VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

KONCEPTY STROJOVÉHO UČENÍ PRO KATEGORIZACI OBJEKTŮ V OBRAZU
MACHINE LEARNING CONCEPTS FOR CATEGORIZATION OF OBJECTS IN IMAGES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Marek Hubený**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Karel Horák, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**
Ústav automatizace a měřicí techniky

Student: Bc. Marek Hubený ***ID:*** 106478
Ročník: 2 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Koncepty strojového učení pro kategorizaci objektů v obrazu

POKYNY PRO VYPRACOVÁNÍ:

1. Detailně nastudujte a popište základní teorii strojového učení v počítačovém vidění, pozornost věnujte pojmům generativní a diskriminativní model.
2. Proveďte zevrubnou analýzu tři základních fází konceptu strojového učení tj. reprezentace (pixely nebo příznaky), učení (dávkové/inkrementální) a rozpoznávání.
3. Seznamte se s metodou bag-of-words pro příznakové rozpoznávání objektů a analyzujte vlastnosti.
4. Navrhněte a implementujte úlohu demonstrující princip vybrané metody strojového učení pro kategorizaci objektů v obrazu.
5. Funkci ověřte na dostatečně velké množině veřejně přístupných nebo vlastních dat získaných analýzou obrazu.

DOPORUČENÁ LITERATURA:

1. GONZALES, R.C., WOODS, R.R.: Digital image processing (3rd edition). Prentice-Hall, Inc., 2008. 954 pages. ISBN 978-0131687288.

2. YOUNG,I.T., GERBRANDS,J.J., VLIET,L.J.: Fundamentals of Image Processing. TU Delft, 1998. 113 pages. ISBN 9075691017.

3. SZELINSKI, R.: Computer Vision: Algorithms and Applications. Springer,2010. ISBN 978-1848829343.

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 15.5.2017

Vedoucí práce: Ing. Karel Horák, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

Abstrakt

Tato práce se zabývá problémem rozpoznávání objektů a scén pomocí nástrojů strojového učení a počítačového vidění. Před řešením tohoto problému byly prostudovány základní fáze konceptu strojového učení a statistické modely s důrazem na jejich rozdělení na diskriminativní a generativní metody. Dále byla prostudována a popsána metoda Bag-of-words a její modifikace.

V praktické části práce byla v prostředí Matlab vytvořena implementace metody Bag-of-words s SVM klasifikátorem a daný model byl prověřen na různých množinách veřejně dostupných obrazů.

Klíčová slova

Strojové učení, počítačové vidění, popis objektu, statistické modely, generativní modely, diskriminativní modely, SIFT, SURF, SVM, neuronová síť, boosting, skrytý Markovovův model, omezený Boltzmannův stroj, bag of words, bag of visual words, bag of features

Abstract

This work is focused on objects and scenes recognition using machine learning and computer vision tools. Before the solution of this problem has been studied basic phases of the machine learning concept and statistical models with accent on their division into discriminative and generative method. Further, the Bag-of-words method and its modification have been investigated and described.

In the practical part of this work, the implementation of the Bag-of-words method with the SVM classifier was created in the Matlab environment and the model was tested on various sets of publicly available images.

Keywords

Machine learning, computer vision, object description, statistical model, generative model, discriminative model, SIFT, SURF, SVM, neural network, boosting, hidden markov model, restricted Boltzmann machine, bag of word, bag of features, bag of visual words

Bibliografická citace:

HUBENÝ, M. *Koncepty strojového učení pro kategorizaci objektů v obrazu*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. 84s. Vedoucí práce: Ing. Karel Horák, Ph.D.

Prohlášení

„Prohlašuji, že svou závěrečnou práci na téma Koncepty strojového učení pro kategorizaci objektů v obrazu 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 závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné 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 části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne **15. května 2017** ……

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Karlu Horákovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne **15. května 2017** ……

podpis autora(-ky)

Obsah

Seznam obrázků

Seznam tabulek

1 ÚVOD

Strojové učení (angl. *machine learning*) je součástí oboru umělé inteligence, ve kterém jsou navrhovány algoritmy schopné se autonomně učit ze vstupních dat a v posledních dvou desetiletích patří k jednomu z nejprogresivněji se rozvíjejících oblastí IT. Ve spojení s počítačovým viděním a metodami pro rozpoznávání obrazu nachází stále větší uplatnění v různých oborech každodenního života, jako je doprava, medicína, robotika, kontrola kvality nebo zábavní průmysl.

Historie strojového učení se nejčastěji datuje od šedesátých let dvacátého století, kdy Alan Turing definoval tzv. Turingův test (1950). Dalším významným milníkem bylo potom navrhnutí první neuronové sítě pro počítače (perceptronu), které provedl Frank Rosenblatt v roce 1957. Od té doby docházelo k rozvoji tohoto oboru, vývoji nových, pokročilejších metod a jeho zařazování do komerční sféry. Až například po poražení člověka ve hře Go počítačem AlphaGo firmy Google, využitím kombinace strojového učení a *tree search* technik, v roce 2016.

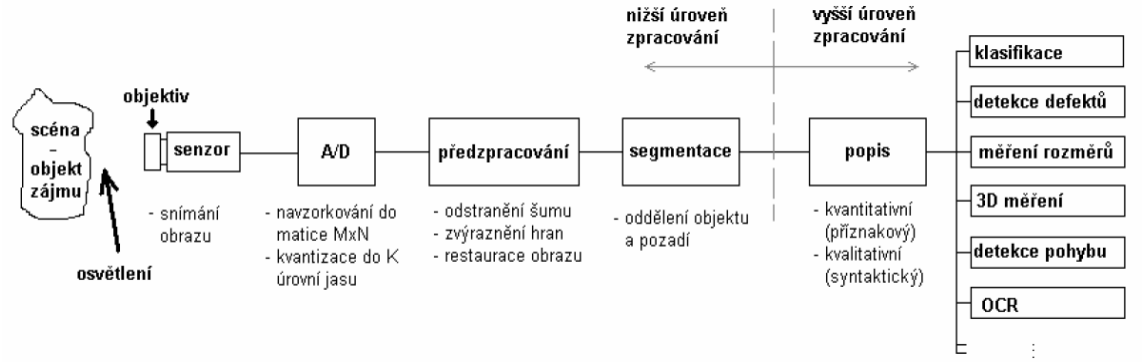
V této práci budou podrobněji probrány koncepty strojového učení, které se v počítačovém vidění využívají, a ukázáno, jak se obory strojového učení a počítačového vidění prolínají. Bude zde uvedeno, jaké metody se využívají pro předzpracování a popis dat získaných z obrazu (zejména *SIFT* a *SURF* deskriptory) a základní metody učení modelů. Velká část kapitoly bude věnována statistickým modelům a jejich rozdělení do dvou skupin – diskriminativních a generativních. Mezi diskriminativní statistické modely můžeme řadit například *Support vector machines,* umělé neuronové sítě, rozhodovací stromy, různé typy diskriminativní analýzy a logistické regrese nebo také *boosting* metody. Do generativních modelů pak spadají různé druhy *mixture* modelů, skryté Markovovy modely, naivní Bayesovský klasifikátor, Boltzmannův stroj a podobně. Na konci druhé kapitoly bude zmíněna oblíbená metoda *Bag of visual Words* (nebo také *Bag of Features*), které měla původně využití zejména v porozumění psanému textu, ale díky jejím vlastnostem se s výhodami používá také v počítačovém vidění.

V praktické části této práce bude provedena implementace metody *Bag of visual Words* a její modifikace, která využívá rozšíření příznakového prostoru pomocí tzv. spatial pyramid. Pro klasifikaci příznakových vektorů, získaných pomocí *Bag of visual Words* bude použito *Support vector machines*. Navržená implementace bude testována na veřejně přístupných množinách obrazů a bude proveden rozbor a vliv různých parametrů modelu na úspěšnost klasifikace.

2 TEORETICKÁ ČÁST

2.1 Koncept strojového učení v počítačovém vidění

Zpracování obrazové informace v počítačovém vidění lze znázornit jako řetězec návazností jednotlivých úkonů, vedoucí od samotného osvětlení či snímání požadované scény až po popis a klasifikaci objektů, měření, detekci pohybu či sledování pohyblivého objektu [3]. Tento řetězec je zobrazen níže (viz. Obrázek 1). Obor strojového učení se v počítačovém vidění využívá ve velké míře zejména kvůli proměnlivosti snímané scény a schopnosti těchto algoritmů zůstat vůči těmto změnám imunní.



Obrázek 1: Řetězec zpracování obrazu. Převzato z [3]

Koncepty strojového učení se v tomto řetězci zpracování obrazu objevují zejména při klasifikaci příznakově popsaných objektů, avšak využití v některých aplikacích nachází již v nižší úrovni. Příklady využití neuronových sítí, které je například možné naleznout literatuře [4] je:

- Předzpracování (Rekonstrukce, restaurace a vylepšování obrazu)
- Redukci (komprese) dat a extrakci příznaků
- Segmentaci obrazu (segmentace založená na pixelech, segmentace založená na lokálních vlastnostech obrazu)
- Rozpoznání objektů (rozpoznání založené přímo na pixelech obrazu, rozpoznání založené na příznacích)
- Porozumění obrazu
- Optimalizace (např. při párování stereo-obrazů)

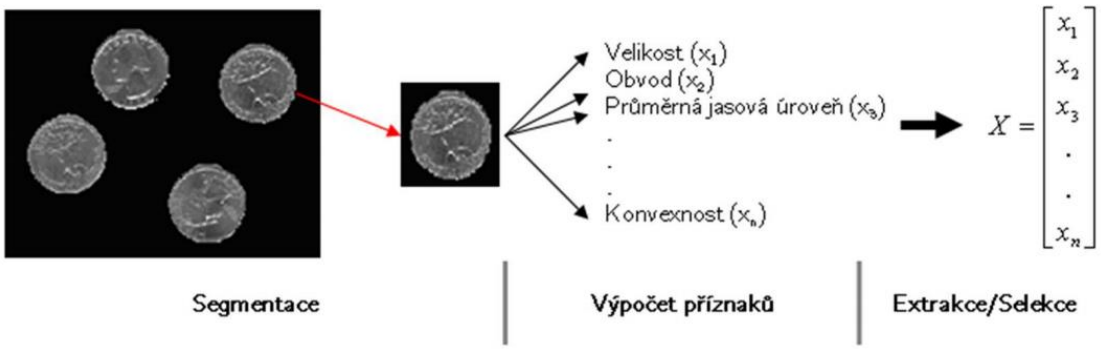
2.1.1 Popis objektů

Popisu objektu nejčastěji předchází předzpracování a segmentace obrazu, čímž je získána oblast zájmu, ve které se objekt nachází. Získanou část obrazu je poté nutné převést na vhodnou reprezentaci, která bude sloužit jako vstup do vyšší vrstvy zpracování (viz. Obrázek 1). K popisu objektu je možné použít dvou základních reprezentací:

- Reprezentace pixely
- Reprezentace příznaky

Zájmová oblast, získaná předzpracováním obrazu, je v případě reprezentace pixely popsána přímo hodnotami intenzit pixelů v této oblasti.

Při reprezentování obrazových dat příznaky, je ze zájmové oblasti obrazu vyextrahován tzv. příznakový vektor, který tuto oblast popisuje pomocí jejich charakteristických rysů [5]. Jedna z možností, jak vytvořit příznakový vektor, je uvedena na obrázku níže (viz. Obrázek 2).



Obrázek 2: Extrakce příznakového vektoru. Převzato z [5]

Na tyto příznaky jsou kladeny zvýšené nároky a měly by splňovat mnohé vlastnosti, mezi které patří [5]:

- Invariantnost
- Diskriminabilita
- Spolehlivost
- Efektivita výpočtu
- Časová invariance

Mezi deskriptory, jak nazýváme nástroje pro extrahování příznaků, můžeme řadit například [5]:

- Radiometrické deskriptory – Popis objektu pomocí jeho geometrických vlastností (kompaktnost, konvexnost, obvod, orientace, Eulerovo číslo,…)

- Fotometrické deskriptory – Popis objektu pomocí jeho optických vlastností (minimální, maximální a průměrná jasová úroveň, jasová disperze,…)
- Deskriptory založené na regionu – Příznakový vektor je vypočten podle hodnot pixelů objektu.
- Deskriptory založené na hranici – Příznak je vypočten podle hraniční reprezentace objektu (např. Freemanův kód, Fouriérovy deskriptory, B-spline deskriptory,…)

V praktické části této práce i v teoretickém rozboru metody „Bag of Words“ (viz. kapitola Bag of Words (BoW)), bude zmíněn SURF detektor a deskriptor. V následující části tedy bude popsána funkce jak jeho předchůdce, SIFT detektoru a deskriptoru, tak i SURF.

2.1.2 SIFT

SIFT (*Scale Invariant Feature Transform)* algoritmus sloužící pro detekci a popis lokálních příznaků v obrazu, která využívá DoG (*Difference of Gaussians*). Princip spočívá v popisu klíčových bodů a jejich okolí pomocí 128 - dimenzionálního vektoru na základě histogramu lokálně orientovaných gradientů [7].

