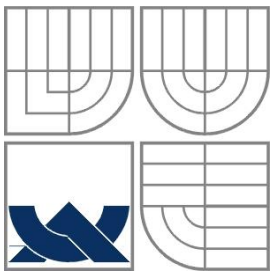
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

APLIKACE ALGORITMU ADABOOST

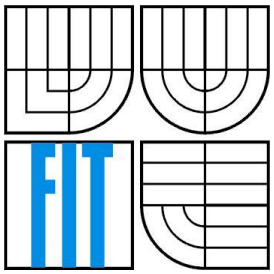
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE VLADIMÍR WRHEL
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

APLIKACE ALGORITMU ADABOOST

APPLICATION OF ADABOOST

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR
VLADIMÍR WRHEL

VEDOUCÍ PRÁCE
SUPERVISOR
ING. MICHAL HRADIŠ

BRNO 2008

Abstrakt

V této práci jsou uvedeny základy klasifikace a rozpoznávání vzorů. Zaměříme se především na algoritmus AdaBoost, který slouží k vytvoření silné klasifikační funkce pomocí několika slabých klasifikátorů. Seznámíme se taktéž s některými modifikacemi AdaBoostu. Tyto modifikace zlepšují některé z vlastností AdaBoostu. Podíváme se taktéž na slabé klasifikátory a příznaky k nim použitelné. Zvláště se podíváme na Haarovy příznaky. Probereme možnosti použití zmíněných algoritmů a příznaků při rozpoznávání výrazu obličeje. Popíšeme si situaci mezi databázemi výrazů obličejů. Nastíníme možnou implementaci aplikace rozpoznávání výrazů obličeje.

Klíčová slova

Klasifikace, rozpoznávání vzorů, neuronové sítě, AdaBoost, klasifikátor, Haarovy příznak, Gáborovy vlnky, rozpoznávání výrazu obličeje, OpenCV

Abstract

Basics of classification and pattern recognitions will be mentioned in this work. We will focus mainly on AdaBoost algorithm, which serves to create a strong classifier function by some weak classifiers. We shall get acquainted with some modifications of AdaBoost. These modifications improve some of AdaBoost attributes. We shall also look into weak classifiers and features applicable to them. We shall especially look into the Haar- likes features. We shall discus possibilities of using the mentioned algorithms and features in facial expression recognition. We shall describe the situation between facial expression databases. We shall draw out a possible implementation of application of facial expression recognition.

Keywords

Classification, pattern recognition, Neural nets, AdaBoost, Haar-like features, Gabor wavelet, facial expression recognition, classifier, OpenCV

Citace

Wrhel Vladimír: Aplikace algoritmu AdaBoost. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Aplikace algoritmu AdaBoost

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Hradiše
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Vladimír Wrhel
13. 5. 2008

Poděkování

Rád bych poděkoval Ing. Michalu Hradišovi, za jeho pomoc, ochotu, trpělivost, odborné vedení a připomínky týkající se této práce.

Obsah

1 Úvod

Počítačová technika je v současné době velmi dynamicky se rozvíjející odvětví. Lidé se pomocí této techniky snaží automatizovat čím dál větší množství úkonů. Z pohledu historie byly ještě nedávno počítače primitivní a veškerá snaha se soustřeďovala na zvládnutí jednoduchých úkonů. S příchodem lepších technologií se však počítače stávaly složitější. Zároveň zvládaly čím dál náročnější úkony. Jak rostla výpočetní síla, byly vymýšleny náročnější algoritmy a nové přístupy k řešení specifických problémů. Jedním z těchto vyvíjejícím se odvětvím je i klasifikace a rozpoznávání vzorů. Výsledky této oblasti se v dnešní době již běžně setkáváme. Příkladem můžou být například OCR programy, či jednoduché detektory obličeje ve fotoaparátech. Zajímavou úlohou z této oblasti je i rozpoznávání výrazu obličeje (Facial Expression), kterému se věnuje tato práce.

Výraz obličeje je jeden ze základních prvků lidské komunikace. Tato nonverbální komunikace nám dokonce v mnoha případech dokáže říct víc než spousta slov. Je to taky první vjem, kterého si při setkání s jinou osobou všimneme a již tomuto vjemu přizpůsobujeme celou komunikaci. Ve světě počítačů je to jeden z kroků ke kvalitní umělé inteligenci. Jak jinak by inteligentní stroj mohl zjistit, jak se cítíme, než že to rozpozná mimo jiné podle našich výrazů. Teoreticky by se dalo rozpoznávání výrazu obličeje použít k měření spokojenosti např. návštěvníků kina.

V druhé kapitole si probereme základy klasifikace a rozpoznávání vzorů. Všeobecně si popíšeme klasifikátor a vyhodnocení jeho chyby. Seznámíme se také s některými klasifikačními algoritmy, jako sou neuronové sítě, Support vector machine nebo metody tyto boosting.

Ve třetí kapitole se podrobněji podíváme na algoritmus AdaBoost. Detailně se zde probírá trénovací algoritmus. Dále si popíšeme některé jeho modifikace, jako např. WaldBoost nebo FloatBoost. V této kapitole se podíváme také na obrazové příznaky. Budeme se zabývat zejména Haarovými vlnkami, které jsou často základem slabých klasifikátorů. Zmíníme i integrální obraz, pomocí kterého se Haarovy příznaky dají implementovat.

Databázemi výrazů obličejů se věnuje čtvrtá kapitola. Rozebereme si zde situaci mezi těmito databázemi. Podmínky a úskalí získání přístupu do nich. Dále se podrobněji podíváme na strukturu a pravidla některých těchto databází.

Pátá kapitola má za úkol rozebrat teoretický přístup k aplikaci na detekci výrazu obličeje. Podíváme se na výběr a úpravu dat, vybereme typ příznaků pro naši aplikaci a vybereme metody na trénování dat.

Naproti tomu šestá kapitola popisuje praktický přístup k této aplikaci. Popíšeme způsob implementace nástrojů nastíněných v páté kapitole. Zmíníme také problémy provázející tuto implementaci.

Předposlední kapitola, tedy sedmá, popíše provedené testy a bude diskutovat jejich výsledky. Podíváme se na vliv počtu slabých klasifikátorů na chybu výsledného klasifikátoru, nebo na vliv počtu trénovacích dat na kvalitu výstupních klasifikátorů.

Osmá kapitola (závěr) pouze shrne a zhodnotí výsledky práce a naznačí další možný vývoj aplikace pro detekci výrazu obličeje.

2 Klasifikace a rozpoznávání vzorů

Klasifikace a rozpoznávání vzorů je typickým využitím oblasti umělé inteligence. Ve velké většině jde o učení s učitelem. Úkolem klasifikace je získávání informací ze vstupních signálů. Toto získávání informací se děje na základě znalosti nebo statistických informací. Signály určené ke klasifikaci jsou typicky výsledkem měření. Signály jsou v prostoru representována příznakem vektorů. Systém klasifikace začíná již získáním dat. Po získání dat je nutné tato data zpracovat a extrahovat z nich vektor příznaků. Po získání tohoto vektoru příznaků můžeme přistoupit k samotnému trénování klasifikátorů a následně ke klasifikaci. Výsledek klasifikace vždy závisí na kvalitě vstupních dat, vektoru příznaků z nich naměřených a v neposlední řadě taktéž na metodě použité k trénování klasifikátoru. Výběr metody je důležitý z důvodu kvality výstupu ale taktéž požadavkům na množství trénovacích dat, výpočetní sílu či jiné zdroje.

Klasifikace a rozpoznávání vzorů jsou metody použitelné v širokém spektru úloh. Použitelnost těchto metod šahá od dolování dat až po rozpoznávání řeči. Klasifikace je taktéž masivně nasazena v oblasti počítačového vidění. V oblasti počítačového vidění je úkolem klasifikace zařadit obraz do jedné ze dvou, výjimečně i více, tříd. Tím se dá například určit, zda se nějaký tvar v obraze nachází nebo nikoliv. První z těchto tříd je hledaný obraz, tedy například auto, druhou třídou je pak opak tohohle obrazu (pozadí), tedy vše co nechceme, aby náš klasifikátor jako auto detekoval. Příkladem vícetřídní klasifikace v počítačovém vidění můžou být tzv. OCR programy. Třída hledaných obrazů není nijak omezena. Klasifikaci lze využít například k rozpoznávání výrazu obličeje.

Klasifikace se dá využít ve velkém množství komerčních nebo průmyslových oblastech. V mnoha případech jde o aplikace, ve kterých je klasifikace pouze jako součást širšího celku. Tento celek může využívat řadu jiných metod. Příkladem využití klasifikace můžou být jednoduché detektory obličeje ve fotoaparátech, OCR programy, nebo kontrola dopravy (rozpoznávání SPZ).

2.1 Klasifikátor

Klasifikátor je funkce, která, co možná nejlépe, rozděluje klasifikační třídy. Ve velkém množství případů se jedná o jednoduchý práh tzv. threshold. Threshold rozděluje klasifikační třídy podle některé hodnoty vektoru příznaků. Daný threshold by měl být doplněn ještě o polaritu, aby bylo zřejmé, do které třídy máme prvek zařadit. Pomocí polarity odlišíme případy, kdy jsou prvky první třídy menší než práh klasifikátoru a naopak, kdy jsou prvky této třídy větší než práh.

2.1.1 Trénování klasifikátoru

Abychom byli schopni použít klasifikátor pro rozhodování o příslušnosti prvku do tříd, musíme jej nejprve natrénovat. To se typicky provádí pomocí metody učení s učitelem. Pod pojmem učení s učitelem se chápe takové učení klasifikátoru, kdy mu předkládáme jak příznakové vektory trénovacích dat, tak i jejich správnou příslušnost ke třídě. Protože trénovací sada nedokáže, až na některé speciální případy, plně popsat všechny možné hodnoty vstupních dat, které má klasifikátor zařadit, bude mít klasifikátor vždy určitou chybu. Tato chyba se dá minimalizovat vhodným výběrem trénovací dat. Čas potřebný pro trénování a vyhodnocení klasifikátoru narůstá se složitostí klasifikátoru. Taktéž platí, že jednodušší klasifikátory jsou rychlejší. To je vykoupeno jejich vyšší chybovostí.

Před samotným trénováním musíme trénovací data nejdříve vhodně transformovat do vektoru příznaků. Trénování je pak iterační proces, ve kterém se opakují 2 následující kroky[5]:

1. *Klasifikace vzorků*: Vektor příznaků se použije ke klasifikaci vzorků aktuální verzí klasifikátoru. Takto se zjistí jeho chyba.
2. *Úprava klasifikátoru dle chyby*: Podle chyby klasifikace z předchozího kroku se trénovací algoritmus pokusí opravit klasifikační funkce.

Iterace se vlastně skládá ze dvou kroků. V prvním z nich se použije aktuální klasifikátor pro zjištění chyby. V druhém kroku se tento klasifikátor upraví tak, aby se chyba co nejvíce snížila. Jednotlivé kroky iterace se opakují, dokud není splněna podmínka zastavení trénování. Existuje několik různých typů podmínek. Nejjednodušší kritérium je počet iterací. Dalším možným je například pokles chyby pod určitou mez. Metoda, která dosahuje nejlepších výsledků, je použití validační sady vzorků. Trénování pokračuje, dokud chyba na trénovací i validační sadě klesá. Ve chvíli, kdy začne chyba na validační sadě stoupat, je potřeba zastavit trénování, protože klasifikátor začíná být přetrénovaný.

