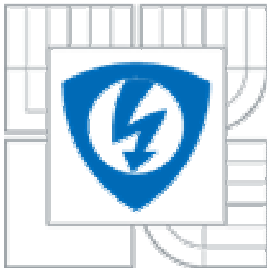
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF BIOMEDICAL INGINEERING

KLASIFIKAČNÍ METODY ANALÝZY VRSTVY NERVOVÝCH VLÁKEN NA SÍTNICI

CLASSIFICATION METHODS FOR RETINAL NERVE FIBRE LAYER ANALYSIS

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. PETR ZAPLETAL
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN ODSTRČILÍK
SUPERVISOR

BRNO 2010

ZDE VLOŽIT ORIGINÁL ZADÁNÍ

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:	Petr Zapletal
Bytem:	Komenského 73, Židlochovice, 667 01
Narozen/a (datum a místo):	27. listopadu 1983 v Brně

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 53, Brno, 602 00
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Jiří Jan,CSc, předseda rady oboru Biomedicínské a ekologické inženýrství
(dále jen „nabyvatel“)

Čl. 1

Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):

   ☐ disertační práce
   ☒ diplomová práce
   ☐ bakalářská práce
   ☐ jiná práce, jejíž druh je specifikován jako .....
   (dále jen VŠKP nebo dílo)

   Název VŠKP: Klasifikační metody analýzy vrstvy nervových vláken na sítnici

   Vedoucí/ školitel VŠKP: Ing. Jan Odstrčilík
   Ústav: Ústav biomedicínského inženýrství
   Datum obhajoby VŠKP: _____

   VŠKP odevzdal autor nabyvateli*:

   ☒ v tištěné formě – počet exemplářů: 2
   ☒ v elektronické formě – počet exemplářů: 2

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.

3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

* hodící se zaškrtněte

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.

2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.

3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti

   ☒ ihned po uzavření této smlouvy
   ☐ 1 rok po uzavření této smlouvy
   ☐ 3 roky po uzavření této smlouvy
   ☐ 5 let po uzavření této smlouvy
   ☐ 10 let po uzavření této smlouvy
   (z důvodu utajení v něm obsažených informací)

4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.

2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.

3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.

4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: 21. května 2010

…….. ……

Nabyvatel Autor

Abstrakt

Tato práce se zabývá klasifikací vrstvy nervových vláken na sínici. Pro klasifikaci jsou použita data získaná šesti různými metodami texturní analýzy. Každá metoda vypočítá ze vstupních obrazů vektor příznaků, který je pro danou skupinu charakteristický. Vlastní třídění je realizováno třemi algoritmy učení s učitelem a jedním algoritmem učení bez učitele. Jako první je otestován algoritmus Ho-Kashyap. Poté Bayessovský klasifikátor NDDF (Normal Density Discriminant Function) a pro třetí klasifikátor je použita metoda nejbližších sousedů (Nearest Neighbors) k-NN. Jako poslední je zde odzkoušen klasifikátor K-means, který pracuje na principu shlukové analýzy. Pro větší kompaktnost jsou použity tři metody výběru testovacích dat pro algoritmy učení s učitelem. Jsou to „Repeated random subsampling cross validation“, „K-fold cross validation“ a „Leave one out cross validation“. Všechny použité třídící algoritmy jsou nakonec porovnány podle výsledné chyby klasifikace.

Klíčová slova: sítnice, klasifikátor, algoritmus, třídění, Ho-Kashyap, NDDF, nejbližší soused, K-means, cross validation

Abstract

This thesis is deal with classification for retinal nerve fibre layer. Texture features from six texture analysis methods are used for classification. All methods calculate feature vector from inputs images. This feature vector is characterized for every cluster (class). Classification is realized by three supervised learning algorithms and one unsupervised learning algorithm. The first testing algorithm is called Ho-Kashyap. The next is Bayess classifier NDDF (Normal Density Discriminant Function). The third is the Nearest Neighbor algorithm k-NN and the last tested classifier is algorithm K-means, which belongs to clustering. For better compactness of this thesis, three methods for selection of training patterns in supervised learning algorithms are implemented. The methods are based on Repeated Random Subsampling Cross Validation, K-Fold Cross Validation and Leave One Out Cross Validation algorithms. All algorithms are quantitatively compared in the sense of classication error evaluation.

Keywords: retina, classificator, algorithm, classification, Ho-Kashyap, NDDF, Nearest Neighbor, K-means, cross validation

ZAPLETAL, P. *Klasifikační metody analýzy vrstvy nervových vláken na sítnici*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 60s, 3 přílohy. Vedoucí diplomové práce: Ing. Jan Odstrčilík.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Klasifikační metody analýzy vrstvy nervových vláken na sítnici jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 21. května 2010 .....
podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Janu Odstrčilíkovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne 21. května 2010 .....
podpis autora

O B S A H

Seznam obrázků

Seznam tabulek

ÚVOD

Klasifikace vrstvy nervových vláken na sítnici může být v mnoha případech užitečným nástrojem při diagnostice řady očních chorob. Mezi nejzávažnější a nenapravitelné onemocnění patří zelený zákal (glaukom). Tato práce by mohla pomoci při diagnostice tohoto závažného onemocnění.

Tato diplomová práce se zabývá klasifikací obrazů sítnice z databáze ÚBMI, FEKT v Brně. Obrazy byly pořízeny digitální fundus kamerou Canon CF-60UDi s digitálním fotoaparátem Canon D20 na oční klinice Tomáše Kuběny ve Zlíně. Z těchto snímků byly vybrány charakteristické obrazy o rozměrech 41×41 pixelů a různými metodami texturní analýzy byly vytvořeny texturní příznaky. Ty byly za účelem testování jejich schopnosti klasifikovat zdravou a nemocnou tkáň vrstvy nervových vláken na sítnici podrobeny několika klasifikačním algoritmům.

Cílem je implementovat vybrané přístupy klasifikace a vytvořit metodiku pro klasifikaci vzorů. Výsledkem každé metody je dosažená chyba klasifikace.

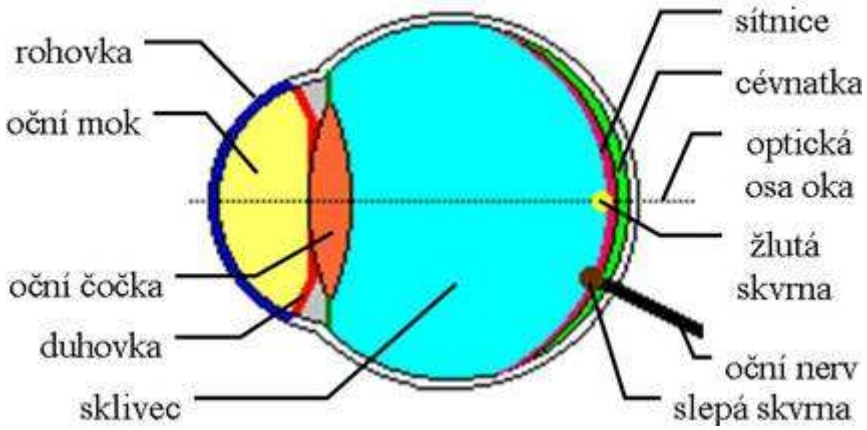
Nejprve je zde uveden algoritmus Ho-Kashyap, který patří mezi lineární klasifikátory. Tento algoritmus nepatří mezi nejznámější, ale jeho výsledky jsou velice dobré. Dalším testovaným algoritmem je NDDF (Normal Density Discriminant Function). Tento algoritmus patří mezi Bayessovské klasifikátory, protože jeho základem je Bayessova věta. Jako následující klasifikátor je zde použit algoritmus Nearest Neighbor. Metoda je známá jako metoda nejbližších sousedů. Posledním testovaným algoritmem je metoda K-means, která patří do metod učení bez učitele. Jedná se o metodu založenou na shlukování dat (clustering).

1 Lidské oko

1.1 Anatomie lidského oka

Lidské oko je párový orgán zraku a zároveň nejsložitější smyslový orgán. Vnímáme jím až 80% informací ze svého okolí. U dospělého člověka má oční koule průměr asi 24 mm a přibližně kulovitý tvar [9].

Při vstupu do oka prochází světlo nejprve rohovkou a očním mokem. Přes duhovku pokračuje do oční čočky, pak prochází přes sklivec a dopadá na sítnici. Popis částí oka je na obrázku 1.1.



Obr.1.1 Stavba oka (zdroj: [9]).

Oční koule - jedná se o asymetrickou kouli o průměru 24 – 25 mm. Hloubka přední komory je 3,5 mm. Před oční čočkou je duhovka, slouží jako clona a určuje barvu očí. Její svalová vlákna dokážou měnit průměr zornice pro procházející světlo od 2 do 6 mm. Uvnitř oční koule je vyplněna bělimou s nitroočním tlakem 2-3 kPa. Stálost tlaku udržuje produkce komorové vody. Pohyb očí je zajištěn šesti pohybovými svaly.

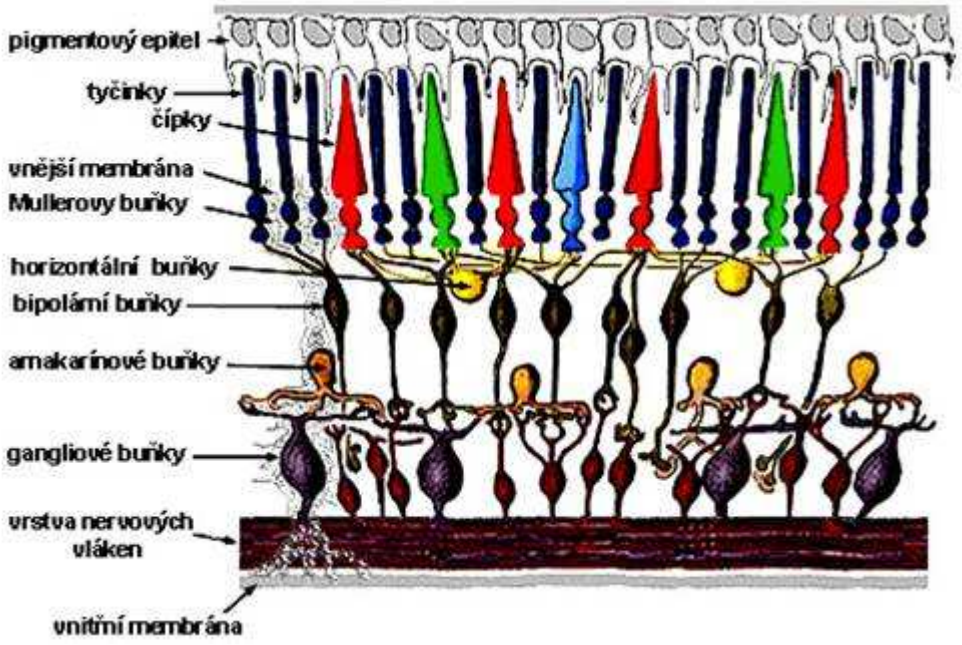
Bělima - je to tuhá vazivová blána o tloušťce 0,3 – 1,0 mm. Tvoří 80% povrchu oční koule. V přední části přechází v průhlednou rohovku.

Rohovka - asi 0,5 mm silná průhledná blána, která musí být neustále zvlhčována. Skládá se z několika nepostradatelných vrstev. Patří mezi nejcitlivější části lidského těla.

Cévnatka - pigmentová vrstva, která vyživuje cévy na vnitřní straně oční koule. Zabraňuje rozptylu světelných paprsků uvnitř oka. V přední části přechází v řasnaté tělísko. Podkladem je sval, z jehož okrajů vybíhají tenká vlákna, která připojují pouzdro čočky. Smrštění svalu povolí tah vláken a dojde k vyklenutí čočky. Při uvolnění svalu dojde ke zploštění čočky. Takto se mění optická mohutnost čočky (ohnisková vzdálenost).

Oční čočka - nehomogenní těleso o tvaru dvojvypuklé čočky, tvořené dokonale průhlednou, tuhou rosolovitou hmotou o tloušťce asi 4 mm. Povrchové části mají index lomu 1,38 a vnitřní 1,41.V průběhu života přibývají na čočce vrstvy, které tvrdnou a snižují akomodaci oka. Optická mohutnost vlastní čočky je asi 18 dioptrií.

Sítnice - je silná 0,2 – 0,4 mm. Tvoří ji 11 vrstev (viz Obr.1.2.). Sítnice obsahuje 1,2 milionu tyčinek a 5 – 7 milionů čípků. Tyčinky slouží pro skotopické vidění (za šera) a dokáží reagovat na dopad světla již jednoho až dvou fotonů. Čípky slouží pro fotopické vidění (barevné). Jsou méně citlivé na světlo, ale jsou schopné vnímat barvy. Sítnice je citlivá na světlo s vlnovou délkou 390 – 790 nm. Největší citlivost je v okolí 555 nm a odpovídá zelené barvě. Poruchami barvocitu trpí asi 4 % lidí (9 % mužů a 0,4 % žen).



Obr.1.2 Jednotlivé vrstvy sítnice (zdroj: [9]).

Žlutá skvrna - místo nejostřejšího vidění. Má průměr asi 1 mm a obsahuje převážně čípky.

Slepá skvrna - místo, kde zrakový nerv opouští oční kouli. Je vzdálená asi 5 mm od žluté skvrny. Na rozdíl od ní neobsahuje žádné čípky ani tyčinky.

Oční komory - štěrbinovité prostory ve kterých cirkuluje komorová voda tvořená převážně krevní plazmou. Nacházejí se mezi rohovkou a duhovkou a mezi duhovkou a čočkou.

Sklivec - gelová tkáň. Kromě jiného zachovává oku jeho tvar a také slouží pro fixaci sítnice na cévnatce.

Oční nerv - párový senzorický mozkový nerv. Slouží pro vedení impulsů ze sítnice do mozku.

1.2 Oční vady a nemoci

Proto, aby rovnoběžné paprsky světla přicházející do oka dopadaly po průchodu jeho optickou soustavou přesně do místa nejostřejšího vidění – žlutá skvrna, musí být v oku zachován správný poměr délky a síly optické soustavy. Řada dědičných a vývojových faktorů může vést ke strukturálnímu nepoměru. Je-li oko příliš dlouhé nebo krátké, nedopadají zaostřené paprsky přímo na sítnici. To se může projevovat v podobě dioptrických vad. [10].

Krátkozrakost (myopie) - patří mezi nejčastější oční dioptrickou vadou. Postihuje až 30% obyvatel. U myopie je oko příliš dlouhé nebo lomivost optické soustavy oka je příliš veliká a světelné paprsky se sbíhají před místem nejostřejšího vidění. Na sítnici potom dopadá rozostřený obraz. Tuto vadu obvykle napravuje čočkou - rozptylkou. Ta upraví svazek paprsků tak, aby se znovu setkávaly na sítnici.

Dalekozrakost (hypermetropie) - postihuje asi 10 % obyvatel. U tohoto nemocnění je oko příliš krátké nebo lomivost oka je nedostatečná. Světelné paprsky se proto sbíhají až za místem nejostřejšího vidění a na sítnici dopadá opět obraz rozostřený. Vada se napravuje čočkou - spojkou, které upravují průběh paprsků tak, aby se znovu setkávaly na sítnici.

Astigmatismus - neschopnost vidět ostře na jakoukoli vzdálenost pro nepravidelný tvar oční rohovky. Rohovka má nepravidelný polokulovitý tvar. Je v některých osách zploštělá nebo naopak více zakřivená a část světelných paprsků se sbíhá mimo místo nejostřejšího vidění a na sítnici dopadá rozostřený, zamlžený a deformovaný obraz. Astigmatismus se může vyskytovat samostatně nebo v kombinaci s krátkozrakostí či dalekozrakostí. Vadu obvykle napravujeme cylindrickými čočkami (lomí paprsky jen v jedné rovině).

Aberace vyššího řádu - jsou odchylky a nepravidelnosti optického systému oka. Mezi hlavní aberace vyššího řádu řadíme koma, sférickou aberaci, sekundární astigmatismus, atd. Aberace vyššího řádu nejsou brýlemi, kontaktními čočkami a také tradiční laserovou chirurgií korigovány, ale mohou být zdrojem zrakových obtíží např. horší vidění za tmy a za šera, dvojité vidění apod.

V**etchozrakost** (presbyopie) - tato vada je s spojená s věkem. Obvykle postihuje lidi starší 45-ti let a projevuje se obtížemi při čtení na blízko. Čočka uvnitř oka nemá již dostatečnou pružnost k tomu, aby účinkem zaostřovacích svalů měnila svůj tvar a tím i optickou mohutnost dle vzdálenosti pozorovaného předmětu. S věkem pružnosti čočky ubývá a starší člověk tím ztrácí schopnost zaostřit na blízko. Tato vada se obvykle řeší brýlemi na čtení. Je možná i metoda monovision, při které kontaktními čočkami či zákrokem korigujeme jedno oko na dálku a druhé na blízko.