SIFT detektor

Konvolucí Gaussovy funkce $G(x, y, \sigma)$se vstupním obrazem $I(x, y)$ vznikne škálový prostor obrazu (angl.. *scale-space*), který definujeme jako funkci $L(x, y, \sigma)$.

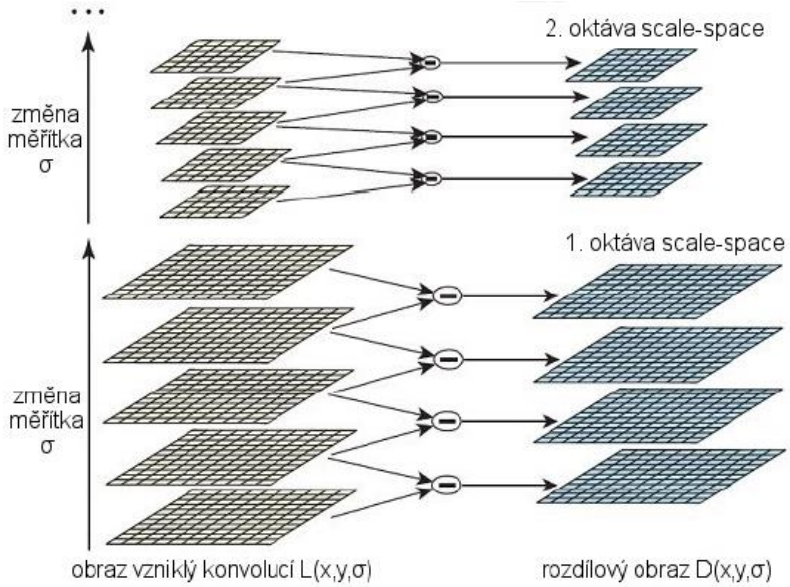
$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{1}$$

Kde $G(x, y, \sigma)$ je Gaussova funkce ve tvaru:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

Aby byly klíčové body ve škálovém prostoru stabilní, vybírají se pouze místa s maximy a minimy DoG funkce. Toho se dosahuje převzorkováním snímku (*resampling)* a stavění tzv. pyramidy, ve které má každá úroveň jinou vzorkovací frekvenci. DoG se následně vypočítá jako rozdíl dvou po sobě jdoucích škál Gaussovy funkce, které se liší konstantou *k* podle vztahu:

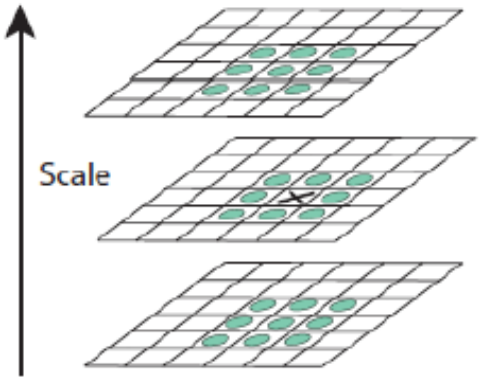
$$D(x,y,\sigma) = \big(G(x,y,k\sigma) - G(x,y,\sigma)\big) * I(x,y) \\ = L(x,y,k\sigma) - L(x,y,\sigma) \tag{3}$$



Obrázek 3: Měřítkově nezávislá forma – získání DoG. Převzato z [7]

Jak je znázorněno na Obrázek 3, snímky získané pomocí konvoluce Gaussových funkcí v různých škálách se rozdělují do oktáv, přičemž každá oktáva má oproti předchozí poloviční velikost. Přilehlé snímky se od sebe následně odčítají, čímž dostaneme DoG snímky (*Difference of Gaussians*). Obecně se doporučuje použít 4 oktávy po pěti snímcích. Dalším důležitým parametrem je pak σ, která se volí v intervalu 1 až 2 (nejčastěji $\sigma = 1{,}6$).

Lokální minima a maxima $D(x,y,\sigma)$ jsou získána porovnáváním každého bodu snímku s jeho sousedními body v aktuálním měřítku a v měřítku o jedno nižším a vyšším (viz. obrázek níže). Označení bodu se pak provede jen v případě, že je jeho hodnota menší, resp. větší, než hodnota všech ostatních.



Obrázek 4: Detekce lokálních extrémů. Převzato z [7]

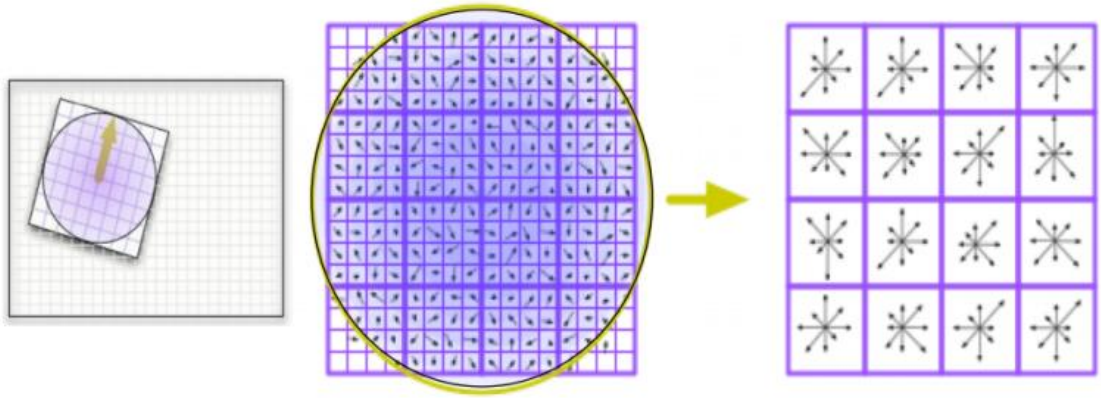
SIFT deskriptor

Pro body nalezené pomocí detektoru je nejprve nutné vypočítat jejich dominantní směr (jeden nebo více), díky čemuž bude možné v dalším kroku provést jejich popis, který bude invariantní vůči rotaci. Každý bod v n-pixelovém okolí vyšetřovaného bodu z prostoru $L(x, y)$ je podle vztahů:

$$m(x, y) = \sqrt{\big(L(x+1, y) - L(x-1, y)\big)^2 + \big(L(x, y+1) - L(x, y-1)\big)^2} \tag{4}$$

$$\theta(x, y) = \operatorname{atan}\big(L(x+1, y) - L(x-1, y)\big) / \big(L(x, y+1) - L(x, y-1)\big) \tag{5}$$

Kde $m(x, y)$ je amplituda a $\theta(x, y)$ je směr gradientu. Z těchto vypočítaných hodnot je následně sestaven z histogram okolí klíčového bodu, obsahujícího 36 binů, které pokrývají rozsah 360°. Vrcholy v tomto histogramu pak odpovídají dominantnímu směru lokálních gradientů



Obrázek 5: Výpočet SIFT deskriptoru klíčového bodu. Převzato z [6]

Jak bylo zmíněno výše, výsledný vektor, popisující klíčový bod nezávisle na jeho poloze, orientaci i měřítku má 128 položek. Ten se získá tak, že se okno o velikosti 16x16 pixelů okolo klíčového bodu rozdělí na 4x4 oblasti, přičemž z každé oblasti je sestaven histogram orientovaných gradientů (4x4x8 směrů = 128 položek).

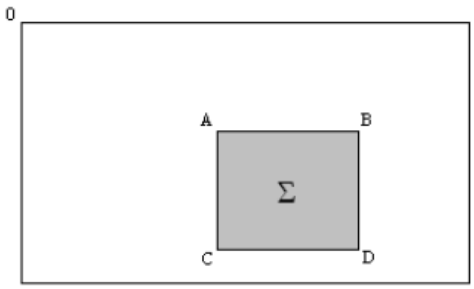
2.1.3 SURF

SURF (*Speed-Up Robust Features)* je robustní detektor i deskriptor, který byl motivovaný SIFT, ale byla u něj snaha snížit výpočetní náročnost a současně udržet spolehlivost a citlivost na detekci klíčových bodů. SURF pracuje s integrálním obrazem [8], ve kterém každý bod $I_\Sigma(x)$ obsahuje součet všech pixelů od počátku (levého horního rohu) a lze jej popsat jako:

$$I_\Sigma(\boldsymbol{x}) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i,j) \qquad (1)$$

Kde $\boldsymbol{x} = (x,y)^T$. Výhoda integrálního obrazu je v tom, že jakmile je vytvořen, stačí k výpočtu sumy intenzit na libovolné obdélníkové oblasti pouze tři operace a čtyři přístupy do paměti.

$$\Sigma = A - B - C + D \qquad (2)$$



Obrázek 6: Získání integrálního obrazu vymezeného body ABCD. Převzato z [8]

SURF detektor

Pro detekci významných bodů se používá matice $H(\boldsymbol{x}.\sigma)$, která se nazývá Hessián a pro bod $\boldsymbol{x} = (x,y)$**,** obraz I a škálu σ je definována jako:

$$H(\boldsymbol{x}.\sigma) = \begin{bmatrix} L_{xx}(\boldsymbol{x},\sigma) & L_{xy}(\boldsymbol{x},\sigma) \\ L_{xy}(\boldsymbol{x},\sigma) & L_{yy}(\boldsymbol{x},\sigma) \end{bmatrix} \qquad (3)$$

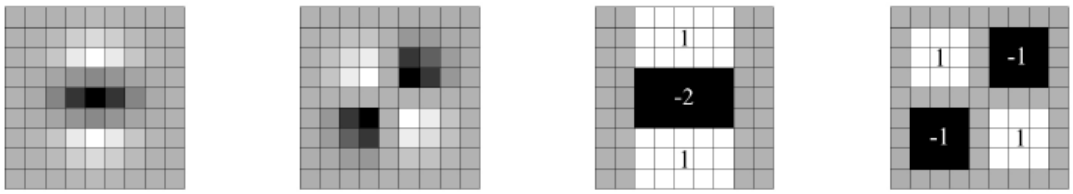
Kde $L_{xx}(\boldsymbol{x},\sigma)$ je konvoluce obrazu Is druhou derivací Gaussovy funkce$\frac{\partial^2}{\partial x^2} g(\sigma)$ v bodě $\boldsymbol{x}$ a podobně i $L_{xy}(\boldsymbol{x},\sigma)$ a $L_{yy}(\boldsymbol{x},\sigma)$.

Přesto že je Gaussova funkce optimální pro analýzu ve škálovém prostoru, obecnou nevýhodou detektorů založených na Hessiánu je jejich snížená

opakovatelnost pro rotaci snímků o liché násobky $\pi/4$ způsobená čtvercovým tvarem filtru.

Použití těchto reálných a neořezaných filtrů by nebylo optimální, proto autoři práce navrhli kvůli snížení výpočetních nároků tzv. čtvercové filtry , které druhou derivaci gaussovy funkce aproximují. (viz. pravá část Obrázek 7). Tyto filtry o velikosti 9x9 pixelů jsou aproximací Gaussianu s $\sigma = 1{,}2$ a reprezentují nejnižší škálu pro výpočet mapy odezev detektoru) Díky integrální reprezentaci obrazu a čtvercovým oblastím ve filtru je výpočetní čas nezávislý na velikosti jádra

Na následujícím obrázku jsou zobrazeny druhé parciální derivace gaussovy funkce (diskretizovaná a oříznutá) ve ve směru osy y- (L_{yy}) a xy-(L_{xy}) a používaná aproximace této derivace ve směru osy y- (D_{yy}) a xy-(D_{xy}). Šedé oblasti jsou rovny nule.



Obrázek 7: Čtvercové filtry. Převzato z [8]

Determinant aproximovaného Hessiánu pak představuje odezvu filtru na blob (*skvrnu*) v bodě x:

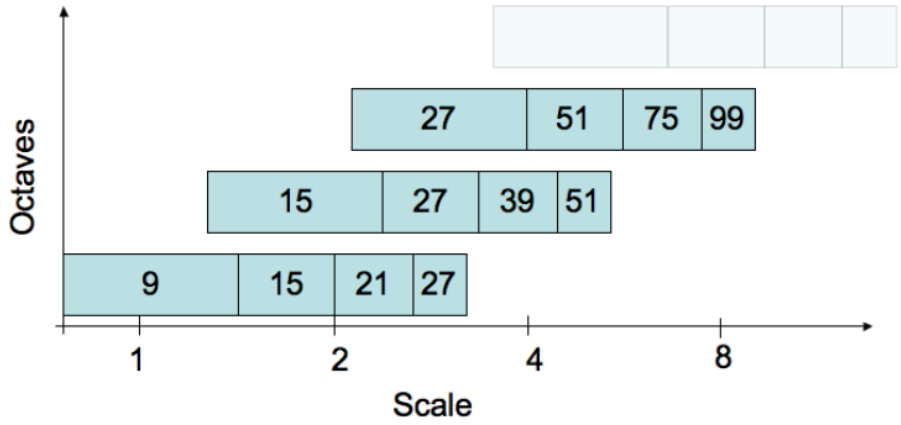
$$det(H_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \qquad (4)$$

$$w = \frac{|L_{xy}(1,2)|_F |D_{yy}(9)|_F}{|L_{yy}(1,2)|_F |D_{xy}(9)|_F} = 0{,}912 \ldots \cong 0{,}9 \qquad (5)$$

Kde w je relativní váha, která vyvažuje determinant Hessiánu, což je nutné vzhledem k použití aproximovaných Gaussových jader, a $|x|_F$ je Frobeniova norma.