2.1.2 Chyba klasifikátoru

Pro vyhodnocení kvality klasifikátoru musíme znát jeho chybu. Chyba klasifikátoru se určuje na testovací sadě vzorků. Existuje několik typů chyb, které jsme schopni zjistit [28]. Základní znalostí pro zjišťování chyb jsou hodnoty True Positive (TP, správné přijetí), True Negative (TN, Správné odmítnutí), False Positive(FP, chybné přijetí) a False Negative (FN, chybné odmítnutí). Z těchto hodnot můžeme vypočítat relativní četnost chyby (**er**ror **r**ate). Tato chyba je pro dané nastavení klasifikátoru poměr chybně klasifikovaných vzorků k celkovému počtu vzorků (1).

$$err = \frac{FP + FN}{TP + TN + FP + FN} \tag{1}$$

Další hodnotou může být celková správnost (overall **acc**uracy viz (2)). Jde o přesný opak četnosti chyby, tj. poměr správně hodnocených vzorků k celkovému počtu vzorků. Přesnost (**prec**ision, (3)) charakterizuje tu část získaných výsledků, která je pro uživatele relevantní. Senzitivita (**rec**all, (4))

odpovídá té části všech možných výsledků, kterou metoda nalézá. Poslední zmíněnou bude specificita (5), která charakterizuje úspěšnost klasifikace u negativních příkladů.

$$acc = \frac{TP + TN}{TP + TN + FP + FN} = 1 - err \tag{2}$$

$$prec = \frac{TP}{TP + FP} \tag{3}$$

$$rec = \frac{TP}{TP + FN} \tag{4}$$

$$spec = \frac{TN}{TN + FP} \tag{5}$$

Pokud se budeme dále bavit o chybě klasifikátoru a nebude uvedeno jinak, bude myšlena relativní četnost chyby.

Dalším možným způsobem representace chyby je tzv. Receiver Operating Characteristic křivka (ROC křivka). Ve své podstatě se jedná o závislost FP a FN při různých nastaveních klasifikátoru. O kvalitě modelu rozhoduje plocha pod ROC křivkou. Na této křivce lze určit hodnotu Equal Error Rate, kdy má klasifikátor stejnou míru chybných přijetí a chybných odmítnutí[5].

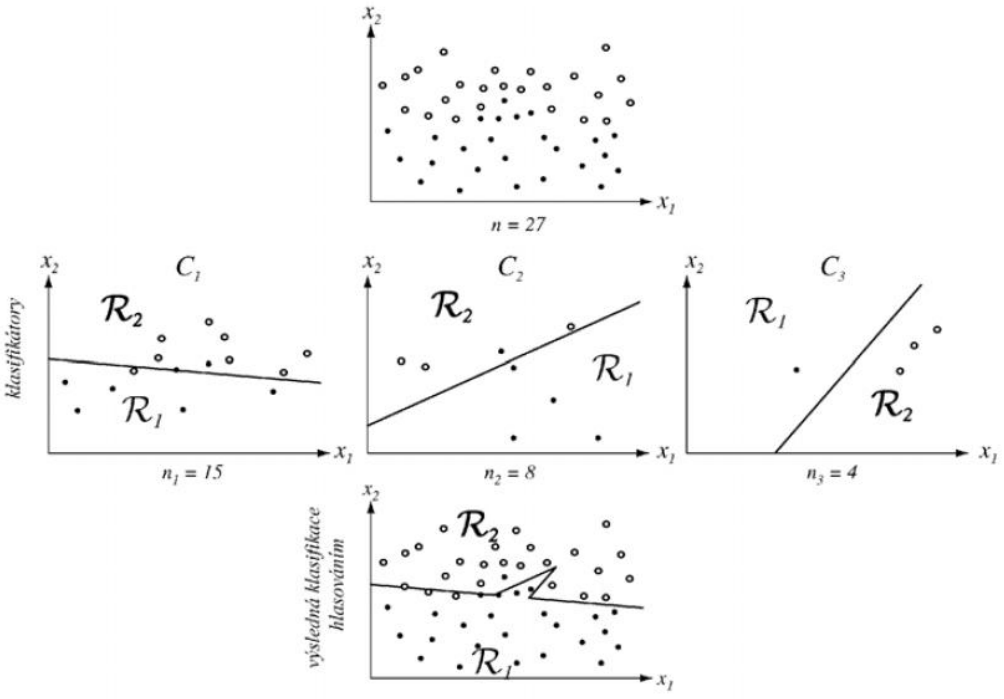
2.2 Bagging

Metoda bagging [12] (***b**ootstrap **agg**regation*, „rozdělování do sáčků“) patří k nejjednodušším postupům. Využívá se více trénovacích množin. Každá z nich je vytvořena výběrem $n' < n$ příkladů ze základní trénovací množiny D. Každá z takto vzniklých podmnožin $d_i \subset D$ je použita k natrénování jednoho z více klasifikátorů. Výsledná klasifikace je provedena „hlasováním“ všech takto vytvořených klasifikátorů. To znamená, že vzorek je zařazen do třídy určené většinou.

Bagging obecně zlepšuje vlastnosti nestabilních klasifikátorů vzhledem k tomu, že se bere do úvahy průměr z názorů jednotlivých klasifikátorů, avšak teoretický důkaz tohoto zlepšení pro libovolný případ neexistuje. Metoda vychází z experimentálních zkušeností. Velmi často vede ke zlepšení, ale není záruka, že bude fungovat vždy.

2.3 Boosting

Cílem metody boosting[12] je zlepšení klasifikační přesnosti libovolného algoritmu strojového učení. I zde je základem vytvoření více klasifikátorů pomocí výběru vzorků ze základní trénovací množiny. Boosting vychází z vytvoření prvního klasifikátoru, jehož klasifikační přesnost je lepší než 50%. Dále jsou přidávány další klasifikátory mající stejnou klasifikační vlastnost, takže je vygenerován soubor klasifikátorů, jehož celková klasifikační přesnost je libovolně vysoká vzhledem ke vzorkům v trénovací množině. Tímto byla klasifikace zesílena (Boosted).



Obr. 1: Příklad klasifikace pomocí metody Boosting[12]

Předcházející obrázek ukazuje případ klasifikace dvourozměrného problému, kde jsou dvě klasifikační třídy (symbolizované znaky R_1 a R_2) a tři lineární klasifikátory (zde typu Widrow-Hoff, tj. gradientní hledání lineární hranice oddělující třídy vzhledem k nejmenšímu čtverci chyb). Hlasující trojice má v tomto případě vždy lepší klasifikační přesnost než jednotlivé klasifikátory trénované na podmnožinách. Přesnost je také vyšší, než kdyby byl vygenerován jediný klasifikátor na všech trénovacích vzorcích.

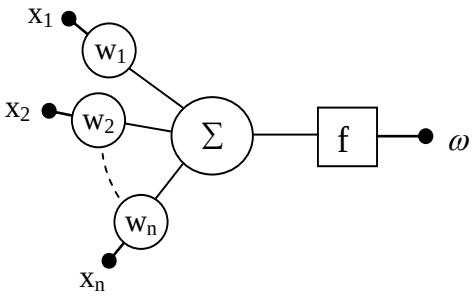
Z metody boosting vychází celá řada metod. Jednou z nejznámějších je AdaBoost. Existuje i celá řada modifikací algoritmu AdaBoost. Algoritmu AdaBoost se budeme věnovat v následující kapitole 3.

2.4 Neuronové sítě

Vzorem neuronových sítí je chování odpovídajících biologických struktur. Umělá neuronová síť je struktura určena pro distribuované paralelní zpracování dat. Základním stavebním prvkem neuronových sítí je umělý neuron. Tyto neurony jsou v síti propojeny tak, aby vykonávali požadovanou funkci. Kladnou vlastností je, že pokud v rozsáhlé síti přestane pracovat několik neuronů, celkový chod sítě zůstane téměř nezměněn. Pomocí neuronových sítí je možné realizovat aplikace z oblasti predikce, rozpoznávání, filtrace aj.

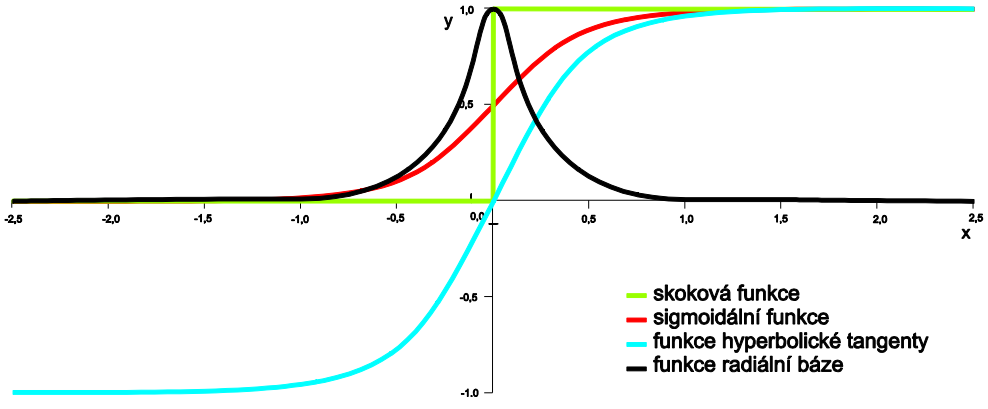
2.4.1 Neuron

První matematický model neuronu vytvořili McCulloch a Pitts[11] v roce 1943 a tento model se dodnes používá pro běžné aplikace. V umělém neuronu (obr. 2) dochází k náhradě funkcí biologického neuronu funkcemi matematickými. Obecně se můžeme setkat s velkým počtem různých typů modelů umělých neuronů, které se navzájem liší vlastní topologií a typy matematických funkcí, které jejich chování popisují. V podstatě se umělý neuron skládá z vektoru vstupů, z vektoru vah vstupů a z aktivační funkce. Jeho výstup je pak dán skalárním součinem vektorů transformovaný aktivační funkcí.



Obr. 2: Model umělého neuronu

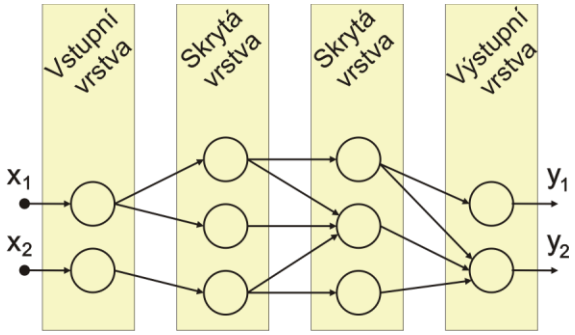
Důležitou vlastností aktivační funkce je její nelinearita. V neuronových sítích se používá několik typů aktivačních funkcí. Pro příklad si uvedeme několik základních aktivačních funkcí [11] (obr. 3). Skoková přenosová funkce vrací pro vstup menší než daná mez nulu, pro větší vrací jedna. Sigmoidální přenosová funkce je ve tvaru $f(x) = {}^{1}/_{1 + e^{-kx}}$. Její hodnoty se blíží nule v minus nekonečnu a jedničce v nekonečnu. Pro nulu je hodnota 0,5. Výhodou sigmoidální přenosové funkce oproti skokové je existence rozumné derivace v každém bodě. Přenosová funkce hyperbolické tangenty je ve tvaru $f(x) = {}^{2}/_{1 + e^{-kx}} - 1$. Její hodnoty se blíží k -1 v minus nekonečnu a k jedničce v nekonečnu. Pro nulu je hodnota 0. Přenosová funkce radiální báze je ve tvaru $f(x) = e^{-kx^2}$. Její hodnoty se blíží nule v minus nekonečnu i v nekonečnu. Pro nulu je maximální hodnota 1.



Obr. 3: Některé aktivační funkce

2.4.2 Typy neuronových sítí

Neuronové sítě se dají dělit podle několika kritérií. Jedním z těchto kritérií je i systém proudění informací uvnitř sítě. Podle tohoto kritéria rozdělujeme neuronové sítě na dopředné sítě a rekurentní sítě. U dopředných sítí putují informace vždy od výstupů přímo k vstupům následující vrstvy. Nejjednodušším případem dopředné sítě je jednoduchý perceptron, který je složen pouze z jediné vrstvy neuronů. Vícevrstvý perceptron je již mnohem komplexnější. Na rozdíl od jednoduchého perceptronu je schopen řešit např. XOR problém. Vícevrstvý perceptron (obr. 4) obsahuje jednu vstupní vrstvu, jednu nebo více skrytých vrstev a právě jednu výstupní vrstvu. Vícevrstvý perceptron dokáže aproximovat libovolnou reálnou funkci[7].



Obr. 4: Vícevrstvý perceptron

Rekurentní sítě na rozdíl od dopředených obsahují zpětné vazby. Typickým příkladem je Hopfieldova síť. Tato síť obsahuje skupinu neuronů, které jsou propojené každý s každým, tj. tvoří úplnou (cyklickou) topologii sítě. Výpočet rekurentních sítí se provádí do ustálení stavu, což je zaručeno. Použití rekurentních sítí je např. jako asociativní paměť, klasifikátor, nebo se používá při optimalizaci[29].

