Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Bc. Michal Ševčík

[**Klasifikátor binárnych deskriptorov**](https://147.175.146.86/project/show/5560)

Diplomový projekt II

**Študijný program: Informačné systémy**

**Študijný odbor: 9.2.6 Informačné systémy**

**Miesto vypracovania: Bratislava, Slovensko**

**Vedúci diplomovej práce: Ing. Ján Kvak**

**December, 2014**

ANOTÁCIA

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program: Informačné systémy

Autor: Bc. Michal Ševčík

Diplomový projekt: Klasifikátor binárnych deskriptorov

Vedenie diplomového projektu: Ing. Ján Kvak

December 2014

Cieľom tejto diplomovej práce je výskum existujúcich klasifikátorov použiteľných v oblasti rozpoznávania obrazu. Zameriava sa hlavne na tie klasifikátory, ktoré dokážu pracovať s binárnymi deskriptormi. Tieto klasifikátory sú v práci bližšie opísané. Výsledok práce je nájdenie a prispôsobenie klasifikátora, ktorý by dosahoval dobré výsledky pri použití v prostredí real-time. Spomenuté sú taktiež výsledky klasifikácie pri rôznych konfiguráciách. Hlavne sa opisuje ich úspešnosť a rýchlosť ktoré sa dosiahli.

ANNOTATION

Slovak University of Technology Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Degree Course: Informational systems

Author: Bc. Michal Ševčík

Diploma project: Classification of binary descriptors

Supervisor: Ing. Ján Kvak

2014, December

The focus of this diploma thesis is to research the existing image classificators. It focuses mainly on those classificators that work with binary descriptors. These classificators are furtherdescribed in the work. The output of the work is a classificator that will achieve good results while used in a real-time environment. It also contains results of the classificator with different configurations. It is describing mainly the success and the speed.

[1. Úvod 1](#_Toc387607613)

[2. Analýza 2](#_Toc387607614)

[2.1. Klasifikátor 2](#_Toc387607615)

[2.2. Trénovacia množina 2](#_Toc387607616)

[2.2.1. Dáta pod dohľadom/ bez dohľadu 3](#_Toc387607617)

[3. Algoritmy klasifikácie 3](#_Toc387607618)

[3.1. Nearest Neighbours (K-nearest neighbours) 4](#_Toc387607619)

[3.1.1. Vzdialenostné metriky 6](#_Toc387607620)

[3.1.2. Výhody a nevýhody k-NN 7](#_Toc387607621)

[3.2. Naive Bayes Klasifikátor 8](#_Toc387607622)

[3.2.1. Výhody Naive Bayes Klasifikátora 8](#_Toc387607623)

[3.2.2. Generatívny klasifikátor -Trieda priorít 10](#_Toc387607624)

[3.2.3. Naive Bayes predpoklad 11](#_Toc387607625)

[3.2.3. Prístupy Naive Bayes klasifikátora 12](#_Toc387607625)

[3.3. Semi-Naive Bayes 13](#_Toc387607626)

[3.3.1. Trénovanie fern-u 15](#_Toc387607627)

[3.4. Rozhodovacie stromy 16](#_Toc387607628)

[3.4.1. Trénovanie 17](#_Toc387607629)

[3.4.2. Výhody a nevýhody rozhodovacích stromov  18](#_Toc387607630)

[3.5. Náhodné lesy 19](#_Toc387607631)

[3.5.1. Porovnanie výkonu pri použití viacerých stromov 20](#_Toc387607632)

[3.5.2. Porovnanie náhodných lesov a fern-ov 21](#_Toc387607633)

[4. Návrh 22](#_Toc387607634)

[5. Implementácia 2](#_Toc387607635)3

[5.1. Postup implementácie 2](#_Toc387607615)4

[5. Záver 2](#_Toc387607635)7

[Bibliografia 28](#_Toc387607636)

Skratky

k-NN k-Nearest Neighbours

MLE Maximum Likehood estamination

FAST Features from Accelerated Segment Test

SURF Speeded Up Robust Feature

BRIEF Binary robust independent elementary features

OpenCV Opensource Computer Vision library

1. Úvod

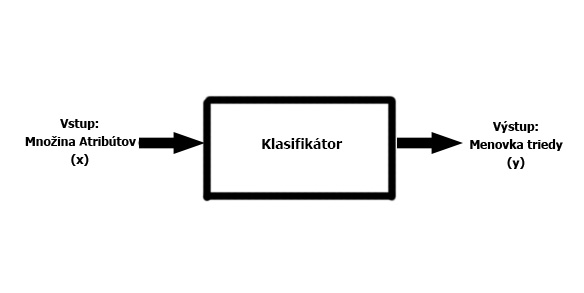
Ako sa technológie vyvíjajú, programy sa stávajú viac sofistikované a majú tendenciu sa stať autonómnymi. Odvetvie umelej inteligencie sa zaoberá vytváraním inteligentného agenta, schopného pozorovať prostredie. Z tohto pozorovania sa program snaží vyvodiť najlepšie riešenie situácie. Aby získal údaje o vonkajších podnetoch, potrebuje senzory ako sú mikrofón, kamera atď, ktoré premenia vonkajšie podnety na dáta. Pri pozorovaní vonkajšieho prostredie kamerou, musí program spracovať dáta, ktoré vidí pred sebou. Typický príklad, kedy je toto spracovanie veľmi dôležité sú Google Automobily (GoogleVehlices). Ide o snahu zautomatizovať automobilovú dopravu, kedy človek len zadá cieľový bod a automobil sa tam automatický dostane. No na ceste sa môžu nachádzať rôzne nečakané prekážky. Na tieto prekážky musí automobil reagovať v čo najkratšej odozve. Pri tomto rozpoznávaní je rýchlosť prvoradá.

1. Analýza

Pri rozpoznávaní obrazu, klasifikácií, sa využíva strojové učenie. Strojové učenie znamená že sa program naučí sám klasifikovať čo na obrázku, ktorý práve sníma, vidí. Ide o transformáciu prijatých dát na informácie. Program sa z týchto informácií naučí rozpoznávať jednotlivé objekty z obrázkov. Rozpoznávanie pracuje na princípe vyhodnotenia nájdeného objektu, ktorý sa potom spracuje a porovná. Porovnáva sa s trénovacou množinou a ako výsledok sa berie ten objekt, na ktorý sa neznámy objekt najviac podobá. Aby sme našli objekt na ktorý sa objekt najviac podobá, používame klasifikátor, ktorý ho rozpozná [1].