Aby byla zachována invariance bodu vůči měřítku, je nutné použit stejně jako u metody SIFT více měřítek a vhodný měřítkový prostor. Díky využití výše zmíněných čtvercových filtrů a integrálního obrazu je ale vzhledem k rychlosti výpočtu vhodnější měnit velikost filtrů. První vrstva je výstup filtru 9x9 a pro další vrstvy první oktávy se volí velikosti filtrů 15x15, 21x21, 27x27. Další oktáva se následně počítá s filtrem 15x15 a rozdíly mezi filtry pro její další vrstvy jsou dvakrát větší, tudíž 27x27, 39x39 a 51x51. Jednotlivé oktávy se pak překrývají, čímž souvisle pokryjí všechna možná měřítka. Kromě výpočetní efektivity má tato metoda také

výhodu v tom, že nevzniká aliasing efekt, protože nedochází k podvzorkování obrazu.



Obrázek 8: Velikosti filtrů tří prvních oktáv. Převzato z [8]

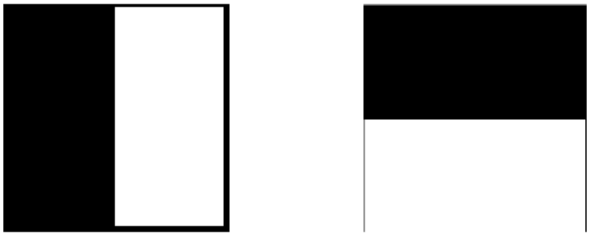
Při hledání klíčových bodů se pak postupuje následujícími kroky:

1) Odezvy s hodnotou menší než zvolený práh se odstraní. Snížením prahu zvýšíme počet významných bodů. Zvýšením prahu se počet významných bodů sníží, ale zůstanou jen ty nejsilnější (s největší hodnotou odezvy).
2) Vyhledání lokálních maxim v okolí každého obrazového bodu i jeho dvou sousedních úrovních zvětšení stejné oktávy měřítkového prostoru. Každý bod se porovnává se svými 8-mi sousedy a dalšími devíti sousedy v dalších vrstvách a pokud je z nich největší, je označen jako lokální maximum.
3) Interpolace dat získaných předchozími kroky do subpixelových pozic.

SURF deskriptor

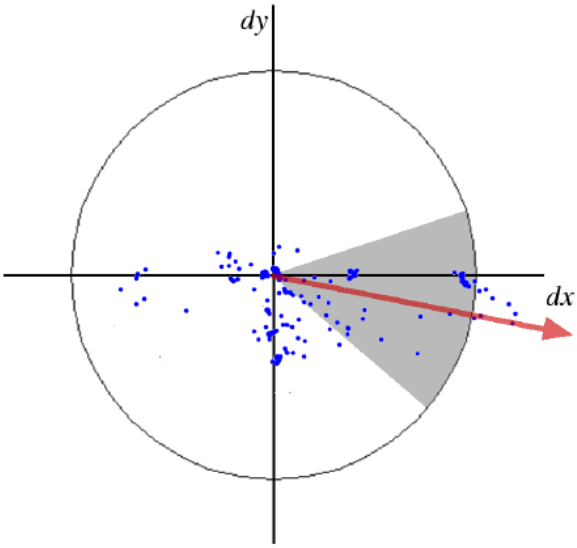
Podobně jako je tomu u SIFT deskriptoru, SURF deskriptor popisuje pomocí 64 - rozměrného vektoru hodnot to, jak jsou intenzity pixelů rozmístěné v okolí daného klíčového bod [8]. Pro zajištění reprodukovatelnosti a invariantnosti vůči rotaci je nutné získat orientaci klíčového bodu z kruhové oblasti okolo něj před extrahováním samotného deskriptoru. Deskriptor se poté extrahuje z čtvercové oblasti zarovnané s danou orientací.

Pro zjištění orientace klíčového bodu se používají jednoduché Haarovy vlnky, což jsou filtry, kterými je možné najít gradienty obrazu ve směru x i y. Vypočítáním odezvy Haarových vlnek o velikosti 4σ v okruhu 6σ okolo klíčového bodu pro danou kruhovou oblast vzorkovanou krokem o velikosti σ. Pro zrychlení výpočtu se používá již dříve získaný integrální obraz. Pro získání odezev filtrů ve směru x nebo y je potřeba pouhých 6 operací v jakémkoliv měřítku.



Obrázek 9: Haarovy vlnky. Převzato z [8]

Na obrázku výše (viz. Obrázek 9) jsou znázorněny Haarovy vlnky, používané k výpočtu odezvy ve směru x (vlevo) a ve směru y (vpravo). Tmavá část má hodnotu -1, světlá +1. Odezvy vlnek jsou váhovány gaussianem ($\sigma = 2s$) s centrem v klíčovém bodu, přičemž jednotlivé odezvy vlnek jsou zaneseny jako body do prostoru s vlastní hodnotou v horizontálním i vertikálním směru. Uvnitř takto získané kruhové oblasti (viz. Obrázek 10) rotujeme kruhovou výsečí s úhlem $\pi/3$ a v každé pozici výseče sečteme všechny uvnitř se nacházející odezvy, z nichž vytvoříme nový vektor. Z takto získaných vektorů se vybere ten největší, který odpovídá dominantní orientaci klíčového bodu.



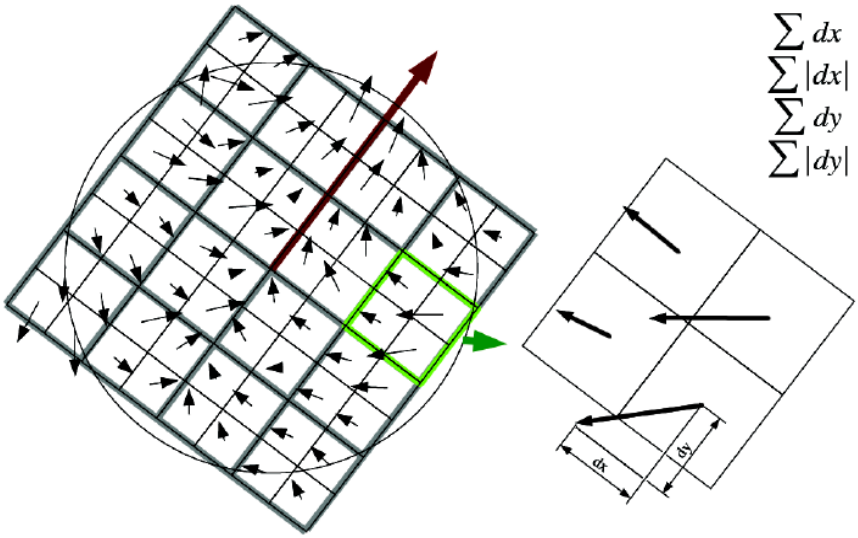
Obrázek 10:Určení orientace klíčového. Převzato z [8]

Pro určení orientace daného klíčového bodu se konstruuje samotný deskriptor. K tomuto účelu se sestaví čtvercové okno kolem klíčového bodu, které obsahuje pixely z nichž bude deskriptor vypočítán. Velikost tohoto okna se volí 20σ a je orientováno podle již vypočítané orientace klíčového bodu. Toto okno je rozděleno na 4x4 oblastí a v každém z nich je vypočítána odezva Haarových vlnek o

velikosti 2σ na 5x5 rovnoměrně rozdělených vzorkových bodů. Pro těchto 25 bodů se vytvoří uspořádaná čtveřice:

$$v = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right) \tag{6}$$

Kde d_x (resp. d_y) jsou odezvy Haarových vlnek ve směru x (resp. y). Každý region nám tedy dává čtveřici hodnot, což ve výsledku vede na konečnou velikost SURF vektoru 4x4x4=64.



Obrázek 11: Konstrukce SURF deskriptoru. Převzato z [8]

2.1.4 Učení

Metody strojového učení můžeme dělit například podle režimu učení na [9]:

- Dávkové (off-line učení) – Při tomto druhu učení jsou modelu postupně předkládána trénovací data a jeho modifikace nastává až po předložení všech trénovacích vzorů. Jednou z výhod tohoto druhu učení je rychlejší konvergence.
- Inkrementální – Inkrementální modely jsou modifikovány po průchodu každého vzoru, čímž stochasticky konverguje k lokálnímu minimu. Nezaručují tedy konvergenci ke globálnímu minimu, ale jejich výhodou je, že se dokáží „přiučit“, tj. upravit model bez nutnosti opětovně předložit všechna tréninková data.

Dalším možným dělením algoritmů učení, využívaného při strojovém učení, může být například [9]:

- učení s učitelem a učení bez učitele, zpětnovazebné (posilované) učení
- Asociativní a neasociativní učení, atd.

2.1.5 Rozpoznávání

Rozpoznávání (jinak také klasifikace) objektů spočívá v jejich zařazení do tříd (=podmnožina prvků se společnými rysy). Na základě této klasifikace je možné potom s objekty pracovat jako se samostatnými jednotkami, charakterizovaných třídou, do které náleží. Tyto třídy můžou reprezentovat například rozdělení dopravních prostředků na osobní, nákladní, motocykly, letadla, atd. [11]

Dělení metod rozpoznávání může být například [12]:

- Rozdílové metody – Jedná se o výpočet vzdálenosti příznakového vektoru vzoru od vyšetřovaného signálu pomocí některé z metrik (Hmmingova, Euklidova,…)
- Korelační metody – Vzájemný vztah dvou kvantitativních veličin je zjišťován pomocí korelační nebo regresní analýzy (Lineární regrese, diskriminační analýza, logistická regrese,…)[1]
- Analýza histogramu – Vzájemný vztah mezi vzorem a objektem je vyjádřen na základě jejich histogramů a jejich vzájemné vzdálenosti.
- Statistické metody – SVM, shluková analýza, RANSAC
- Syntaktické rozpoznávání – „*Úkolem syntaktického rozpoznávání objektu je určit, zda analyzovaný obraz odpovídá obrazům dané gramatiky, tj. zda gramatika může tento obraz generovat. Obraz je reprezentován řetězcem jazyka, který je generován danou gramatikou.*“ [13]
- Neuronové sítě
- Fuzzy rozpoznávací systémy

2.2 Diskriminativní modely

Máme-li objekt popsaný vektorem $\boldsymbol{x}$, který chceme klasifikovat do třídy y, pak diskriminativní model poskytuje takový proces klasifikování, při kterém je přímo určena náležitost do dané třídy $P(y|\boldsymbol{x})$ [10]. Není tedy nutná žádná jiná explicitní znalost $P(\boldsymbol{x})$, $P(y, \boldsymbol{x})$ ani $P(\boldsymbol{x}|y)$. Mezi nejčastěji používané diskriminativní modely patří například:

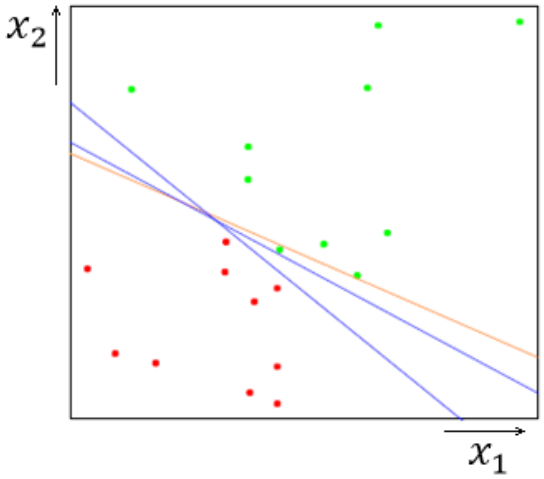
- *Support Vector Machines (SVM)*
- Umělé neuronové sítě (angl. *Artificial Neural Networks*)
- Rozhodovací stromy (angl. *Decision Trees*)

- *Logistic regression*
- *Discriminant analysis*
- *Boosting methods*, atd.

V této kapitole budou dále některé diskriminativní modely popsány podrobněji. Nejpodrobněji bude rozebrán *Support vector machines*, který je využit v praktické části této práce.

2.2.1 Support vector machines

Support vector machines (SVM) je algoritmus učení s učitelem, který je primárně určen pro klasifikaci dat do dvou skupin, tzn. binární klasifikátor. V takovém případě algoritmus hledá oddělující nadrovinu a k ní co nejširší možné hranice (angl. *margin*), aby prvky tříd ideálně oddělil [10]. Nejjednodušším případem jsou tzv. lineárně separabilní množiny, které se prvky tříd v příznakovém prostoru nepřekrývají. Tento případ je znázorněn na obrázku níže. Zelené body jsou prvky jedné třídy, červené body jsou prvky druhé třídy. Úkolem je najít takovou dělící nadrovinu, která prvky oddělí.