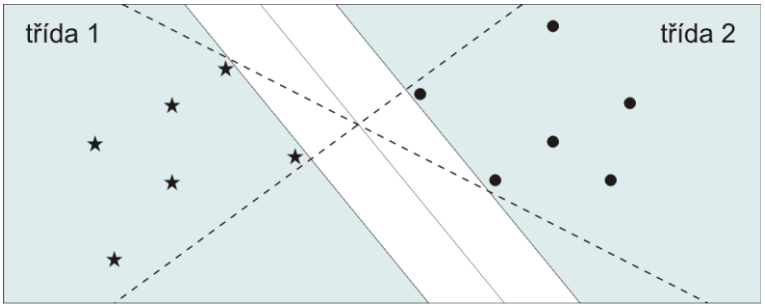
2.4.3 Trénování

Často využívaným algoritmem pro trénování dopředných sítí je algoritmus zpětného šíření chyby (Back-Propagation [8]). Jedná se o metodu učení s učitelem. Na vstup sítě se postupně vkládají vektory vstupních dat. V každém výstupním neuronu se vypočítá chyba. Nastavením vstupních vah vektoru se chyba minimalizuje. Postup se opakuje postupně s vektory předchozích vrstev, dokud se nedojde ke vstupní vrstvě. U neuronových sítí si musíme dávat pozor, abychom síť nepřetrénovali.

Další metodou může být například SON (Self-Organizing Network [29]) Jedná se o algoritmus učení bez učitele. Využívá metody shlukování. Podobná vstupní data aktivují neurony, které jsou blízko sebe. Díky schopnosti projekce se SON používají pro snížení dimensionality a vizualizaci dat.

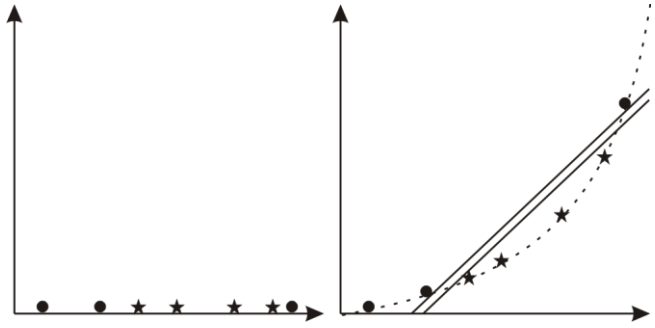
2.5 Support vector machine

Support vector machine (SVM) [9] je algoritmus realizující princip minimalizace strukturálního risku. Problém minimalizace strukturálního risku je převeden na problém maximalizace vzdálenosti rozdělující nadroviny klasifikátoru k bodům z trénovací množiny. V podstatě konstruuje dělící nadrovinu v prostoru vyšší dimenze, než kterou měl prostor vstupu. Na obrázku 5 jsou data ze dvou tříd. Čárkované čáry jsou hyperplochy s minimální dělící vzdálenosti. Plnou čárou je zvýrazněna hyperplocha, která data dělí nejlépe.



Obr. 5: Ukázka maximalizace vzdálenosti dělící hranice od dat.

Důležitou součástí techniky SVM je jádrová transformace prostoru příznaků dat do prostoru transformovaných příznaků typicky vyšší dimenze. Tato jádrová transformace umožňuje převést původně lineárně neseparovatelnou úlohu na úlohu lineárně separovatelnou, na kterou lze dále aplikovat optimalizační algoritmus pro nalezení rozdělující nadroviny (obr. 6). Použitím transformací do více dimenzí můžeme docílit i vícetřídní klasifikaci dat[10].



Obr. 6: Lineárně neseparovatelná data a jejich transformace.

2.6 Gausovské modely

V případech, kdy je vhodné použít statistické rozpoznávání vzorů, se uplatňuje modelování hledaných tříd pomocí směsi gausovských funkcí ve vektorovém prostoru příznaků. Tahle metoda je založena na tom, že prvky jedné třídy budou mít určité statistické vlastnosti společné. Využití je například ve zpracování řeči pro rozeznávání fonémů nebo pro detekci části lidského těla pomocí barvy kůže[5].

3 AdaBoost a jeho modifikace

AdaBoost (Adaptive Boosting) je v současnosti nejpoužívanější variantou metody boosting. Jeho autory jsou Yoav Freund a Robert E. Schapire. AdaBoost umožňuje při návrhu přidávat slabé klasifikátory tak dlouho, dokud není dosažena určitá požadovaná hodnota chyby silného klasifikátoru. Algoritmus AdaBoost je schopen exponenciálně snižovat chybu výsledného klasifikátoru na libovolně nízkou úroveň [1]. AdaBoost dokáže produkovat klasifikátory s velmi dobrými vlastnostmi i za použití jednoduchých klasifikátorů. Při použití v reálných podmínkách jsou klasifikátory natrénované pomocí metody AdaBoost přesné a jejich vyhodnocení se dá provést velice rychle, což se dá s výhodou využít například v real time aplikacích.

Použití algoritmu AdaBoost je vhodné například pro úlohu detekce obličeje v obraze či detekce obličeje ve videosekvenci v reálném čase [2]. Dále se dá s úspěchem použít pro hledání „čehokoliv“ v obraze, rozpoznávání výrazů obličeje, rozpoznávání SPZ nebo rozpoznávání řeči.

Slabé klasifikátory mohou být libovolně složité funkce. Nejčastěji se používá tzv. threshold zmíněný v kapitole 2.1. V dřívějších dobách se používaly např. neuronové sítě[5] atd.

Existuje celá řada modifikací algoritmu AdaBoost. Tyto modifikace obecně zlepšují některé vlastnosti algoritmu. Rychlost výpočtu snižuje např. algoritmus WaldBoost. Chybu výsledného klasifikátoru zmenšují algoritmy FloatBoost a TCAcu.

3.1 Algoritmus

AdaBoost je učící algoritmus, jehož vstupem je trénovací množina a výstupem klasifikátor do dvou tříd. Existují i modifikace algoritmu, které umožňují všetřídní klasifikaci[1]. V této kapitole si popíšeme základní variantu algoritmu od Freunda a Schapira [1].

Algoritmus AdaBoost je založen na učení s učitelem. Proto jako vstup potřebujeme jak sadu vzorků x, tak jejich ohodnocení y. V případě diskrétního AdaBoostu platí, že $y \in Y$ a $Y = \{1, -1\}$, s tím, že hledané objekty jsou označeny hodnotou $+1$ a protipříklady hodnotou -1. Algoritmus uchovává distribuci vah trénovacích vzorků. Díky této distribuci je schopen přizpůsobit se těžko klasifikovatelným vzorkům. To provede tím, že po každé iteraci algoritmu AdaBoost správně zařazeným vzorkům váhu sníží a špatně zařazeným vzorkům váhu zvýší. Vzorky s vyšší vahou více ovlivňují výběr dalšího slabého klasifikátoru.

V každém kroku učení je do silného klasifikátoru přidán jeden slabý klasifikátor h_t z množiny všech možných klasifikátorů. Výběr v každém kroku učení je realizován hladovým způsobem tak, aby byl minimalizován horní odhad chyby klasifikátoru. Chyba klasifikátoru se dá vyjádřit jako součet vah chybně klasifikovaných vzorků (6).

$$\varepsilon_t = \sum_{i:\, h_t(x_i) \neq y_i} D_t(i) \tag{6}$$

Pro každý slabý klasifikátor vypočítáme parametr α_t, který nám říká důležitost klasifikátoru. Platí, že α_t je nepřímo úměrné chybě klasifikátoru tzn. čím nižší je jeho chyba, tím vyšší je α_t. Následující aktualizace distribuce pro další iteraci algoritmu má za úkol zvýšit důležitost u chybně klasifikovaných vzorků a snížit u správně klasifikovaných.

Algoritmus lze zapsat pseudokódem:

Vstup: $(x_1, y_1) \ldots (x_m, y_m)$, $x \in X$, $Y = \{1, -1\}$

Inicializace: $D_1 = \frac{1}{m}$ pro všechny vzorky

Pro $t = 1 \ldots T$

1. Učení klasifikátorů. Nalezení optimálních parametrů každého slabého klasifikátoru tak, aby měl co nejmenší chybu ε_j na aktuální distribuci D.
2. Nalezení klasifikátoru s nejmenší chybou ε_t pro distribuci D_t.

   $$h_t: X \to \{-1, +1\}$$
3. Výpočet váhy klasifikátoru

   $$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$
4. Aktualizace vah vzorků v trénovací množině

   $$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

   Kde Z_t je normalizační faktor, zvolený tak, aby D_{t+1} zůstala pravděpodobnostním rozložením.

Alg 1: Pseudokód Diskrétního algoritmu AdaBoost [5]

Výsledný klasifikátor je lineární kombinací slabých klasifikátoru. Funkce $H: X \to \{-1, +1\}$ je po T krocích vyhodnocena jako součet odezev klasifikátorů násobených jejich váhou. Lze ji vyjádřit i vektorově jako skalární součin $\vec{a}.\vec{h}$ (7), kde $\vec{a}$ je vektor vah klasifikátorů a $\vec{h}$ je vektor jejich odezev.

$$H(x) = sgn\left(\vec{a}.\vec{h}\right) = sgn\left(\sum_{t=0}^{T} a_t h_t(x)\right) \tag{7}$$

Pokud lze nalézt slabý klasifikátor takový, že jeho chyba je nižší než 0.5 což odpovídá náhodné funkci, chyba vždy klesá. Algoritmus AdaBoost se téměř nedá přetrénovat, což je nesporná výhoda proti některým jiným metodám použitelných ke klasifikaci.

3.1.1 AdaBoost dle Violy a Jonese

Algoritmus od těchto dvou autorů[2] vychází z diskrétního AdaBoostu. Autoři využili pro jejich detektor obličeje tento upravený AdaBoost. Jako příznaky použili Haarovy vlnky s využitím

integrálního obrazu. Oproti původní verzi AdaBoostu se AdaBoost Violy a Jonese liší v rozdílné inicializaci a aktualizaci vah vzorků. Rozložení distribuce vah zde nemusí být pravděpodobnostním rozložením, proto se musí na začátku každé iterace normalizovat.

Inicializace: $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ pro $y_i = 0, 1$, kde m a l je počet negativních, resp. Pozitivních vzorků.

Pro $t = 1 \ldots T$

1. Normalizace vah, aby w bylo pravděpodobnostní rozložení.

$$w_{t,i} = \frac{w_{t,i}}{\sum_j w_{t,j}}$$

2. Učení klasifikátorů. Nalezení optimálních parametrů každého slabého klasifikátoru tak, aby měl co nejmenší chybu ε_j na aktuální distribuci w.

$$\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$$

3. Výběr slabého klasifikátoru h_j s nejnižší chybou.

4. Aktualizace vah.
   $w_{t+1,i} = w_{t,i}\beta_t^{1-\varepsilon_i}$, kde $e_i = 0$, pokud je vzorek i klasifikován správně a ε_i = 1, pokud je klasifikována špatně. $\beta_t = \frac{e_t}{1-e_t}$

Alg 2: Pseudokód algoritmu AdaBoost dle Violy a Jonese [5]

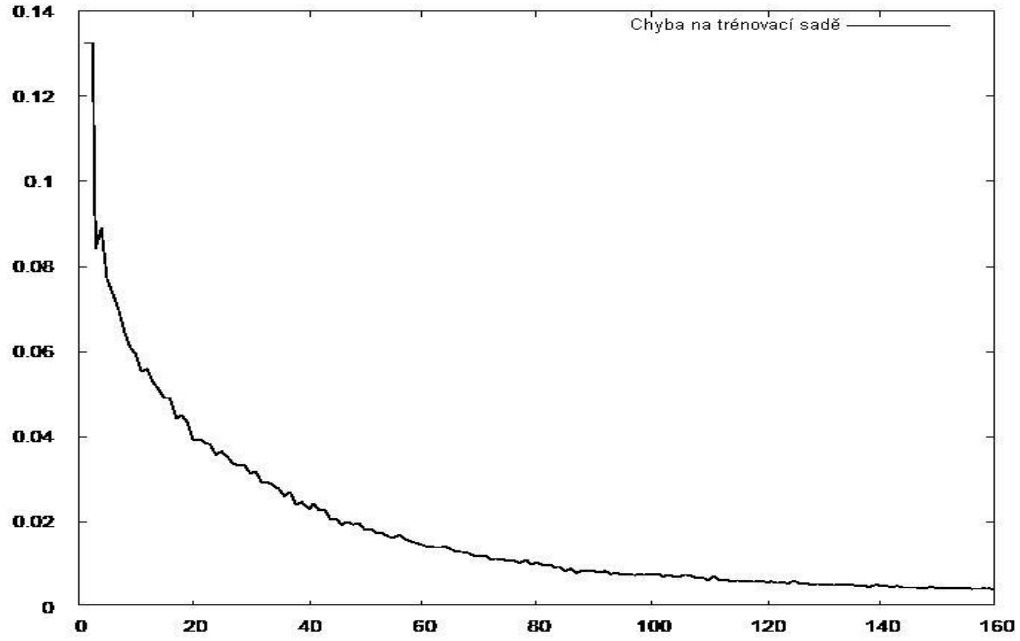
Hodnota výsledného silného klasifikátoru se vypočítá podle vztahů (8).