K patologickým změnám na oku může docházet z různých důvodu. Choroby oka můžou být způsobené například viry, bakteriemi nebo plísněmi, důsledkem nějakého jiného onemocnění (cukrovka). Velmi časté jsou věkem podmíněné degenerativní změny, nebo vrozené postižení a v neposlední řadě úrazy oka [2,11].

Šedý zákal (katarakta) - patří k častým onemocněním starších lidí. Asi 50% lidí starších 60-ti let má určitý stupeň šedého zákalu. Jedná se o oční nemoci, které se projevují zákalem oční čočky. Katarakta se řeší operačním zákrokem, při kterém je zkalená čočka nahrazená čirou umělou.

Keratokonus - degenerativní, nezánětlivé onemocnění rohovky, u kterého se určitá část rohovky ztenčuje a vyklenuje. Tím dochází k její deformaci a postupně vzniká tzv. nepravidelný astigmatizmus, což má za následek ztrátu zrakové ostrosti.

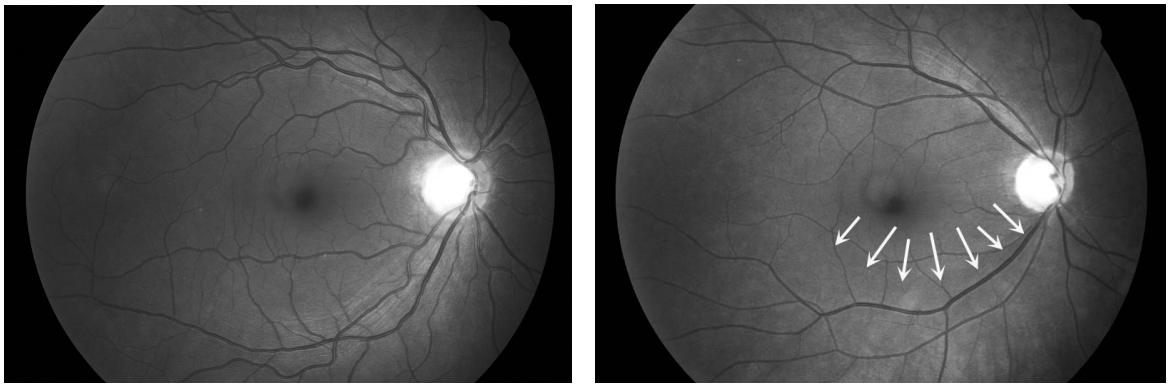
Syndrom suchého oka - nemoc, při které je povrch oka suchý a oko není chráněno slzným filmem. Jedná se tedy o poruchu smáčení oka slzy. Důsledkem může být zhoršené vidění, záněty očí, únava očí apod.

Zánět spojivek - má velmi širokou škálu forem. Od několikadenního pocitu cizího tělíska v oku, přes hnisavou, ale krátkodobou konjunktivitídu, až po dlouhodobé chronicky obtěžující potíže spojené s poruchou tvorby slz.

Diabetická retinopatie - asi 5 - 6 % lidí trpí cukrovkou, která je příčinou vzniku diabetické retinopatie. Jedná se o poškození sítnice, které vzniká důsledkem celkového postižení cév. Dochází tak k poškození cév vyživujících sítnici.

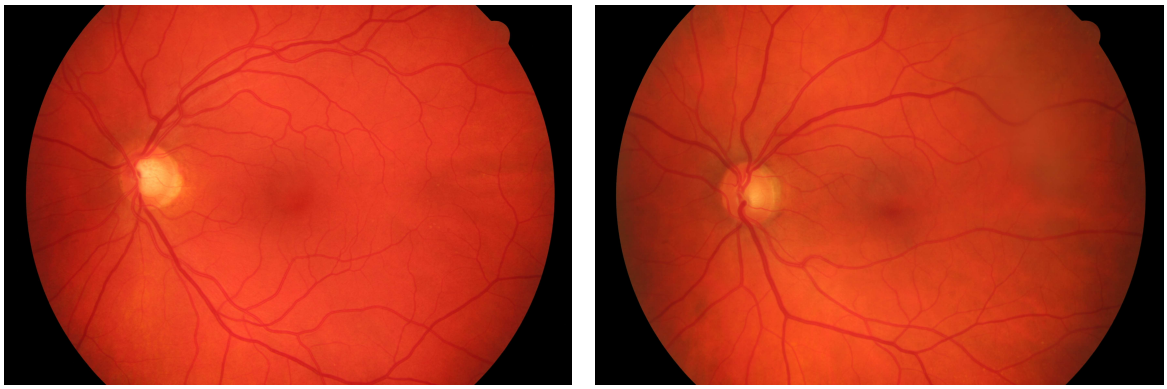
Šilhání (strabizmus) - jedná se o zrakové postižení, u kterého je porušena vzájemná spolupráce obou očí. Každé oko míří jiným směrem. Ve většině případů se jedno oko kouká rovně a druhé je stočené jiným směrem.

Zelený zákal (glaukom) - zelený zákal je závažné oční onemocnění způsobené především zvýšeným nitroočním tlakem. Doprovázejí ho změny na zrakovém nervu. Poškození vláken tohoto nervu častokrát vede až k výpadkům v zorném poli. U glaukomu pomalu odumírají nervové buňky sítnice a jejich vlákna. To vede k přerušení spojení mezi okem a mozkem. Na obr.1.3 a) vidíme snímek oční sítnice zdravého pacienta. Obrázek byl zhotoven bez použití červeného světla. Nervová vlákna jsou patrná kolem očních cév (bílé pruhy). Na obr.1.3 b) vidíme sítnici pacienta s glaukomem. Nervová vlákna v místě označeném šipkami chybí.



a) b)

Obr.1.3 Snímky sítnice a) snímek zdravého oka, b) snímek poškozeného oka (zdroj: [ÚBMI])



a) b)

Obr.1.4 Barevné snímky sítnice - nejsou zde viditelná nervová vlákna a) snímek zdravého oka, b) snímek poškozeného oka (zdroj: [ÚBMI])

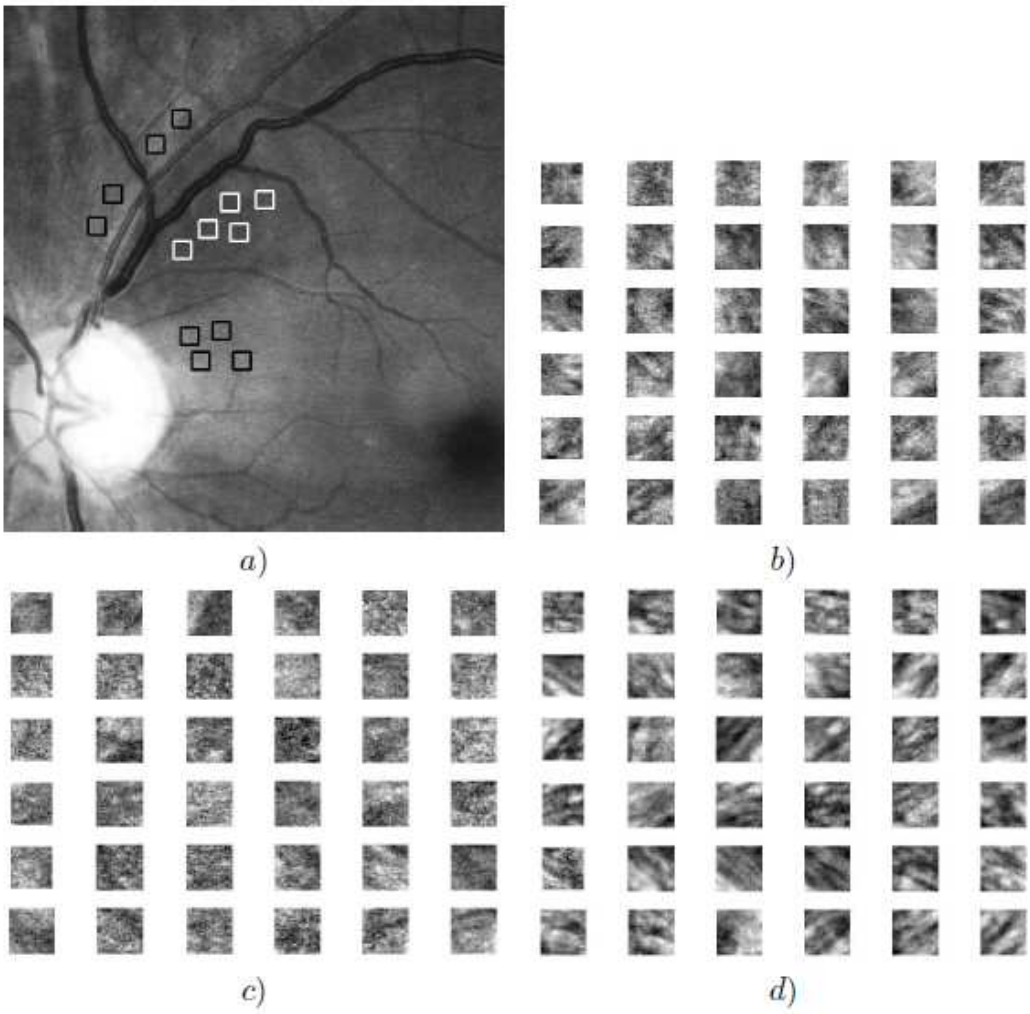
2 Vstupní data

Oftalmologické obrazy sítnice v databázi ÚBMI byly získány během let 2005-2007 na oční klinice Dr. Kuběny ve Zlíně. Tyto obrazy byly pořízeny digitální fundus kamerou Canon CF-60UDi s digitálním fotoaparátem Canon D20.

Databáze obsahuje jednak snímky sítnice zdravých pacientů a jednak retinální snímky pacientů s glaukomem. Vrstva nervových vláken zdravého pacienta je charakteristická světlou proužkovou strukturou (žíháním). U pacientů s glaukomem toto žíhání chybí.

Snímky byly převedeny na šedotónové a následně z nich bylo vybráno několik vzorků o rozměrech 41×41 pixelů, viz obr.2.1. Ty byly rozděleny do následujících tříd [3]:

Třída A - vzorky obsahující vrstvu nervových vláken od pacientů s glaukomem
Třída B - vzorky bez vrstvy nervových vláken od pacientů s glaukomem
Třída C - vzorky vrstvy nervových vláken od pacientů se zdravými oči



Obr.2.1 a) Detail obrazu sítnice s viditelnými nervovými vlákny, b) vzorky zdravé tkáně od nemocných pacientů (černé čtverečky v detailu), c) vzorky tkáně pacientů s glaukomem (v detailu bílé čtverečky), d) kontrolní vzorky od zdravých pacientů. (zdroj: [3])

3 Použité metody texturní analýzy

Pro výpočet vstupních dat bylo využito texturní analýzy. První skupinou metod texturní analýzy jsou metody založené na strukturálním přístupu. Ty nahlížejí na texturu jako na soubor konkrétních primitiv v prostoru. V této skupině se popisuje textura pomocí lokálních binárních vzorů, např. strukturální metoda. Druhou skupinou texturní analýzy jsou metody statistické. Ty popisují texturu pomocí statistických parametrů, které jsou závislé na rozložení pravděpodobnosti nebo vztazích mezi pixely. Mezi tyto metody patří statistiky prvního a druhého řádu. V rámci práce do této skupiny patří základní statistika a metody využívající co-occurrence a run-length matice. Třetí skupinou jsou metody založené na vytváření modelu textury. Metody popisují textury pomocí odpovídajícího matematického modelu, např. výpočet fraktálních koeficientů a Markovových náhodných polí. Do poslední skupiny patří metody texturní analýzy založené na transformaci z prostorové oblasti do oblasti koeficientů dané transformace, např. Fourierova či vlnková transformace [13].

V práci byly použity pouze vektory příznaků získané příslušnými metodami texturní analýzy. Proto je zde uveden jen jejich stručný popis. Zmíněné metody jsou převzaty z dřívějších prací na ÚBMI, FEKT v Brně [3,13].

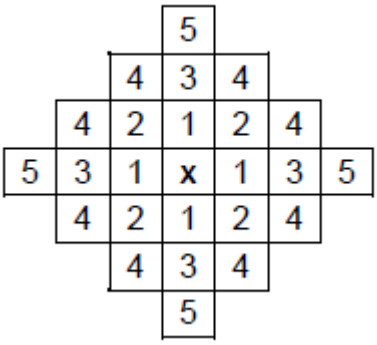
3.1 Markovovy náhodná pole

Modelování pomocí Markovových náhodných polí popisuje pravděpodobnost prostorových interakcí mezi pixely v texturním obraze. Na snímky sítnice byl použit Gaussovský model Markovových náhodných polí. Ten předpokládá, že lokální textura je tvořena sadou pozorování s nulovou střední hodnotou. Pro analýzu textury byl použit model pátého řádu se strukturou symetrického okolí kolem centrálního pixelu, obrázek 3.1. Jednotlivá pozorování lze vyjádřit diferenční rovnicí [13]:

$$y(s) = \sum_{r \in N_S} \varphi_r \, y(s+r) + e(s), \qquad (3.1)$$

kde N_s reprezentuje okolí centrálního pixelu s, φ_r je parametr modelu náležící konkrétnímu pixelu v okolí r a $e(s)$ je stacionární Gaussovský šumový proces s nulovou střední hodnotou a známým rozptylem σ.

Výstupem metody je 6 texturních příznaků. 5 příznaků reprezentuje vliv okolí na centrální pixel a zbývající 1 parametr charakterizuje Gaussovský šumový proces daného modelu. Více o této metodě texturní analýzy je možné nalézt v literatuře [13].



Obr.3.1 Struktura symetrického okolí 5. řádu (zdroj: [13])

3.2 Fraktální koeficient

Fraktální modely se používají pro zpracování obrazu od osmdesátých let. Jejich použití můžeme rozdělit do dvou skupin. Do první patří komprese obrazů a druhou skupinou je modelování textur a jejich segmentace s následnou klasifikací. Podrobněji je tato metoda popsaná v literatuře [3].

Jako fraktál můžeme označit objekt, který je soběpodobný. To znamená, že pokud daný útvar pozorujeme v jakémkoliv měřítku či rozlišení, pozorujeme stále opakující se určitý charakteristický tvar. Na první pohled mívá fraktál velmi složitý tvar, ale je generován opakovaným použitím jednoduchých pravidel [14].

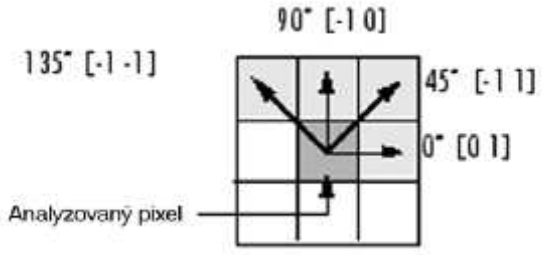
Vlastní příznaky jsou vypočítány z 1D a 2D výkonového spektra obrazu [3].

3.3 Metoda co-occurrence

Tato metoda patří do statistik 2. řádu. Jedná se o matice současného výskytu pixelů v určitém směru. Patří mezi nejpoužívanější metody pro texturní analýzu nervových vláken. Základem je sdružený 2D histogram. Ten vyjadřuje četnost opakujících se kombinací jasu na odpovídajících si pozicích v obraze, [6,8]. Z matice co-occurrence lze vyčíst až 14 parametrů.

V našem případě je použito deset příznaků, jejichž detailní popis je uveden v [3]:

- entropie (texture entropy)
- kontrast (texture kontrast)
- homogenita (texture homogenity)
- korelace (texture correlation)
- maximum pravděpodobnosti (maximum probability)
- uniformita energie (uniformity of energy)
- rozdíl entropií (diference entropy)
- součet průměru (sum average)
- součet entropie (sum entropy
- míra korelace (information measure of correlation)



a)

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \end{bmatrix}$$

b)

$$\mathbf{P}_{0,1} = \begin{bmatrix} 4 & 2 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

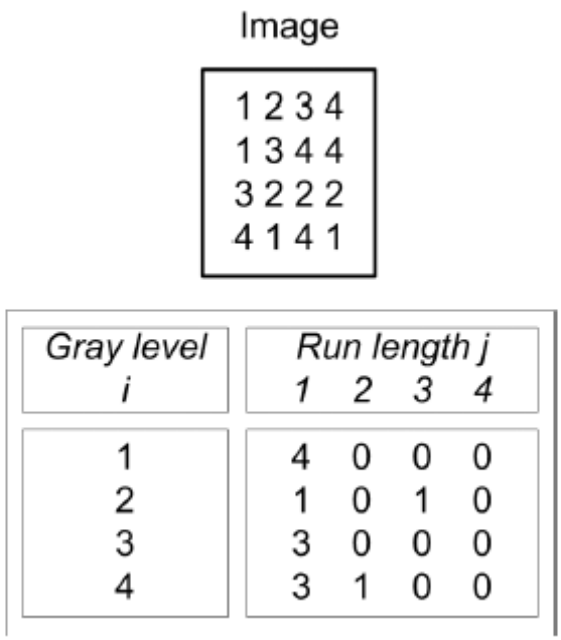
c)

Obr. 3.2 a) směry pro výpočet matice co-occurrence, b) příklad obrazu se čtyřmi úrovněmi šedi, c) výsledná co-occurence matice pro směr d = [0,1], (zdroj: [8]).