* 1. Klasifikátor

Klasifikátor je funkcia f, ktorá namapuje vložený vektor príznakov na výstupnú triedu kde X je Trénovacia množina. Tento základný prístup klasifikátorov vysvetlí obrázok. [1]



Obrázok Proces klasifikácie [1]

* 1. Trénovacia množina

Pri rozpoznávaní sú získané dáta predspracované na príznaky. Aby sme získali správne výstupy (rozpoznanie objektu) používame algoritmy strojového učenia. Tieto algoritmy analyzujú zozbierané dáta. Jednou z metrík určujúce úspešnosť klasifikátora je veľkosť trénovacej množiny.[1] Rozpoznávanie prebieha tak, že sa nasníma neznámy objekt. Tento objekt sa rozpozná podľa klasifikácie čŕt a trénovacej množiny, ktorá ho naučila klasifikovať vlastnosti, podľa získaných čŕt. Môže nastať situácia že odhad nebude postačujúci. V tomto prípade sa môže zvýšiť počet objektov v trénovacej množine, alebo sa zvolí iný klasifikátor a proces klasifikácie sa zopakuje.

* + 1. Dáta pod dohľadom/ bez dohľadu

Dáta v trénovacej množine môžu byť označené menovkami (meno, vek,..) alebo neoznačené. Pri označených údajoch ide o dáta s dohľadom.[1] Tieto dáta sa posielajú klasifikátoru aj s nejakým údajom, ktorý ich opisuje. Učenie klasifikátora, ktorý sa učí s dátami s dohľadom, sa nazýva učenie s učiteľom. Môže ísť o kategorické učenie s učiteľom, kedy sa asociuje názov k objektu (meno k rozpoznávanej tvári), alebo dáta môžu mať číselné alebo usporiadané menovky (vek k rozpoznávanej tvári). Ak dáta majú menovky v podobe mien (kategórie), tak hovoríme že ide o klasifikáciu. Ak menovky sú číselné, hovoríme že ide o regresiu. Učenie môže byť jedno k jednému: párovanie menoviek s dátovými vektormi, alebo môže ísť o odložené učenie (zosilnené učenie).[1]

Pri zosilnenom učení, sa používa dátová menovka[1] (taktiež nazývaná odmena alebo trest). Rozpoznávanie prebieha tak, že systém získa oneskorený signál (odmenu alebo trest) a podľa toho sa snaží odvodiť ďalšie kroky pre rozpoznávanie. Učenie s učiteľom môže mať takisto čiastočne menovky, kde niektoré menovky chýbajú alebo sú zlé. Ide o tzv. semi supervised učenie. Väčšina algoritmov strojového učenia rieši jednu alebo dve situácie z opísaných. Napríklad algoritmy strojového učenia môžu riešiť klasifikáciu ale nie regresiu, algoritmy dokážu robiť semi supervised učenie, ale nie zosilnené učenie, algoritmus dokáže pracovať s číselnými dátami ale nie s kategorickými súčasne. [1]

Ďalšie z algoritmov rozpoznávania pracujú nad dátami, ktoré nie sú pod dohľadom, sú neopísané. Ide len o zistenie čí skúmane dáta spadajú do skupín. Skupina týchto algoritmov sa volá clusteringové algoritmy. Cieľom je zoskupiť neoznačené dátové vektory, ktoré sú si podobné.[1]

1. Algoritmy klasifikácie

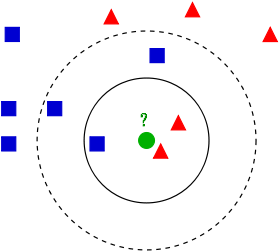
V nasledujúcej kapitole sa budeme zaoberať opisom jednotlivých algoritmov strojového učenia a klasifikácie. Väčšina týchto algoritmov, ktoré spomenieme, sú naimplementované v knižnici OpenCV.

* 1. Nearest Neighbours (k-nearest neighbours)

Klasifikácia NN je založená na porovnávaní neznámych objektov s identifikovanými objektmi z trénovacej množiny. Podľa toho, ku ktorému z jeho “susedov” sa najviac blíži vieme určiť že o čo s najväčšou pravdepodobnosťou pôjde. Je to často krát užitočné vziať viac ako jedného suseda pre získanie relevantnejších výsledkov. Kvôli tomuto sa často krát odkazuje tento klasifikátor na k-NN. Trénovacie vzorky sú potrebné počas spusteného programu (run-time). Celý čas musia byť v pamäti počítača, a preto sa táto klasifikácia označuje aj ako Memory-based classification (klasifikácia založená na pamäti). Pretože začiatok programu je spomalený z dôvodu predspracovania dát, je táto technika učenia považovaná za Lazy Learning technique (Technika lenivého učenia)[2]. A pretože klasifikácia je založená priamo na trénovacej množine, ide o Example-Based Classification (Klasifikácia založená na príkladoch) alebo Case-BasedClassification (Prípadovo založená klasifikácia).

Samotná klasifikácia k-NN má dve fázy: prvá je rozhodnutie, ktorý zo susedných prvkov je najbližší k hľadanému prvku. Druhá fáza – finálna je rozhodnutie triedy do ktorej spadá podľa najbližších susedov. Toto možno vidieť na obrázku. Okrúhly objekt je rozpoznávaný, a trojuholník spolu so štvorcom sú už rozpoznané. Klasifikátor sa podľa vzdialeností (sústredné kružnice) rozhoduje, ako ho klasifikovať. Berie do úvahy len tie prvky, ktoré sú po určenú vzdialenosť.

D je trénovacia množina, ktorá obsahuje všetky rozpoznané prvky, F je množina príznakov každého prvku a Y je množina menoviek.



Obrázok Klasifikácia neznámeho objektu pomocou k-NN [2]

Máme trénovaciu množinu D, ktorá obsahuje trénovacích vzoriek. Vzorky sú opísané podľa príznakov F a všetky tieto numerické príznaky boli normalizované s rozsahom [0,1]. Každá trénovacia vzorka má menovku triedy Cieľom je klasifikovať neznámy prvok q. Pre každý môžeme pomocou výpočtov vzdialenosti medzi q a podľa vzorca:

Existuje veľa spôsobov vzdialenostných metrík. Základná verzia pre kontinuálne a diskrétne atribúty by bola:

* 0 – f je diskrétna a
* 1 - f je diskrétna a
* || - f je kontinuálna

V algoritme k-NN je selekcia založená na vzdialenostných metrikách. Existuje množstvo variácií spôsobov v ktorých k-NN môže byť použitá na rozpoznanie triedy q. Jednoduchým prístupom je priradiť majoritnú triedu hľadanému prvku. Pri vkladaní váh okolitým susedom, dáva väčší zmysel priradiť vyššiu váhu susedovi ktorý je bližšie, ako tomu, ktorý je ďalej. Technika, ktorá sa na takéto hodnotenie používa, sa nazýva vzdialenostné volenie váh.[2] Susedia „volia“ s hlasmi, ktoré majú váhu inverznú k ich vzdialenosti od neznámeho prvku.