$$\begin{aligned} H(x) &= 1 \quad pro \sum_T \alpha_t h_t(x) > \frac{1}{2}\sum_t \alpha_t \\ H(x) &= 0 \quad jinak \\ \alpha_t &= \frac{1}{\beta_t} \end{aligned} \tag{8}$$

3.2 Trénovací chyba

Jednou ze základních vlastností algoritmu AdaBoost je schopnost exponenciálně snižovat chybu na trénovacích datech. Klasifikátor, který vybírá třídu pro každý prvek náhodně, má u binárních

problémů chybu 0,5. Chyba slabého klasifikátoru h_t se dá zapsat vztahem $\varepsilon_t = \frac{1}{2} - \gamma_t$. Pak γ_t vyjadřuje jakou mírou je h_t lepší jak náhodný výběr. Na obrázku 7 je ukázka chování chyby silného klasifikátoru na trénovací sadě, při zvyšujícím se počtu slabých klasifikátorů. Jak je vidět, chyba až na lokální výkyvy exponenciálně klesá. Můžeme tvrdit, že pokud se klasifikátor $H(x)$ skládá ze slabých klasifikátorů, které mají chybu jen o něco málo menší než náhodná funkce, chyba na trénovací sadě vzorků exponenciálně klesá s počtem slabých klasifikátorů.



Obr. 7: Pokles chyby s počtem slabých klasifikátorů[5]

3.3 WaldBoost

První modifikace algoritmu AdaBoost, kterou si probereme je algoritmus WaldBoost. WaldBoost zrychluje vyhodnocení silného klasifikátoru. Zrychlení docílí tím, že umožňuje nevyhodnocovat klasifikátor celý. Vyhodnocování klasifikátoru je zastaveno ve chvíli, kdy vzroste pravděpodobnost příslušnosti prvku do jedné z klasifikačních tříd nad určitou předem danou mez. Algoritmus WaldBoost je ve své podstatě kombinací 2 různých algoritmů. Algoritmus AdaBoost používá pro výběr a řazení slabých klasifikátorů. Druhým algoritmem je metoda SPRT (Sequential Probability ratio Test [3,4]), která je založena na sekvenční rozhodovací strategii. Využití metody WaldBoost je především v real time aplikacích, které netolerují zpoždění ve zpracování dat.

3.3.1 Sekvenční rozhodovací strategie

Sekvenční rozhodovací strategie je metoda, která se uplatňuje v mnoha oblastech i mimo informační technologie. Je založena na tom, že definuje 2 prahy A a B, přičemž platí $A < B$. Pokud je hodnota spočtená pomocí klasifikátoru (R_t)v daném kroku menší jak A, je prvek klasifikován jako -1. Pokud je R_t větší jak B, je prvek klasifikován $+1$. V případě, ve kterém platí $A \leq R_t \leq B$, je nutné provést minimálně ještě jednu iteraci algoritmu AdaBoost. Hodnoty definovaných prahů A a B se volí podle maximální přípustné velikosti chyb (FP a FN viz kap. 2.1). Tyto prahy se určují podle odhadu [4].

3.3.2 Trénování

Trénování metodou WaldBoost probíhá ve 2 fázích. V první fázi se vybere klasifikátor metodou AdaBoost. V druhé fázi se podle prahů vyřadí prvky, u kterých lze bez problému rozhodnout o její příslušnosti. Aby nebyly znehodnoceny další iterace, musíme vyřazené prvky nahradit takovými, kterým stávající klasifikátor nedokáže přiřadit třídu.

3.4 FloatBoost

Algoritmus FloatBoost [13] je založený na algoritmu AdaBoost. FloatBoost přidává fázi zpětného vyhledávání. Této fázi odstraní všechny klasifikátory, které z pohledu nových klasifikátoru už nemají pro klasifikaci význam. Tím vytváří silné klasifikátory s menším počtem slabých klasifikátorů. Snižuje tím i chybu výsledného klasifikátoru. Lepší vlastnosti jsou ale vykoupené podstatně delší dobou trénování. Trénovací doba je zhruba 5x větší než u klasického AdaBoostu[25].

3.5 TCAcu

Další z modifikací algoritmu AdaBoost dle Šochmana a Matase[14] spočívá v opakovaném vylepšování předpovědí slabých klasifikátorů, které byli vybrané v předchozí iteraci. Tento algoritmus byl pojmenován Totally corrective algorithm with coefficient updates (TCAcu). Předpověď z vybraných slabých klasifikátorů se stává neoptimální s dalšími iteracemi AdaBoostu. V každé iteraci však slabé klasifikátory mohou být optimalizovány. Tato optimalizace není příliš výpočetně náročná. TCAcu zajišťuje, že v každé iteraci AdaBoostu je vybraný slabý klasifikátor, co možná nejvíc, nezávislý na všech slabých klasifikátorech vybraných v předchozích kolech, což je podstatné zlepšení oproti AdaBoostu[25], ve kterém je nově vybraný klasifikátor nezávislý pouze vůči klasifikátoru vybranému v minulém kole. TCAcu prokazatelně snižuje trénovací chybu se stoupající složitostí klasifikátoru.

3.6 Příznaky a slabé klasifikátory

Vlastnosti klasifikátorů jsme si nastínili v kapitole 2.1. Silný klasifikátor je složen z určitého množství slabých klasifikátorů odpovídající tamní definici klasifikátoru. Ve zpracování obrazu se velmi často jako základy slabých klasifikátorů využívají tzv. obrazové příznaky.

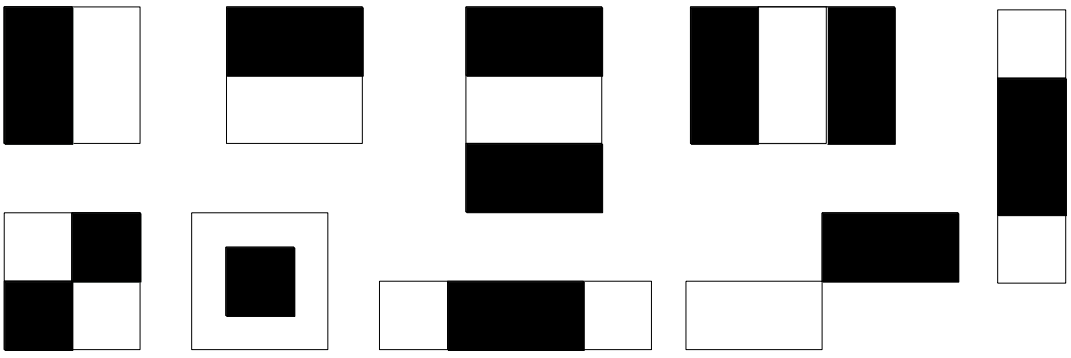
Obrazové příznaky jsou ve valné většině konvoluce o různé pozici a velikosti v obraze. Jde vlastně o spojité příznaky. Velmi často jsou využívány Haarovy vlnky. V aplikacích, které nekladou takový důraz na rychlost výpočtu, se dají využít Gáborovy vlnky. Existují i diskrétní příznaky. Každý spojitý příznak má svojí diskrétní verzi, ale naopak to pravdou být nemusí.

Typy příznaků jsou dány tvarem konvolučního jádra. Velikost a pozice v obraze pak určují jednotlivé instance příznaku daného typu. Pokud chceme využít dané příznaky jako základ slabých klasifikátorů, musíme před začátkem trénování všechny příznaky z jednotlivých obrazů vypočítat. Tím nám vzniknou vektory příznaků, které jsou vstupem samotného trénování. V obrazu velikosti $24x24$ px je při použití prvních čtyř Haarových vlnek z obrázku 8 přesně 141 600 příznaků. Ačkoliv je příznaků v obraze mnohem více než pixelů, je jejich použití výhodnější kvůli jejich lepším vlastnostem, než mají samotné pixely.

Při učení slabých klasifikátorů musíme nalézt jejich optimální parametry. Může se jednat o paměťově a výpočetně náročnou operaci, kterou musíme provést pro všechny slabé klasifikátory a v každé iteraci trénovacího algoritmu.

3.6.1 Haarovy příznaky

Tato podkapitola se zaměřuje na popis slabých klasifikátorů založených na Haarových vlnkách. Tyto klasifikátory byly použity například v pracích [2, 3] pro úlohu detekce obličeje.



Obr. 8: Tvary Haarových příznaků

Na předcházejícím obrázku je zobrazeno několik typů Haarových bází. Výpočet hodnoty konkrétního příznaku v obraze je roven rozdílu sum hodnot pixelů pod bílými a černými oblastmi. To by se dalo vyjádřit vzorcem (10).

$$f(x) = \sum_{w \in W} x(w) - \sum_{b \in B} x(b) \quad (9)$$

V předcházejícím vzorci x representuje vzorek dat. Hodnoty W (white, bílá) a B (black, černá)jsou množiny pixelů, které přísluší jednotlivým „barvám“ Haarova příznaku. Na obrázku 9 vlevo jsou ukázány 2 instance jednoho typu Haarova příznaku v obraze. Na stejném obrázku vpravo je ukázána odezva na 2 různé typy Haarových příznaků na stejném obraze. Byly použity příznaky velikosti $5x2$px resp. $2x5$px. Každý pixel výsledného obrazu je tedy hodnota patřičného příznaku[5].

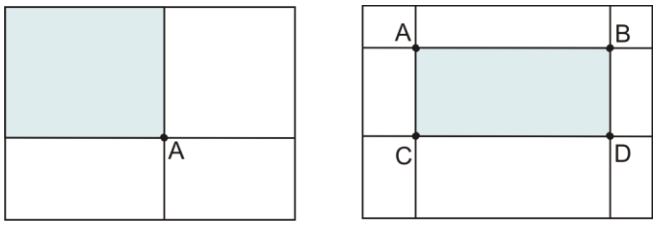
Obr. 9: Různé instance jednoho typu příznaku ve vzorku a odezvy příznaků na vzorku[5]

Typickým klasifikátorem využívající Haarovy příznaky je Threshold (kap. 2.1). Ten obsahuje prahovou hodnotu θ, která určuje rozhodovací hodnotu pro klasifikaci do určité třídy. Obsahuje taktéž polaritu p. Polarita určuje význam klasifikovaných tříd. Například při kladné polaritě je hodnota větší jak práh klasifikována jako třída 1 a hodnota menší jak práh jako třída 2, při záporné polaritě je tomu přesně naopak, tedy hodnota větší jak práh je klasifikována jako třída 2 a hodnota menší jak práh jako třída 1. Vyhodnocení těchto slabých klasifikátorů nám ukazuje vzorec 10.

$$\begin{aligned} h_j(x) &= 1 \quad pro\ p_j f_j(x) < p_j \theta_j \\ h_j(x) &= 0 \end{aligned} \quad (10)$$

Integrální obraz

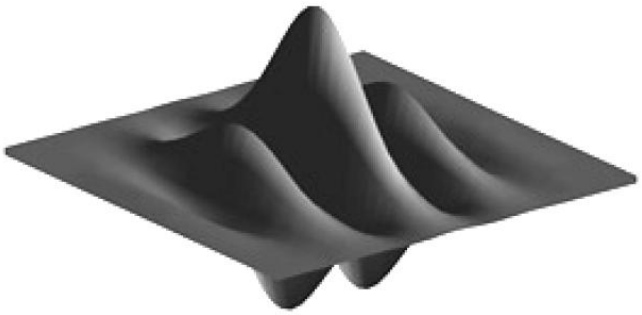
Výpočet Haarových příznaků je v podstatě rozdíl sum pixelů jednotlivých oblastí. Tyto sumy můžeme vypočítat různými způsoby. Prvním způsobem může být obyčejné sčítání pixelů. Tento přístup je časově velice náročný a tím i nevhodný. Dalším způsobem může být násobení vzorku příslušným vektorem masky příznaku. Tímto způsobem vyřešíme rovnou i rozdíl obou oblastí. Výhodou tohoto přístupu může být i jednoduchá výměna typů příznaků (např. za Gáborovy vlnky). Nejrychlejším přístupem je využití integrálního obrazu[2]. Integrální reprezentace obrazu obsahuje v každém pixelu A hodnotu, která představuje sumu všech pixelů vlevo a nahoru od A (Obr. 10 vlevo). Díky integrálnímu uložení obrazu můžeme získat hodnotu kterékoliv obdélníkové oblasti v obrazu v konstantním čase. Sumu pixelů v označené oblasti obrázku 10 vpravo vypočítáme jako $A - B - C + D$.