3.4 Metoda run-length

Metoda „grey level run length“ patří do metod statistiky vyššího řádu. Výsledkem metody je dvourozměrná matice $p_\theta(i,j)$, kde θ je daný směr (úhel) , i je počet úrovní šedi a j značí délku běhu (počet opakujících se úrovní v daném směru). Rozměry matice závisí jednak na počtu stupňů šedi (obvykle 16 až 256), a také na maximální délce výskytu jednotlivých úrovní (run length). Matice se obvykle počítají pro čtyři úhly (0°, 45°, 90°a 135°). Výsledná matice je potom vypočítána jako průměr těchto čtyř matic. Z této matice je vypočteno následujících 9 příznaků, převzatých z [3].

- zdůraznění krátkých řad (short runs emphasize)
- zdůraznění dlouhých řad (long runs emphasize)
- nerovnoměrnost stupňů šedi (rey level non uniformity)
- nerovnoměrnost délky řad (run length non-uniformity)
- procentuální vyjádření řad (run percentage)
- zdůraznění nízkých hodnot odstínu šedi (low grey level runs emphasize)
- zdůraznění vysokých hodnot odstínu šedi (high grey level runs emphasize)
- entropie krátkých řad (short run length entropy)
- zdůraznění ultra krátkých řad (ultra short runs emphasize)

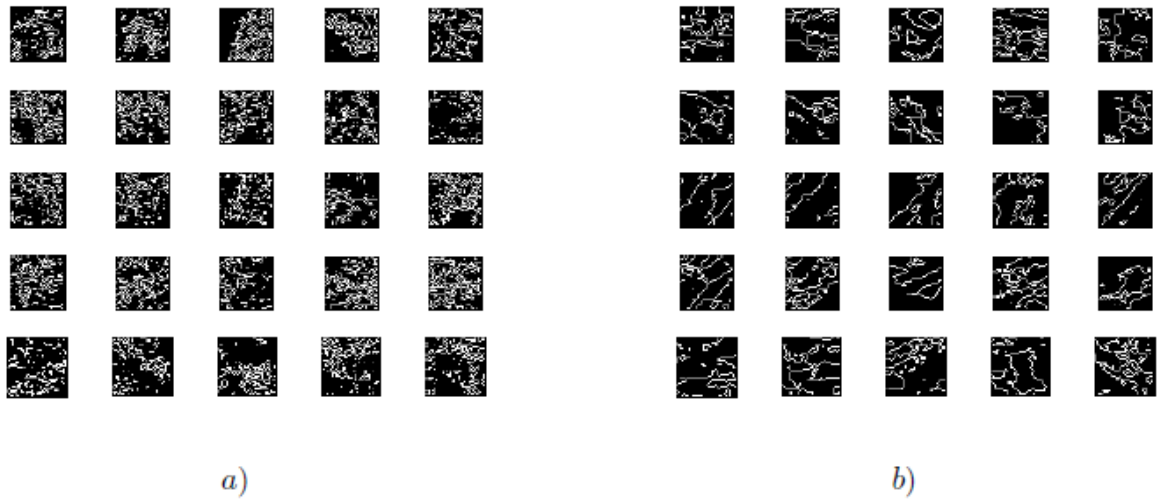


Obr. 3.3 Ukázka metody grey level run length, (zdroj: [3])

3.5 Strukturální metoda

Zmíněná metoda popisuje texturu pomocí primitiv. Primitiva jsou tvořena strukturami. Jako strukturu označujeme skupiny pixelů se stejnou úrovní šedi.Popsaná metoda je detailněji rozebrána v [3].

V našem případě se jedná o hranovou reprezentaci s prahem k = 0,9. Na následujícím obrázku 3.4. můžeme vidět takto zpracované vzorky. V případě a) chybí vrstva nervových vláken a struktura odpovídá pouze šumu v obraze. Zato na obrázku b) vidíme strukturu, která odpovídá zmíněným nervovým vláknům.



a) *b)*

Obr. 3.4 Vzorky po hranové detekci a) vzorky nemocné tkáně, chybí vrstva nervových vláken, b) vzorky tkáně zdravé, (zdroj:[3])

Vzory použité v této práci obsahují z této metody texturní analýzy následující 4 příznaky [3]:

- relativní počet hraničních pixelů (relative number of level crossing pixel)
- počet nespojitých struktur (number of broken structures)
- střední délka nespojitých částí (mean length of broken components)
- medián délky nespojitých částí (median from the lengths of broken components)

3.6 Základní statistika

Tato metoda patří do statistik 1. řádu. Pro tuto statistiku potřebujeme znát rozložení hustoty pravděpodobnosti. Ze vstupního šedotónového obrazu se spočítá normovaný histogram četnosti výskytu jednotlivých úrovní šedi. Následně vypočteme potřebné příznaky, jejichž podrobný popis je uveden v [3].

Použité vzory obsahují těchto pět texturních příznaků:

- centrální moment 1. řádu
- centrální moment 2. řádu
- koeficient šikmosti
- koeficient strmosti
- entropie

4 Metody pro klasifikaci texturních příznaků

4.1 Klasifikace

Hlavním cílem klasifikace dat je jejich zařazení do předem stanovených tříd. Jako klasifikátor označujeme algoritmus, který je schopen úspěšně rozdělovat vstupní data do předem definovaných tříd. Celý proces třídění je možné rozdělit do tří základních operaci. Nejprve je nutné získaná data předzpracovat, potom následuje tvorba příznaků a nakonec vlastní klasifikace. Hranice mezi tvorbou příznaků a klasifikací je velmi tenká. V ideálním případě, vytvořením vhodných příznaků, může být klasifikace velmi jednoduchá. Naopak dokonalý klasifikátor nepotřebuje sofistikovaný algoritmus na vytvoření příznaků.

Vstupem systému na rozpoznávání vzorů je obvykle nějaký snímač (kamera, mikrofon, apod.). Celý systém je tedy limitován charakteristikami a mezními hodnotami použitých snímačů.

Hlavním úkolem tvorby příznaků je charakterizovat objekt tak, aby změřené hodnoty byly pro danou třídu co nejpodobnější a co nejrozdílnější pro kategorie ostatní.

Úlohou klasifikace je dle vstupního vektoru příznaků zařadit objekt do správné kategorie (třídy). Složitost klasifikátoru závisí na variabilitě hodnot příznaků pro objekty stejné třídy a rozdílu hodnot příznaků pro odlišné třídy.

Výsledkem klasifikace je zařazení objektu do jedné z požadovaných tříd. Výkonnost klasifikátoru je možné určit chybou klasifikace. Určí se jako procento vzorů, které jsou zařazeny do špatné kategorie. Z toho tedy vyplývá, že správně nastavený klasifikátor má minimální chybu klasifikace. V ideálním případě je chyba rovna nule. Dle výsledné chyby klasifikace je možné porovnávat schopnosti jednotlivých klasifikátorů [1].

4.2 Metody učení

V širším slova smyslu můžeme říct, že všechny metody, které přiřazují informace nějaké trénovací skupině, používají učení. Ve většině případů třídění neznáme výsledky předem, a proto návrh učícího algoritmu zabírá mnoho času.

4.2.1 Učení s učitelem

Metoda učení s učitelem je technika strojového učení, která využívá trénovací data. Trénovací data obsahují párové hodnoty. Za vstupní hodnoty považujeme vektor příznaků, který popisuje potřebný objekt. Výstupní hodnoty popisují příslušnost k dané třídě. Výstupní data mohou být i spojitá (regrese). Hlavním úkolem klasifikace je vypočítat hodnotu výstupu (výstupní třídu) pro vstupní data, která přijdou po skupině dat trénovacích.

4.2.2 Učení bez učitele

Metoda je známá i pod názvem clustering (shlukování). V této metodě jsou známy pouze výsledné třídy. Systém sám roztřídí vstupní vzory do určitých shluků. To znamená, že jiný algoritmus může zařadit vzor do jiné třídy. Uživatel může definovat potřebný počet shluků pro klasifikaci. Tyto klasifikátory bývají založeny na pravděpodobnosti a statistice.

4.2.3 Zpětnovazební učení

Metoda též nazývána jako učení s posilováním. Odlišnost od učení s učitelem spočívá v tom, že zde klasifikátor nezná třídu vstupního vzoru. Musí sám rozhodnout o třídě vzoru. Metodu lze označit jako učení s kritikem, který rozhoduje zda je vzor klasifikován do správné třídy či nikoliv. Ovšem neříká proč.

4.3 Křížová validace

Metoda křížové validace („cross-validation“) je technika pro otestování výsledků klasifikace na nezávislém souboru dat. Používá se pro posouzení, jak přesně se bude daný model chovat v praxi. Prvním krokem křížové validace je rozdělení vstupních dat na dvě podskupiny. Na první podskupině (trénovací data) provedeme natrénování klasifikátoru. Druhou skupinu (testovací data) použijeme pro otestování již natrénovaného klasifikátoru. Pro snížení variability tuto metodu několikrát zopakujeme a výsledky zprůměrujeme [4].

4.3.1 Repeated random subsampling cross validation

Tato metoda náhodně rozdělí data na trénovací a testovací množinu. Pro každé takové dělení se spočítá chyba klasifikace. Celkovou chybu získáme zprůměrováním dílčích výsledků. Výhodou je, že metoda není nijak omezena počtem iterací oproti metodě “k-fold cross validation” (kap. 4.3.2). Nevýhodou této metody je, že skupiny jsou rozděleny náhodně. Z toho vyplývá, že určitý vzor může být stále vybírán do trénovací množiny a nemusí být vybrán pro testování. Totéž může nastat i opačně [4].

4.3.2 K fold cross validation

V metodě “k fold cross validation” jsou vstupní data náhodně rozdělena na *k* podskupin. Každá podskupina je použita právě jednou pro testování modelu. Na trénování je použito zbývajících *k-1* podskupin. Celý proces se zopakuje *k*-krát, aby byla na otestování použita všechna data. Následné výsledky se zprůměrují, a tím získáme výslednou chybu klasifikace. Výhodou oproti “repeated random subsampling” je, že pro testování použijeme každý vzor právě jednou. Při třídění do dvou skupin je vhodné rozdělit vstupní data tak, aby každá podskupina obsahovala přibližně stejný poměr vzorů z každé třídy [4].

4.3.3 Leave one out cross validation

Jak už název napovídá, tato metoda používá jeden vzor pro testování. Zbývající data jsou použita pro natrénování systému. Tato metoda se opakuje tolikrát, kolik je vstupních vzorů, aby pro otestování byl použit každý vzor právě jednou. Je to vlastně zvláštní případ metody "k fold cross validation", pro *k* rovno počtu vzorů. Tato metoda je poměrně zdlouhavá, protože trénovací proces je mnohokrát opakován. Výslednou chybu spočítáme z počtu špatně zařazených vzorů [4].

4.4 Popis použitých klasifikačních algoritmů

Zde popsané algoritmy, které jsou využity pro vlastní třídění vstupních obrazů sítnice, byly převzaty z literatury [1]. Vlastní zdrojové kódy jsou v přílohách této práce. Algoritmus K-means byl naprogramován dle [5].

4.4.1 Algoritmus Ho-Kashyap

Tento algoritmus najde dělící nadrovinu, jestliže vzory jsou lineárně oddělitelné. Základem je minimalizovat $|| Ya - b ||^2$. Pro lineárně oddělitelné vzory existují vektory *a* a *b*, takové, že

$$Y\vec{a} = \vec{b} > 0 . \qquad (4.1)$$

Vektor *b* nazýváme *margin* a všechny jeho složky musí být kladné. *Margin* označuje vzdálenost mezi jednotlivými shluky dat. Vektor *a* je dělící vektor. Do proměnné *Y* jsou uloženy vstupní trénovaní vzory. Struktura algoritmu Ho-Kashyap je následující [1]:

1. **begin initialize** a,b,η(.) < 1, threshold b_{min}, k_{max}
2. **do** k ← (k + 1) mod n
3. e ← Ya – b
4. e+ ← 1/2 (e + Abs(e))
5. b ← b + 2 η (k) e+
6. a ← Y^{+}b
7. **if** Abs(e) ≤ b_{min} **then return** a, b and **exit**
8. **until** k = k_{max}
9. print "NO SOLUTION FOUND"
10. **end**

Popis použitého algoritmu:

1. Nejprve se nastaví vstupní proměnné
   - k_{max} … maximální počet opakování cyklu
   - b_{min} … kritérium konvergence (práh)
   - η ….. konvergenční rate
   - *b* …… margin – prvotní nastavení na hodnotu 1
   - *a* …… dělící vektor
2. Inkrementace počtu iterací.

3. Výpočet chybového vektoru

$$e(k) = Ya(k) - b(k). \tag{4.2}$$

4. Výpočet kladné časti chybového vektoru

$$e^{+}(k) = \frac{1}{2}\left(e(k) + |e(k)| \right). \tag{4.3}$$

5. Výpočet vektoru *b*

$$b = b + 2\eta(k)e^{+}. \tag{4.4}$$

6. Výpočet dělícího vektoru *a* jako součin pseudoinverzní matice *Y* s vektorem *b*

$$a = Y^{+}b. \tag{4.5}$$

7. Omezovací kritérium, jestliže Abs(e) $\leq b_{min}$, tak vypíše proměnné a skončí.
8. Další omezení je počtem iterací, v našem případě k_{max} = 2000.
9. Jestliže nenalezne řešení, tak to oznámí.
10. Konec.

Výsledné cíle testovacích vzorů získáme jako součin dělícího vektoru *a* s testovacími vzory. Jako třídu 0 označíme výsledky menší než nula. Třída 1 odpovídá výsledkům větším než nula. Výpis zdrojového kódu pro Matlab je v příloze 1.

4.4.2 Algoritmus NDDF

Diskriminační funkce pro normální rozložení pravděpodobnosti (Normal Density Discriminant Function). Tento algoritmus vychází z Bayessovské teorie [7].V našem případě použijeme dvě diskriminační funkce $g_i(x)$, i = 1,2.
Klasifikátor přiřadí vektor příznaků *x* do třídy ω_i, jestliže

$$g_i(x) > g_j(x) \qquad pro\ j \neq i. \tag{4.6}$$

Když označíme $g_i(x) = P(\omega_i \mid x)$, potom maximální diskriminační funkce odpovídá maximální aposteriorní pravděpodobnosti. Pro výpočet použijeme Bayessův vzorec [1].

$$g_i(x) = P(\omega_i \mid x) = \frac{p(x \mid \omega_i)P(\omega_i)}{\sum_{j=1}^{c} p(x \mid \omega_j)P(\omega_j)}, \tag{4.7}$$

kde $p(x|\omega_i)$ značí podmíněnou pravděpodobnost pro vektor příznaků *x*, $P(\omega_i)$ je apriorní pravděpodobnost pro třídu ω_i a ve jmenovateli je celková pravděpodobnost. Jelikož nás zajímá, kdy je funkce maximální, můžeme jmenovatele zanedbat.

$$g_i(x) = p(x \mid \omega_i)P(\omega_i). \tag{4.8}$$

Volba diskriminační funkce není jednoznačná. Můžeme ji násobit libovolnou konstantou a pořád získáme stejný výsledek. Jestliže nahradíme $g_i(x)$ funkcí $f(g_i(x))$, kde f (.) je monotónně rostoucí funkce, výsledek klasifikace se nezmění.

$$g_i(x) = \ln p(x \mid \omega_i) + \ln P(\omega_i). \tag{4.9}$$

Toto vyjádření můžeme spočítat, jestliže rozložení hustoty pravděpodobnosti $p(x|\omega_i)$ odpovídá normálnímu rozložení se střední hodnotou μ a rozptylem Σ.

$$p(x \mid \omega_i) \sim No(\mu_i, \Sigma_i). \tag{4.10}$$

V tomto případě vyjádříme rovnici (3.9) jako [1]

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1}(x-\mu_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i). \tag{4.11}$$

Vlastní klasifikační algoritmus spočívá pouze v implementaci vzorce (3.11) do vývojové prostředí Maltab. (příloha 2)

4.4.3 Algoritmus Nearest Neighbor

Patří mezi neparametrické metody klasifikace. U těchto metod neznáme pravděpodobnost zařazení do jednotlivých tříd.

Známe trénovací množinu $\{(x_i, \omega_i)\}_{i = 1...K}$, kde x_i je vzorek, kterému je přiřazena třída ω_i a K je velikost trénovací množiny [12]. Pro neznámý prvek x hledáme x'_k takové, že

$$\left\| x'_K - x \right\| = \min \left\| x'_K - x \right\|_{i=1...K}. \tag{4.12}$$

Prvek x pak zařadíme do stejné třídy, do jaké náleží x'_k.