Vzorec vzdialenostného volenia váh má tvar [2]:

Voľba prislúcha premennej podľa susedov podieľu jednotky podľa vzdialenosti . Táto vzdialenosť vráti 1 ak sa menovky triedy zhodujú a inak 0.

Ďalší z prístupov je voľba založená na Shepardovej práci a používa exponenciálnu funkciu radšej ako inverznú vzdialenosť[2]:

* + 1. Vzdialenostné metriky

Metrika podobnosti a metrika vzdialenosti sú často používané na zmeranie afinity medzi dvoma objektmi. Afinitou sa myslí príbuznosť. Metrika má formálne vysvetlenie v matematike. Vysvetľuje čo vypočítane hodnoty znamenajú. V prípade k-NN hovorí o vzdialenosti objektu:

* d(x,y) > 0 nezáporná
* d(x,y) = 0 iba ak x=y; identita
* d(x,y) = d(y,x) symetria
* d(x,z) > d(x,y) + d(y,z) trojuholníková nerovnosť

Podľa týchto prvých dvoch metrík vzniká špeciálny prípad Minkowského vzdialenostnej metriky[2] :

Minkowského vzdialenostná metrika je veľmi generický príklad metriky, ktorý možno použiť v k-NN klasifikátoroch. Táto vzdialenosť sa môže použiť pre rôzne dáta, ktoré reprezentujú vektor príznakov. Obrázok, ktorý sa väčšinou pri rozpoznávaní používa, je čiernobiely a histogram H s N levelmi odtieňov, alebo so zásobníkmi, kde je počet pixelov, ktoré spadajú do intervalu reprezentovaného zásobníkom i (vektor h je príznakový vektor). Minkowského vzdialenostný vzorec môže byť použitý na porovnanie dvoch obrázkov, opísaných histogramom. Pri práci s obrázkami existujú aj iné vzorce na porovnanie obrázkov. Ide napríklad o Kullback-Leiblerovu divergenciu a štatistiku [2]:

Pričom H a K sú dva histogramy, h a k sú vektory, ktoré korešpondujú vektorom zo zásobníkových hodnôt a . Tieto vzorce sa zameriavajú na informačnú a štatistickú teóriu a majú isté nevýhody. Prvá z nich je že nejde o metriky a tak nedokážu uspokojiť symetrické požiadavky. Avšak toto sa dá ľahko prekonať pri definovaní modifikovanej vzdialenosti medzi x a y, čo je spôsob vytvorenia priemerov a – Jeffreyho divergencia.[2] Jeffreyho divergencia je symetrická verzia Kullback-Leiblerovej divergencie. Ďalšia z nevýhod je, že tieto meranie sú náchylne na chyby kvôli ohraničeniam zásobníkov. Rozdiel medzi obrázkom a trošku tmavším tým istým obrázkom môže byť veľký ak pixely spadajú do rozdielnych zásobníkov.

* + 1. Výhody a nevýhody k-NN

k-NN je veľmi jednoduchý na pochopenie a implementáciu. Takže by sa dal použiť na každý klasifikačný problém. Výhody tohto prístupu sú:

* Keďže tento proces je transparentný, je ho jednoduché implementovať a ladiť.
* V situáciách, kde vysvetlenie výstupu klasifikátora je užitočné, k-NN môže byť veľmi efektívne ak analýza susedov je užitočná ako vysvetlenie.
* Existujú techniky redukcie hluku, ktoré pracujú len pre k-NN, ktoré môžu zlepšiť efektívnosti klasifikátora.

Na ďalšej strane tento prístup má aj svoje nevýhody:

* Pretože všetka práca sa vykonáva počas behu programu, k-NN môže byť dosť pomalý ak trénovacia množina je veľká.
* Klasifikátor k-NN je veľmi senzitívny na irelevanciu alebo redundanciu príznakov, pretože všetky príznaky prispievajú k podobe a ku klasifikácií. Toto môže byť zlepšené pri opatrnom výbere príznakov.
* Pri veľmi zložitých klasifikačných úlohách, k-NN môže byť prekonaný niektorými exotickými technikami ako napríklad Podpornými vektorovými strojmi alebo neurónovými sieťami.[2]
  1. Naive Bayes Klasifikátor

Naive Bayes klasifikátor je štatistický klasifikátor, ktorý sa zameriava na pravdepodobnosť. To znamená že využíva metódy, ktoré zobrazujú pravdepodobnosť že pri rozpoznávaní ide o daný objekt. Existujú dva spôsoby ako takýto výstup dosiahnuť. Prvý spôsob je priame naučenie funkcie, ktorá vypočíta triedu pravdepodobností. Tento model výpočtu sa nazýva diskriminatívny model, pretože diskriminuje spomedzi tried získaných pri vstupe. Alternatíva k tomuto modelu je class-conditional density (hustota tried-podmienok) pre každú hodnotu y a pre naučenie classpriors (priorít tried) . Potom sa môže aplikovať Bayesovo pravidlo na výpočet pravdepodobnosti. [3]

Toto sa nazýva generatívny model, keďže špecifikuje ako možno generovať príznakové vektory x, pre každú možnú triedu y. Alternatívou ku generatívnemu a diskriminativnému učeniu je rozdelené učenie so všetkými pravdepodobnosťami spolu. Táto metóda priamo mapuje vstupy na výstup:

* + 1. Výhody Naive Bayes Klasifikátora

Medzi výhody Naive Bayes klasifikátora patrí:

* Odmietnutie výberu. Podľa distribúcie pravdepodobnosti vieme, či výsledok klasifikácie je veľmi pravdepodobný alebo nepravdepodobný. Podľa toho sa môže rozhodnúť že neposkytne klasifikáciu a prenechá to na človeka aby rozhodol o čo sa jedná. Tento princíp má uplatnenie napríklad v medicíne, kedy doktor môže rozhodnúť ak pravdepodobnosť nie je dostatočne veľká. [3]
* Zmena užitočnej funkcie. Môže sa skombinovať pravdepodobnostná distribúcia s funkciou užitočnosti aby sa minimalizoval risk. Hoci môžeme diskriminant naučiť aby priamo minimalizoval riziko. Výhodou je, že naša funkcia užitočnosti sa mení a nemusíme znova učiť . [3]
* Kompenzácia pre nevyrovnanosť tried. Ak je jedna trieda omnoho viac pravdepodobná ako druhá (trieda2 = zriedkavá - 1 prípad z 1000). Ak učíme klasifikátor s nevyrovnanými dátami, môže získať triviálnu 99,9% presnosť keď jednoducho sa naučí pravidlo f(x) = trieda1, čiže vždy bude výstup rovnaký bez ohľadu na to, čo je vstupom. Aby sa vyhlo tomuto degradovanému riešeniu, môžeme ho natrénovať s trénovacou množinou, ktorej dáta majú rovnomerne pravdepodobnosti. Zavolá sa výsledný model . Na začiatku behu programu, môžeme tento model modifikovať, aby pracoval nad skutočnou distribúciou dát [3]:

Keďže

Táto konverzia pravdepodobnosti sa nazýva škálovaný pravdepodobný trik.