Obr. 10: Příklady integrálních obrazů

3.6.2 Gáborovy příznaky

Alternativou Haarovým vlnkám jsou Gáborovy vlnky[6]. Ty mají sice větší požadavky na zpracování, ale v aplikacích, kde požadavek na čas zpracování není tak rozhodující, jsou upřednostňovány, kvůli jejich větší popisovací síle. Gáborovy vlnky poskytují ideální kompromis mezi frekvenčním a prostorovým rozkladem. Zajímavým důvodem k použití 2D Gáborových vlnek je, že souvisí se stylem zpracování obrazu v lidském mozku[25].



Obr. 11: 2D Gáborova vlna

3.6.3 Analýza hlavních komponent

Analýza hlavních komponent neboli PCA má za cíl odvodit relativně malý počet lineárních kombinací (hlavních komponent) množiny náhodných proměnných, které nesou co nejvíce informací původních proměnných. Cílem, který je pro nás nejdůležitější, je přitom především určení lineárních kombinací proměnných, redukce počtu dimenzí a výběr nejužitečnějších proměnných. Jinak řečeno, hlavním účelem PCA je komprese dat a výběr příznaků, tj. signálů nesoucích nejvíce užitečné informace.

4 Databáze výrazů obličejů

Jako pro každé trénování i pro rozpoznávání výrazu obličeje je důležitá kvalita a množství trénovacích dat. Pokud máme možnost vyfotit větší množství lidí, ve více výrazech tváře a pokud možno i ve více verzích jednoho výrazu, můžeme si tento dataset udělat sami. Byla by to ovšem příliš náročná práce. Jednodušší je využití již hotových databází těchto obličejů. Ty jsou s nejrůznějšími podmínkami přístupné na internetu.

Databází fotografií je k dispozici velké množství, ale většina se věnuje jiným oblastem. Příkladem můžou být tyto stránky[15][16][17], které obsahují seznam několika desítek databází, které se zaměřují na různé oblasti. Většina se věnuje pouze datům pro detekci obličeje. Další velkou skupinou jsou „mrtvé“ databáze. Jedná se o projekty, na které pořád existují reference z jiných stránek, ale dané projekty už skončily a nikdo se jim již nevěnuje. Příkladem může být PIE Database[18].

Pokud se objeví vhodná databáze, nastává další problém. Získání přístupu není občas triviální záležitost. Některé databáze jsou volně stažitelné, ale jejich kvalita silně zaostává. Příkladem volně stažitelné databáze je JAFFE[19]. JAFFE je, oproti jiným volně šiřitelným databázím, na dobré úrovni. Většina majitelů databází vyžaduje poslání naskenované podepsané smlouvy, ve které se zavazujete nekomerčnímu použití databází, popř. dalším podmínkám. Vyskytují se však takové, které tuto smlouvu chtějí poslat poštou.

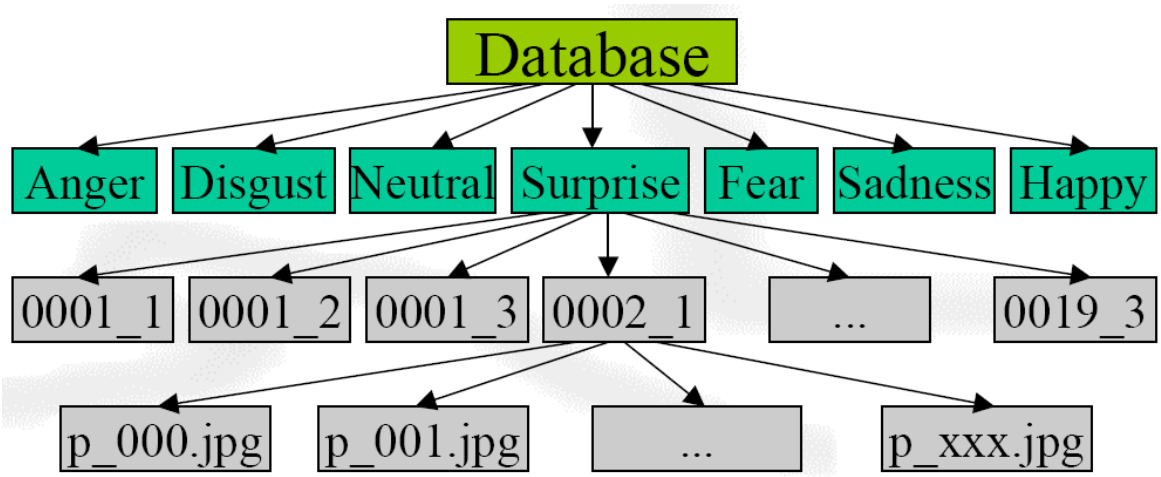
Téměř všechny databáze věnující se výrazu obličeje obsahují 7 základních výrazů. Jedná se o neutrální (normální, neutral), štěstí (happy), překvapení (surprise), strach (fear), smutek (sadness), odpor (disgust) a hněv (anger).

V následujících podkapitolkách se podíváme blíže na dvě databáze obličejů, do kterých se mi podařilo získat přístup. Tyto 2 databáze jsem použil jako zdroj dat pro můj dataset k rozpoznávání výrazu obličeje.

4.1 Facial Expressions and Emotion Database

Facial Expressions and Emotion Database (FEED) [20] je databáze vytvořená na technické univerzitě v Mnichově. Pro nekomerční účely je přístupná na požádání. Výhodou této databáze je rychlé a jednoduché jednání. Dalo by se říct, že z celé škály databází, ke kterým jsem se snažil získat přístup, se tady chovali nejlépe.

Databáze se skládá jak ze sekvencí obrázků ve formátu JPEG, tak z videosekvencí, které byli základem pro sekvence obrázků, ve formátu MPEG. Obsahuje data od 19 osob. Ke každé osobě je přístupno všech sedm výrazů, vždy ve třech sekvencích ke každému výrazu. Používá barevné obrázky. Strukturu uložení dat v databázi asi nejlépe presentuje obrázek 12.



Obr. 12: Struktura adresářů databáze FEED[20]

Jak je vidět na obrázku 12, databáze je rozdělena podle výrazů. Každý výraz je rozdělen podle sekvencí jednotlivých osob. Každá sekvence pak obsahuje řádově stovku obrázků obličejů osoby.

Obr. 13: Příklad části sekvence databáze FEED

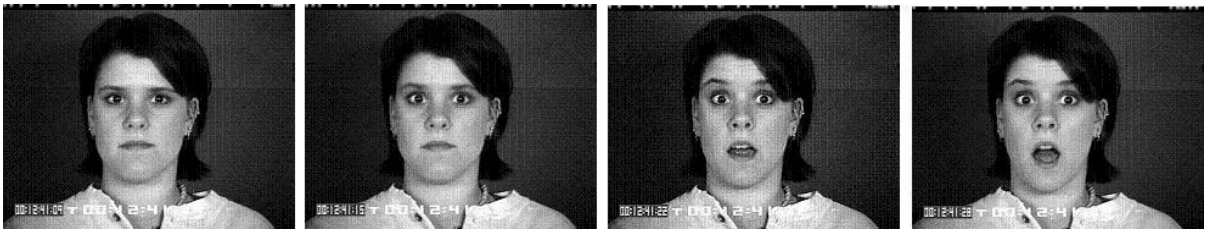
Nutno podotknout, že lidem v této databázi občas daný výraz nevyjde, a místo toho aby se např. báli, tak se začnou neskrývaně smát. Občas se stane, že v sekvenci z některých snímků úplně zmizí. Právě proto je nutné snímky z této databáze pečlivě přebrat. I přes pečlivý výběr však tyto výkyvy zhorší chybu klasifikátoru natrénovaným na těchto datech.

4.2 Cohn-Kanade Facial Expression Database

Cohn-Kanade Facial Expression Database [21] vznikla na univerzitě Carnegie Mellon. Databáze je poskytnuta na žádost do 4-5 pracovních dní. Je poskytnuta pouze k nekomerčním účelům. Obsahuje téměř 500 sekvencí od zhruba 100 subjektů.

I tato databáze se skládá ze sekvencí snímků. Je ale řazena dle osob. Ke každé osobě je určitý počet sekvencí. Ty nejsou rozeznávány dle výrazu obličeje, ale je k nim dodán jejich FACS [22]kód. FACS (Facial Action Coding System) systém je založený na stavech svalů lidské hlavy. Výsledný výraz obličeje se potom určuje dle typických stavů. Tyto stavy jsou ale občas neprůkazné. Může se

stát, že ve FACS kódu jsou dva rozdílné výrazy zakódovány téměř identicky. Kvalitativně je data v této databázi na velice dobré úrovni. Neexistují zde „úlety“ jako např. ve FEED.



Obr. 14: Příklad části sekvence databáze Cohn-Kanade

Obrázky jsou uloženy ve valné většině pouze v odstínech šedi. Data v této databázi jsou přístupné ve 2 formách. Bezztrátové PNG a na JPEG s 90% kvality. Data ve formátu JPEG jsou ale více než 4x menší.

5 Návrh řešení

Kapitola se bude věnovat návrhu jednotlivých součástí programu. Rozebereme zde výběr dat pro trénování a testování a jejich úpravu do vektoru příznaků. Nastíníme taky možné řešení klasifikace výrazů obličejů.

5.1 Příprava a výběr dat

Jedním z nedůležitějších kroků při trénování je vybrat kvalitní data. V těchto datech by se neměly moc opakovat jednotlivé osoby. Důležitou podmínkou je, aby se jedna osoba nevyskytovala jak v trénovacích tak v testovacích datech. To by nám mohlo zlepšit výsledky testování, protože klasifikátor by danou osobu již „znal“.

Základem přípravy dat je vybrat trénovací a testovací data z databází. Kvalita vybraných dat nám ovlivní kvalitu celého výsledného klasifikátoru. Vybrané data je vhodné přehledně odlišit pokud možno již názvem.

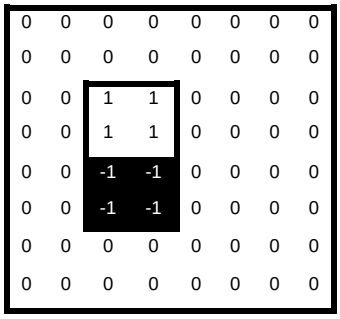
5.2 Úprava dat a extrakce příznaků

Data pro extrakci příznaků musí být transformována. Použijeme pouze šedotónový obraz. Protože naším cílem je rozeznávat výraz obličeje, musíme v první řadě z obrázku tento obličej vyříznout. K lokalizaci obličeje můžeme použít některý z již existujících detektorů obličeje. Detektory obličejů jsou již natrénované klasifikátory uložené do externích souborů. Velmi často se používá formát xml. Tyto klasifikátory pak pouze načteme do svého programu a po jejich vyhodnocení daného obrázku nám určí pozice, na kterých se nacházejí obličeje.

Druhým úkolem je normalizace velikosti obrázku. Jak bylo zmíněno v kapitole 3.6, příznaků může být mnohem více než pixelů. Vhodnou velikostí může být například 24x24px.