Nejčastěji je klasifikace podle jednoho nejbližšího souseda (1-NN), ale existují i klasifikace pro obecně k sousedů (k-NN). Základem algoritmu je spočítat vzdálenost každého testovacího vzoru od všech trénovacích.

Jelikož známe trénovací množinu $\{(x_i, \omega_i)\}_{i = 1 \ldots K}$, kde x_i je trénovací vzor, který je přiřazen do třídy ω_i. K je velikost trénovací množiny. Pro testovací vzor x hledáme x'_k takové, že

$$\left\| x'_k - x \right\| = \min \left\| x'_i - x \right\|_{i=1..k}. \tag{4.13}$$

Testovací vzor x zařadíme do takové třídy, do které přísluší vzor x'_k.

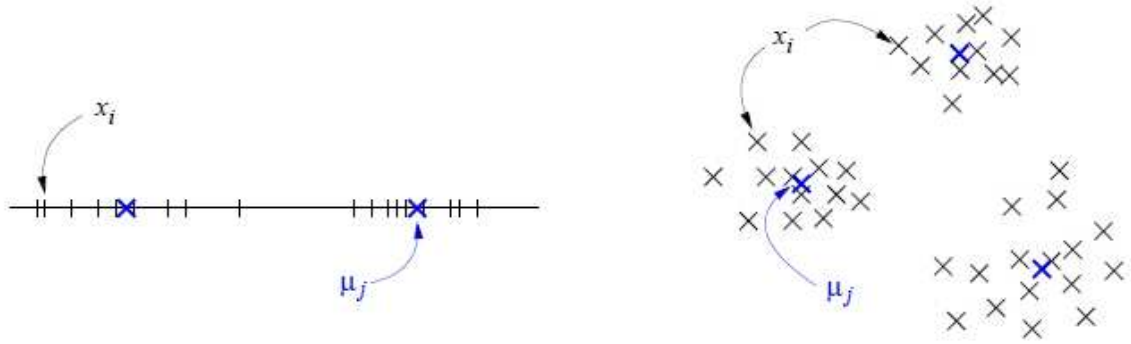
V našem případě používáme k nejbližších sousedů. To se označuje jako k-NN (Nearest Neighbors). Je to rozšíření, kdy vybereme k nejbližších trénovacích vzorů a testovací vzor zařadíme do třídy, která odpovídá většině tříd mezi k sousedy.

Ve vlastní funkci je nutné nastavit počet nejbližších sousedů *Knn*. Výpis zdrojového kódu v programu Matlab je v příloze 3.

4.4.4 Algoritmus K-means

Tato metoda spadá do kategorie učení bez učitele. Jedná se o tzv. shlukování (clustering) . Každý shluk je charakterizován svým středem μ_j [5].

Algoritmus tedy hledá pro množinu vstupních vektorů x_i ($i = 1 \ldots n$) vektory μ_j ($j = 1...k$) , kde k je počet tříd, takové, aby byla minimalizována střední kvadratická odchylka.



Obr. 4.1 Příklad nalezení středů pro jednorozměrný a dvourozměrný příklad, (zdroj: [5])

Metoda třídění:

1. Náhodně zvolíme středy jednotlivých shluků μ_j.
2. Každý vektor příznaků x_i (i = 1…k), zařadíme do nejbližšího shluku dle minima euklidovské vzdálenosti. Z toho plyne, že vzor x_i zařadíme do třídy
$$y_i = \arg\min_{j=1...k} \left\| x_i - \mu_j \right\| \ . \qquad (3.14)$$
3. Po zařazení všech vzorů do tříd se vypočítají nové hodnoty středů jako střední hodnoty dat x_i které byly klasifikovány do třídy určené vektorem μ_j .
$$\mu_j = \frac{1}{n_j} \sum_{i:y_i=j} x_i \, . \qquad (3.15)$$
kde n_j je počet vzorů x_i klasifikovaných v prvním kroku do třídy určené příslušným vektorem μ_j.
4. Kroky 2 a 3 opakujeme do té doby, dokud je alespoň jeden vzor x_i zařazen do jiné třídy než do jaké byl klasifikován v předcházejícím kroku.

4.5 Vzory použité k testování implementovaných algoritmů

Z každé použité metody texturní analýzy (kapitola 3) máme k dispozici výsledný vektor příznaků různé délky. Pro vlastní třídění bylo použito šest souborů dat z této analýzy. Každý soubor obsahuje tři matice. Tyto matice jsou nazvány *A_ne*, což odpovídá tkáni nemocných pacientů, *A_zd*, který odpovídá tkáni pacientů zdravých a *A_zn* odpovídající zdravé tkáni od nemocných pacientů.

V následující tabulce 4.1. je zobrazeno, jakou metodou byly získány jednotlivé soubory vzorů. Pro pořádek jsou zde uvedeny rozměry jednotlivých matic. První číslo udává počet příznaků daného vzoru a druhé udává počet vzorů (vektorů) pro danou skupinu.

Název souboru	**Použitá metoda**	**NE**	**ZD**	**ZN**
vzory1.mat	1. Markovovy náhodná pole	6 x 142	6 x 283	6 x 141
vzory2.mat	2. Fraktální koeficient	5 x 261	5 x 261	5 x 261
vzory3.mat	3. Metoda co-occurrence	10 x 261	10 x 261	10 x 261
vzory4.mat	4. Metoda run-length	9 x 261	9 x 261	9 x 261
vzory5.mat	5. Strukturální metoda	4 x 261	4 x 270	4 x 270
vzory6.mat	6. Základní statistika	5 x 261	5 x 270	5 x 270

Tab. 4.1 Vstupní data

5 Popis vytvořených experimentálních programů

V následující části jsou podrobně popsány vytvořené programy pro třídění zadaných dat. Nejprve je použit algoritmus Ho-Kashyap. Potom NDDF a metoda nejbližších sousedů k-NN. Jako poslední je vytvořen algoritmus K-means, který patří mezi klasifikátory bez učitele. Všechny algoritmy jsou použity pro třídění dat do dvou možných skupin. Abychom obsáhly všechny kombinace třídění, máme pro každou klasifikaci tři možné výsledky. Konkrétně pro kombinace tříd nemocní × zdraví, nemocní × zdraví nemocní, zdraví nemocní × zdraví.

Cílem této kapitoly je podrobně popsat vytvořené programy a vytvořit tak jednoduchý návod na jejich obsluhu.

5.1 Algoritmy učení s učitelem

Popsané časti programů jsou pro všechny tři zmíněné algoritmy totožné. Tudíž je zde popsán pouze jeden program.

Popis začátku programu

```
load vzory1;                % načte potřebné vzory

format long;                % delší formát čísel

L_ne = length(A_ne);        % počet nemocných
L_zd = length(A_zd);        % počet zdravých
L_zn = length(A_zn);        % počet zdravých nemocných

    % počet trénovacich vzorů - nutno nastavit ručně
zdravych     = 200;
nemocnych  = 100;
zdravnemoc = 100;
```

Nejprve načteme vzory příkazem *load*. Používané vzory se nazývají *vzory1* až *vzory6*. Výběr požadovaných vzorů je třeba nastavit manuálně. Na následujícím řádku se pouze změní formát čísel z původních 5 na 15 platných číslic. Na další tři řádky se zadá počet nemocných, zdravých a zdravých_nemocných.

5.1.1 Výběr trénovacích vzorů metodou repeated random subsampling cross validation

Pro výběr vzorů touto metodou je nutné nastavit manuálně počty trénovacích vzorů pro jednotlivé skupiny.

Popis výběru trénovacích a testovacích vzorů

```
        % výběr trénovacích a testovacích vzorů - NEMOCNÝCH
train_ne        = nemocnych;
celkem_ne       = L_ne;
test_ne         = celkem_ne - train_ne;

[B,C]  = delicka( train_ne,test_ne,celkem_ne );      %  funkce pro výběr pořadí
                                                     %  trénovacích a testovacích vzorů

        % Cyklus na výběr trénovacích  vzorů B - pořadí vzoru
for i  = 1:train_ne
   A_ne_train(:,i)  =  A_ne(:,B(i));                 %  trénovací vzory nemocných
end
        % Cyklus na výběr testovacích  vzorů C - pořadí vzoru
for i = 1:test_ne
   A_ne_test(:,i)  =  A_ne(:,C(i));                  %  testovací vzory nemocných
end
```

Další částí programu je výběr trénovacích a testovacích vzorů. Zde je ukázka pouze pro nemocné. Pro zdravé a zdravé_nemocné je to stejné.

Do proměnné *train_ne* se uloží požadovaný počet nemocných pro trénování, do *celkem_ne* se uloží celkový počet nemocných. Proměnná *test_ne* odpovídá počtu trénovacích vzorů. Na následujícím řádku je volána funkce *delicka*, která slouží pro výběr pořadí trénovacích a testovacích vzorů (proměnné *B* a *C*).

Následují dva cykly *for***.** První uloží do proměnné *A_ne_train* trénovací vzory pro nemocné a ve druhém *for* cyklu dojde k uložení testovacích vzorů nemocných do proměnné *A_ne_test*.

Popis funkce delicka

```
function [B,C] = delicka( train,test,celkem )

        % funkce na rozdělení pořadových čísel testovacích a trénovacích vzorů
        % B pořadí trénovacích vzorů
        % C pořadí testovacích vzorů

pocet = train + test;   % kdyby nebyla vybírána pořadová čísla všech vzorů pro zpracování

B1 = randomm( pocet, celkem );          %  vybere požadovaný počet pořadových čísel
                                        %  vzorů z celkového počtu
C1 = randomm( test , pocet  );          %  vybere pořadová čísla testovacích vzorů z
                                        %  požadovaného počtu
C   =  B1(C1);              % vybere pořadová čísla testovacích vzorů z celkového počtu
B2 =  B1;
B2(C1) =  0;                % vynulování již použitých pořadových čísel (testovacích)
B2      =  sort(B2);        % poskládání
B   =  B2(test+1:pocet);    % výběr pořadí trénovacích vzorů z celkového počtu
```

Vstupem funkce je počet vzorů pro trénování *train*, pro testování *test* a proměnná *celkem* označuje celkový počet vzorů. (není nutné používat všechny vzory) Na následujícím řádku je do *počet* uložen počet vzorů pro testování a trénování. Pro náhodný výběr pořadových čísel vzorů je vytvořena funkce *randomm*.

Do proměnné *B1* se náhodně vygenerují pořadová čísla všech potřebných vzorů. *C1* odpovídá pouze testovacím vzorům z *pocet*. Do *C* se vybere pořadí testovacích vzorů z *celkem*. Následné dojde k jejich vynulování a po poskládání dojde k výběru pořadových čísel trénovacích vzorů.

Výstupem funkce je proměnná *B*, která označuje pořadová čísla trénovacích vzorů, zatímco proměnná *C* odpovídá pořadovým číslům vzorů pro testování.

Popis funkce randomm

```
function [A]  =  randomm( pocet, celkem )

        %  funkce na náhodný výběr několika čísel /pocet/
        %  z určitého množství /celkem/

A = unique(sort(ceil(rand(1,pocet)*celkem)));      % seřazená unikátní řada čísel
                                                   % pro výběr pořadí vzoru
delka = pocet-(length(A));                         % kolik čísel chybí do 'pocet'

while delka > 0                          % dokud není nejsou vybrána všechna potřebná čísla
 A1     = unique(sort(ceil(rand(1,delka)*celkem)));
 A      = unique(sort([A A1]));          % přidá a seřadí další čísla
 delka  = pocet-(length(A));             % kolik čísel chybí do 'pocet'
end
```

Vstupem funkce jsou proměnné *pocet*, která označuje počet náhodných čísel z požadovaného rozsahu *celkem*. Na prvním řádku se pomocí matlabovské funkce *rand* vygeneruje požadovaný počet čísel. Do proměnné *A* se uloží pouze unikátní, poskládaná a zaokrouhlená čísla. Proměnná *delka* označuje počet čísel zbývajících do *pocet*. Následující cyklus *while* doplňuje zbývající náhodná čísla, dokud *delka* > 0.

Výstupem je seřazená řada náhodných čísel od 0 do *celkem*. Jejich počet odpovídá vstupní proměnné *pocet*.

Výpočet chyby funkcí Ho_Kashyap pro nemocné a zdravé

```
        % vzory NE a ZD
train_patterns  = [ A_ne_train  A_zd_train];    % trénovací vzory NE a ZD
test_patterns   = [ A_ne_test   A_zd_test  ];   % testovací vzory  NE a ZD

        % třídy pro NE a ZD 'Nemocní - 0' a 'Zdraví - 1'
train_targets   = [ zeros(1,train_ne) ones(1,train_zd) ];
test_targets    =  [ zeros(1,test_ne)   ones(1,test_zd)  ];

[ test_targetH_NE_ZD ] = Ho_Kashyap(train_patterns, train_targets, test_patterns, []);

chyba_NE_ZD = 100*sum(abs(test_targetH_NE_ZD - test_targets))/length(test_patterns);
```

Nejdůležitější částí programu je vlastní klasifikace vzorů pomocí vybraného algoritmu. Zde je ukázka pro funkci Ho_Kashyap.

Nejprve je nutné vytvořit trénovaní a testovací vzory. Ty se vytvoří sloučením vzorů nemocných a zdravých. Dále je nutné vytvořit jim odpovídající třídy. Zde je zvoleno pro nemocné třída 0 a zdravé třída 1. Vlastní algoritmus spočítá k testovacím cílům odpovídající třídy. Pro ověření schopnosti algoritmu je zde vypočtena chyba klasifikace jako poměr nesprávně klasifikovaných vzorů ku skutečné třídě vzoru.

Výpis skriptu pro opakování daného algoritmu - *opakuj*

```
% script na opakování daného algoritmu
% nastaví se 'iopak' jako počet opakování daného algoritmu
% do proměnné CH_**** se uloží chyba pro jednotlivá opakování
% následně se vypočítá průměrná chyba a směrodatná odchylka z daného počtu
% opakování
clc; clear all; close all;

format long;          % delší formát čísel

iopak = 100;          % počet opakování algoritmu

for i=1:iopak
   [CH_NE_ZD(i) CH_ZDNE_ZD(i) CH_NE_ZDNE(i)] = hokasap2;      % výpočet chyb
   i                                                          % vypíše aktuální počet opakování
end

prumer_NE_ZD = mean (CH_NE_ZD,2);
odchyl_NE_ZD = std    (CH_NE_ZD)   ;

prumer_ZDNE_ZD = mean (CH_ZDNE_ZD,2);
odchyl_ZDNE_ZD = std    (CH_ZDNE_ZD)   ;

prumer_NE_ZDNE = mean (CH_NE_ZDNE,2);
odchyl_NE_ZDNE = std    (CH_NE_ZDNE)   ;

konecne = 'skoncil'              % vypíše daný text na Command Window Matlabu, aby
                                 % uživateli oznámil konec daného programu
```

Pro možnost několikanásobného opakování algoritmu je vytvořen skript *opaku***j**. Zde stačí nastavit pouze počet opakování *iopak* a tento program po proběhnutí zadaného počtu opakování vypíše průměrnou chybu a směrodatnou odchylku pro nemocní x zdraví, zdraví_nemocní x zdraví a nemocní x zdraví_nemocní.

5.1.2 Výběr trénovacích vzorů metodou leave one out cross validation

Touto metodou dojde k otestování výkonnosti algoritmu pro všechny vzory. Pro natrénování se použije *k-1* vzorů, kde *k* je celkový počet vzorů. Pro otestování algoritmu se použije vždy pouze jediný vzor. To znamená, že dojde ke *k* opakování.

Základem algoritmu je cyklus *for*. Ten se opakuje dle počtu vzorů. Jako testovací vzor použijeme první z nich. Na dalším řádku je proměnná *V*, která obsahuje čísla od 1 do celkového počtu vzorů. V následujícím řádku dojde k posunutí této proměnné o *i-1* pozic doleva. Tím se zajistí, že jako testovací vzor požijeme vždy ten první, a pro natrénování se použijí vzory od druhého do posledního.

Následně může dojít k natrénování příslušného klasifikátoru a jeho vlastní klasifikaci. Přesnost jedné klasifikace může být pouze 100% a nebo 0%. Proto vypočítáme průměrnou chybu a směrodatnou odchylku ze všech klasifikací.

Zde je ukázka části programu na výpočet chyby klasifikace. Konkrétně funkcí Ho_Kashyap pro nemocné a zdravé.