Obrázek 12: Dvě lineárně separabilní třídy. Převzato z [1]

Jak je patrné, prvky lze oddělit nekonečně mnoha způsoby (modré čáry). SVM hledá tuto nadrovinu pomocí maximalizování *margin*. Červeně je vyznačen výsledek získaný metodou nejmenších čtverců [1]

Lineárně separabilní případ

[2] Máme-li k dispozici trénovací množinu $T = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{N}$ s vektory příznaků $\boldsymbol{x_i} \in R^P$, které jsou rozděleny do dvou tříd $y_i \in \{-1,1\}$, pak je cílem najít optimální nadrovinu

$$\{x: f(x) = x^T\beta + \beta_0 = 0 \qquad (7)$$

tak, aby prvky co nejlépe rozdělila. Pokud mají vstupní vektory dva prvky, pak se jedná o hledání přímky, která oddělí body do dvou polorovin. Optimalizační problém je v tomto případě maximalizování *margin* M dělící nadroviny tak, aby všechny prvky trénovací množiny T ležely nejméně ve vzdálenosti M od nadroviny:

$$\max_{\beta,\beta_0,\|\beta\|=1} M \tag{8}$$

pro

$$y_i(x_i^T\beta + \beta_0) \geq M, i = 1, \dots, N \tag{9}$$

Pokud bude β a β_0 vyhovovat této nerovnici, zavedením normály $\|\beta\| = 1$ zajistíme, že jakýkoliv kladný násobek těchto proměnných jí bude vyhovovat také. Výraz upravíme:

$$\frac{1}{\|\beta\|} y_i(x_i^T\beta + \beta_0) \geq M \tag{10}$$

nebo ekvivalentně

$$y_i(x_i^T\beta + \beta_0) \geq M\|\beta\| \tag{11}$$

Normálu můžeme s výhodami zvolit jako $\|\beta\| = \frac{1}{M}$ a optimalizační problém hledání maximálního *marginu* převedeme na minimalizaci normály $\|\beta\|$.

$$\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 \tag{12}$$

Což odpovídá:

$$y_i(x_i^T\beta + \beta_0) \geq 1, i = 1, \dots, N \tag{13}$$

Jedná se o konvexní optimalizační problém – hledání nadroviny definované β a β_0 s prázdným místem (*margin*) o šířce $\frac{1}{\|\beta\|}$ okolo ní, který je možné řešit pomocí Lagrangeových multiplikátorů $\alpha_i, i = 1, \dots, N$. Primární Lagrangeova funkce je ve tvaru [1]:

$$L_P = \frac{1}{2}\|\beta\|^2 - \sum_{i=1}^{N} \alpha_i[y_i(x_i^T\beta + \beta_0) - 1] \tag{14}$$

Rovnice vede na úlohu konvexního kvadratického programování, jehož řešením za podmínek [2] $\frac{\partial}{\partial \alpha_i} L_P = 0, \alpha_i \geq 0$ pro primární a $\frac{\partial}{\partial \beta_k} L_P = 0$, $\frac{\partial}{\partial \beta_0} L_P = 0$, $\alpha_i \geq 0$ pro duální úlohu dostaneme:

$$\beta = \sum_{I=1}^{N} \alpha_i y_i x_i \tag{15}$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i \tag{16}$$

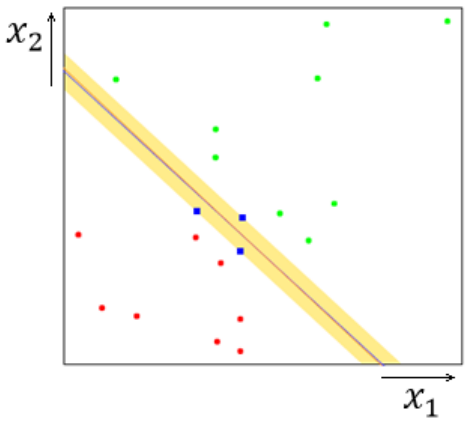
a dosazením do primární Lagrangeovy funkce dostanem duální (Wolfe) funkci [1]:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k \tag{17}$$

„V takto formulované úloze odpovídá každému trénovacímu příkladu $(\boldsymbol{x}_i, y_i)$ *jeden Lagrangeův multiplikátor* α_i*. Ty vektory* $\boldsymbol{x}_i$*, pro které* $\alpha_i > 0$*, se nazývají podpůrné vektory a leží na jedné z nadrovin* H_1 *nebo* H_2*. Tyto podpůrné vektory leží nejblíže rozhodovací nadrovině* H *a jsou nezbytnou součástí trénovací množiny* T*. Pokud by veškeré ostatní trénovací příklady (pro které* $\alpha_i = 0$*) byly z množiny* T *odstraněny, stéle by řešením optimalizačního problému byla táž nadrovina* H*“* [2].

Vzhledem k tomu, že oddělující nadrovina je definována jako funkce $\hat{f}(x) = \boldsymbol{x}^T \hat{\beta} + \hat{\beta}_0$, je možné klasifikovat nový prvek prostou funkcí signum:

$$\hat{G}(x) = \text{sing}\, \hat{f}(x) \tag{18}$$

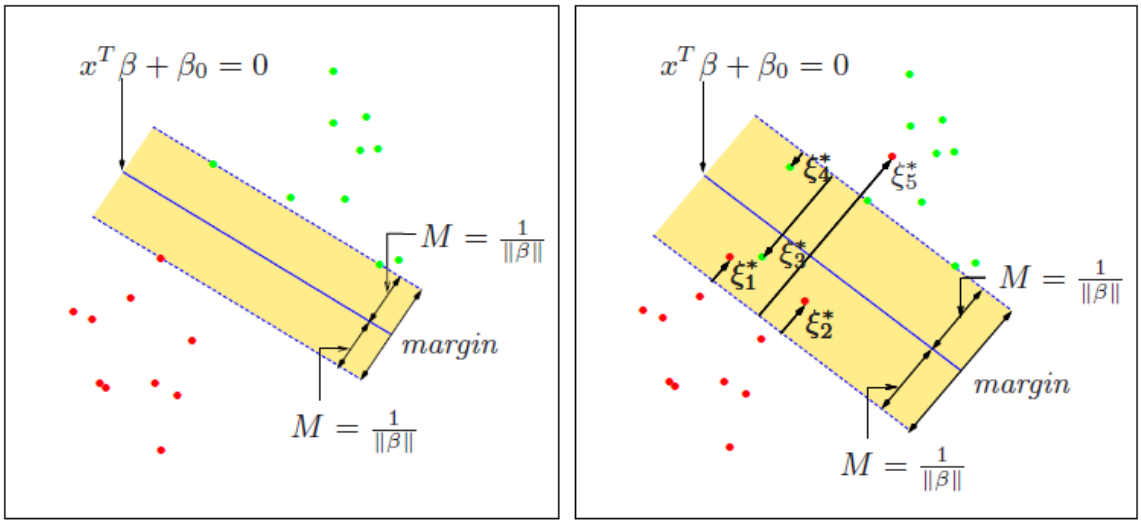


Obrázek 13: Data separovaná nadrovinou a největším možným marginem. Převzato z [1]

Na obrázku jsou znázorněna data separovaná nadrovinou (modrá čára) a největším možným marginem (žlutá plocha). Modře jsou vyznačeny body ležící na hranici marginu (support points). Červená čára je výsledek logistické regrese.

Lineárně neseparabilní případ

Výše uvedený způsob oddělení dvou tříd pomocí nadroviny je vhodný v případě, pokud se prvky nepřekrývají, tzn. lze je všechny perfektně oddělit. Máme-li dvě lineárně neseparabilní množiny, duální Lagrangeova funkce L_D poroste nade všechny meze (diverguje), proto se zavádí tzv. *slack proměnné* ξ_j, určující míru chybného zařazení prvku vzhledem k *margin* oddělující nadroviny H (viz. obrázek). Body označené jako ξ_j^* jsou na špatné straně hranice *margin* s mírou $\xi_j^* = M\xi_j$ (body na správné straně hranice mají $\xi_j^* = 0$).



Obrázek 14: Lineárně separabilní (vlevo) a neseparabilní (vpravo) případ. Oddělující nadrovina je vyznačena modrou čarou. Převzato z [1].

Při hledání oddělující nadroviny v příznakovém prostoru, ve kterém dochází k přesahu bodů je podle [1] možné přistupovat tak, že budeme stále maximalizovat margin M, ale úpravou původních rovnic dovolíme některým bodům ležet na špatné straně této hranice. Existují dvě možnosti:

- $y_i(x_i^t\beta + \beta_0) \geq M - \xi_i$
  - Vzdálenost přesahujícího bodu je určena v jako reálná vzdálenost od *marginu*
  - Vede na nonkonvexní optimalizační problém
- $y_i(x_i^t\beta + \beta_0) \geq M(1 - \xi_i)$
  - Vzdálenost přesahujícího bodu je určena jako relativní vzdálenost (násobení M)
  - Vede na konvexní optimalizační problém

- Vede na „standardní“ standardní *Support Vector* klasifikátor, proto se bude v následujícím textu pracovat s touto nerovností

Za podmínek $\xi_i \geq 0, \sum_{i=1}^{N} \xi_i \leq konst.$ pro $\forall i$. Hodnota ξ_i určuje proporcionální míru chybné klasifikace při predikci $f(x_i) = x_i^T \beta + \beta_0$. K tomu dochází když $\xi_i > 1$. Druhá podmínka slouží k omezení chybných klasifikací a cílem modifikované kriteriální funkce bude tuto horní mez minimalizovat:

$$\min_{\beta,\beta_0} \left[\frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{N} \xi_i \right] \tag{19}$$

při dodržení podmínek

$$\{\xi_i \geq 0, y_i(x_i^t \beta + \beta_0) \geq 1 - \xi_i \; \forall i \tag{20}$$

Parametr C nahradil konstantu určující maximální počet chybných klasifikací (pro separabilní případ je $C = \infty$). Po této úpravě je primární Lagrangeova funkce minimalizující $\beta, \beta_0 \; a \; \xi_i$:

$$\begin{aligned} L_P = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^{N} \xi_i \\ - \sum_{i=1}^{N} \alpha_i [y_i(x_i^T \beta + \beta_0) - (1 - \xi_i)] \\ - \sum_{i=1}^{N} \mu_i \xi_i \end{aligned} \tag{21}$$

Minimalizací primární Lagrangeovy funkce podle $\beta, \beta_0 \; a \; \xi_i$a položením příslušných proměnných nule dostaneme [1]:

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i, \tag{22}$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i, \tag{23}$$

$$\alpha_i = C - \mu_i, \forall i \tag{24}$$

Uvažujeme-li omezení $\alpha_i, \; \mu_i, \xi_i \geq 0$ pro $\forall i$, dostaneme dosazením rovnic výše do rovnice pro primární Lagrangeovu funkci duální Lagrangeovu funkci. Ta díky výše

uvedenému tvaru při zavedení *slack* proměnných neobsahuje ani hodnoty slack proměnných ξ_i, ani Lagrangeovy multiplikátory μ_i odpovídající těmto proměnným [2].

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} x_i^T x_{i'} \tag{25}$$

Maximalizováním L_D při dodržení podmínek $0 \leq \alpha_i \leq C$, $\sum_{i=1}^{N} \alpha_i y_i = 0$ dostaneme nejmenší hranice (*bound*) odpovídající zadané kriteriální funkci $\min_{\beta,\beta_0} \left[\frac{1}{2}\|\beta\|^2 + C \sum_{i=1}^{N} \xi_i\right]$ a omezením, vyplívajícím z Karush-Kuhn-Tuckerových podmínek:

$$\alpha_i [y_i (x_i^T \beta + \beta_0) - (1 - \xi_i)] = 0 \tag{26}$$

$$\mu_i \xi_i = 0 \tag{27}$$

$$y_i (x_i^T \beta + \beta_0) - (1 - \xi_i) \geq 0 \tag{28}$$

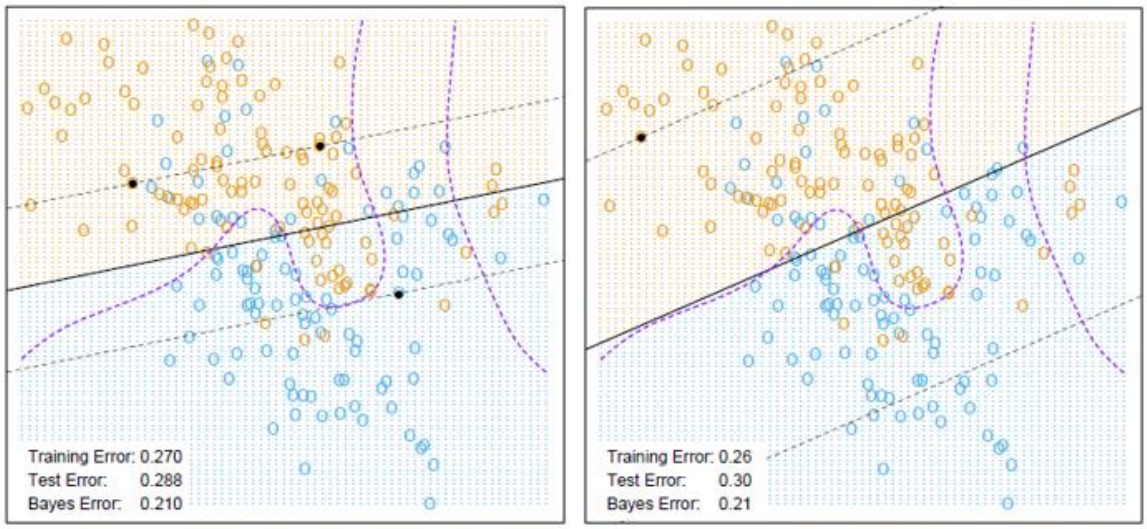
Obdržíme pro $i = 1 \ldots N$ normálu β definující nadrovinu jako:

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i \tag{29}$$

Pozorování i, které přesně splňují $y_i (x_i^T \beta + \beta_0) - (1 - \xi_i) \geq 0$ pro nenulové α_i, nazýváme podpůrné vektory (*support vectors)*.
Rozhodovací funkce je obdobně jako pro lineárně separabilní případ dána funkcí:

$$\hat{G}(x) = sign\left[\hat{f}(x)\right] = sing\left[x^T \hat{\beta} + \hat{b}_0\right] \tag{30}$$

V úvodu byl jako součást kriteriální funkce zaveden parametr C, který nahradil konstantu určující maximální počet chybných klasifikací. Na následujícím obrázku je znázorněn vliv tohoto parametru na adaptaci oddělující nadroviny bodů trénovací množiny. Vyšší hodnota má za následek to, že špatně klasifikovaným bodům je přidělena vyšší váha chyb a tudíž se oddělující nadrovina lépe adaptuje. To má za následek tzv. *přetrénování* klasifikátoru. Pro volbu parametru C se doporučuje používat metodu křížové validace.