Jako příznaky použijeme Haarovy vlnky (kap. 3.6. Použijeme první 4 typy příznaků z obr. 8Obr. 8: Tvary Haarových příznaků. Využijeme masku Haarova příznaku. Vynásobením masky a obrázku obličeje získáme hodnotu příznaku. Jedná se vlastně o linearizaci obrázku skenováním po řádcích a skalární součin s vektorem masky Haarova příznaku. Vektor masky získáme linearizací masky Haarova příznaku skenováním po řádcích. Na obrázku 15 máme uvedený příklad masky Haarova příznaku. Maska má velikost 8x8(px). Má naznačené hodnoty jednotlivých bodů masky.

Při využití masky příznaku je velice jednoduché změti typ používaných příznaků. Stačí pouze vyměnit generátor masek za jiný. Tím můžeme velice rychle zaměnit Haarovy příznaky např. za Gáborovy příznaky.



Obr. 15: Příklad masky Haarova příznaku

Výpočtem hodnot všech instancí Haarových příznaků v daném obrázku získáme vektor příznaků obrázku. V obrázku 24x24px při použití čtyř základních bází Haarova příznaku bude tento vektor mít 141 600 prvků (viz kap. 3.6).

5.3 Trénování

Pokud máme vyextrahovány příznaky z jednotlivých obrázků testovacích dat, můžeme přistoupit k samotnému trénování klasifikátoru. Využijeme algoritmu AdaBoost popsané v kapitole 0. Vždy se natrénuje klasifikátor, který bude umět rozeznávat jeden výraz od všech ostatních. Vstupem trénování budou vždy vektory příznaků jednotlivých obrázků vypočtené dle bodu 5.2 a ohodnocení příslušnosti do dané třídy, tj. např. $+1$ pokud jde o hledaný výraz a -1 pokud se jedná o některý z jiných výrazů. Ke konečnému rozhodnutí, do které třídy výraz patří, se může použít neuronová sít (kap. 2.2). My však využijeme o něco jednodušší metodu. Výraz zařadíme do třídy, jejíž klasifikátor bude mít nejvyšší odezvu.

6 Realizace

V této kapitole se seznámíme s implementací řešení problému rozpoznávání výrazu obličeje. Probereme také problémy, se kterými jsem se při implementaci setkal. Některé z těchto problémů změnily dokonce původní návrh práce.

6.1 Data

Ze zpřístupněných databází si v první řadě musíme vybrat testovací a trénovací data. To jde učinit i tak, že vybereme jedny data, která poté rozdělíme v půlce (osob).

Data budeme vybírat z databází zmíněných v kap. 4. Tyto databáze jsou tvořeny sekvencemi snímků. Každá sekvence patří k jednomu výrazu obličeje. V každé sekvenci je zaznamenána změna výrazu obličeje dané osoby od neutrálního k výrazu, daném příslušností sekvence.

Výběr dat může být poloautomatickou operací, kdy vybereme některé z prvních snímků sekvence jako neutrální, a některé z posledních jako snímky daného výrazu. Poté ale musíme zkontrolovat a popřípadě upravit výběr dat, některé snímky nemusí odpovídat přesně výrazu, který chceme detekovat. Tabulka 1 popisuje počty vybraných obličejů dle jednotlivých výrazů a databází.

Výraz	FEED			Cohn-Kanade			celkem		
	Test.	Trén.	celkem	Test.	Trén.	celkem	Test.	Trén.	celkem
překvapení	42	58	**100**	79	85	**164**	121	143	**264**
strach	39	52	**91**	52	62	**114**	91	114	**205**
veselý	46	60	**106**	84	93	**177**	130	153	**283**
smutek	42	54	**297**	57	60	**117**	99	114	**414**
odpor	48	60	**108**	32	51	**83**	80	111	**191**
hněv	46	59	**105**	33	42	**75**	79	101	**180**
neutrální	160	199	**359**	327	364	**691**	487	563	**1050**
Celkem	**423**	**542**	**1166**	**664**	**757**	**1421**	**1087**	**1299**	**2587**

Tab. 1: Počty obrázků vybraných z jednotlivých databází a k jednotlivým výrazům

Po výběru dat je vhodné jednotlivé obrázky přehledně pojmenovat, aby bylo na první pohled z názvu obrázku poznat, do jaké patří databáze, jaký má daný obličej výraz a podobně. Já jsem zvolil pojmenování ve tvaru DD_PXXX_VVCCC, kde DD vyjadřuje databázi, ze které je tento obrázek vybraný, PXXX značí číslo osoby z této databáze, VV je výraz obličeje a CCC je pořadové číslo obrázku dané osoby u daného výrazu. Takto upravené data můžou postoupit do dalšího zpracování.

6.2 Operace s daty

Operace s daty, neboli příprava dat pro trénování se dá rozložit do 2 problémů. Prvním je najít v obrázku samotný obličej a změnit výřez obličeje z obrázku na požadovanou velikost. Druhým problémem je samotná extrakce příznaků. Pro vyřešení těchto problémů byly vytvořeny 2 aplikace. Požadavek na více aplikací je spjat s nutnosti překontrolovat výstup první z nich. Detektor obličeje zde použitý bude mít vždy nějakou chybu. Nemůžeme si dovolit trénovat výraz obličeje na náhodném pozadí. Pro práci s obrázky se používá knihovna OpenCV [23]. Taktéž knihovna se používá i pro detekci obličeje. Původně se měla použít i pro trénování klasifikátoru pomocí metody AdaBoost, ale její část týkající se machine learning(ML) má určité nedostatky a postavila nám do cesty nepřekonatelné problémy.

K prvotní úpravě dat slouží program Odetektor. Jeho vstupy z příkazové řádky jsou dvě cesty adresářů. První z nich určuje zdrojové data a druhá složku pro cílová data. Aplikace bere v úvahu všechny soubory zdrojového adresáře se všemi podsložkami. Právě proto musí mít cílový adresář stejnou hierarchii podsložek.

Pro vyhledání obličeje použijeme detektor obličeje, který nám nabízí OpenCV. OpenCV nám dává celou třídu pro detekci objektů v obraze. Tato třída se nazývá `CvHaarClassifierCascade`. Jak již název třídy napovídá, pro detekci obličeje používá Haarovy příznaky. K natrénování klasifikátorů používá algoritmus AdaBoost. Klasifikátory jsou uložené u nainstalovaného OpenCV ve složce `OpenCV/data/haarcascades/`. Námi použitý klasifikátor pro detekci obličejů je uložen v souboru `haarcascade_frontalface_alt.xml`.

Na každém obrázku se očekává právě jeden obličej. V případě, kdy by se jich vyskytlo více, se vybere jen ten, který je „nejvýraznější“. To je zajištěno tím, že hledáme obličej s největším možným počtem detekovaných sousedů. S použitím funkce `cvHaarDetectObjects(…)` získáme rohové body čtverce obličeje. Tento obličej pak z obrázku vyřízneme. Po vyříznutí obrázku změníme jeho velikost na požadovaný počet pitelů. Jak již bylo zmíněno dříve v našem případě je to 24x24 px. Po změně velikosti ho uložíme pod stejným jménem na příslušné místo v cílové složce.

O extrakci příznaků se nám postará program Priznaky. V původním návrhu měla tato aplikace zároveň trénovat klasifikátory, ale potíže s knihovnou ML z OpenCV mě donutili změnit postup. Tento program musíme spustit pro vzorky jednotlivých výrazů obličeje zvlášť. Tím dostaneme pro každý výraz zvláštní soubor. Tento soubor obsahuje údaje o počtu trénovacích dat, o počtu příznaků na jeden obrázek a hodnoty těchto příznaků. Formát souboru vyjadřuje tabulka 2. Poněvadž byl tento formát původně navržen k jinému účelu [24], může se zdát lehce nesmyslný.

Aplikace má taktéž dva vstupní parametry. První parametr je název výstupního souboru. Druhým parametrem je cesta k datům daného výrazu. Po spuštění programu se inicializuje třída Haarových příznaků. Do této třídy se automaticky negenerují patřičné Haarovy příznaky. Poté začne načítat jednotlivé obrázky. Z každého obrázku vypočítá vektor příznaků. Následně se vektory

příznaků zapíší do souboru. Soubory s jednotlivými výrazy se pak stávají vstupem trénování jednotlivých klasifikátorů.

offset	délka	hodnota	popis
00	4 Byty	„S2D“	Hlavička používaná pro rozlišení verze
04	4 Byty	??	Bezznaménkové číslo určující počet vzorků
08	4 Byty	??	Bezznaménkové číslo, opět příznaků na řádek
18	4 Byty	??	Bezznaménkové číslo, počet řádků na vzorek
20	??	??	8 bitové hodnoty příznaků

Tab. 2: Struktura raw souboru

6.3 Trénování a testování

Původní předpoklad byl, že pro trénování metodou AdaBoost bude využita knihovna ML z OpenCV. Tamní třída CVBoost se však nedokáže vypořádat s větším množstvím dat. Chyba alokace paměti se objevovala již pro cca sto vzorků, jejichž příznakové vektory byly dlouhé 141 600 prvků. Proto je pro nás absolutně nepoužitelná. Nakonec byl pro trénování a vyhodnocení klasifikátorů použit externí program[24].

Tento program má za vstup raw soubory. Vytvoření těchto souborů je popsáno v předchozí kapitole. Dalším neméně důležitým vstupem jsou konfigurační soubory. V těchto souborech se dají nastavit podmínky trénování a testování. Výstupem aplikace jsou klasifikátory, odezvy klasifikátorů na testovací data, některé statistické výsledky, jako například ROC nebo DET křivka, informace o průběhu trénování a popřípadě chybový výstup.

Rozhodnutí do jaké třídy vzorek patří, provedeme pomocí programu Vyhodnoceni. Tento program vyžaduje na vstupu soubory odezev klasifikátorů na jednotlivé vzorky trénovací sady. Výstupem jsou statistiky obsahující data chybovosti a úspěšnosti zařazení do jednotlivých tříd. Jsou vygenerovány statistiky z pohledu jednotlivých tříd i celkové statistiky.

7 Experimenty

Kapitola rozebere výsledky testů provedených s daty. Zhodnotí výsledky a prodiskutuje jejich příčinu. Kvůli lepší transparentnosti výsledků si po prvním testu vybereme do dalších testů jen ty výrazy, které se mezi sebou nejvíce rozlišují. To poznáme tak, že klasifikátory natrénované mezi těmito výrazy mají, pokud možno, co nejmenší chybu. Chybou klasifikátoru se v této kapitole myslí poměr špatně klasifikovaných vzorků ke všem vzorkům v daném nastavení klasifikátoru (viz kap. 2.1). K trénování a testování se používá dataset zmíněný v kapitole 6.1.

7.1 Všichni proti všem

Pro tento pokus bylo natrénováno 2x21 silných klasifikátorů. Každý silný klasifikátor má vždy 100 slabých hypotéz. Vždy byl natrénován klasifikátor pro každý výraz vůči každému z množiny sedmi výrazů, které máme k dispozici. Poprvé byly natrénovány vždy na omezené trénovací sadě. Velikost trénovací sady byla omezena na 100 prvků. Podruhé byly tyto klasifikátory natrénovány na celé trénovací sadě. V tomto případě je rozdíl v počtu vzorků mezi jednotlivými výrazy několik set (viz tab. 1). První tabulka se tedy věnuje klasifikátorům natrénovaných na omezené trénovací sadě. Jsou zde zobrazeny chyby klasifikátorů natrénovaných mezi jednotlivými výrazy. Chyby jsou vyjádřeny v procentech a zaokrouhleny na dvě desetinná čísla.

	neutrální	hněv	odpor	smutek	veselý	strach	překvapení
překvapení	14,75 %	13,37 %	10,34 %	16,21 %	6,72 %	18,81 %	
strach	26,54 %	25 %	27,12 %	30,1 %	22,91 %		
veselý	4,69 %	8,06 %	11,79 %	4,33 %			
smutek	28,06 %	33,33 %	11,60 %				
odpor	14,58 %	33,54 %					
hněv	27,82 %						
neutrální							

Tab. 3: Chyby klasifikátorů při omezeném počtu prvků při trénování

Z tabulky 3 můžeme vypozorovat, že některé výrazy obličejů se od ostatních rozpoznávají velice špatně. To může být způsobeno podobností těchto výrazů. Nejhůř ze všech dopadl strach. Taky by se tento výsledek dal odůvodnit tím, že dobrovolníci, využití při focení obličejů, se nedovedou tvářit zastrašeně bez příčiny. Oproti tomu některé výrazy dosahují až překvapivě dobrých výsledků. Jde vlastně o přesný opak strachu. Smát se očividně dokáže každý.