Výpočet chyby funkcí Ho_Kashyap pro nemocné a zdravé

```
        % Výpočet chyby funkcí Ho_Kashyap pro NE a ZD
patterns = [ A_ne A_zd ];                    % vzory pro NE a ZD
targets  =  [ zeros(1,L_ne) ones(1,L_zd) ];  % cíle pro 'NE - 0' a 'ZD - 1'
L = length(patterns);                        % celková délka vzorů

for i=1:L
   test_pattern  = patterns(:,i);            % výběr testovacího vzoru
   test_target(i)= targets  (:,i);           % výběr testovacího cíle
   V    = (1:L);                             % ukazatel vzorů
   VV = circshift(V,[0,-(i-1)]);             % posunutí ukazatele vzorů o i-1 doleva
   train_patterns = patterns(:,VV(2:end));   % zbývající ~ trénovací vzory
   train_targets   = targets (:,VV(2:end));  % zbývající ~ trénovací cíle
   [test_targetH_NE_ZD(i) ] = Ho_Kashyap(train_patterns, train_targets, test_pattern, []);
   chyba_NE_ZD(i) = 100*sum(abs(test_targetH_NE_ZD(i) - test_target(i)));
                                             % výpočet jednotlivé chyby
   i                                         % aktuální pořadí výpočtu
end

prumer_NE_ZD = mean(chyba_NE_ZD,2);          % průměrná chyba pro NE a ZD
odchyl_NE_ZD = std (chyba_NE_ZD)  ;          % směrodatná odchylka pro NE a ZD
```

5.1.3 Výběr trénovacích vzorů metodou K fold cross validation

Základem této metody je rozdělit jednotlivé skupiny (nemocní, zdraví a zdraví_nemocní) na *k* podskupin. To je možné udělat následovně. Zde je popsán postup pouze pro skupinu nemocných, protože pro zbývající dvě skupiny je to stejné.

Před vlastním dělením musíme zadat *K*. Nejprve zjistíme velikost vstupní matice, kde *r_ne* odpovídá počtu příznaků a *c_ne* je počet vzorů. Následně spočítáme počet vzorů, které připadnou do každé skupiny *K*. Protože požadované submatice musí mít stejné rozměry, je nutno před vlastním dělením doplnit vstupní matici nulami.

Základem dělení je cyklus *for*, který postupně rozepisuje do příslušné submatice vstupní matici tak, že první vzor zapíše do první submatice, druhý do druhé a tak dále. Takže *K*-tý vzor zapíše do *K*-té submatice a vzor *K+1* do první submatice atd.

Dělení vzorů do submatic

```
        % Nemocni

[r_ne c_ne]= size(A_ne);                  %  r- počet řádků, c- počet sloupců
CK_ne      = ceil(c_ne/K);                %  počet vzorů ve skupině
CKK_ne     = CK_ne*K;                     %  maximální počet vzorů ve skupině
ch_ne      = CKK_ne - L_NE;               %  počet chybějících vzorů
A_ne       = [A_ne zeros(r_ne,ch_ne)];    %  doplní původní matici nulami, na nejbližší
                                          %  násobek K
```

```
for j = 1:CK_ne;                          %  rozdělí vzory na K submatic
   for k = 1:K
      NE(:,j,k) =A_ne(:,(((j-1)*K)+k));
   end
end
```

Dále jen nutné tyto submatice rozdělit na testovací a trénovací. Jako testovací použijeme vždy první submatici. Je třeba odstranit nulové vektory (pokud se vyskytují), které jsme přidali při třídění. K takto upraveným testovacím vzorům vytvoříme testovací třídy.

Trénovací vzory získáme ze zbývajících submatic tím, že je jednoduše poskládáme vedle sebe (příkazem *squeeze*). Takto získáme jedinou matici. Ještě je nutné vytvořit vektor trénovacích tříd. Poté dojde k posunu jednotlivých submatic o -1, což znamená, že druhá submatice se přesune na první místo a první na místo poslední.

Ukázka výběru testovacích a trénovacích vzorů pro nemocné

```
        % K-tá skupina NE
   test_patterns_NE    = NE(:,:,1);                  % výběr testovacích vzorů nemocných
   V0                  = find(test_patterns_NE(1,:) ~=0 );
   test_patterns_NE    = test_patterns_NE(:,V0);          % výběr pouze nenulových vzorů
   test_targets_NE     = zeros(1,size((test_patterns_NE),2));  % testovací cíle pro nemocné-'0'

   train_patterns_NE = NE(:,:,(2:end));              % výběr trénovacích vzorů nemocných
   train_patterns_NE = squeeze(train_patterns_NE(:,:));  % poskládání vzorů do jediné matice
   V0                = find(train_patterns_NE(1,:) ~=0 );
   train_patterns_NE = train_patterns_NE(:,V0);          % výběr pouze nenulových vzorů
   train_targets_NE  = zeros(1,length(train_patterns_NE)); % trénovací cíle pro nemocné-'0'

   NE = circshift(NE,[0,0,-1]);                  % posun jednotlivých submatic o -1 (= doleva)
```

Klasifikace vzorů a vypočet chyby pro K-té opakování

```
        %  NE_0 x ZD_1    Výpočet chyby pro K-té opakování
   train_patterns =  [train_patterns_NE  train_patterns_ZD];   % trénovací vzory
   train_targets  =  [train_targets_NE   train_targets_ZD   ];   % trénovací cíle
   test_patterns  =  [test_patterns_NE   test_patterns_ZD  ];   % testovací vzory
   test_targets   =  [test_targets_NE    test_targets_ZD    ];   % testovací cíle

   [test_targetH_NE_ZD ]    = Ho_Kashyap(train_patterns, train_targets, test_patterns, []);
   chyba_NE_ZD(h)           = 100*sum(abs(test_targetH_NE_ZD - test_targets)) / …
                                 length(test_patterns);
```

Výpočet průměrné chyby a směrodatné odchylky pro nemocné a zdravé

```
    prumer_NE_ZD  = mean(chyba_NE_ZD,2);
    odchyl_NE_ZD  = std (chyba_NE_ZD)  ;
```

5.2 Program k_means_2

Tento algoritmus patří do kategorie učení bez učitele. To znamená, že data nemusíme dělit na trénovaní a testovací množinu.

Inicializační část programu obsahuje načtení potřebných vzorů příkazem *load*. Program pracuje se dvěma možnými shluky *p11* a *p22*. Těm je nutné manuálně přiřadit vzory, které chceme třídit. Možné jsou libovolné kombinace těchto tří: *A_ne*, *A_zd* a *A_zn*. Následně je sloučíme, abychom získaly vzory pro třídění.

Poprvé je nutno zvolit první středy shluků. V našem případě náhodně vybereme jeden vzor z každé třídy, který budeme považovat za střed.

Začátek programu k_means

```
load vzory1;                % načte potřebné vzory

format long ;               % delší formát čísel

p11 = A_ne;                 % do p11 přiřadí první vzory
p22 = A_zd;                 % do p22 přiřadí druhé vzory

patterns = [ p11 p22 ];     % vzory pro třídění
r        = size(patterns,1); % počet příznaků
L        = length(patterns); % celkový počet vzorů
L_p1 = length(p11);         % počet vzorů p11
L_p2 = length(p22);         % počet vzorů p22

        % náhodný výběr středů shluků
r_p1  = ceil(rand*L_p1);                % náhodně vybrané pořadí prvního vzoru
r_p2  = L_p1+ceil(rand*L_p2);           % náhodně vybrané pořadí druhého vzoru
mi_p1 = patterns(:,r_p1);               % výběr středu prvního clusteru
mi_p2 = patterns(:,r_p2);               % výběr středu druhého clusteru
```

Nejdůležitější částí programu je výpočet euklidovské vzdálenosti od každého středu. V Matlabu je to možné realizovat následovně.

Výpočet euklidovské vzdálenosti

```
        % výpočet euklidovské vzdálenosti všech vzorů od prvního středu
  for i=1:L
      d_p1(i) = sqrt(sum((patterns(:,i) - mi_p1(:,1)).^2));
  end
        % výpočet euklidovské vzdálenosti všech vzorů od druhého středu
  for i=1:L
      d_p2(i) = sqrt(sum((patterns(:,i) - mi_p2(:,1)).^2));
  end
```

V následující části programu je dojde k zařazení vzorů do nových tříd, dle minimální vzdálenosti. Poté je nutno spočítat nové středy těchto vzorů. To je v Matlabu je možné realizovat následovně. Zde je ukázka pro jednu třídu vzorů, protože pro druhou je to stejné.

Klasifikace do příslušné třídy a výpočet nového středu

```
        % zařadí nové vzory do třídy 1
  patterns_p1  = zeros(r,L_p1);
        for i = 1:L_p1
                patterns_p1(:,i) = (patterns(:,p1(i)));
        end
         % spočítá nový střed vzorů třídy 1
         for m = 1:r
                mi_p1(m) = 1/L_p1*sum(patterns_p1(m,:));
        end
```

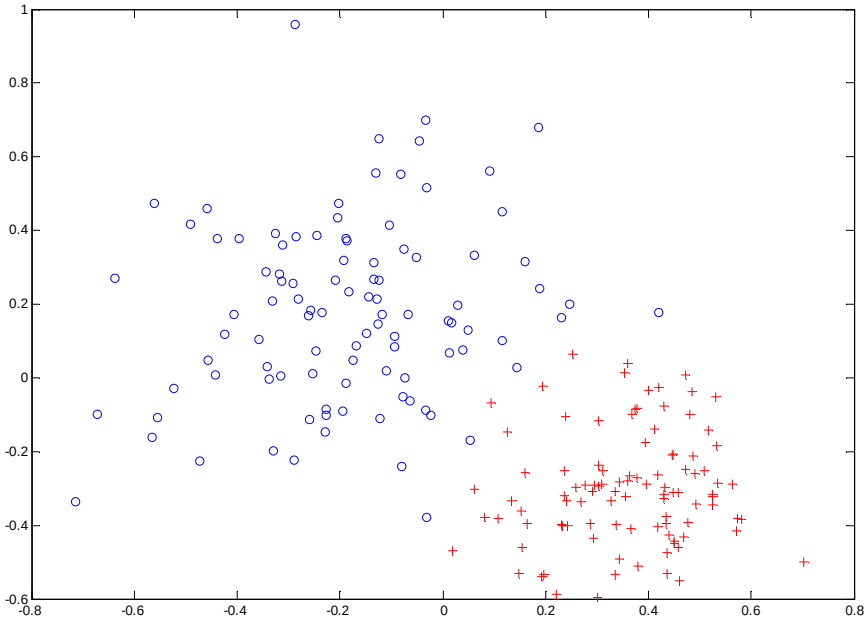
Poslední dvě popsané části opakujeme do té doby, dokud se mění počet vzorů v jednotlivých shlucích. V Matlabu realizováno *while* cyklem.

Posledním úkolem je zjistit chybu klasifikace. Spočítáme ji jako počet nesprávné klasifikovaných vzorů. Jediným problémem je určit, kterému shluku přiřadit kterou třídu. Jelikož při třídění na dvě skupiny by chyba neměla být větší než 50%, uvažujeme vždy tu menší chybu.

Protože tento algoritmus je možné aplikovat pro klasifikaci na libovolný počet tříd, byl program k_means_2 rozšířen na program k_means_3. Ten je schopen rozdělit vstupní data na tři třídy.

5.3 Zkouška na testovacích datech

Pro otestování funkce programů byla použita data ze souboru *test_data2.mat*, který obsahuje testovací data k literatuře [1]. Po následném rozdělení do daných tříd byly takto vzniklé skupiny pojmenovány *A_zd* a *A_ne*, a uloženy do souboru *test_data2_upravena.mat*, pro otestování schopnosti třídění jednotlivých třídících metod. Požitá data jsou dvourozměrná, kvůli snadnému zobrazení. Vstupní data jsou rozdělena do dvou tříd. V každé třídě je sto vzorů se dvěma příznaky. Na obrázku 5.1 znázorněno kolečky a křížky.

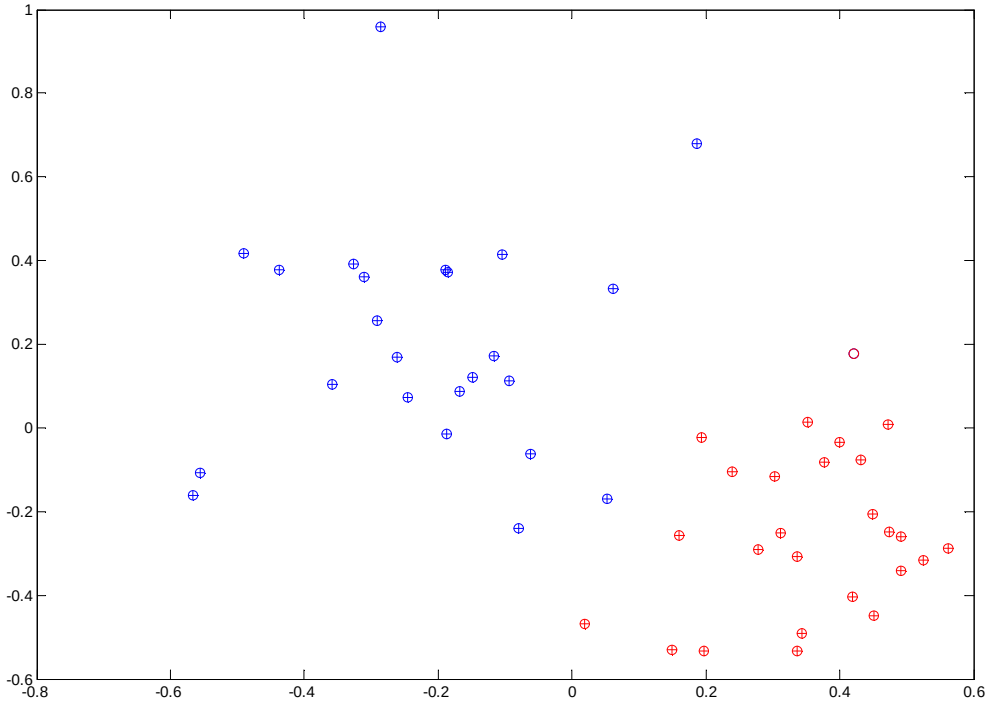


Obr. 5.1 Testovací data ze souboru test_data2.mat

5.3.1 Testování algoritmem Ho-Kashyap

Pro trénování zvolíme z každé třídy 75 vzorů a na testování jich zbývá 25. Pro přehlednost označíme třídy určené algoritmem Ho_Kashyap opačnými znaky než jak jsou označeny původně. To znamená, kolečka křížky a naopak. Tím bude na první pohled patrné, které vzory jsou špatně klasifikovány.

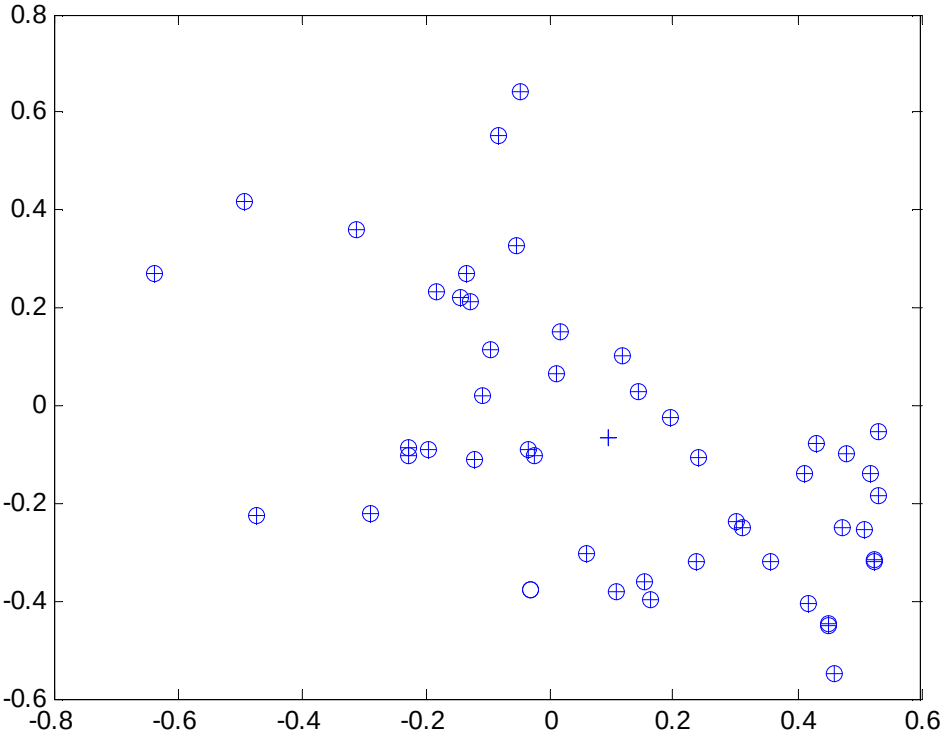
Jak je patrné z obrázku 5.2, tak tímto algoritmem byl nesprávně klasifikován jediný vzor (prázdné červené kolečko). Z toho můžeme určit chybu klasifikace. Jeden špatně zařazený vzor z padesáti nám určuje chybu klasifikace rovnou 2%.