Obrázek 15: Vliv parametru C stanovení hranic dvou lineárně neseparabilních tříd. Převzato z [1].

Na obrázku výše je znázorněn vliv parametru C stanovení hranic dvou lineárně neseparabilních tříd. Vlevo C=10000, vpravo C=0,01.Černé přerušované čáry označují margin, kde $f(x) = \pm 1$. Černé tečky označují body ležící přímo na margin ($\xi_i = 0, \alpha_1 = 0$). Fialová přerušovaná čára je Bayesovská rozhodovací hranice.

Jádrové funkce a nelineární separace

Klasifikátor popisovaný v předchozích kapitolách se v literatuře nazývá *Suppor Vector classifier* a jeho hlavními rysy jsou právě lineární hranice (*boundaries*) ve vstupním příznakovém prostoru – vektory trénovací množiny se vyskytují ve skalárním součinu. *Support Vector Machines (SVM)* klasifikátor je jeho dalším rozšířením, kde je dimenze příznakového prostoru mnohonásobně zvětšena (v některých případech až k nekonečnu [1]) s předpokladem, že původně lineárně neseparabilní data budou v tomto zvětšeném prostoru opět lineárně separabilní. Proto se zde na místo vektoru x ve vztahu $f(x) = x^T\beta + \beta_0$ zavádí tzv. bázová funkce (*basis function)* $h_m(x), m = 1, \dots, M$. Postup je potom stejný, rozdíl je jen v použití bázových funkcí na vstupní příznaky $h(x_i) = \big(h_1(x_i), h_2(x_i), \dots, h_M(x_i)\big), i = 1, \dots, N$ a výsledkem bude nelineární funkce $\hat{f}(x) = h(x)^T\hat{\beta} + \hat{\beta}_0$. Rozhodovací funkce pro klasifikaci zůstane stejná: $\hat{G}(x) = sign\big[\hat{f}(x)\big]$. Řešení optimalizačního problému definovaného primární Lagrangeovou funkcí je možné počítat přímo přes transformovaný příznakový vektor $h(x_i)$ a při různých volbách h je řešení možné obdržet snadno [1].
Duální Lagrangeova funkce bude:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle \qquad (31)$$

Z předchozího řešení jsme obdrželi vztah $\beta = \sum_{i=1}^{N} \alpha_i y_i x_i$, takže v tomto případě bude řešení pro $f(x)$:

$$\begin{aligned} f(x) &= h(x)^T \beta + \beta_0 \\ &= \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0 \end{aligned} \qquad (32)$$

V obou předchozích vztazích se objevuje $h(x)$ pouze jako vnitřní produkt a jeho transformaci vůbec nemusíme znát. Pro výpočet tohoto vnitřního produktu v transformovaném prostoru potřebujeme pouze tzv. jádrové (*kernel*) funkce $K(x, x')$:

$$K(x, x') = \langle h(x), h(x') \rangle \qquad (33)$$

Funkce K by měla být pozitivní a (semi-) definitní. Mezi nejčastěji používané jádrové funkce pro SVM patří:

- *Polynomální d-tého řádu -* $K(x, x') = (1 + \langle x, x' \rangle)^d$
- *Radiální bázová funkce -* $K(x, x') = exp(-\gamma \|x - x'\|^2)$
- *Neuronová síť -* $K(x, x') = tanh(\kappa_1 \langle x, x' \rangle + \kappa_2)$

Pro objasnění jádrových funkcí si uveďme příklad výpočtu pro příznakový prostor se dvěma vstupy X_1 a X_2. Chceme použít polynomiální jádro druhého řádu:

$$\begin{aligned} &K(X, X') = (1 + \langle X, X' \rangle)^2 \\ &= (1 + X_1 X_1' + X_2 X_2')^2 \\ &= 1 + 2X_1 X_1' + X_2 X_2' + (X_1 X_1')^2 + (X_2 X_2')^2 + 2X_1 X_1' X_2 X_2' \end{aligned}$$

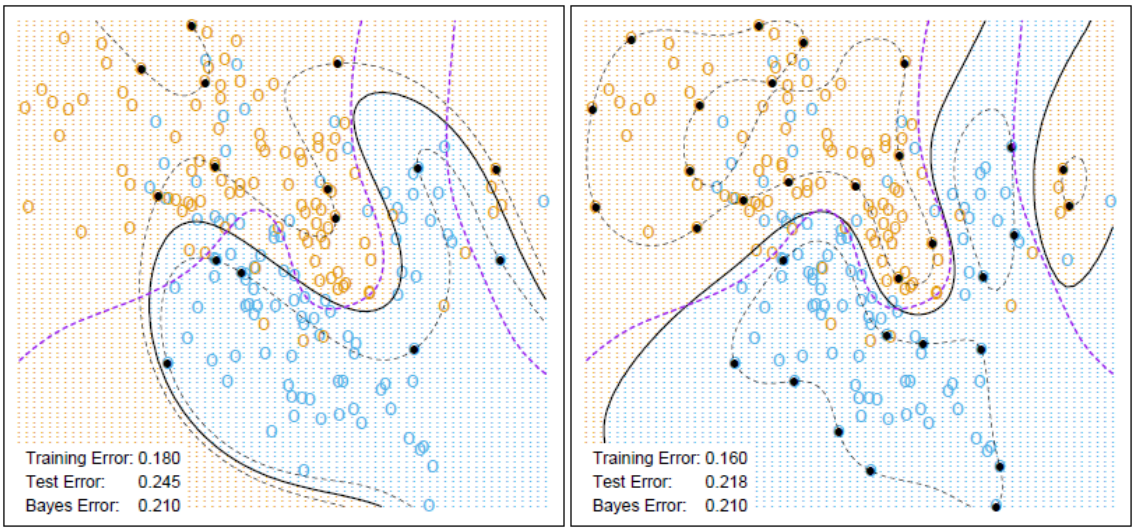
V tomto případě je tedy $M = 6$ a jednotlivé basis funkce:

$$\begin{aligned} h_1(X) &= 1 \\ h_2(X) &= \sqrt{2} X_1 \\ h_3(X) &= \sqrt{2} X_2 \\ h_4(X) &= X_1^2 \\ h_5(X) &= X_2^2 \\ h_6(X) &= \sqrt{2} X_1 X_2 \end{aligned}$$

Potom je $K(X, X') = \langle h(X), h(X') \rangle$ a funkce $\hat{f}(x)$ můžeme zapsat ve tvaru:

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{\alpha}_i y_i K(x, x_i) + \hat{\beta}_0$$

Stejně jako u lineární separace, i zde hraje parametr C, použitý v optimalizační funkci, důležitou roli. Vzhledem k tomu, že je příznakový prostor uměle zvětšen, je dokonalá separace často dosažitelná. Vysoká hodnota C způsobí dokonalejší separaci a vede k tomu, že hranice v původním příznakovém prostoru budou více „pokroucené“. Naproti tomu nízká hodnota C bude zmenšovat $\|\beta\|$, což bude mít za následek „hladší“ hranice.



Obrázek 16: Vliv jádra SVM. Převzato z [1].

Na obrázku výše jsou zobrazena dvě nelineární SVM, které byly trénovány na stejná data -vlevo je použito polynomiální jádro 4-tého řádu, vpravo radiální báze ($\gamma = 1$). $C = 1$. Fialová přerušovaná čára je Bayesovská rozhodovací hranice

Klasifikace do více tříd pomocí binárních klasifikátorů

Pro trénování klasifikátorů do více tříd existují tři základní postupy: *One-against-all, One-against-one* a *Orientovaný acyklický graf.* V principu je takovýto druh klasifikací založen na využití více binárních klasifikátorů. Jak je uvedeno v [2], při statistickém porovnání kvality klasifikace těchto tří přístupů na různých datových množinách, neexistují významnější rozdíly. Doporučuje se však využít metod *One-against-one* nebo *Orientovaného acyklického grafu*, které při trénování klasifikátoru nemají tak velkou výpočetní náročnost a navíc je natrénování dílčích binárních klasifikátorů dosaženo při menším počtu trénovačích prvků.
Pro klasifikaci do k tříd předpokládejme trénovací množinu T, složenou z dvojic $(x_i, y_i), i = 1 \dots l, x_i \in R^n, y_i \in \{1 \dots k\}$.

- *One-against-all* – k natrénovaných binárních klasifikátorů diskriminují příznakové vektory dané třídy oproti zbývajícím. Výsledkem je k rozhodovacích funkcí, ze kterých je vybrána třída nejvzdálenější

nadrovině $H^{(c)}, c = 1 \dots k$, která klasifikuje bod $\boldsymbol{x}$.

$$\hat{y} = \arg\max_{c=1\dots k} \left[\sum_{i=1}^{l} \alpha_i^{(c)} y_i^{(c)} K(x_i, x) + b^{(c)} \right] \tag{34}$$

Kde

$$y_i^{(c)} = \begin{cases} 1 & pro\ y_i = c \\ -1 & jinak \end{cases} \tag{35}$$

$\alpha_i^{(c)}$ a $b^{(c)}$ jsou parametry jednotlivých binárních klasifikátorů.

- *One-against-one* – Oproti předchozímu postupu je v tomto případě trénováno $\frac{k(k-1)}{2}$ klasifikátorů a každý pak diskriminuje vždy dvě třídy navzájem. Máme-li klasifikátor diskriminující třídy *m* a *n*, pak se při jeho trénování vybírají z trénovací množiny T pouze prvky $y_i \in \{m, n\}, i = 1 \dots l$. Pro výběr náležitosti prvku x do cílové třídy se používá hlasovací strategie, která spočívá v přičítání hlasů pro třídu, kterou predikují jednotlivé klasifikátory a po predikci všemi $\frac{k(k-1)}{2}$ klasifikátory je **x** přiřazeno do třídy s nejvíce hlasy.
- *Orientovaný acyklický graf* – Stejně jako v *One-against-one* je trénováno $\frac{k(k-1)}{2}$ klasifikátorů, ale ve fázi predikce se pro snížení výpočetní náročnosti sestaví klasifikátory do struktury orientovaného acyklického grafu, který má $\frac{k(k-1)}{2}$ uzlů (odpovídajících klasifikátorům) a *k* listových uzlů (odpovídajících třídám). Při predikci prvku **x** se tedy využijí pouze klasifikátory, které leží na cestě z kořene do uzlu třídy náležící tomuto prvku.

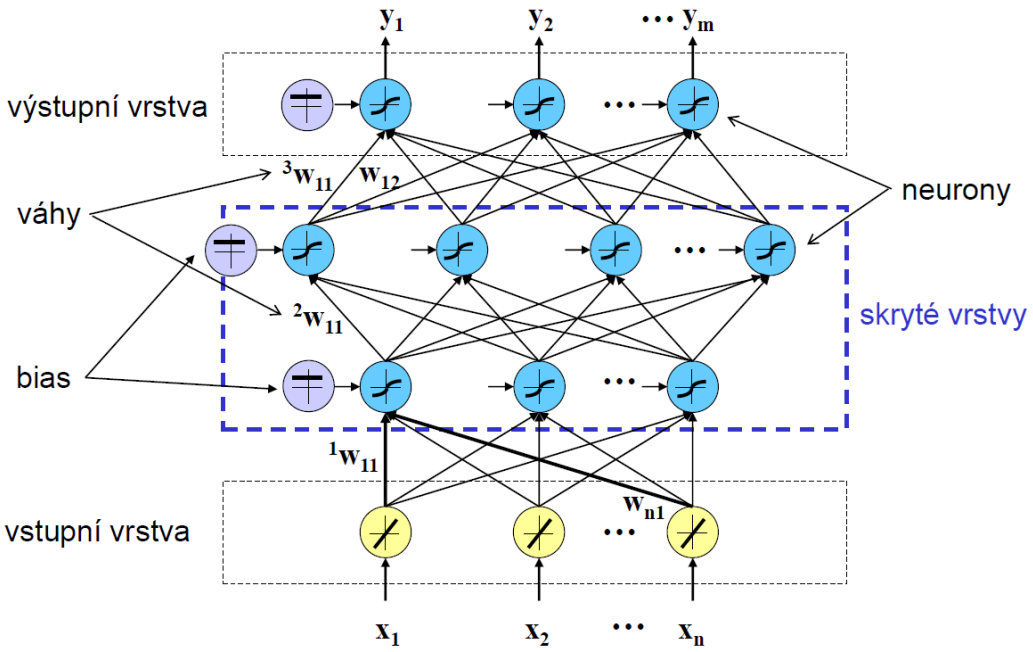
2.2.2 Neuronová síť

Umělé neuronové sítě (UNN) vycházení z biologické neuronové sítě v mozku živých tvorů. Jedná se síť navzájem propojených jednoduchých procesů (neuronů), které mají své vstupy (v biologii dendrity) a výstupy.