Pokud srovnáme tabulky 3 a 4 zjistíme, jaký vliv má rostoucí počet trénovacích dat na chybu klasifikátoru. Většina klasifikátorů svoji chybovost zmenšila. Nejvýraznější to je pro neutrální výrazy. Pravděpodobně to bude způsobeno masivním vzrůstem vzorků tohoto výrazu (je jich cca 600, viz Tab. 1). Vyskytují se i zhoršení rozpoznávací schopnosti, to si jde vysvětlit například tím, že vzorky přidané k trénování jsou těžko klasifikovatelné v rámci celé trénovací sady.

	neutrální	Hněv	odpor	smutek	veselý	strach	překvapení
překvapení	10 %	14,85 %	8,87 %	19,37 %	7,51 %	23,39 %	
strach	20,03 %	27,84 %	25,99 %	26,53 %	18,94 %		
veselý	4,85 %	13,74 %	13,68 %	8,66 %			
smutek	26,19 %	31,11 %	14,36 %				
odpor	11,78 %	39,13 %					
hněv	17,43 %						
neutrální							

Tab. 4: Chyby klasifikátorů při plných trénovacích množinách.

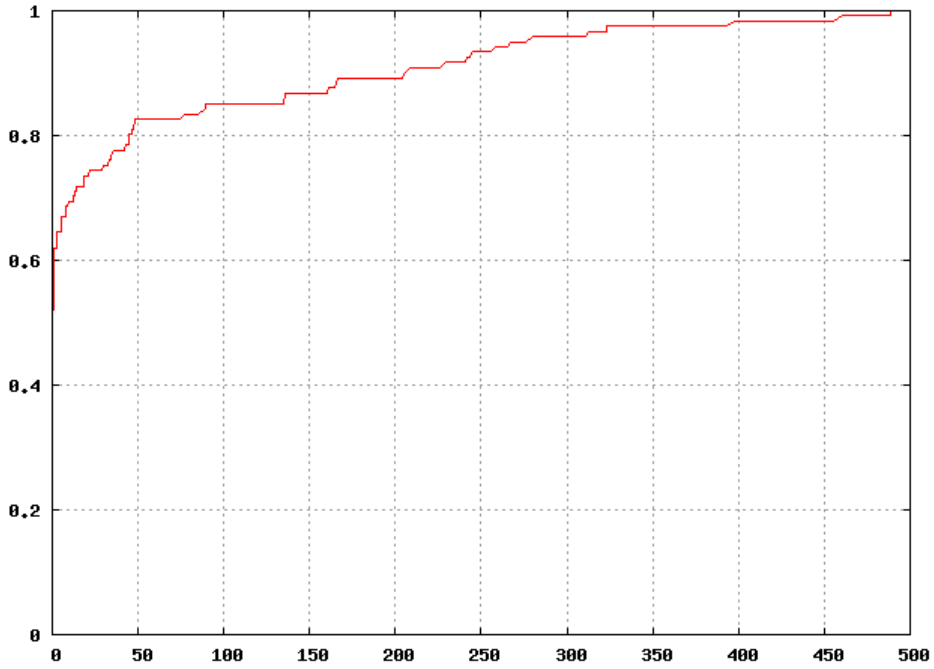
Po prozkoumání tabulek jsme schopní vybrat čtyři nejlépe rozeznatelné výrazy pro další testy. Již na první pohled do téhle skupiny musí patřit výrazy překvapení a radosti. Ty vykazovaly téměř vůči všem výrazům chybu kolem 10 % a méně. Dalším vybraným výrazem bude neutrální. Ten si své místo zasloužil především velkým množstvím vzorků. Posledním výrazem bude odpor. Odpor byl vybrán kvůli nejlepší schopnosti klasifikovat mezi ním a již vybranými výrazy ze všech zbývajících výrazů.

7.2 Jeden vůči všem

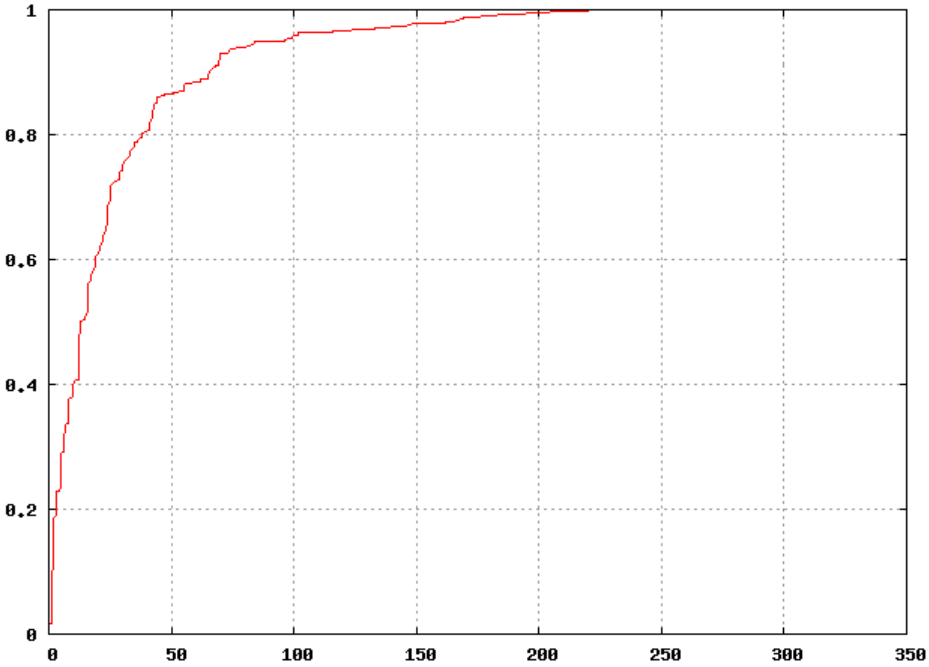
Tento experiment se provedl už pouze se čtyřmi výrazy vybraných dříve. Tyto výrazy tedy jsou překvapení, štěstí, odpor a neutrální výraz. Byly natrénovány čtyři silné klasifikátory. Vždy jeden výraz proti třem zbývajícím. Bylo využito vždy 100 slabých klasifikátorů. Budou zde uvedeny výsledky všech 4 klasifikátorů i s jejich stručným zhodnocením. Ke každému klasifikátoru bude uvedena jeho ROC křivka (kap. 2.1). ROC křivka je závislost míry flase positives a false negatives při různých nastaveních klasifikátoru.

Jako první se podíváme na výraz „překvapení“. Chyba klasifikátoru je 7,57%. Na obrázku 16 se můžeme podívat na ROC křivku daného klasifikátoru. Neutrální výraz dopadl ze všech čtyř výrazů nejhůře. Chyba klasifikátoru je 15,02%. ROC křivka je uvedena na obrázku 17. Výraz odporu dopadl o něco lépe. Chybou klasifikátoru 10,26% se řadí na třetí místo našeho pomyslného žebříčku. ROC křivka je zobrazení na obrázku 18. Výraz štěstí je posledním z testovaných. Z tohoto testu vyšel

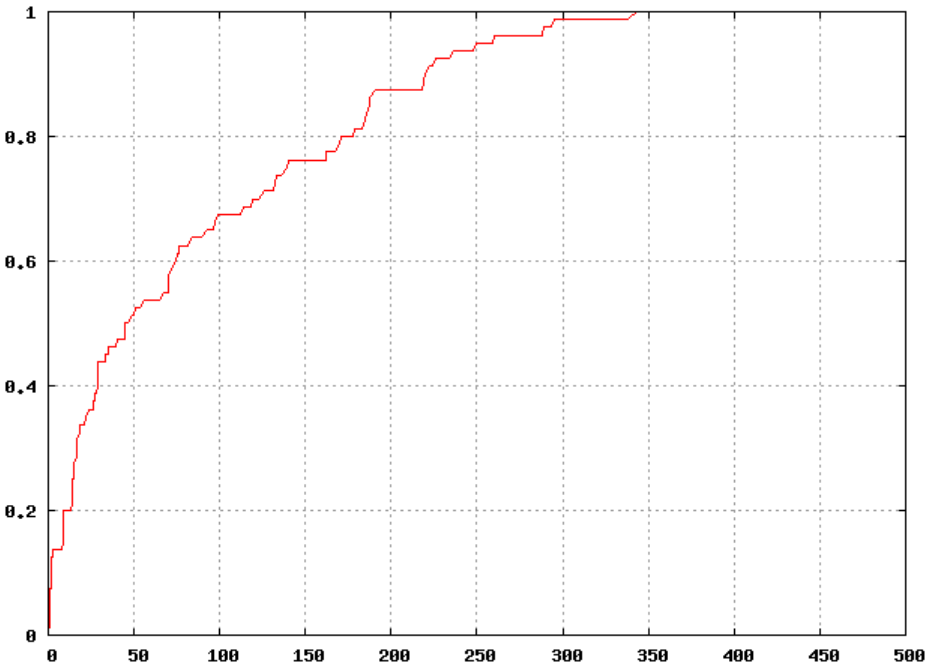
velice dobře. Chyba tohoto klasifikátoru je 5,38%. To je nejmenší chyba ze všech čtyř klasifikátorů. ROC křivka tohoto výrazu jde vidět na obrázku 19. I na této křivce jde poznat dobrá klasifikační schopnost klasifikátoru štěstí.



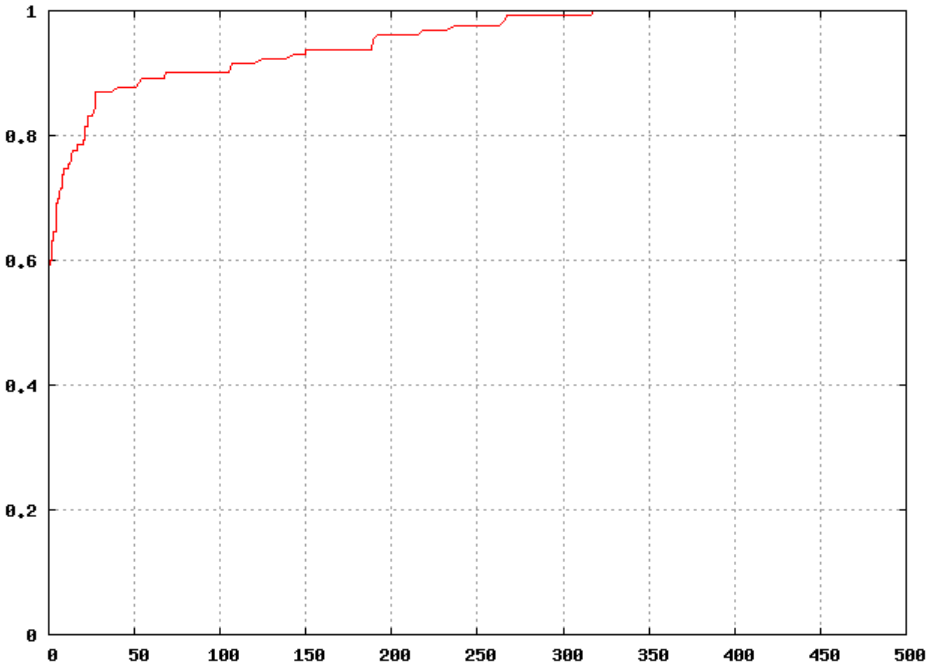
Obr. 16: ROC křivka výrazu překvapení



Obr. 17: ROC křivka neutrálního výrazu



Obr. 18: ROC křivka výrazu odpor



Obr. 19: ROC křivka výrazu štěstí

Z celkového pohledu všechny čtyři klasifikátory nedopadly špatně. Průměrná chyba pod 10 procent je uspokojivá. Pro detekci výrazu obličeje je taková chyba téměř zanedbatelná, a to především z důvodu nejednoznačnosti výrazu lidského obličeje.