Obr. 5.2 Klasifikace testovacích dat pomocí algoritmu Ho-Kashyap

5.3.2 Testování algoritmem NDDF

V tomto algoritmu opět náhodně vybereme z každé třídy 75 vzorů pro trénování a 25 na testování. Jak vidíme na obr.5.3, tento třídící algoritmus chybně klasifikoval dva vzory. První vidíme jako prázdné kolečko a druhý jako samotné plus. Proto je chyba jeho klasifikace rovna 4%.

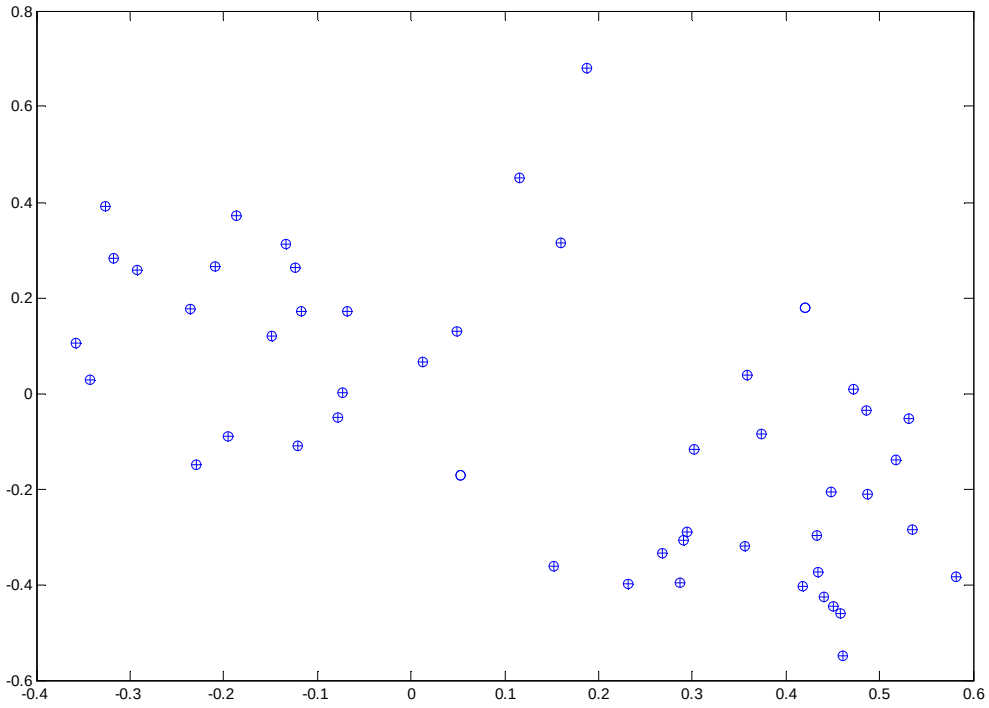


Obr.5.3 Klasifikace testovacích dat pomocí algoritmu NDDF

5.3.3 Testování algoritmem Nearest Neighbor

Následuje klasifikace metodou nejbližších sousedů Knn. Nastavíme počet nejbližších sousedů Knn = 5. Testovacích vzorů bude opět 25 z každé třídy.

Nesprávně určené jsou dva vzory, jak je patrné z obrázku 5.4. Z toho vyplývá, že klasifikační chyba je 4%.

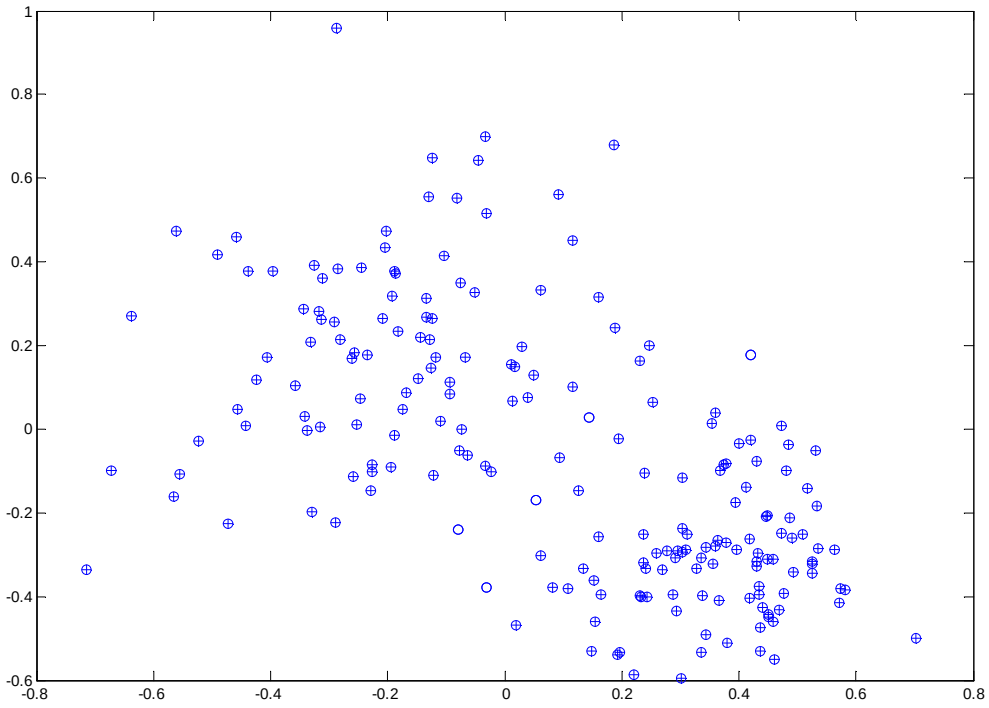


Obr.5.4 Klasifikace testovacích dat pomocí algoritmu Nearest Neighbor

5.3.4 Testování algoritmem K-means

Jako poslední otestujeme klasifikátor bez učitele k-means. Pro tento klasifikátor není třeba žádné nastavení, protože algoritmus neobsahuje trénovaní fázi.

Z výsledného roztřídění, obr.5.5, vidíme pět špatně zařazených vzorů, což při celkovém počtu 200 vzorů, odpovídá klasifikační chybě 2,5 %.



Obr.5.5 Klasifikace testovacích dat pomocí algoritmu k-means

6 Výsledky klasifikace retinálních dat

Hlavní částí práce je vlastní třídění dat reprezentujících texturu vrstvy nervových vláken na fundus fotografiích sítnice. Vstupní data jsou popsána v kap.2. Tyto vstupní vzory jsou postupně tříděny algoritmy Ho-Kashyap, NDDF, k-NN a k-means.

Algoritmus Ho-Kasyap je nastaven následovně.

- type of training type = 'Basic'
- max.iterations Max_iter = 2000
- convergence criterion b_min = 0.0002
- convergence rate eta = 0.005

Pro metodu nejbližších sousedů jsou zde uvedeny tři možná nastavení nejbližších sousedů, a sice Knn = 5,10 a 25.

Všechny metody jsou navíc testovány na tři typy validací [kap.4.3]. Konkrétně Leave one out cross validation, K-fold cross validation pro K = 25 a K = 100 a metodou repeated random subsampling, pro počet opakování 100. Pro poslední typ validace (repeated random subsampling) je před vlastními výsledky uvedena tabulka počtu trénovacích vzorů, protože tato metoda vyžaduje toto nastavení.

V následujících tabulkách jsou dosažené chyby klasifikací (v procentech) vypočtené různými metodami. Pro každou klasifikaci byly použity všechny vypočtené příznaky a roztřídění bylo provedeno pro všechny kombinace tříd vzorů (NE-ZD, NE-ZN, ZN-ZD)

6.1 Výsledky klasifikace příznaků získaných metodou Markovovy náhodná pole

označení	Nemocní	Zdraví	Zdraví_Nemocní
a)	100	200	100
b)	71	142	71
c)	42	83	41
vzorů celkem	142	283	141

Tab.6.1.1 Počty trénovacích vzorů pro rep_rand_subs

V prvním sloupci tabulky jsou různé metody validace

- L_1_out - Leave one out cross validation
- K_fold /K=25/ - K fold cross validation, pro K = 25
- K_fold /K=100/ - K fold cross validation, pro K = 100
- rep_rand_subs - repeated random subsampling

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	0.2353 ± 4.8507	3.1802 ± 17.5783	8.4906 ± 27.9071
K_fold /K=25/	0.2500 ± 1.2500	3.2970 ± 5.3567	8.2271 ± 4.4679
K_fold /K=100/	0.1667 ± 1.6667	1.5000 ± 4.7937	5.8333 ± 8.6635
rep_rand_subs a)	0.4960 ± 0.6510	3.2530 ± 1.6292	8.5806 ± 1.9839
rep_rand_subs b)	0.4953 ± 0.4416	2.8511 ± 1.0668	8.3981 ± 1.4637
rep_rand_subs c)	0.5767 ± 0.4467	3.0700 ± 1.0226	8.8400 ± 1.1954

Tab.6.1.2 Výsledky třídění algoritmem Ho-Kashyap

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	0.4706 ± 6.8518	2.8268 ± 16.6033	9.6698 ± 29.5895
K_fold /K=25/	0.4722 ± 1.6375	2.8303 ± 5.6587	9.8497 ± 5.9388
K_fold /K=100/	0.3333 ± 2.3451	1.3333 ± 4.5443	6.8333 ± 9.7945
rep_rand_subs a)	0.5600 ± 0.6869	3.2651 ± 1.5440	9.9274 ± 2.2306
rep_rand_subs b)	0.6321 ± 0.5658	3.156 ± 1.3002	9.8531 ± 1.7634
rep_rand_subs c)	0.7600 ± 0.6179	3.865 ± 1.1867	10.2230 ± 1.3691

Tab.6.1.3 Výsledky třídění algoritmem NDDF

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	21.6471 ± 41.2324	16.2544 ± 36.9603	29.2453 ± 45.5427
K_fold /K=25/	22.1487 ± 10.4600	15.4303 ± 11.7994	28.6046 ± 9.8287
K_fold /K=100/	15.3333 ± 14.7329	7.5000 ± 10.1545	20.5000 ± 13.3701
rep_rand_subs a)	31.1560 ± 2.8700	16.7730 ± 2.1630	26.4740 ± 2.5233
rep_rand_subs b)	27.8320 ± 3.5291	16.2650 ± 3.4271	27.2260 ± 2.9584
rep_rand_subs c)	34.5670 ± 2.7636	16.8800 ± 2.2124	26.8870 ± 2.1764

Tab.6.1.4 Výsledky třídění algoritmem Nearest Neighbor Knn = 5

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	32.9412 ± 47.0554	15.9011 ± 36.6334	22.8774 ± 42.0539
K_fold /K=25/	32.0621 ± 13.9189	15.3939 ± 9.4908	22.2255 ± 5.8688
K_fold /K=100/	23.0000 ± 19.2129	7.3333 ± 8.9643	16.0000 ± 13.1724
rep_rand_subs a)	35.4910 ± 2.9055	15.9930 ± 2.2773	25.3600 ± 2.2478
rep_rand_subs b)	33.4080 ± 3.7817	16.0240 ± 3.5074	23.8790 ± 2.8790
rep_rand_subs c)	37.6300 ± 2.7690	16.0700 ± 1.5875	25.2900 ± 2.1334

Tab.6.1.5 Výsledky třídění algoritmem Nearest Neighbor Knn = 10

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	34.3529 ± 47.5446	15.5477 ± 36.3001	25.4717 ± 43.6217
K_fold /K=25/	34.1748 ± 8.9168	15.1939 ± 8.2657	24.3186 ± 5.8126
K_fold /K=100/	24.5000 ± 15.0672	7.3333 ± 8.3148	17.8333 ± 12.3671
rep_rand_subs a)	34.4810 ± 2.2362	15.5390 ± 1.9704	24.5500 ± 2.2883
rep_rand_subs b)	34.0240 ± 2.6287	15.9880 ± 3.1801	23.6530 ± 2.8886
rep_rand_subs c)	34.8470 ± 2.1453	15.6600 ± 1.6328	26.2670 ± 2.5271

Tab.6.1.6 Výsledky třídění algoritmem Nearest Neighbor Knn = 25

6.2 Výsledky klasifikace příznaků získaných metodou fraktální koeficient

označení	Nemocní	Zdraví	Zdraví_Nemocní
a)	200	200	200
b)	130	130	130
c)	61	61	61
vzorů celkem	261	261	261

Tab.6.2.1 Počty trénovacích vzorů pro rep_rand_subs

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	5.5556 ± 22.9281	13.2184 ± 33.9015	20.3065 ± 40.2667
K_fold /K=25/	5.9636 ± 4.9417	13.1636 ± 7.4233	20.0727 ± 7.0780
K_fold /K=100/	5.1667 ± 8.9690	12.2333 ± 12.9148	18.6333 ± 15.8917
rep_rand_subs a)	5.5492 ± 1.8562	13.5331 ± 2.5404	20.1886 ± 3.1297
rep_rand_subs b)	5.7366 ± 1.0427	13.2330 ± 1.2896	19.8134 ± 1.9234
rep_rand_subs c)	5.8946 ± 0.7986	13.8791 ± 1.2429	20.0849 ± 1.3410

Tab.6.2.2 Výsledky třídění algoritmem Ho-Kashyap

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	5.7471 ± 23.2964	13.9847 ± 34.7161	16.2835 ± 36.9569
K_fold /K=25/	5.5273 ± 4.4569	14.0727 ± 6.8314	15.9636 ± 5.9388
K_fold /K=100/	5.4000 ± 9.1229	12.9000 ± 12.3697	15.3667 ± 14.4863
rep_rand_subs a)	5.4344 ± 1.8627	14.1229 ± 3.1538	17.2623 ± 3.1430
rep_rand_subs b)	5.6069 ± 1.2198	14.1183 ± 1.6305	17.4809 ± 1.7350
rep_rand_subs c)	6.1500 ± 1.0618	14.9175 ± 1.2487	18.3800 ± 1.4648

Tab.6.2.3 Výsledky třídění algoritmem NDDF

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	15.1341 ± 35.8725	17.0498 ± 37.6431	31.2261 ± 46.3860
K_fold /K=25/	15.2727 ± 6.9433	17.4000 ± 6.3670	31.2000 ± 9.9013
K_fold /K=100/	13.6667 ± 14.1223	15.5667 ± 13.7073	29.5000 ± 19.2122
rep_rand_subs a)	16.2541 ± 3.0138	16.7951 ± 3.1011	32.7296 ± 4.0963
rep_rand_subs b)	16.9618 ± 2.0155	17.9122 ± 1.5790	34.9732 ± 2.2107
rep_rand_subs c)	19.2575 ± 1.8861	19.8725 ± 1.5298	38.2250 ± 2.4159

Tab.6.2.4 Výsledky třídění algoritmem Nearest Neighbor Knn = 5

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	15.1341 ± 35.8725	16.6667 ± 37.3035	34.0996 ± 47.4499
K_fold /K=25/	15.3455 ± 6.0534	17.0000 ± 4.8974	33.9091 ± 8.1713
K_fold /K=100/	14.3333 ± 12.9274	15.6000 ± 14.0136	31.1333 ± 19.5759
rep_rand_subs a)	16.0984 ± 2.9806	17.4262 ± 3.5739	34.6557 ± 3.5739
rep_rand_subs b)	17.2290 ± 1.9851	17.7366 ± 1.9298	36.0191 ± 2.6179
rep_rand_subs c)	19.9475 ± 2.2877	20.4952 ± 2.2877	37.4175 ± 2.2877

Tab.6.2.5 Výsledky třídění algoritmem Nearest Neighbor Knn = 10

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	16.8582 ± 37.4742	17.0498 ± 37.6431	34.2912 ± 47.5138
K_fold /K=25/	17.4364 ± 7.4028	16.2000 ± 5.4858	32.4545 ± 9.2700
K_fold /K=100/	15.9333 ± 13.7688	15.4000 ± 12.9817	31.2333 ± 21.3971
rep_rand_subs a)	17.2705 ± 3.4131	17.3934 ± 2.9463	33.3852 ± 3.8319
rep_rand_subs b)	18.7557 ± 1.8397	19.2939 ± 2.1477	34.9924 ± 2.4717
rep_rand_subs c)	21.4300 ± 1.4035	21.2750 ± 1.5343	37.9500 ± 2.9263

Tab.6.2.6 Výsledky třídění algoritmem Nearest Neighbor Knn = 25

6.3 Výsledky klasifikace příznaků získaných metodou co-occurrence

označení	Nemocní	Zdraví	Zdraví_Nemocní
a)	200	200	200
b)	130	130	130
c)	61	61	61
vzorů celkem	261	261	261

Tab.6.3.1 Počty trénovacích vzorů pro rep_rand_subs

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	4.0299 ± 19.6686	14.7509 ± 35.4953	29.5019 ± 45.6489
K_fold /K=25/	4.2364 ± 4.7938	14.3273 ± 5.8551	28.7636 ± 10.4714
K_fold /K=100/	2.1000 ± 4.3333	7.7000 ± 7.8951	15.6000 ± 10.9471
rep_rand_subs a)	3.9016 ± 1.5047	14.8688 ± 2.4896	29.6393 ± 3.9016
rep_rand_subs b)	4.1679 ± 0.8441	14.2290 ± 1.8603	29.5496 ± 2.2216
rep_rand_subs c)	4.9150 ± 0.9723	14.7050 ± 1.4705	30.2050 ± 1.4801