Vícevrstvá neuronová síť

Vícevrstvá neuronová síť je jednou z nejčastějších topologií neronových sítí [14]. Síť je tvořena minimálně třemi vrstvami –vstupní, jednou nebo více skrytých a výstupní

vrstvou (viz. Obrázek 17). Dvě sousední vrstvy jsou propojeny tak, že každý neuron z vrstvy nižší je propojen se všemi nurony vrstvy vyšší.



Obrázek 17: Topologie vícevrstvé neuronové sítě. Převzato z [14]

Jak vyplývá z uvedené topologie, šíření signálu je dopředné (feedforward) a probíhá následujícím způsobem:

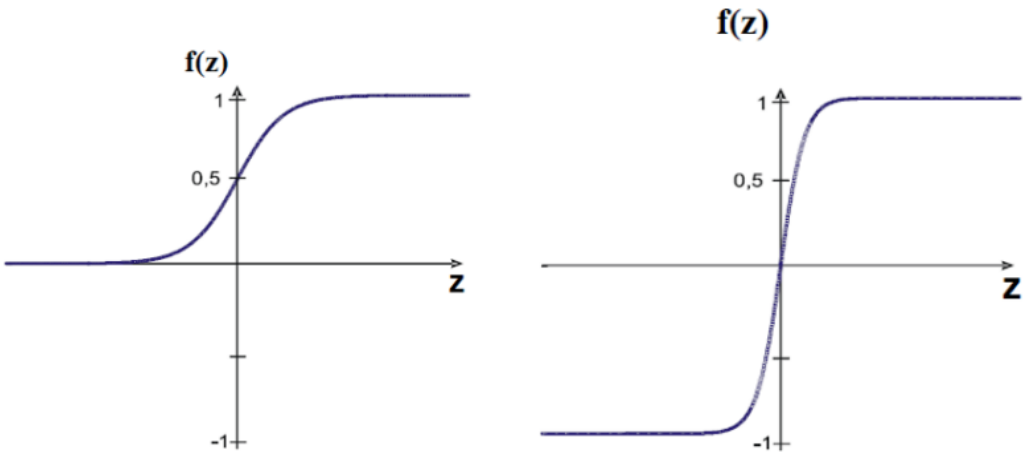
- Na neurony ve vstupní vrstvě je přiveden signál, čímž dojde k jejich exitaci na odpovídající úroveň.
- Pomocí vazeb mezi neurony je signál šířen do následující vrstvy. Signál je pomocí sinaptických vah zesílen nebo zeslaben a přiveden k jednotlivým neuronům v následující vrstvě. Tento proces se opakuje přes všechny ostatní vrstvy až k vrstvě výstupní.
- Výsledek odpovídá excitačním stavům neuronů ve výstupní vrstvě.

Neurony v tomto typu sítě používají přenosové funkce, které musí být spojité a derivovatelné (viz. Obrázek 18). Tyto podmínky splňuje například sigmoida, jejíž přenosová funkce $f(z)$ je ve tvaru:

$$f(z) = \frac{1}{1 + e^{-\sigma z}} \tag{36}$$

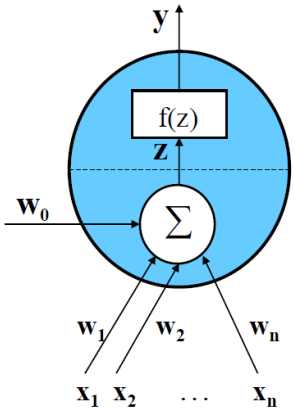
nebo hyperbolická tangenta:

$$f(z) = \frac{1 - e^{-\sigma z}}{1 + e^{-\sigma z}} \tag{37}$$



Obrázek 18: Sigmoida a hyperbolická tangenta. Převzato z [14]

Koeficient σ je strmost funkce (obvykle se volí pevně předem, ale existují i algoritmy učení, které strmost v průběhu upravují) a z je vážená suma vstupů neuronu (viz. obrázek).



Obrázek 19: Neuron. Převzato z [14]

Přímý výpočet hodnot synaptických vah (popř. i strmostí přenosových funkcí, biasů,…) je pro tento typ sítě nevhodný, proto se zde používá tzv. zpětného šíření chyby (angl.. *back-propagation*). Při tomto typu učení se nastavení vah provádí iterativním způsobem tak, že síti je předložena sada vzorů a poté jsou zpětným šířením chyby upraveny hodnoty synaptických vah tak, aby byla minimalizována chyba výstupu sítě.

Tréninková množina je vektor uspořádaných dvojic:

$$T = \{[^1X, {}^1D], [^2X, {}^2D], \dots, [^pX, {}^pD]\} \tag{38}$$

Kde $[{}^sX, {}^sD]$ je s-tý vzor, ve kterém je ${}^sX = [{}^sx_1, {}^sx_2, \dots, {}^sx_n]$ vektor vstupů a ${}^sD = [{}^sd_1, {}^sd_2, \dots, {}^sd_m]$ vektor požadovaných výstupních hodnot. n je pak počet vstupních neuronů a m počet neuronů ve výstupní vrstvě sítě.

Princip metody back-propagation spočívá ve snižování odchylky výstupů sítě od požadovaných hodnot pro danou tréninkovou množinu. Pro výpočet celkové chyby E_c je nutné počítat chyba sítě E_s vzhledem k s-tému tréninkovému vzoru podle vzorce:

$$E_s = \frac{1}{2}\sum_{j=1}^{m}\left(d_{sj} - y_{sj}\right)^2 \tag{39}$$

Celkovou chybu sítě E_c je pak možné vypočítat jako sumu dílčích chyb E_s jednotlivých vzorů:

$$E_c = \sum_{s=1}^{p} E_s = \frac{1}{2}\sum_{s=1}^{p}\sum_{j=1}^{m}\left(d_{sj} - y_{sj}\right)^2 \tag{40}$$

Na základě celkové chyby E_c je nutné hledat takové nastavení vah w, které bude tuto chybu minimalizovat a váhy upravit podle:

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk}(t+1) \tag{41}$$

Adaptování váhy mezi neurony j a k je možné spočítat jako parciální derivaci celkové chyby podle aktuálního nastavení váhy. α je koeficient učení.

$$\Delta w_{jk} = -\alpha \frac{\partial E_c}{\partial w_{jk}} \tag{42}$$

Pokud máme k dispozici sadu trénovacích dat $T = \{[{}^1X, {}^1D], [{}^2X, {}^2D], \dots, [{}^pX, {}^pD]\}$ kde vektory ${}^SX$ představují vstupní hodnoty a vektor ${}^SD$ jsou požadované výstupní hodnoty, pak je možné algoritmus back-propagation popsat v následujících krocích:

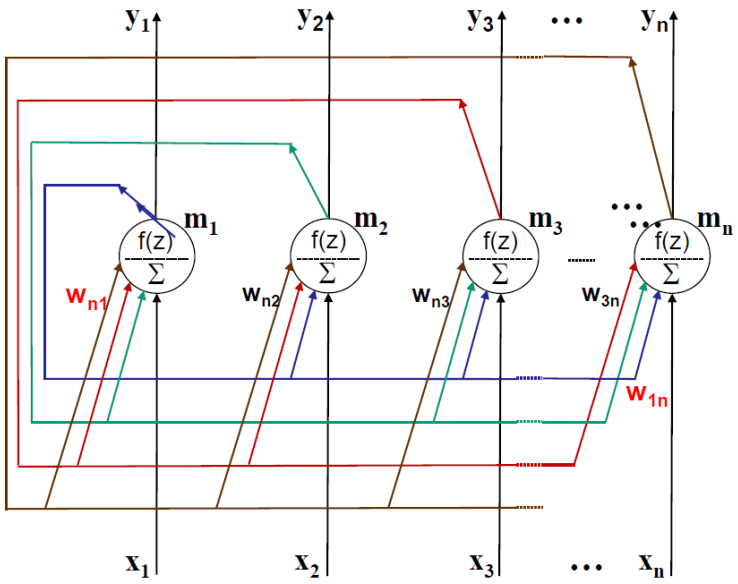
1. Inicializace algoritmu
    a. volba strmosti σ
    b. nastavení vah na náhodné hodnoty blízké nule (např.: $-0{,}3 < w_{ij} < 0{,}3$)
    c. volba koeficientu učení α
    d. volba biasu w_0
    e. stanovení ukončovací podmínky E_{END}

2. Předložení vzoru síti
3. Výpočet výstupu sítě dopředným šířením vstupních hodnot
4. Výpočet chyby sítě vzhledem k aktuálnímu tréninkovému vzoru E_s
5. Pokud byly síti předloženy všechny vzory z tréninkové množiny T, pokračovat na krok 6, jinak zpět na krok 2
6. Výpočet celkové chyby sítě E_c a pokud je menší než zvolená ukončovací hodnota E_{END}, pak ukončit algoritmus učení. Jinak krok 7
7. Adaptace vah
    a. Výpočet Δw_{ij}
    b. Aktualizace hodnoty váhy $w_{ij}(t+1) = w(t) + \Delta w_{ij}(t+1)$ a po aktualizaci všech vah znovu začít předkládat vzory krokem 2

Vybavování naučené neuronové sítě při předložená neznámého vzoru X pak probíhá pouze pomocí jednoho průchodu.

Hopfieldova neuronová síť

Hopfieldova neuronová síť (HNN) je speciální případ neuronové sítě, ve které jsou výstupy každého neuronu přivedeny na vstup všech ostatních, čímž umožňuje postupné zpřesňování výsledku. Použití tato síť nachází například jako asociativní paměť (postupné vybavování informace na základě její částečné znalosti), jako klasifikátor či při řešení optimalizačních problémů. [15]



Obrázek 20: Topologie Hopfieldovy neuronové sítě. Převzato z [15]

Topologie HNN je zobrazena na obrázku. Její charakteristické rysy je možné shrnout v následujících bodech:

1. Počet neuronů sítě je roven počtu vstupů x_i, příčemž do každého neuronu vstupuje pouze jedna část vstupního vzoru.
2. Zpětnovazební topologie je vytvořena tak, že výstup každého neuronu je přiveden na vstup všech ostatních neuronů pomocí vazeb s vahami w_{ij} a tím vzniká uzavřená smyčka.
3. Matice vah W je diagonálně symetrická ($w_{ij} = w_{ji}$) a na diagonále má nuly – neuron není spojen sám se sebou ($w_{ii} = 0$).
4. Učení probíhá jednorázově postupným předkládáním tréninkových vzorů. Lze ho popsat následujícími kroky:
    a. Předložení vzoru ${}^{s}X = [{}^{s}x_1, {}^{s}x_2, \dots, {}^{s}x_n]$ z tréninkové sady $T = \{[{}^{1}X, {}^{1}X], [{}^{2}X, {}^{2}X], \dots, [{}^{p}X, {}^{p}X]\}$
    b. Nastavení vah. Pro perceptrony s nespojitou přenosovou funkcí unipolární signum:

$$w_{ij} = \begin{cases} \sum_{s=1}^{p} (2 \cdot {}^{s}x_i - 1)(2 \cdot {}^{s}x_j - 1), pro\ i \neq j \\ 0, \qquad\qquad pro\ i = j,\ \ 1 \leq (i;j) \leq n \end{cases} \tag{43}$$

Pro přenosovou funkci bipolární signum pak podle vztahů:

$$w_{ij} = \begin{cases} \sum_{s=1}^{p} {}^{s}x_i {}^{s}x_j, \qquad pro\ i \neq j \\ 0, \qquad pro\ i = j,\ \ 1 \leq (i;j) \leq n \end{cases} \tag{44}$$

5. Vybavování je iterativní proces:
    a. Na vstup sítě je přiveden neznámý vzor
    b. Vypočítjí se nové stavy (výstupy) sítě podle

$$m_i(t+1) = f\left[\sum_{j=1}^{n} w_{ij} m_j(t)\right], 1 \leq i \leq n \tag{45}$$