7.3 Spojené klasifikátory

V této kapitole se podíváme jakou úspěšnost přiřazení výrazu obličeje do správné třídy měla aplikace Vyhodnoceni zmíněna v kapitole 6.3. Tato aplikace pracuje s výstupy předešlého experimentu. Vstupem jí jsou odezvy na testovací data. Načte výstupy všech čtyř klasifikátorů a určí ke kterému výrazu vzorek, který representují hodnoty výstupu klasifikátorů, patří. Výsledek vyhodnocení je uveden v tabulce 5. Je zde uveden výsledek na všech trénovacích datech. Výsledek je také rozebrán na jednotlivé výrazy. Je zde uvedeno, s jakou chybou byl daný výraz zařazován, nebo kam byl chybně zařazen. Objevuje se zde i pojem úspěšnosti přiřazení. Ta vyjadřuje, v kolika procentech případů byl vzorku správně určen jeho výraz. Úspěšnost přiřazení je vlastně přesným opakem chyby klasifikátoru.

Výraz	štěstí	neutrální	odpor	překvapení	souhrn
Celkový počet	130	487	80	121	818
Počet správně zařazených	100	441	51	97	689
Počet chybně zařazených	30	46	29	24	129
Zařazen jako štěstí		6	5	5	
Zařazen jako neutrální	14		18	15	
Zařazen jako odpor	15	14		4	
Zařazen jako překvapení	1	26	6		
Úspěšnost přiřazení	76,92 %	90,55 %	63,75 %	80,17 %	84,23 %
Chyba přiřazení	23,08 %	9,45 %	36,25 %	19,83 %	15,77 %

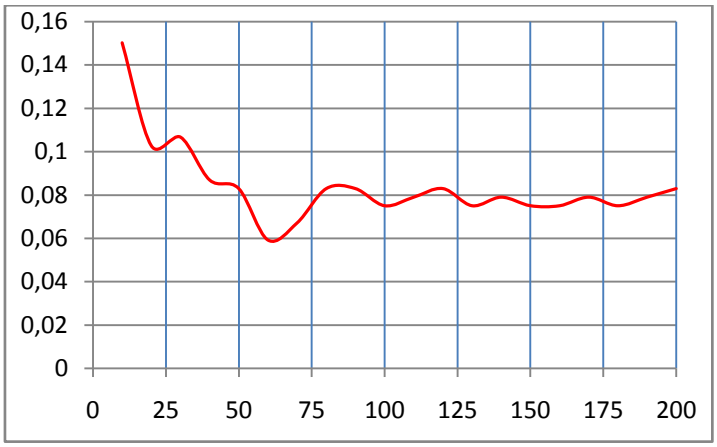
Tab 5: Výsledky spojených klasifikátorů

Jak je z předchozího výsledku vidět chyba spojených klasifikátorů je 15,77%. Nejčastěji je správně zařazen neutrální výraz, a to v 90,55% případů. Ostatní výrazy mají podstatně horší úspěšnost přiřazení. Nejhůře dopadl výraz odporu, který byl chybně zařazen v 36,25% případů. Z naměřených vyplívá, že chyba spojených klasifikátorů je necelých 16 % jen díky neutrálnímu výrazu, který je klasifikován nejlépe. Neutrální výraz množstvím vzorků výrazně zlepšuje průměr.

7.4 Pokles chyby s počtem slabých klasifikátorů

V této podkapitole se zaměříme na pokles chyby silného klasifikátoru s rostoucím počtem slabých hypotéz. Pro demonstraci tohoto jevu si vybereme pouze dva výrazy. Vybral jsem třídy výrazu překvapení a štěstí. Při trénování klasifikátoru pro kterýkoliv dva jiné výrazy by výsledek dopadl pravděpodobně velice podobně. Bude natrénováno celkem 20 silných klasifikátorů. První má 10

slabých hypotéz. Dvacátý jich má 200. Všechny mezi nimi mají vždy o 10 slabých klasifikátorů víc jak předchozí.



Obr. 20: Graf závislosti počtu klasifikátorů na chybu silného klasifikátoru

Předpokladem je, že chyba bude klesat s počtem klasifikátorů (viz kap. 3). Dle obrázku 20 však tato chyba klesá pouze prvních 60 slabých klasifikátorů. Poté chyba lehce stoupne a až do 200 slabých klasifikátorů na jeden silný klasifikátor lehce stagnuje v rozmezí 1%. Obrázek zhruba potvrzuje klesání chyby. V tab. 6 jsou uvedené naměřené chyby jednotlivých silných klasifikátorů.

Počet slabých klasifikátoru	Chyba silného klasifikátoru
10	15,02 %
20	10,28 %
30	10,67 %
40	8,70 %
50	8,30 %
60	5,93 %
70	6,72 %
80	8,30 %
90	8,30 %
100	7,51 %

Počet slabých klasifikátoru	Chyba silného klasifikátoru
110	7,91 %
120	8,30 %
130	7,51 %
140	7,91 %
150	7,51 %
160	7,51 %
170	7,91 %
180	7,51 %
190	7,91 %
200	8,30 %

Tab. 6: Chyby Klasifikátorů v závislosti na počtu slabých hypotéz

8 Závěr

V úvodu této práce byly shrnuty některé algoritmy využitelné pro klasifikaci a rozpoznávání vzorů. Byly uvedeny neuronové sítě, Support vector machine, metody typu boosting atd. Výrazně jsem se zaměřil na učící algoritmus AdaBoost a některé jeho modifikace. Bylo uvedeno teoretické pozadí algoritmu. Kladnou vlastností tohoto klasifikátoru je, že se během trénování adaptuje na těžko klasifikovatelné vzorky. Byly uvedeny i některé modifikace algoritmu AdaBoost, které zvětšují jeho přesnost nebo zvyšují rychlost vyhodnocení. Tyto modifikace jsou WaldBoost, FloatBoost a TCAcu. Rozebrali jsme vlastnosti slabých klasifikátorů. Popsali jsme si některé typy příznaků. Zaměřili jsme se především na Haarovy příznaky.

Úkolem práce ovšem bylo implementovat aplikaci pro rozpoznávání výrazu obličeje. Databáze, které nabízejí data k tomuto problému, jsme si popsaly ve čtvrté kapitole. Jsou zde uvedeny i dva datasety, z kterých byli využity data, na kterých jsme potom trénovali klasifikátory sloužící k rozpoznávání jednotlivých výrazů. Návrh možného řešení aplikace pro rozpoznávání výrazu obličeje byl uveden v kapitole 5. Pro rozpoznání výrazu obličeje byl vybrán algoritmus AdaBoost s využitím Haarových příznaků. Implementace programů pro rozpoznávání byla popsána o kapitolu později. Byly provedeny experimenty, které všeobecně potvrzují vlastnosti trénovací metody AdaBoost. Bylo ukázáno, že s rostoucím počtem trénovacích dat stoupá přesnost výsledního klasifikátoru. Testy bylo zjištěno, že některé výrazy se rozpoznávají znatelně hůře než jiné. V končeném výsledku byly vzorky z testovací sady špatně zařazeny v 15,77% případů. V porovnání s jinými aplikacemi[26][27] zabývající se rozpoznáváním výrazu obličeje to je zhruba střed.

Další vývoj této práce by se mohl ubírat dvěma směry. Prvním směrem by mohlo být zlepšování kvality výsledného klasifikátoru. To by se dalo docílit použitím Gáborových vlnek místo Haarových, nebo implementací některého rozšíření algoritmu AdaBoost. Druhým směrem by mohlo být zakomponování rozpoznávání výrazu obličeje do videa. K tomuto real time přístupu by se musel zrychlit výpočet příznaků např. použitím integrálního obrazu. Dalším krokem pro dosažení real time detekce je použití např. algoritmu WaldBoost.

Literatura

[1] Freund Y., Schapire R., A Decision-theoretic Generalization of On-line Learning and an Application to Boosting, Journal of Computer and System Sciences, 1997

[2] Viola P., Jones M., Robust Real Time Object Detection, SCTV, Vancouver, Canada, 2001

[3] Šochman J., Matas J., WaldBoost - Learning for Time Constrained Sequential Detection, Center for Machine Perception, Dept. of Cybernetics, Faculty of Electrical Engineering, CVUT Prague, 2005

[4] Wald A., Sequential Analysis, Dover, New York, 1947

[5] Bc. Juránek R., Rozpoznávání Vzorů v Obraze Pomocí Klasifikátorů, diplomová práce, Brno, FIT VUT v Brně, 2007

[6] Lee, T.S.:.Image representation using 2D Gabor wavelets. IEEE Transaction of Pattern Analysis and Machine Intelligence. Vol. 18, No. 10, 1996.

[7] Španěl M., Rozpoznávání Gest ve Video Sekvencích, Vysoké Učení Technické v Brně, 2003

[8] *Back-Propagation*, 2006, http://en.wikipedia.org/wiki/Back_propagation

[9] Vapnik V. N., Lerner A.,, Pattern Recognition Using Generalized Portrait Method, Automation and Remote Control, 1963

[10] Boser B. E., Guyon I. M., Vapnik V. M., A Training Algorithm for Optimal Margin Classifiers, 5th Annual ACM Workshop on COLT, 1992

[11] Neuronová síť, http://cs.wikipedia.org/wiki/Neuronov%C3%A1_s%C3%AD%C5%A5

[12] R. O. Duda, P. E. Hart, D. G. Stork: Pattern Classification, 2nd Edition, John Wiley & Sons, 2001

[13] Li, S.Z., Zhang, Z.Q., Shum, H, Zhang, H.J.: FloatBoost learning for classification. In S. Thrun S. Becker and K. Obermayer, editors, NIPS 15. MIT Press, December 2002.

[14] Sochman, J., Matas, J.: Adaboost with totally corrective updates for fast face detection. AFGR04, 2004.

[15] Database Overview, http://www.nue.tu-berlin.de/wer/goldmann/Research/Database.html

[16] Seznam databází, http://peipa.essex.ac.uk/benchmark/databases/index.html#faces

[17] Seznam databází, http://www.face-rec.org/databases/

[18] PIE Database , Simon Baker, http://www.ri.cmu.edu/projects/project_418.html

[19] The Japanese Female Facial Expression (JAFFE) Database, http://kasrl.org/jaffe.html

[20] Frank Wallhoff: Facial Expressions and Emotion Database http://www.mmk.ei.tum.de/~waf/fgnet/feedtum.html, Technische Universität München 2006

[21] Kanade, T., Cohn, J. F., & Tian, Y. (2000). Comprehensive database for facial expression analysis. Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'00*)*, Grenoble, France

[22] Facial Action Coding Systém, http://en.wikipedia.org/wiki/Facial_Action_Coding_System

[23] Knihovna OpenCV, http://www.intel.com/technology/computing/opencv/

[24] Hradiš Michal: Framework for Research on Detection Classifiers, In: Proceedings of Spring Conference on Computer Graphics, Budmerice, SK, 2008, s. 171-177

[25] Michal Hradiš: AdaBoost in Computer Vision, diplomová práce, Brno, FIT VUT v Brně, 2007.

[26] Facial Expression detection and recognition systém, W.K. Teo, Liyanage C De Silva and Prahlad Vadakkepat, Journal of The Institution of Engineers, Singapore 2004

[27] Real Time Face Detection and Facial Expression Recognition: Development and Applications to Human Computer Interaction. Marian Stewart Bartlett, Gwen Littlewort, Ian Fasel, Javier R. Movellan. Machine Perception Laboratory, Institute for Neural Computation, University of California, San Diego and Intelligent Robotics and Communication Laboratories, ATR, Kyoto, Japan. 2003

[28] Důvěryhodnost naučených modelů, Olga Štěpánková, datel.felk.cvut.cz/xui1/2006ZS/X06.ppt

[29] Umělé neuronové sítě z pohledu rozpoznávání, Václav Hlaváč, Centrum strojového vnímání, Katedra kybernetiky FEL ČVUT Praha

Seznam příloh

Příloha 1. Obsah přiloženého DVD

Složka	Popis obsahu
\BP	Tato práce ve formátu PDF a DOCX
\data\image	Trénovací a testovací data použité v této aplikaci + stručný popis struktury. Jde o vyříznuté obličeje velikosti 192x192
\data\vyb	Obrázky obličejů před úpravou, pouze vybraný a pojmenovaný data.
\data\raw	Raw soubory vektorů příznaků
\programy	Zdrojové texty programů se stručným návodem k použití a jednoduchou programovou dokumentací
\vysledky	Obsahuje výsledky trénování klasifikátorů se stručným popisem