Tab.6.3.2 Výsledky třídění algoritmem Ho-Kashyap

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	4.7893 ± 21.3744	13.7931 ± 34.5158	31.8008 ± 46.6149
K_fold /K=25/	4.8000 ± 4.8869	14.1455 ± 5.8245	31.4545 ± 8.5139
K_fold /K=100/	2.5000 ± 4.7937	7.3000 ± 7.7662	16.8000 ± 10.3358
rep_rand_subs a)	4.9672 ± 1.7234	14.1639 ± 3.0315	31.3852 ± 4.9672
rep_rand_subs b)	5.0001 ± 1.0679	14.3282 ± 1.6179	31.2748 ± 2.2201
rep_rand_subs c)	5.4175 ± 0.8733	15.3825 ± 1.3937	31.5250 ± 1.7879

Tab.6.3.3 Výsledky třídění algoritmem NDDF

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	41.7625 ± 49.3641	48.6590 ± 50.0300	48.8506 ± 50.0347
K_fold /K=25/	42.1818 ± 11.9226	47.7636 ± 10.9971	49.9636 ± 7.0844
K_fold /K=100/	21.9000 ± 11.5203	25.2000 ± 11.2349	25.5000 ± 11.0440
rep_rand_subs a)	41.2541 ± 4.1633	45.9836 ± 4.0734	47.0655 ± 4.0429
rep_rand_subs b)	41.6298 ± 2.7580	45.7786 ± 2.4004	47.0191 ± 2.8198
rep_rand_subs c)	42.8400 ± 2.5953	46.9475 ± 2.2344	46.5975 ± 2.6914

Tab.6.3.4 Výsledky třídění algoritmem Nearest Neighbor Knn = 5

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	42.1456 ± 49.4266	46.7433 ± 49.9417	45.4023 ± 49.8359
K_fold /K=25/	43.7273 ± 9.9214	47.3636 ± 9.4010	44.7273 ± 8.1343
K_fold /K=100/	22.3000 ± 10.9963	24.1000 ± 11.4676	23.3000 ± 11.1060
rep_rand_subs a)	41.1557 ± 3.9872	46.6311 ± 3.7767	45.6557 ± 3.9961
rep_rand_subs b)	42.3779 ± 2.5378	46.2328 ± 2.4632	45.7023 ± 2.6397
rep_rand_subs c)	44.1725 ± 2.1901	46.0950 ± 2.0688	45.6525 ± 2.4505

Tab.6.3.5 Výsledky třídění algoritmem Nearest Neighbor Knn = 10

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	37.9310 ± 48.5681	44.6360 ± 49.7591	41.1877 ± 49.2645
K_fold /K=25/	38.2464 ± 9.8973	44.6545 ± 10.6117	41.7818 ± 11.6649
K_fold /K=100/	19.8000 ± 10.7290	23.0000 ± 11.8492	21.6000 ± 11.3458
rep_rand_subs a)	41.3361 ± 4.0446	44.8771 ± 3.5402	43.7951 ± 3.5876
rep_rand_subs b)	42.2366 ± 2.4719	45.2291 ± 2.3885	43.7824 ± 2.3398
rep_rand_subs c)	44.7675 ± 2.8712	47.1425 ± 2.8713	48.885 ± 3.2083

Tab.6.3.6 Výsledky třídění algoritmem Nearest Neighbor Knn = 25

6.4 Výsledky klasifikace příznaků získaných metodou run-length

označení	Nemocní	Zdraví	Zdraví_Nemocní
a)	200	200	200
b)	130	130	130
c)	61	61	61
vzorů celkem	261	261	261

Tab.6.4.1 Počty trénovacích vzorů pro rep_rand_subs

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	5.1724 ± 22.1682	6.7050 ± 25.0348	24.7126 ± 43.1755
K_fold /K=25/	5.3455 ± 3.6180	6.5091 ± 5.5573	24.0909 ± 8.4285
K_fold /K=100/	3.0000 ± 5.4376	3.8889 ± 6.1924	14.0000 ± 10.6668
rep_rand_subs a)	5.0655 ± 1.6450	6.7213 ± 2.1400	24.0327 ± 3.1786
rep_rand_subs b)	5.3282 ± 1.1305	7.1221 ± 1.3978	24.3435 ± 1.9065
rep_rand_subs c)	5.6900 ± 0.9361	7.6500 ± 1.1495	24.2450 ± 1.1220

Tab.6.4.2 Výsledky třídění algoritmem Ho-Kashyap

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	4.9808 ± 21.7758	6.9050 ± 25.0348	22.0307 ± 41.4851
K_fold /K=25/	5.2182 ± 3.9520	6.5091 ± 5.4997	21.4727 ± 10.4983
K_fold /K=100/	3.000 ± 5.4376	3.7778 ± 5.5207	10.8889 ± 10.9095
rep_rand_subs a)	5.4344 ± 1.9060	7.1803 ± 2.1359	21.9590 ± 3.6027
rep_rand_subs b)	5.2404 ± 0.9778	7.1336 ± 1.3613	22.0954 ± 1.9536
rep_rand_subs c)	5.6050 ± 0.7639	7.7025 ± 1.0575	23.0975 ± 1.5895

Tab.6.4.3 Výsledky třídění algoritmem NDDF

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	4.7893 ± 21.3744	16.2835 ± 36.9569	34.0996 ± 47.4499
K_fold /K=25/	4.8181 ± 4.3261	16.4545 ± 6.1977	34.0182 ± 8.6579
K_fold /K=100/	2.7778 ± 5.3264	9.5556 ± 9.3457	19.4444 ± 11.3196
rep_rand_subs a)	4.8604 ± 1.5379	16.4508 ± 3.0117	33.0327 ± 3.7362
rep_rand_subs b)	4.7904 ± 1.0963	16.0610 ± 1.6344	32.5192 ± 2.3352
rep_rand_subs c)	5.3525 ± 0.9956	16.0275 ± 1.3816	33.2500 ± 2.2761

Tab.6.4.4 Výsledky třídění algoritmem Nearest Neighbor Knn = 5

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	4.2146 ± 20.1114	15.5172 ± 36.2416	29.8851 ± 45.8193
K_fold /K=25/	4.2364 ± 4.3790	15.4727 ± 5.8439	29.9818 ± 9.0809
K_fold /K=100/	2.4444 ± 4.8881	9.0000 ± 8.7496	17.1111 ± 10.5255
rep_rand_subs a)	4.5492 ± 1.6117	15.8197 ± 2.5802	30.4754 ± 3.9754
rep_rand_subs b)	4.7862 ± 1.1030	15.5534 ± 1.7632	31.6488 ± 2.1210
rep_rand_subs c)	5.2225 ± 0.7776	15.2825 ± 1.2801	32.4625 ± 2.0509

Tab.6.4.5 Výsledky třídění algoritmem Nearest Neighbor Knn = 10

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	4.9808 ± 21.7758	16.0920 ± 36.7809	30.2682 ± 45.9859
K_fold /K=25/	4.9818 ± 4.1670	16.0545 ± 5.6112	30.5273 ± 9.8470
K_fold /K=100/	2.8889 ± 5.1466	9.3333 ± 9.0336	17.3333 ± 11.6395
rep_rand_subs a)	5.1311 ± 1.8185	15.0902 ± 2.4616	31.5657 ± 3.3776
rep_rand_subs b)	5.1031 ± 0.9819	14.6297 ± 1.4917	31.5573 ± 2.5311
rep_rand_subs c)	5.0000 ± 0.7334	14.3600 ± 0.8283	30.89250 ± 1.8001

Tab.6.4.6 Výsledky třídění algoritmem Nearest Neighbor Knn = 25

6.5 Výsledky klasifikace příznaků získaných strukturální metodou

označení	Nemocní	Zdraví	Zdraví_Nemocní
a)	200	200	200
b)	130	135	135
c)	61	70	70
vzorů celkem	261	270	270

Tab.6.5.1 Počty trénovacích vzorů pro rep_rand_subs

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	8.8512 ± 28.4306	27.3069 ± 44.5956	26.6667 ± 44.2627
K_fold /K=25/	8.6113 ± 5.9309	27.1879 ± 9.3712	26.2182 ± 8.2819
K_fold /K=100/	8.8433 ± 12.5598	26.9500 ± 19.6876	26.6667 ± 18.3876
rep_rand_subs a)	8.7634 ± 2.2835	27.8931 ± 3.3192	26.1857 ± 2.9736
rep_rand_subs b)	9.3158 ± 1.4457	28.0376 ± 1.9125	25.8741 ± 1.9469
rep_rand_subs c)	9.5300 ± 1.1863	27.6300 ± 1.4779	25.8750 ± 1.3816

Tab.6.5.2 Výsledky třídění algoritmem Ho-Kashyap

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	12.9944 ± 33.6558	34.8399 ± 47.6913	27.4074 ± 44.6460
K_fold /K=25/	12.9896 ± 5.0834	34.4632 ± 7.6486	27.3818 ± 9.7439
K_fold /K=100/	13.1000 ± 14.1183	34.0500 ± 19.1122	28.0833 ± 20.3661
rep_rand_subs a)	13.0153 ± 2.4226	35.7023 ± 3.4462	27.5714 ± 3.2677
rep_rand_subs b)	12.7782 ± 1.5033	34.2669 ± 2.8170	27.6148 ± 2.3158
rep_rand_subs c)	12.7275 ± 1.1823	33.7375 ± 3.0373	27.8500 ± 1.7742

Tab.6.5.3 Výsledky třídění algoritmem NDDF

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	9.6045 ± 29.4931	30.6968 ± 46.1671	28.3333 ± 45.1035
K_fold /K=25/	9.7645 ± 4.9935	31.2649 ± 9.6768	26.8000 ± 7.5600
K_fold /K=100/	9.9000 ± 13.3582	29.9667 ± 19.5645	28.1667 ± 19.3823
rep_rand_subs a)	9.9465 ± 2.2822	31.5267 ± 3.9486	28.1214 ± 3.0637
rep_rand_subs b)	10.2952 ± 1.4185	32.5092 ± 2.03160	28.2750 ± 2.1314
rep_rand_subs c)	10.7325 ± 1.1521	31.3100 ± 2.3058	29.6100 ± 1.9499

Tab.6.5.4 Výsledky třídění algoritmem Nearest Neighbor Knn = 5

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	9.0395 ± 28.7018	30.8851 ± 46.2455	26.8519 ± 44.3600
K_fold /K=25/	8.8294 ± 4.9206	30.1853 ± 9.3448	26.6182 ± 9.2580
K_fold /K=100/	8.8167 ± 12.0379	30.4667 ± 20.4839	26.7500 ± 18.5507
rep_rand_subs a)	9.3740 ± 2.2369	29.4273 ± 3.4573	28.4786 ± 3.5629
rep_rand_subs b)	9.9511 ± 1.4412	29.0526 ± 2.0736	28.7185 ± 2.2815
rep_rand_subs c)	10.6625 ± 1.2290	29.4025 ± 1.7957	29.6575 ± 2.0096

Tab.6.5.5 Výsledky třídění algoritmem Nearest Neighbor Knn = 10

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	9.9812 ± 30.0031	27.8719 ± 44.8792	27.7778 ± 44.8319
K_fold /K=25/	9.9195 ± 6.0106	27.3775 ± 10.0041	27.8909 ± 10.5095
K_fold /K=100/	9.9000 ± 13.4106	27.4500 ± 18.5659	27.8333 ± 19.6497
rep_rand_subs a)	10.2748 ± 2.6519	27.1452 ± 3.2100	28.0143 ± 3.3450
rep_rand_subs b)	10.8571 ± 1.6627	28.0526 ± 1.9682	27.2889 ± 2.2284
rep_rand_subs c)	10.9000 ± 0.9968	29.5350 ± 1.6655	28.3625 ± 1.4356

Tab.6.5.6 Výsledky třídění algoritmem Nearest Neighbor Knn = 25

6.6 Výsledky klasifikace příznaků získaných metodou základní statistika

označení	Nemocní	Zdraví	Zdraví_Nemocní
a)	200	200	200
b)	130	135	135
c)	61	70	70
vzorů celkem	261	270	270

Tab.6.6.1 Počty trénovacích vzorů pro rep_rand_subs

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	5.6497 ± 23.1097	12.0527 ± 32.5884	28.5185 ± 45.1921
K_fold /K=25/	5.6589 ± 4.5440	12.1636 ± 6.1485	28.4364 ± 9.9664
K_fold /K=100/	5.3667 ± 9.9595	11.0667 ± 13.2986	26.8667 ± 17.7324
rep_rand_subs a)	5.5878 ± 1.4421	12.0305 ± 2.1512	28.4286 ± 3.3253
rep_rand_subs b)	5.6466 ± 1.0647	12.5338 ± 1.5742	28.8371 ± 1.9208
rep_rand_subs c)	5.9700 ± 0.7604	13.3450 ± 0.9555	29.2250 ± 1.4614

Tab.6.6.2 Výsledky třídění algoritmem Ho-Kashyap

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	6.7797 ± 25.1633	12.4294 ± 33.0228	33.8889 ± 47.3771
K_fold /K=25/	8.9545 ± 5.4972	12.0260 ± 8.1631	35.0000 ± 8.9545
K_fold /K=100/	6.1000 ± 9.9387	11.7000 ± 14.8051	32.0333 ± 16.7499
rep_rand_subs a)	6.9771 ± 1.9530	13.3969 ± 2.5592	33.7929 ± 2.9663
rep_rand_subs b)	6.8233 ± 1.4879	13.3571 ± 1.7449	33.9000 ± 2.1412
rep_rand_subs c)	6.8975 ± 1.2949	14.2175 ± 1.6281	33.8750 ± 1.9081

Tab.6.6.3 Výsledky třídění algoritmem NDDF

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	5.2731 ± 22.3706	12.4294 ± 33.0227	30.5556 ± 46.1069
K_fold /K=25/	5.2675 ± 5.6929	12.5532 ± 7.8337	29.7091 ± 10.4627
K_fold /K=100/	4.8333 ± 9.9543	11.7000 ± 13.6453	28.4333 ± 18.8886
rep_rand_subs a)	6.3740 ± 1.6790	13.6488 ± 1.8937	31.0357 ± 3.3416
rep_rand_subs b)	7.3609 ± 1.6445	14.5714 ± 2.0375	32.2667 ± 2.0492
rep_rand_subs c)	11.7700 ± 2.2306	19.1300 ± 2.7088	36.0750 ± 2.3920

Tab.6.6.4 Výsledky třídění algoritmem Nearest Neighbor Knn = 5

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	6.4030 ± 24.5037	12.4294 ± 33.0227	32.0370 ± 46.7052
K_fold /K=25/	6.3766 ± 5.5546	12.9161 ± 7.9007	31.2364 ± 9.9287
K_fold /K=100/	6.0000 ± 9.9381	11.7667 ± 15.2896	29.9333 ± 18.3309
rep_rand_subs a)	7.9084 ± 2.5642	15.3511 ± 2.9711	31.8143 ± 3.1364
rep_rand_subs b)	10.2481 ± 2.2177	17.2518 ± 2.7308	34.4963 ± 2.1590
rep_rand_subs c)	15.0425 ± 2.4487	23.1025 ± 2.3336	39.3750 ± 2.7639

Tab.6.6.5 Výsledky třídění algoritmem Nearest Neighbor Knn = 10

Metoda	NE x ZD	NE x ZN	ZN x ZD
L_1_out	10.1695 ± 30.2532	17.5141 ± 38.0446	32.5926 ± 46.9154
K_fold /K=25/	10.1013 ± 6.4527	17.4130 ± 6.4527	33.0909 ± 11.2487
K_fold /K=100/	9.4000 ± 12.1197	16.4000 ± 16.4305	30.2667 ± 17.5789
rep_rand_subs a)	11.8091 ± 2.4965	20.2214 ± 3.8384	34.8857 ± 3.4663
rep_rand_subs b)	15.2819 ± 3.3251	23.2669 ± 2.9149	37.3333 ± 2.8833
rep_rand_subs c)	21.6275 ± 1.2459	26.3675 ± 1.5194	44.2800 ± 2.9726

Tab.6.6.6 Výsledky třídění algoritmem Nearest Neighbor Knn = 25

6.7 Výsledky klasifikace K-means

Pro metodu K-means není třeba žádného nastavení. Jedná se o metodu třídění bez učitele a není tedy potřeba žádné trénovací skupiny. Algoritmus sám roztřídí vzory do dvou skupin. Obtížné je pouze zjistit, jaké výsledné skupině odpovídá potřebná třída. V prvním inicializačním kroku je sice vybrán z každé třídy jeden vzor, ale klasifikátor jej může v následujícím kroku zařadit do třídy jiné. Proto je obtížné označit příslušné třídy. V našem případě, kdy dělíme data jen na dvě skupiny, počítáme s chybou menší než 50%. Když je chyba větší, tak změníme označení tříd, a tím dostaneme chybu menší než 50%.V následující tabulce 6.7 je výsledná klasifikační chyba v procentech pro všechny tříd vzorů (NE-ZD, NE-ZN, ZN-ZD)

Metoda výpočtu vzorů	NE x ZD	NE x ZN	ZN x ZD
Markovovy náhodná pole	48.000	36.043	37.500
Fraktální koeficient	26.437	23.372	45.019
Metoda co-occurence	46.360	47.510	48.084
Metoda run-length	8.238	14.943	32.376
Strukturální metoda	12.053	30.697	29.815
Základní statistika	26.177	34.463	48.148

Tab.6.7 Výsledky třídění algoritmem K-means

7 Diskuze dosažených výsledků

Všechny použité klasifikátory dokáží nejlépe roztřídit vzorky nemocné (NE) a zdravé tkáně (ZD). To je způsobeno jejich značnou odlišností, protože vzorky nemocné tkáně neobsahují žíhání charakteristické pro nervová vlákna na sítnici, zatímco zdravá tkáň je tím charakteristická. Co se týká klasifikace tkáně zdravé (ZD) s tkání zdravou od nemocných pacientů (ZN), tak pro klasifikátory je velice obtížné ji rozlišit, protože oba typy textury do jisté míry vypadají podobně (obsahují žíhání).