* Kombinácia modelov. Ak máme dva odlišné príznakové vektory x1 a x2 (jeden zobrazuje krvné testy a ďalší zlomeniny). Oplatí sa viac spraviť dva separátne klasfikátory p(y|x1) a p(y|x2) a potom ich skombinovať. Jeden monolitický klasifikátor sa neoplatí robiť. Základ pri kombinácií výstupov z rôznych zdrojov informácií (fúzny senzor) je vedieť ako spoľahlivý každý zdroj je. Toto presne je to, čo nám pravdepodobnosť hovorí. Jednoduchý spôsob ako skombinovať systémy je predpokladať že príznaky sú podmienkovo nezávisle poskytujúce menovky triedú [3]:

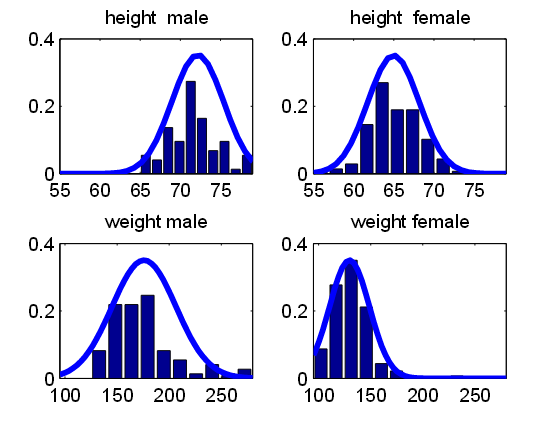
Ide o Naive Bayes predpoklad, ktorý možno vysvetliť využitím triku škálových podobností:

* + 1. Generatívny klasifikátor -Trieda priorít

Pri predpoklade že počet tried C je malý, môžeme jednoducho zistiť prioritnú triedu pri použití y ako multinominálnej náhodnej premennej [3]:

Kde je vektor tried pravdepodobnosti. MLE je [3]:

Pričom je veľkosť trénovacej množiny ktoré majú menovku c. Ak sa použije Dirichletova priorita potom stredná hodnota bude[3]:



Obrázok : Príklad gausovské rozloženie dvoch čŕt(výška a váha u žien a mužov)[3]

* + 1. Naive Bayes predpoklad

Naive Bayes predpoklad alebo idiot Bayes [3] predpoklad je že všetky príznaky sú podmienkovo nezávislé:

Hoci toto tvrdenie je zvyčajne nepravdivé (keďže príznaky sú väčšinou závislé na sebe), výsledný model v prípade Gaussovských dát má tvar [3]:

Takže treba len odhadnúť Gaussovské parametre. V prípade binárnych dát dostaneme [3]:

A v tomto prípade treba odhadnúť separátnyBernoulliho parameter . [3]

Celá Bayesovská pravdepodobnosť, je predpovednie hustoty na triedu s menovkou Y so vstupom X a trénovacou množinou D. Toto je dané nasledujúcim vzorcom[3]:

Čo je často približne:

V prípade kde všetky premenné majú multinominálnu distribúciu s Ditrechtovými prioritami, použitím stredného posteriora druhá rovnica bude úplne korektná.[3]

* + 1. Prístupy Naive Bayes klasifikátora

Pre výpočet pravdepodobnosti pri klasifikácií, existuje viac prístupov a modelov, ktoré možno použiť. Ide o matematické modely založené na rôznych matematických pravidlách. Známe typy sú [8]:

* Gaussian naive Bayes
* Multinominal naive Bayes
* Bernouli naive Bayes

**Gaussian naive Bayes**

Využíva sa pri práci s dátami, ktoré sú kontinuálne. Pri kontinuálnych dátach možno predpokladať že sú rozložené podľa gaussovského modelu. Najskôr sa vykoná segmentácia dát podľa tried a potom sa vypočíta priemer a odchýlka pre x v každej triede. Zo získaných hodnôt – priemeru a odchýlky, možno získať pravdepodobnosť po dosadení do vzorca normálového rozloženia, ktoré obsahuje parametre priemeru a odchýlky:

**Multinominal naive Bayes**

Vzorky v multinominalnom naive Bayes klasifikátore reprezentujú frekvenciu výskytu určitých udalosti, ktoré sú generované multinominálne(,, kde je pravdepodobnosť že udalosť i nastane. Typicky sa používa pre klasifikáciu dokumentov.

**Bernouli naive Bayes**

Hodnoty v tomto modeli, sú booleanove. Taktiež sa používa pre klasifikáciu dokumentov.

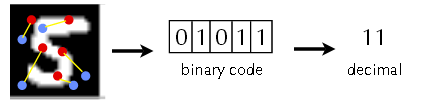
* 1. Semi-Naive Bayes

Semi-Naive Bayes oproti Naive Bayes má viac vyváženú komplexitu. Výmena komplexita/výkon je z dôvodu použitia fernov. Tieto ferny sú o veľkosti S a ich počet je L:

Ferny pracujú tak, že vytvoria sériu binárnych testov na vstupnom vektore. Vždy sa jedná o dvojice bodov, ktoré sa porovnávajú. Platí pravidlo:

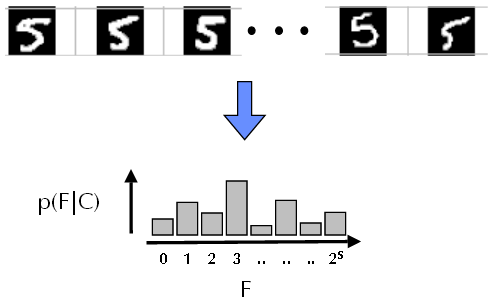
Čo vo všeobecnosti znamená, že ak intenzita prvého bodu je väčšia ako druhého bodu tak výsledok testu je 1 (True) a naopak.

Takéto testy nám potom dajú ako výsledok, S-miestny binárny kód pre príznak, ktorý môže byť interpretovaný ako prirodzené číslo z intervalu. Číslo S sa vyberá podľa počtu dvojíc príznakov resp. podľa počtu binárnych testov.