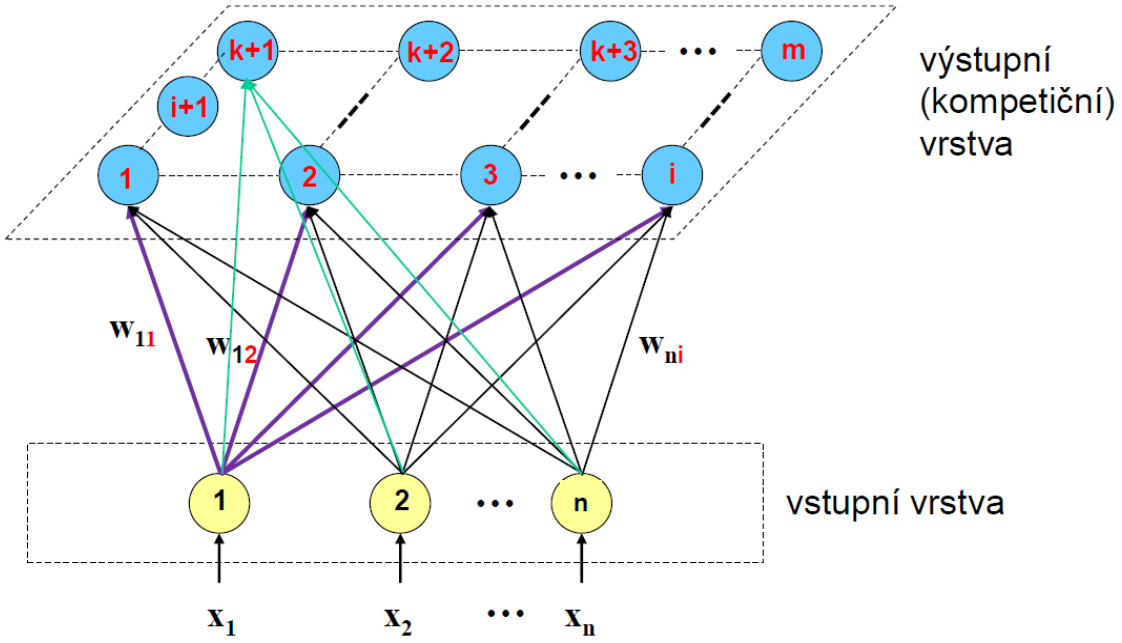
    c. Výpočet stavů sítě se opakuje do doby, dokud se dva po sobě jdoucí stavy neopakují $m_i(t) = m_i(t+1)$. Stav sítě se pak rovná výstupu.

Počet vzorů, které je schopna si Hopfieldova neuronová síť uchovat vychází z počtu neuronů. Údaj, která udává [15] je $p < 0{,}15n$, kde p je počet vzorů a n počet neuronů.

Kohonenova neuronová síť

Kohonenova neuronová síť nebo také Kohonenova samoorganizační mapa (Self-Oraganizing Map) používá soutěžní strategii učení a slouží k identifikaci neznámých vlastností vstupních dat. Po natrénování jsou váhy sítě uvedeny do stavu, který odráží statistické vlastnosti tréninkové množiny. Hlavními rysy kromě zmiňovaná soutěžní strategie učení je dále schopnost samoorganizace a schopnost neustálého doučování [16].

Síť se skládá z dvou vrstev – vstupní vrstva, která slouží pouze pro distribuci signálu (neurony mají lineární přenosovou funkci) a výstupní (také kompetiční) vrstvy, jejíž neurony nemají přenosovou funkci žádnou (v neuronu se provádí pouze výpočet vzdálenosti jeho vah od předloženého vstupního vzoru). Neurony ze vstupní vrstvy jsou propojeny váhovanými spoji se všemi neurony vrstvy výstupní.

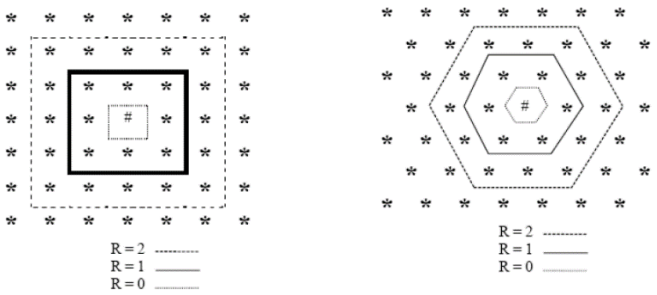


Obrázek 21: Kohonenova samoorganizační mapa. Převzato z [16]

Práce s Kohonenovou samoorganizační mapou lze shrnou v následujících bodech [16]:

1. Volba topologie, uspořádání a inicializace sítě. Tzn.:
    a. Počet neuronů v kompetiční vrstvě
    b. Okolí vítězných neuronů, ve kterém bude probíhat excitace, inhibice, popř. laterární inhibice

c. Nastavení vah na hodnoty blízké nule, určení koeficientu učení α, velikosti okolí R a koeficientu ukončení Φ



Obrázek 22: Okolí neuronů. Převzato z [16]

2. Síti je předložen vzor ${}^{s}X = [{}^{s}x_1, {}^{s}x_2, \dots, {}^{s}x_n]$ z tréninkové množiny $T = \{{}^{1}X, {}^{2}X, \dots {}^{S}X, \dots, {}^{P}X\}$ a vypočítá se vítězný neuron podle jeho vzdálenosti od předloženého vzoru podle vztahu:

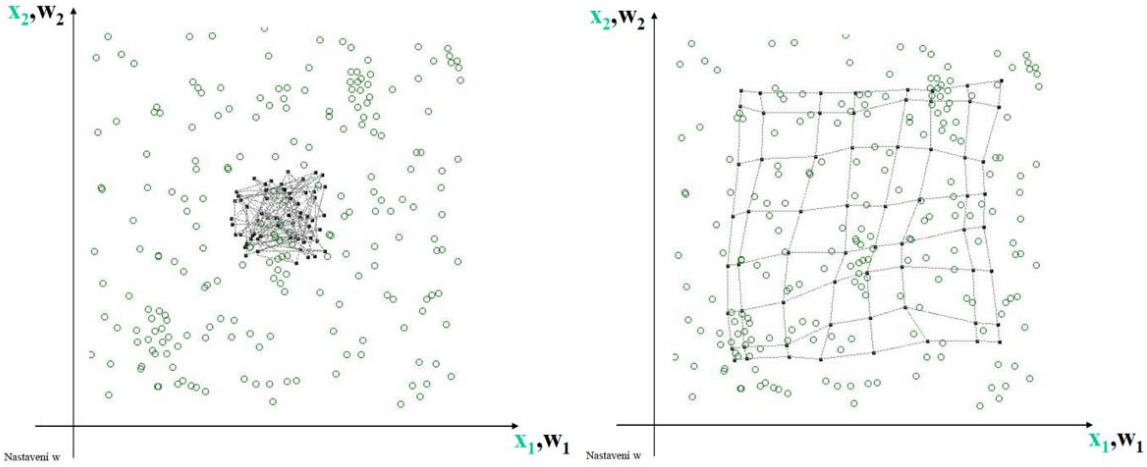
$$d_j = \sum_{i=1}^{n} \left[{}^{s}x_i - w_{ij}(t)\right]^2, \quad 1 \leq j \leq m \tag{46}$$

Kde n je počet neuronů ve vstupní vrstvě, m počet neuronů v kompetiční vrstvě a neuron s nejmenší vzdáleností $d_j^* = min(d_j)$ je určen jako vítězný.

3. Úprava vah vítězného neuronu j^* a jeho okolí podle vztahu

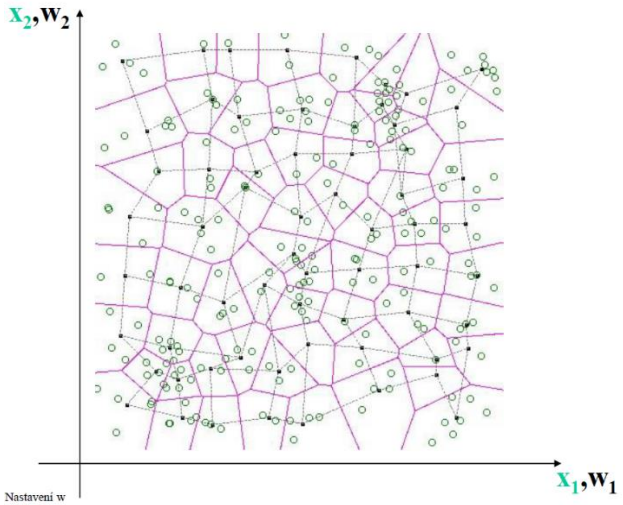
$$w_{ij}(t+1) = w_{ij}(t) + \alpha(k)\left[{}^{s}x_i - w_{ij}(t)\right] \tag{47}$$

4. Po předložení všech tréninkových vzorů se aktualizují parametry učení $\alpha(t+1) = \alpha(t) - e$ a parametry okolí $R(t+1) = R(t) - 1$. Pokud je okolí nebo parametr učení menší než ukončovací podmínka, pak se ukončí cyklus učení. Jinak se provádí znovu od bodu 2.



Obrázek 23: Vizualizace průběhu učení KSOM.a) počáteční nastavení, b) n-tá iterace. Převzato z [16]

Vybavování je jednorázový proces, při kterém se pro předložený neznámý vzor spočítá vzdálenost a je určen vítězný neuron. Vizualizace tohoto procesu může být např. jako na obrázku níže.



Obrázek 24: Vizualizace vybavování KSOM. Převzato z [16]

2.2.3 Boosting

Základní myšlenkou *Boosting* metod je kombinace výstupů více „slabých" klasifikátorů (klasifikátor dává lehce lepší výsledky, než které bychom obdrželi prostým hádáním) a tím dosáhnout přesnějšího výsledku, než by poskytli jednotlivé

klasifikátory samostatně [1]. Nejznámější metodou je tzv. *AdaBoost.M1* algoritmus, prezentován v roce 1997 Freundem a Schapirem.

Při klasifikaci vstupních proměnných X do dvou tříd $Y \in \{-1,1\}$, pak nám klasifikátor $G(X)$ bude predikovat náležitost vstupních dat do jedné ze tříd $\{-1,1\}$. Chyba klasifikátoru je na trénovacích datech dána vztahem:

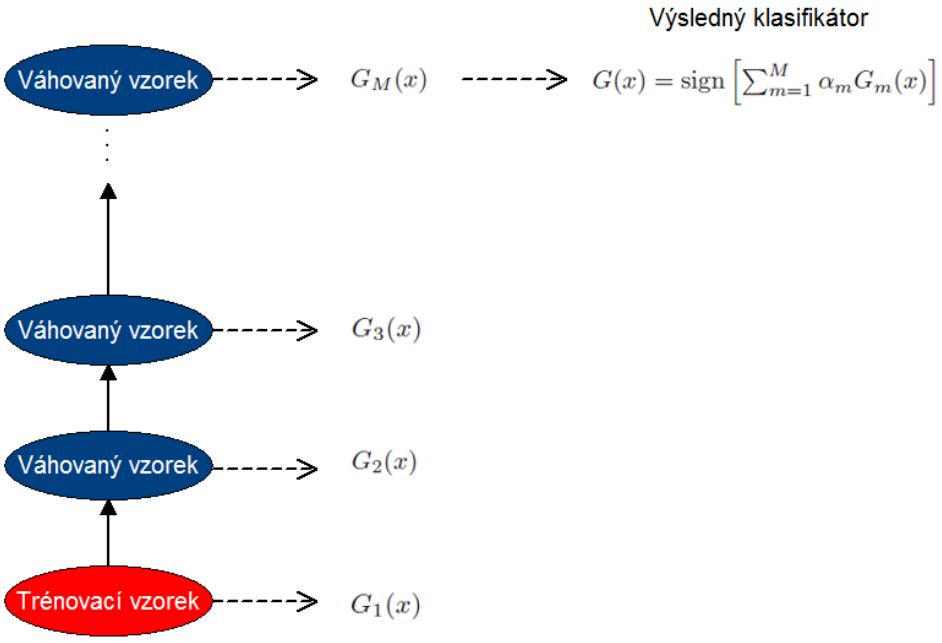
$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} I\big(y_i \neq G(x_i)\big) \qquad (48)$$

A chyba na budoucích predikcích je $E_{XY}I\big(Y \neq G(X)\big)$.

Účelem boosting metod je postupná aplikace slabých klasifikačních metod na opakovaně modifikované verze vstupních dat tak, abychom obdrželi sekvenci slabých klasifikátorů $G_m(x), m = 1,2,\dots,M$. Predikované hodnoty ze všech těchto klasifikátorů jsou kombinovány a váhovány tak, aby produkovali finální predikci:

$$G(x) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right) \qquad (49)$$

Kde $\alpha_1, \alpha_2, \dots, \alpha_M$jsou váhy vypočítané boosting algoritmem a snižují nebo zvyšují hodnotu výsledků jednotlivých slabých klasifikací $G_m(x)$. Tato skutečnost je graficky znázorněna na Obrázek 25.



Obrázek 25: Schématické znázornění AdaBoost. Učení jednotlivých klasifikátorů podle váhovaných trénovacích dat.