Pro vzory vypočtené metodou Markovových náhodných polí se pro třídění nejvíce hodí algoritmus Ho-Kashyap, protože dosahuje nejmenší chyby klasifikace. O něco málo horších výsledků dosáhl algoritmus NDDF. Jako nejméně vhodný se jeví algoritmus Nearest Neighbor. Jestliže zhodnotíme testování úspěšnosti s ohledem na metody výběru vzorů pro trénování, tak jako nejlepší se jeví metoda K fold cross validation. Konkrétně pro K = 100 vychází pro algoritmus Ho-Kashyap a třídění ZD a NE pouze 0.1667 %. Pro K = 25 je chyba klasifikace 0.25%, což je téměř shodné s metodou Leave one out cross validation. Metoda Repeated random subsampling cross validation má v daném případě chybu dvojnásobnou.

Pro vzory vypočtené druhou metodou - fraktální koeficient jsou výsledky následující: Algoritmy Ho-Kashyap a NDDF dosahují srovnatelných výsledků.Výsledky se zásadně neliší ani pro různé typy validace. Pro skupinu ZD x NE se výsledky pohybují okolo 5.5 %, pro NE x ZN se chyba klasifikace pohybuje okolo 13 % a pro ZD x ZN je chyba okolo 20 %. Metoda klasifikace dle nejbližšího souseda za zmíněnými algoritmy poněkud zaostává, protože dosahuje chyby asi 2-3 krát větší.

Pro další vzory, vypočtené metodou co-occurence, se výsledné chyby klasifikace téměř shodují pro algoritmy Ho-Kashyap a NDDF. Jako nevhodný algoritmus pro třídění těchto vzorů se jeví metoda nejbližších sousedů, protože chyba klasifikace se pohybuje nad 40 %. Jako zajímavé se jeví použití hodnocení K fold cross validation pro K = 100. Výsledné chyby pro tuto validaci jsou ve srovnání s ostatními výsledky přibližně poloviční, a to pro všechny metody.

Vzory vypočítané metodou run-length mají následující výsledky: Pro třídění tříd NE x ZD se chyba klasifikátorů pohybuje okolo 5 %. A to pro všechny použité metody validace. Výjimku zde tvoří opět K fold cross validation (K = 100), jejíž výsledky jsou pro všechny tříděné skupiny poloviční. Konkrétně nejlepšího výsledku zde dosáhl algoritmus Nearest Neighbor pro Knn = 10 a to 2.4444 % (NE x ZD). Pro ostatní tříděné skupiny (NE x ZN a ZN x ZD) vychází menší chyba klasifikace pro algoritmy Ho-Kashyap a NDDF, ve srovnání s Nearest Neighbor. Metoda nejbližších sousedů dosahuje klasifikační chyby přibližně 2–krát vyšší než první dvě klasifikace (pro NE x ZN a ZN x ZD).

Pro vzory získané strukturální metodou má největší chyby klasifikace algoritmus NDDF. Zatímco Ho-Kashyap a Nearest Neighbor dosahují srovnatelných výsledků. Pro klasifikaci skupin NE x ZD je nejmenší chyba klasifikace pro algoritmus Ho-Kashyap a K fold cross validation, pro K = 25, konkrétně 8.6113 %. Pro skupinu NE x ZN je nejmenší chyba klasifikace pro Ho-Kashyap a K fold cross validation, pro K = 100, a to 26.95 %. Pro skupinu zdraví ZN x ZD je nejmenší chyba opět pro Ho-Kashyap a validaci repeated random subsampling (pro 135 trénovacích vzorů z 270) , a to 25.8741 %.

Poslední metodou texturní analýzy byla metoda založená na základní statistice. Výsledné chyby klasifikací se od sebe výrazně neodlišují pro algoritmy Ho-Kashyap a NDDF. Pro Ho-Kashyap jsou klasifikační chyby následující: pro NE x ZD je chyba přibližně 5 %, pro NE x ZN se klasifikační chyba pohybuje okolo 12 % a pro skupinu ZN x ZD je klasifikační chyba asi 28 %. Metoda nejbližších sousedů má srovnatelné chyby pro Knn = 5 a Knn = 10 a validace Leave one out a K fold cross validation. Validace repeated random subsampling má pro snižující se počet trénovacích vzorů chybu klasifikace vyšší. Nejvyšší chybu klasifikace dosahuje algoritmus Nearest Neighbor pro Knn = 25.

Pro shlukovou analýzu K means vychází klasifikační chyba nejmenší pro vzory vypočítané metodou run-length. Pro třídy NE x ZD je chyba 8.238 %, pro NE x ZN je chyba klasifikace 14.943 % a pro skupiny ZN x ZD je klasifikační chyba 32.376 %. Největší chyba klasifikace je u klasifikátoru K-means pro vzory vypočítané metodou co-occurrence a pohybuje se pod 50 %.

Závěr

Tato práce se zabývá tříděním retinálních dat několika klasifikačními algoritmy. Jsou to za prvé algoritmus Ho-Kashyap, který patří mezi lineární klasifikátory. Jako další je to Bayessovský klasifikátor NDDF (Normal Density Discriminant Function) a klasifikace metodou nejbližšího souseda k-NN. Tyto metody patří do skupiny učení s učitelem. Jako poslední je implementován klasifikátor K-means, který spadá do shlukové analýzy (clusteringu). Jedná se o formu učení bez učitele.

Pro hodnocení výkonnosti algoritmů jsou v práci použity tři základní metody výběru testovacích dat. První metodou je „repeated random subsampling cross validation“, kdy se testovací vzory vybírají náhodně a vlastní klasifikace se (v našem případě) opakuje stokrát a vyhodnocuje se průměrná klasifikační chyba. Další metodou je „leave one out cross validation“, která používá pro testování pouze jeden vzor, a jako třetí je použita metoda „k-fold cross validation“, kdy se vzory rozdělí na *k* podskupin a poté se postupně použijí pro testování algoritmu všechny tyto podskupiny.

Vlastní klasifikace byla odzkoušena na šesti souborech dat, které byly získány různými metodami texturní analýzy z obrazů sítnice od pacientů s glaukomem a pacientů zdravých. Z vlastních obrazů byly získány tři skupiny vzorů. Vzorky zdravé tkáně od pacientů zdravých, vzorky nemocné tkáně od pacientů s glaukomem a jako poslední vzorky zdravé tkáně od pacientů postižených glaukomem.

Úkolem klasifikace bylo správně roztřídit tyto data do předem daných tříd. Jako nejvhodnější se jeví algoritmus Ho-Kashyap, který dosahuje nejmenších chyb klasifikace. Jako druhý se osvědčil algoritmus NDDF. Nejslabším algoritmem klasifikátoru, který patří do skupiny učení s učitelem, je Nearest Neighbor. I když je zde otestován pro tři možné okruhy nejbližších sousedů k-NN. Jednotlivé výsledky jsou detailně diskutovány v kapitole 7.

Co se týká způsobu výběru dat ke klasifikaci, tak nejlepších výsledků dosahuje metoda „K-fold cross validation“, která ve dvou metodách dosahuje dokonce poloviční chyby klasifikace ve srovnání se zbývajícími dvěma. Metody „Leave-one-out cross validation“ a metoda „Repeated random subsampling cross validation“ dosahují srovnatelných výsledků.

Třídění formou shlukování, K-means není pro klasifikaci těchto dat vhodné, protože dosažená chyba klasifikace je ve srovnání s algoritmy učení s učitelem daleko vyšší.

Seznam použité literatury

[1] Duda O. R.,Hart E. P. , Stork G.D.: *Pattern Clasification (2nd Edition)*, Wilez-Interscience, 2000, ISBN: 0-471-05669-3.

[2] Flammer J.: *Glaukom*. Triton, 1. vydání 2003; ISBN 80-7254-351-2.

[3] Kolář R.: Methods *for image analysis and pattern recognition - Application to early glaucoma diagnosis*: habilitation thesis: Department of Biomedical Engineering,Faculty of Electrical Engineering and Communication Brno, University of Technology , November 2008.

[4] -: *Cross-validation (statistics)*. [cit. 28.dubna 2010]. Dostupné z www: <http://en.wikipedia.org/wiki/Cross-validation_%28statistics%29>.

[5] Šochman J.: Cvičení z *RZP-shlukování k-means*. [cit. 28.dubna 2010]. Dostupné z www: < http://cmp.felk.cvut.cz/cmp/courses/recognition/Labs/kmeans/kmeans.pdf>.

[6] Odstrčilík, J.: *Analýza barevných snímků sítnice se zaměřením na segmentaci cévního řečiště* : diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 82 s., 2 přílohy. Vedoucí diplomové práce byl prof. Ing. Jiří Jan, CSc.

[7] Baštinec J.: *Pravděpodobnost, statistika, operační výzkum*. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií,Ústav matematiky, 2008

[8] Gazárek J. *Texturní analýza snímků sítnice se zaměřením na detekci nervových vláken*. Brno: Vysoké učení technické v Brně. Fakulta elektroniky a komunikačních technologií. Ústav biomedicínského inženýrství, 2008. Počet stran 88, počet stran příloh 15. Diplomová práce. Vedoucí práce byl prof. Ing. Jiří Jan, CSc.

[9] -: *Stavba oka*. [cit. 4.května 2010]. Dostupné z www: <http://fyzika.jreichl.com/index.php?page=486&sekce=browse>.

[10] -: *Oční vady*. [cit. 4.května 2010]. Dostupné z www: <http://www.lexum.cz/ocni-vady.php#kr>.

[11] -:.*Choroby oka*. [cit. 4.května 2010]. Dostupné z www: <http://www.videni.cz/nemoci-oci/choroby-oka>.

[12] Pivoňková A.,Šmerák P.: *Klasifikace podle nejbližších sousedů, Zápis z přednášky*., ČVUT v Praze, 2002/03.

[13] Odstrčilík, J.: *Analýza obrazových dat sítnice pro podporu lékařské diagnostiky*: pojednání. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 29 s.,Vedoucí diplomové práce je doc. ing. Radim Kolář, Ph.D.

[14] -:.*Fraktál*. [cit. 4.května 2010]. Dostupné z www:<http://cs.wikipedia.org/wiki/Frakt%C3%A1l>.

Použité zkratky a symboly

FEKT	Fakulta elektrotechniky a komunikačních technologií
ÚBMI	Ústav biomedicínského inženýrství
VUT	Vysoké učení technické
NDDF	Normal Density Discriminant Function (diskriminační funkce pro normální rozložení pravděpodobnosti)
Knn	K nearest neighbors (K nejbližších sousedů)
ZD	Zdraví (zdravá tkáň od zdravých pacientů)
NE	Nemocní (nemocná tkáň od nemocných pacientů)
ZN	Zdraví nemocní (zdravá tkáň od nemocných pacientů)

Seznam příloh

Příloha 1 – Výpis algoritmu Ho-Kashyap

```
function [ test_targets ] = Ho_Kashyap(train_patterns, train_targets, test_patterns, params)

% Classify using the using the Ho-Kashyap algorithm
% Inputs:
%       train_patterns  - Train patterns
%       train_targets   - Train targets
%       test_patterns    - Test  patterns
%       params          - [Type(Basic/Modified), Maximum iteration, Convergence criterion,
Convergence rate]
%
% Outputs
%       test_targets    - Predicted targets
%       w_percept       - Classifier weights
%       b               - Margin

[c, n]              = size(train_patterns);

% [type, Max_iter, b_min, eta] = process_params(params);
 type ='Basic';
%  type='modified';
Max_iter = 2000;
% b_min = 0.002;
b_min = 0.0002;
% eta = 0.05;
eta = 0.005;

train_patterns  = [train_patterns ; ones(1,n)];
train_zero      = find(train_targets == 0);

%Preprocessing (Needed so that b>0 for all patterns)
processed_patterns = train_patterns;
processed_patterns(:,train_zero) = -processed_patterns(:,train_zero);
processed_targets  = 2*train_targets-1;

b  = ones(1,n);
Y = processed_patterns;
a  = pinv(Y')*b';
k  = 0;
e  = 1e3;
found = 0;

while ((sum(abs(e) > b_min)>0) & (k < Max_iter) &(~found))

   %k <- (k+1) mod n
   k = k+1;
```

```
    %e <- Ya - b
   e       = (Y' * a)' - b;

   %e_plus <- 1/2(e+abs(e))
   e_plus  = 0.5*(e + abs(e));

   %b <- b + 2*eta*e_plus
   b       = b + 2*eta*e_plus;

   if strcmp(type,'Basic'),
      %a <- pinv(Y)*b;
      a = pinv(Y')*b';
   else
      %a <- a + eta*pinv(Y)*|e_plus|;
      a = a + eta*pinv(Y')*abs(e_plus)';
   end

end

if (k == Max_iter),
%    disp(['No solution found']);
  sum(abs(e) );
else
  disp(['Did ' num2str(k) ' iterations'])
end

test_targets = a' * [test_patterns; ones(1, size(test_patterns,2))];

if (length(unique(train_targets)) == 2)
   test_targets = test_targets > 0;
end
```

Příloha 2 – Výpis algoritmu NDDF

```
function [test_targets ] = NDDF(train_patterns, train_targets, test_patterns, cost)

% Classify using the normal density discriminant function
% Inputs:
%       train_patterns  - Train patterns
%       train_targets   - Train targets
%       test_patterns   - Test  patterns
%       cost            - Cost for class 0 (Optional, Unused yet)
%
% Outputs
%       test_targets    - Predicted targets
%       g               - The discriminant function for the test examples
format long;
[d, L] = size(train_patterns);

%Estimate mean and covariance for each class
mu      = zeros(d,length(unique(train_targets)));
sigma   = zeros(d,d,length(unique(train_targets)));
p       = zeros(length(unique(train_targets)),1);

classes = unique(train_targets);
for i = 1:length(classes),
  indices       = find(train_targets == classes(i));
  mu(:,i)       = mean(train_patterns(:,indices)')';
  sigma(:,:,i)  = cov(train_patterns(:,indices)',1)';
  p(i)          = length(indices)/length(train_targets);
end

test_targets = zeros(1, size(test_patterns,2));
for i = 1:size(test_patterns,2),
   sample = test_patterns(:,i);
   for j = 1:length(classes),
      g(i,j) = -0.5*(sample - mu(:,j))'*inv(squeeze(sigma(:,:,j)))*(sample - mu(:,j)) - ...
                        d/2*log(2*pi)-0.5*log(det(squeeze(sigma(:,:,j))))+log(p(j));
   end
   [m, best]      = max(g(i,:));
   test_targets(i) = classes(best);
end
```

Příloha 3 – Výpis algoritmu Nearest_Neighbor

```
function test_targets = Nearest_Neighbor(train_patterns, train_targets, test_patterns, Knn)

% Classify using the Nearest neighbor algorithm
% Inputs:
%       train_patterns  - Train patterns
%       train_targets   - Train targets
%       test_patterns   - Test  patterns
%       Knn             - Number of nearest neighbors
%
% Outputs
%       test_targets    - Predicted targets

L = length(train_targets);
Uc = unique(train_targets);

if (L < Knn),
  error('You specified more neighbors than there are points.')
end

N = size(test_patterns, 2);
test_targets = zeros(1,N);
for i = 1:N,
   dist = sum((train_patterns - test_patterns(:,i)*ones(1,L)).^2);
   [m, indices] = sort(dist);              % m - nejmenší vzdálenost indices- pořadí vzoru

   n = hist(train_targets(indices(1:Knn)), Uc);

   [m, best]  = max(n);

   test_targets(i) = Uc(best);
end
```