Obrázok : Binárne testy [5]

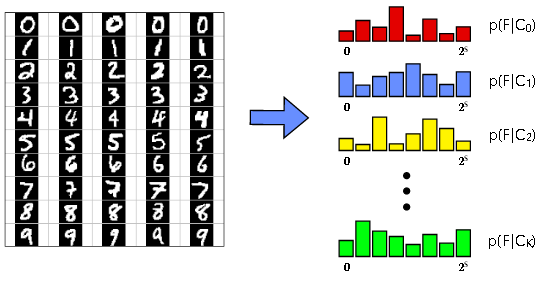
Potom ako sa uskutoční viacero testov na vstupných vektoroch (z trénovacej množiny) rovnakej triedy získame výstup, ktorý je multinominálne rozloženie.



Obrázok Multinominálne rozloženie triedy[5]

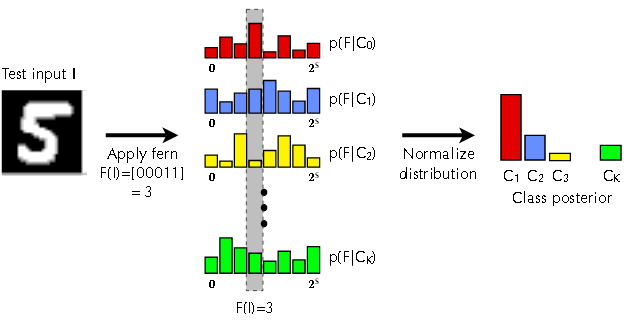
* + 1. Trénovanie fern-u

Ako bol fern opísaný v predchádzajúcej kapitole, na každý prvok z trénovacej množiny je vykonaný binárny test a vypočíta sa výstup . Po natrénovaní celej trénovacej množiny, týmto spôsobom sa naučí multinominálne hustoty v podobe histogramového výstupu pre každú triedu.



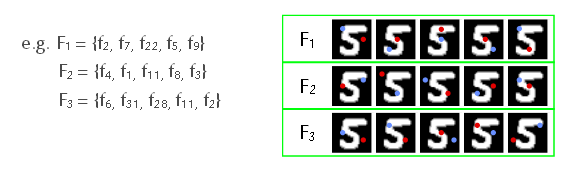
Obrázok Vytvorenie histogramov pre celú trénovaciu množinu [5]

Pri klasifikácií aplikujeme fern na neznámy objekt, a následne vyhľadáme v histogramoch tried ku ktorému naučenému objektu sa tento neznámy objekt najviac podobá:



Obrázok Klasifikácia [5]

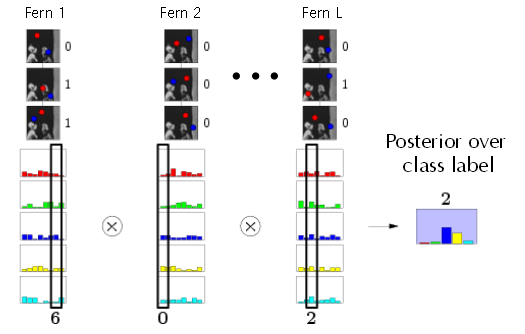
Tento spôsob klasifikácie sa dá vylepšiť pomocou pridania náhodnosti. Pri klasifikácií nebudeme používať iba jeden fern a stále tie isté príznačné body, ale vytvoríme množinu fernov. Každý fern bude obsahovať náhodne zvolené body z množiny príznakov. Tieto ferny budú nezávislé, čiže každý bude obsahovať podmnožinu príznakov, ktorá bude unikátna.



Obrázok Množina fernov [5]

Ferny sú označené F1, F2,... a ich príznaky f1,f2,... Po tomto skombinujeme výstupy pomocou Semi-Naive Bayesovho prístupu:

Pri rozpoznávaní sa používaju viaceré fern-y, ktoré vznikli náhodným výberom príznakov. Každý fern rozpoznávaný objekt očísluje podľa spomínaných binárnych testov. Výstup každého fernu (stĺpec v histograme) sa vektorovo vynásobí. Podľa výsledku sa získa hodnota triedy z trénovacej množiny.[5]



Obrázok Rozpoznávanie [5]

Na obrázku je možné vidieť že prvý binárny test má hodnotu 6 čiže vyberáme 6. hodnotu histogramu, pričom číslujeme od 0. Po vynásobení týchto histogramov, nám vznikne výsledný histogram, ktorý možno vidieť vpravo. V tomto vzniknutom histograme vidíme, že prevahu má 2. stĺpec a tak sa objekt klasifikuje ako trieda 3.

Keďže veľkosť fernov môže byť veľmi veľká (pre fern veľkosti 10 je výstupný histogram v intervale od [0…210] v desiatkovej sústave [0…1024]). Je tu vysoká šanca, že niektoré prvky histogramu zostanú prázdne. Toto bude problém pri Semi-NaiveBayesovi. Takže, ak budeme predpokladať Dirichletovu prioritu na distribúcií P(F|Ck), môžeme priradiť minimálnu pravdepodobnosť každému prvku histogramu. Táto pravdepodobnosť sa vypočíta nasledovne.[5]

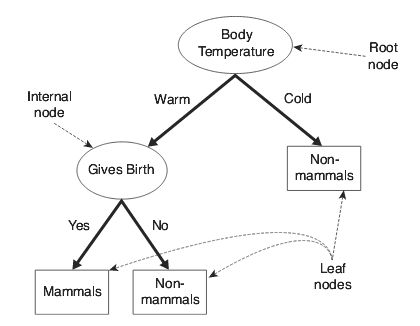
kde určuje koľko krát sme pozorovali výstup Fern, ekvivalentný k z trénovacej množiny triedy . Výhoda fernov je, že sa porovnávajú vždy intenzity dvoch bodov. Ak je obraz pod inými svetelnými podmienkami (preniká naňho silnejšie svetlo), rozpoznávanie bude rovnaké lebo intenzita každého jedného bodu sa zvýši o rovnakú hodnotu.[4][5]

* 1. Rozhodovacie stromy

Ďalším  klasifikátorom sú tzv. rozhodovacie stromy. Ide ako už z názvu vyplýva o stromovú štruktúru. Skladá sa z týchto častí:

* Koreň – ktorý nemá žiadne vstupné hrany a má výstupné hrany, ktoré z neho vychádzajú.
* Uzol – má jedinú vstupnú hranu a presne dve výstupne hrany
* List alebo Terminál – má jednu vstupnú hranu a žiadnu výstupnú hranu

V každom rozhodovacom strome je listom pridelená menovka triedy. Uzly, obsahujú podmienky, podľa ktorých sa ďalej rozhoduje, ktorou vetvou sa ďalej vybrať pri klasifikácií. Toto separuje od seba rozdielne rozpoznávané objekty. Nasledujúci príklad rozhodovacieho stromu zobrazuje klasifikáciu živočícha medzi cicavce/necicavce:



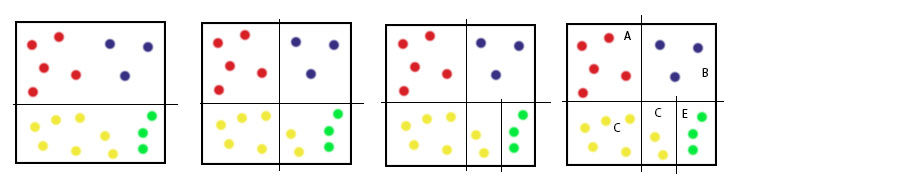
Obrázok Príklad rozhodovacieho stromu pre klasifikáciu cicavcov [6]

Klasifikačný proces pri rozhodovacích stromoch je jednoduchý a priamy. Keď už máme vybudovaný rozhodovací strom začíname v koreni a postupujeme podľa podmienok z jedného uzla do ďalšieho, až kým neprídeme do Terminálu. V termináli  zistíme triedu, ku ktorej sa objekt klasifikuje.

* + 1. Trénovanie

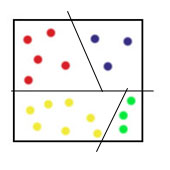
Vytvoriť rozhodovací strom resp. natrénovať takýto strom, znamená rekurzívne delenie trénovacích dát podľa testu typu áno/nie na príznakové vektory. Toto delenie prebieha dovtedy, kým každá trieda obsahuje jej typické príznaky. Potom podľa týchto typických čŕt možno overiť, o ktorú triedu sa jedná.

* Prvé delenie je založené na osovom zarovnaní. Možno vidieť, že sa príznaky delia podľa čiar, ktoré sú rovnobežné s osami. Threshold sa robí na jediný príznak pre každý uzol. Tento prístup je veľmi rýchly na vyhodnotenie.



Obrázok Delenie príznakov

* Druhé delenie sa nazýva všeobecné rovinné delenie. V tomto delení sa neopiera o osi, ale delí sa podľa toho, ako sa to hodí. Čiže deliť sa môže osami, ktoré nie sú rovnomerné s osou. Threshold sa robí s lineárnou kombináciou príznakov. Ide o náročnejší prístup, ale vygenerujú sa menšie stromy. [6]



Obrázok Všeobecné rovinné delenie

* + 1. Výhody a nevýhody rozhodovacích stromov 

Výhody RS:

* Trénovanie môže byť veľmi jednoduché na implementáciu
* Jednoducho zvláda veľké množstvo vstupných premenných

Nevýhody RS:

* Je jasné, že vždy sa dá vytvoriť rozhodovací strom, ktorý má 100% úspešnosť na trénovacej množine, ale majú tendenciu byť veľmi veľké a negeneralizovať veľmi dobre.

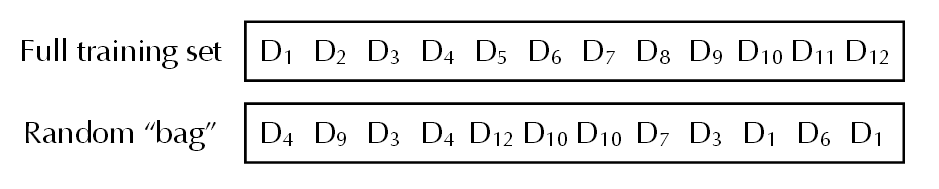
Aby sa predišlo týmto nevýhodám, boli vytvorené tzv. Náhodné lesy (Random Forests).[6]

* 1. Náhodné lesy

Základná myšlienka pri vytvorení tejto štruktúry, bola snaha zakomponovať náhodu do klasifikácie. [3] Táto náhoda bola pridaná do procesu stromového učenia. Keďže každý takýto strom obsahuje prvok náhody, vytvára sa ich viac a všetky sú založené na trénovacej množine. Po tom ako tento klasifikátor dostane vstup, každý strom dospeje k nejakému výstupu. Keďže tieto výstupy nie sú rovnaké, každý strom má hlas. S týmto hlasom prispeje ku konečnému výstupu rozpoznaného objektu. Ak sú stromy naozaj nezávisle, výkon sa zvýši s použitím viacerých stromov.[5]

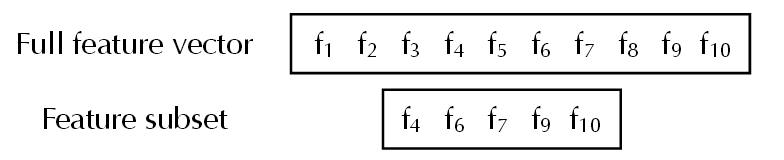
Pri vytváraní sa používajú tieto metódy:

* Bagging (balenie) – vytvorenie novej trénovacej množiny (bag), použitím pôvodnej trénovacej množiny. Z pôvodnej trénovacej množiny sa náhodné vyberú prvky(tieto prvky sa môžu aj opakovať – bootstrap sampling).



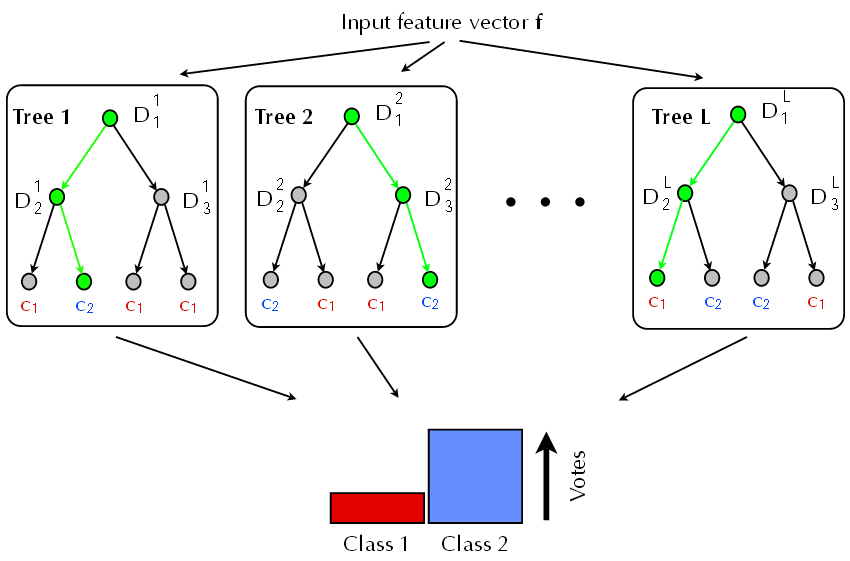
Obrázok Výber prvkov pri bagging [6]

* Feature subset selection (Príznaková podmnožina výberu) – Z celkového príznakového vektora, sa vyberú náhodne podmnožiny príznakov na generovanie každej triedy. Keďže ide o podmnožiny, počet prvkov môže byť menší.



Obrázok Výber príznakov [6]

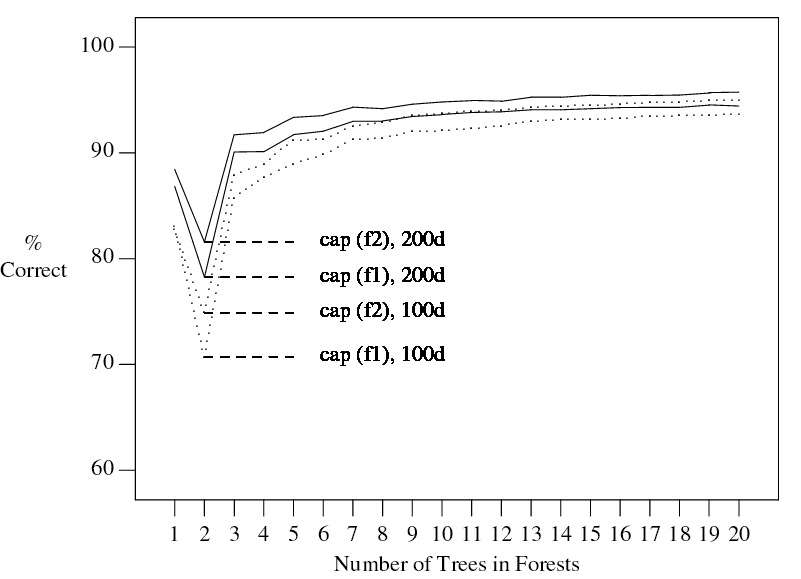
Na nasledujúcom obrázku je zobrazený príklad rozpoznávania objektu podľa vstupného vektora príznakov f. Každý strom ponúkne svoju odpoveď a na konci sa hlasy spočítajú. Podľa toho, ktorá trieda získa najväčší počet hlasov, za tú bude považovaný rozpoznávaný objekt. Na obrázku dostala najviac hlasov trieda Class 2 a tak neznámy objekt bude považovaný za Class 2:



Obrázok Hlasovanie stromov [6]

* + 1. Porovnanie výkonu pri použití viacerých stromov

V príklade z článku [6] boli ručne písané čísla takže išlo o 10 tried. Čísla boli o veľkosti 20x20 pixelov. Trénovacia množina obsahovala 60000 prvkov a testovacia množina 10000. Množina príznakov f1 obsahuje 400 príznakov a f2 obsahuje množinu f1 + gradient a to spolu tvorí 852 príznakov. Každý zo stromov obsahuje celú trénovaciu množinu (nepoužilo sa balenie). Porovnávajú sa rozdielne príznakové podmnožiny (100 a 200 príznakov):



Obrázok Porovnanie použitia viacerých stromov [6]

Z grafu možno vidieť, že skutočne počet stromov do istej miery zvýšil efektívnosť. Miestami je táto efektívnosť zvýšená až o 10%. [6]

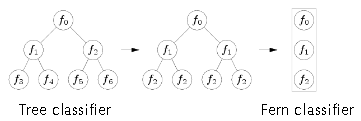
* + 1. Porovnanie náhodných lesov a fern-ov

Lesy

* Rozhodovacie stromy sa hneď naučia pravdepodobnostný posterior
* Aplikujú sa rozdielne sekvencie testov pre každý uzol
* Trénovací čas rastie exponenciálne s hĺbkou stromu

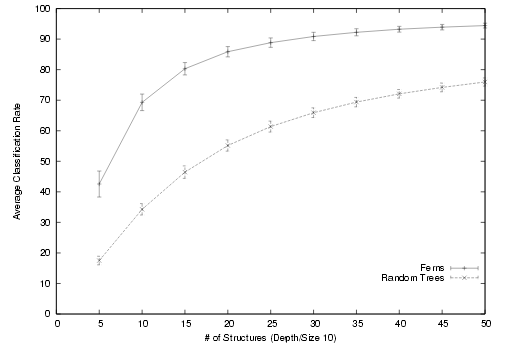
Ferny

* Učenie sa distribúciou trieda-podmienka
* Aplikujú sa rovnaké sekvencie testov, pre každý vstupný vektor
* Trénovací čas rastie lineárne s veľkosťou Fernu S
* Kombinuje hypotézy použitím Bayesovho pravidla (násobenie)[6]



Obrázok Porovnanie stromov a fernov [6]

Pri používaní viacerých stromov sa zvýši miera správneho rozpoznávania a tak isto to je aj pri fern-och. Toto možno vidieť na nasledujúcom grafe.



Obrázok Porovnanie efektívnosti stromov a fernov[7]

1. Návrh

Vyvíjaná aplikácia bude desktopová. Na klasifikáciu sa použijú klasifikátory Naive Bayes a Semi Naive Bayes. Postup klasifikácie bude nasledujúci:

* Extrakcia význačných bodov obrázkov z trénovacej množiny
* Spustenie trénovania
* Extrakcia význačných bodov z neznámeho obrázka
* Spustenie rozpoznávania
* Získajú sa časy a úspešnosti rozpoznávania
* Zmení sa počet fern-ov a pokračuje sa od druhého bodu pričom počet začne na čísle 5 a horná hranica bude určená podľa výsledkov. A teda zvyšovať sa bude pokiaľ výsledky sa budú zlepšovať.

Množinu na testovanie budú tvoriť obrázky z trénovacej množiny, ktoré budú čiastočne deformované. Rovnaký postup rozpoznávania bude pre každý z klasifikátorov.

1. Implementácia

Na začiatku mala byť aplikácia implementovaná v jazyku Java. No neskôr sa jazyk zmenil na C++ z nasledujúcich dôvodov:

* Jazyk C++ je rýchlejší
* Neúplná dokumentácia knižnice OpenCV pre jazyk Java

Po finálnom výbere vývojového jazyka sa prešlo k implementácii. Prostredie IDE pre vývoj je Microsoft Visual Studio 2012.

Pri implementácií sa použila knižnica OpenCV verzie 2.4.9. V samotnej implementácií sú použité dva klasifikátory. A to sú:

* Semi Naive Bayesov klasifikátor za použitia fernovej dátovej štruktúry
* Gaussian Naive Bayesov klasifikátor

Pri spustení aplikácie máme na výber, ktorý klasifikátor použijeme.

* 1. Postup implementácie

Ako prvé sme si vytvorili jednoduché obrázky, ktoré sú súčasťou trénovacej množiny:



Obrázok 9 Obrázky z trénovacej množiny

Ďalej sme vytvorili obrázok, ktorý budeme rozpoznávať – neznámy obrázok.

Po tejto počiatočnej inicializácií a príprave všetkých potrebných rekvizít sa prešlo k implementácií. Ako prvé sme načítali obrázky z trénovacej množiny. Tieto načítané obrázky sme prekonvertovali do čiernobieleho formátu kvôli binárnym deskriptorom, ktoré nepotrebujú informáciu o farbe.

Na detekciu význačných bodov sme použili detektor FAST. Ten zdetegoval všetky význačné body s ktorými sa ďalej pokračovalo v klasifikácií. Aby sme ich mohli použiť, museli sme ich extrahovať. Krok výberu extrakcie je dôležitý. Keďže používame binárne klasifikátory, extrahovať musíme množinu dát, ktoré budú tak isto binárne. A podľa článku [4] v ktorom sa autori taktiež sústredili na využívanie binárnych zdrojov dát pri klasifikácií, rozhodol som sa vybrať BRIEF extraktor pre extrahovanie.



Obrázok 20: Zobrazenie význačných bodov prvý obrázok Obrázok 21: Zobrazenie význačných bodov druhý obrázok

Po detekcií a extrakcií význačných bodov sme zistili, že v knižnici OpenCV sa binárne reťazce rozpoznaných význačných bodov prepočítavajú do dekadickej sústavy. Tieto čísla sú 8 bitové. Vo svojej práci chceme pre Semi-naivný bayesov klasifikátor otestovať rôzne veľkosti fernov a jej dopad na výsledky rozpoznávania význačných bodov. Z tohto dôvodu sme prepočítali tieto dekadické reťazce na binárne a z nich sme vyberali počet bitov podľa veľkosti fernov, ktoré chceme testovať. Ako prvé sme pracovali s veľkosťou 4bity. Po extrakcií a výbere 4 bitov, sme vytvorili histogram. Histogram mal veľkosť podľa veľkosti fernov – 15 a bol dvojrozmerný. Druhý rozmer určoval počet nájdených význačných bodov. Ide o histogram výskytu čísel pre každý význačný bod. Každý riadok vyjadroval jeden význačný bod. To znamená, že ak bolo objavených 30 význačných bodov, tak bolo alokovaných 30 riadkov histogramu. Histogram bol na začiatku inicializovaný na nulu. Po prechádzaní binárnymi reťazcami a ich konverziou na dekadické čísla, stĺpec s číslom, ktoré sa rovnalo číslu prepočítaného binárného reťazca sa inkrementoval. Každý obrázok mal svoj vlastný histogram, ktorý opisoval význačné body. Tieto obrázky spolu s histogramom boli uložené medzi naučené obrázky.

Pre samotné rozpoznávanie neznámeho obrázka bol prvotný postup rovnaký. Rozdiel nastal v tom že dáta neznámeho obrázka sa neučili, ale začali sa klasifikovať podľa dvoch spomínaných klasifikátorov.

Prvý, Semi Naive Bayes klasifikátor, funguje na princípe spomenutom na strane 14. Histogram neznámych bodov sa prechádza bod po bode. Čísla význačných bodov histogramu neznámeho obrázka určujú, ktoré stĺpce natrénovaných histogramov sa majú spočítať. Neznámy bod je rozpoznaný ako ten bod, ktorého súčet bol najväčší. Takýto postup sa aplikuje na každý význačný bod neznámeho obrázka.

Druhý spôsob klasifikácie použitím Gaussian Naive Bayes klasifikátora pracuje na matematickom princípe. Pri tomto spôsobe je potrebné vytvoriť deformácie obrázkov trénovacej množiny. Zo vstupných bodov trénovacieho obrázka sa vytvoria priemery a odchýlky pre jednotlivé význačne body. Pri tejto deformácií treba dbať na to, že význačné body zostanú tie isté aj po deformácií – treba ich pozície tiež deformovať rovnakým spôsobom ako samotné obrázky. Takže pre každý význačný bod obrázka bude existovať množina priemerov a odchýlok, ktoré sa budú používať pri klasifikácií. Priemery vypočítame zrátaním hodnôt pre každý stĺpec význačného bodu obrázka z trénovacej množiny a jeho všetkých deformácií. Odchýlky vypočítame pomocou priemerov. Na každú hodnotu v riadku neznámeho obrázka aplikujeme Gaussov výpočtový model:

Takto sa pre neznámy význačný bod vypočítajú pravdepodobnosti pre každý význačný bod z trénovacej množiny. Neznámy význačný bod je rozpoznaný ako ten, ktorého pravdepodobnosť má najväčšiu hodnotu. Toto sa spraví pre každý význačný bod. Tieto dva spôsoby sa budú v nasledujúcich kapitolách porovnávať a tvoriť štatistiky pre rôzne veľkosti fernov.

1. Záver

Práca sa venuje analýze existujúcich klasifikátorov. Vysvetľujú sa spôsoby akými pracujú a porovnávajú sa výsledky niektorých prístupov. Po tejto analýze je spomenutý návrh riešenia problematiky binárnych klasifikátorov. Posledná kapitola sa sústredí na vysvetlenie implementácie aplikácie, ktorá bola navrhnutá v kapitole návrh. Výstup práce je prototyp aplikácie, ktorý využíva dva klasifikátory na klasifikáciu význačných bodov. Táto klasifikácia bude obohatená o zvýšenie počtu obrázkov v trénovacej množine – vygenerujú sa deformácie existujúcich obrázkov. Po tomto obohatení aplikácie sa začne vykonávať samotné testovanie a vytváranie štatistík úspešnosti pre jednotlivé prístupy klasfikácie pri ich rôznych konfiguráciách.

Bibliografia

[1] G. Bradski and A. Kaehler, *Learning OpenCV*. .

[2] S. J. Delany, “k -Nearest Neighbour Classifiers,” pp. 1–17, 2007.

[3] K. P. Murphy, “Naive Bayes classifiers Generative classifiers,” pp. 1–8, 2006.

[4] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, “Fast keypoint recognition using random ferns.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, pp. 448–461, 2010.

[5] A. Bosch, A. Zisserman, and X. Muoz, “Image Classification using Random Forests and Ferns,” *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. pp. 1–8, 2007.

[6] D. Capel, “Random Forests and Ferns The Multi-class Classification Problem.”

[7] M. Ozuysal, P. Fua, and V. Lepetit, “Fast Keypoint Recognition in Ten Lines of Code,” *2007 IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007.

[8] John, George H.; Langley, Pat (1995). "*Estimating Continuous Distributions in Bayesian Classifiers*". Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence. Morgan Kaufmann. pp. 338–345.

[9] Metsis, Vangelis; Androutsopoulos, Ion; Paliouras, Georgios (2006). "*Spam filtering with Naive Bayes—which Naive Bayes*?". Third conference on email and anti-spam (CEAS) 17.