Modifikace vstupních dat je v každém kroku boosting algoritmu prováděna pomocí vah $w_1, w_2, \dots, w_N$pro každé vstupní trénovací pozorování $(x_i, y_i), i = 1, 2, \dots, N$. Na začátku jsou všechny váhy nastaveny na $w_i = 1/N$, takže v prvním kroku se klasifikátory učí obvyklým způsobem. V každé další úspěšné iteraci $m = 2,3, \dots M$ jsou váhy jednotlivých klasifikátorů upravovány a algoritmus se opakuje s těmito upravenými váhami. Váhy klasifikátorů $G_{m-1}(x)$ naindukované v předchozích krocích, které v kroku m chybně klasifikovala výstupní třídu, jsou zvýšeny a váhy těch správně klasifikujících jsou naopak sníženy. Jak iterace učebního procesu pokračuje, těžko klasifikovatelné pozorování mají stále větší vliv a každý úspěšný klasifikátor je nucen zvýšit pozornost na ta pozorování, která byla předchozím klasifikátorem špatně zařazena.

Samotný algoritmus *AdaBoost.M1* lze shrnout do třech kroků:

1. Počáteční inicializace vah pozorování $w_i = \frac{1}{N}, i = 1, \dots, N$
2. Pro všechny klasifikátory $m = 1, \dots, M$:
    a. Indukce klasifikátoru $G_m(x)$ na trénovací data váhovaná w_i
    b. Vypočítej chybu výstupu
    $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I\left(y_i \neq G_m(x_i)\right)}{\sum_{i=1}^{N} w_i}$$
    c. Vypočítej váhy α_m pro jednotlivé klasifikátory $G_m(x)$ podle vztahu $\alpha_m = \log\left(\frac{1-\text{err}_m}{\text{err}_m}\right)$
    d. Uprav váhy $w_i \leftarrow w_i \cdot e^{\alpha_m \cdot I\left(y_i \neq G_m(x_i)\right)}, i = 1, \dots, N$
3. Vypočítej výstup $G(x) = sing[\sum_{m=1}^{M} \alpha_m G_m(x)]$

2.3 Generativní modely

Jak bylo zmíněno v předchozí kapitole, výstupem diskriminativních modelů je přímo náležitost objektu do třídy $P(y|\boldsymbol{x})$. Generativní modely oproti tomu klasifikují na základě sdružené pravděpodobnosti $P(y, \boldsymbol{x})$ nebo za použití apriorní pravděpodobnosti $P(y)$ a aposteriorní pravděpodobnosti $P(\boldsymbol{x}|y)$ [10]. Odhad výstupu modelu je pak možné vyčíslit z Bayesova vztahu [2]:

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})} \propto P(\boldsymbol{x}|y)P(y) = P(\boldsymbol{x}, y)$$

Mezi modely, generujících tuto pravděpodobnost můžeme řadit například:

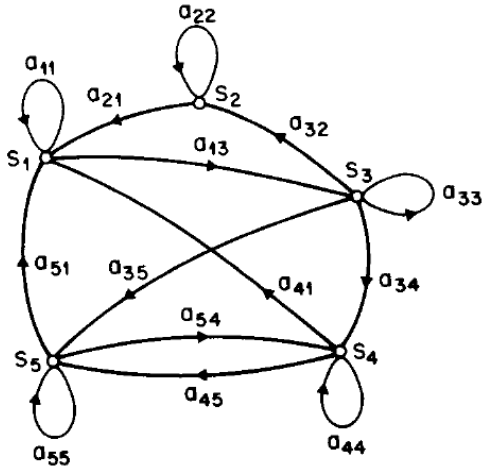
- Gaussian mixture model a další typy mixture modelů
- Skryté Markovovy modely
- Probabilistic context-free grammar

- Naivní Bayesovský klasifikátor
- Bayesian Framework
- Averaged one—dependence estimators
- Latent Dirichlet allocation
- Boltzmanův stroj a omezený Boltzmanův stroj

V následujícím textu budou některé vybrané modely popsány podrobněji.

2.3.1 Skrytý Markovův model

Skrytý Markovův model (HMM) je statistický model používaný k charakterizování statistických vlastností signálu. Jedná se o konečný stavový automat, k jehož zápisu se formálně používá zápis $\lambda = (N, M, A, B, \Pi)$.



Obrázek 26:Markovův řetězec se stavy S_1-S_5 a přechody mezi nimi. Převzato z [17]

Význam jednotlivých symbolů v uvedeném zápisu je [17]:

- N – Počet skrytých stavů modelu. Pokud máme vektor stavů ve tvaru $S = \{S_1, S_2, \dots, S_N\}$, pak je aktuální stav v čase t dán $q_t \in S, 1 \leq t \leq T$. T je délka pozorované sekvence.
- M – Počet pozorovaných symbolů ve stavu. Vektor pozorovatelných symbolů označujeme $V = \{v_1, v_2, \dots, v_M\}$ s jejich hodnotou O_t v čase t. Hodnoty pozorovatelných symbolů přímo odpovídají fyzickému výstupu automatu.
- A – Matice pravděpodobnosti přechodů mezi jednotlivými stavy
  $$a_{ij} = P\left[q_t = S_j | q_{t-1} = S_i\right], 1 \leq i, j, \leq N$$
  kde $0 \leq a_{ij} \leq 1$ a zároveň je součet členů v řádku roven jedné: $\sum_{j=1}^{N} a_{ij} = 1$, $1 \leq i \leq N$

- B – Vektor rozdělení pravděpodobností pozorovaných symbolů v jednotlivých stavech. Platí $B = \{b_j(k)\}$ kde:
$$b_j(k) = P[v_k \; v \; \check{c}ase \; t | q_t = S_j], 1 \leq j \leq N, 1 \leq k \leq M$$
- Π – Vektor počáteční pravděpodobnosti stavů $\Pi = \{\pi_i\}$, pro který platí:
$$\pi_i = P[q_1 = S_i], 1 \leq i \leq N$$

Takový to stavový automat potom generuje pozorovanou sekvenci $O = O_1 \; O_2 \; \dots \; O_T$, kde každé pozorování O_t odpovídá jednomu symbolu z V, následujícím způsobem [17]:

1. Výběr počátečního stavu $q_1 = S_i$ podle rozdělení počátečních pravděpodobností stavů Π
2. Inkrementace času do $t = 1$
3. Podle aktuálního stavu S_i a rozdělení pravděpodobností pozorovaných symbolů v něm se zvolí výstup automatu $O_t = v_k$
4. Podle matice pravděpodobnosti přechodů $A = \{a_{ij}\}$ se přesuň ze stavu S_i do nového stavu $q_{t+1} = S_j$
5. Pokud je aktuální čas menší, než je délka pozorované sekvence $t < T$, nastav $t = t + 1$ a opakuj od bodu 3. Pokud je $t = T$, tak algoritmus končí.

Vyhodnocení – výpočet podmíněné pravděpodobnosti $P(O|\lambda)$

$P(O|\lambda)$ odpovídá pravděpodobnosti, že budeme pozorovat sekvenci $O = O_1 \; O_2 \; \dots \; O_T$ generovanou modelem λ. Přímý výpočet je ovšem výpočetně a časově náročný, proto se zavádí tzv. *forward-backward* procedura s dopřednou (*forward*) proměnnou $\alpha_t(i)$, která odpovídá pravděpodobnosti částečné sekvence pozorování $O_1 \; O_2 \; \dots \; O_t$ a stavu S_i v čase t modelu λ

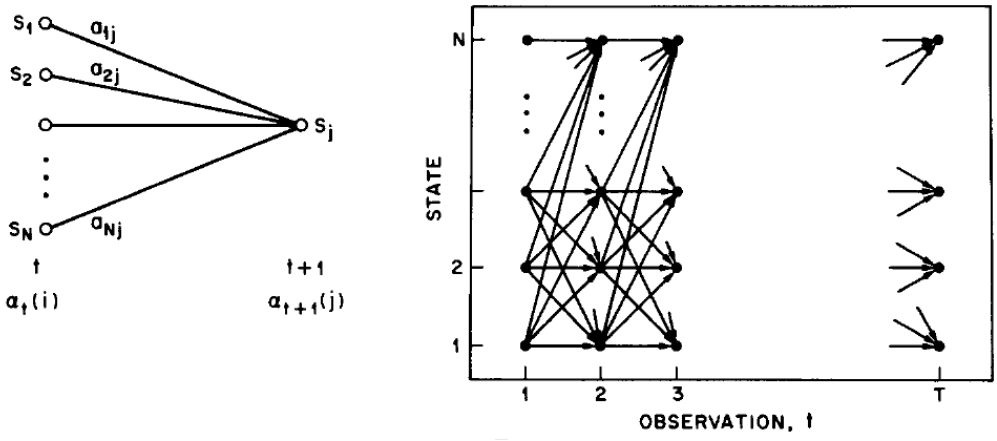
$$\alpha_t(i) = P(O_1 \; O_2 \; \dots \; O_t, q_t = S_i | \lambda) \qquad (50)$$

Induktivní řešení pomocí $\alpha_t(i)$ lze shrnout do následujících kroků:

1) Inicializace dopředné proměnné

$$\alpha_t(i) = \pi_i b_i(O_i), 1 \leq i \leq N$$

2) Pomocí induktivního výpočtu (viz. Obrázek 27) zjistíme, jak je možné dosáhnout stavu S_j v čase $t + 1$ z N možných stavů $S_i, 1 \leq i \leq N$, ve kterých se automat nachází v čase t



Obrázek 27:Vlevo - ilustrace sekvencí operací potřebných pro výpočet dopředné proměnné $\alpha_{t+1}(j)$.Vpravo - Implementace výpočtů $\alpha_t(j)$ z hlediska mřížky pozorování t a stavů i. Převzato z [17]

$\alpha_t(i)$ je pravděpodobnost, že budeme pozorovat sekvenci $O_1\ O_2 \dots O_t$ a stav v čase t bude S_i. Pokud chceme vyjádřit, jaký stav S_j bude následovat v čase $t+1$ (tedy že budeme pozorovat sekvenci $O_1\ O_2 \dots O_t$ do stavu S_i a stav S_j bude následovat), můžeme tuto pravděpodobnost vyjádřit jako $\alpha_t(i)a_{ij}$. Sumací všech možných pozorování přes všechny možné stavy $S_i, 1 \le i \le N$ v čase t získáme pravděpodobnost dosažená stavu S_j v čase $t+1$ se všemi doprovodnými pozorováními. Když známe S_j, $\alpha_{t+1}(j)$ můžeme spočítat:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)a_{ij}\right] b_j(O_{t+1}), 1 \le t \le T-1, 1 \le j \le N$$

3) Výpočet pravděpodobnosti $P(O|\lambda)$ je pak dán sumou dopředných proměnných $\alpha_T(i)$, protože vzhledem k definici
$$\alpha_T(i) = P(O_1\ O_2 \dots O_T, q_T = S_i|\lambda)$$
je pravděpodobnost $P(O|\lambda)$ pouze suma všech $\alpha_T(i)$.

Obdobným způsobem je taktéž možné zavést zpětnou proměnnou $\beta_t(i)$ vyjadřující pravděpodobnost pozorování částečné sekvence od času $t+1$ do konce, daného stavem S_i v čase t a modelem λ jako:

$$\beta_t(i) = P(O_{t+1}\ O_{t+2} \dots O_T|q_T = S_i, \lambda) \qquad (51)$$

Výpočet $\beta_t(i)$ budeme opět řešit induktivně následujícím způsobem:

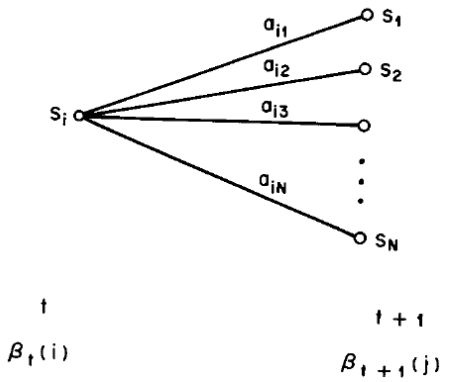
1) Inicializace zpětné proměnné $\beta_t(i)$
$$\beta_t(i) = 1, 1 \le i \le N$$

2) Indukce

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$$

$$t = T-1, T-2, \dots, 1, \;\; 1 \leq i \leq N$$

Tento proces je znázorněn na obrázku níže. Pokud jsme v čase t ve stavu S_i a bereme v úvahu pozorovanou sekvenci od času $t+1$, musíme vzít v úvahu všechny stavy S_j v čase $t+1$. Proto se ve vzorci vyskytuje člen a_{ij} (přechod ze stavu S_i do stavu S_j). Dále musíme vzít v úvahu pozorování O_{t+1} ve stavu j (člen $b_j(O_{t+1})$) a zbývající částečné sekvence pozorování od stavu j (člen $\beta_{t+1}(j)$)



Obrázek 28: Ilustrace výpočtu zpětné proměnné. Převzato z [17]

Učení – odhad parametrů modelu $\lambda = (A, B, \Pi)$

Učení modelu spočívá v určení parametrů (A, B, Π) tak, aby byla maximalizována pravděpodobnost sledované posloupnosti dané modelem. Analytická metoda ani optimální metoda, které by vedla k této maximalizaci, neexistuje [17]. Existují ale postupy výběru modelu $\lambda = (A, B, \Pi)$, které pravděpodobnost $P(O|\lambda)$ lokálně maximalizují pomocí iterativních procedur. Například:

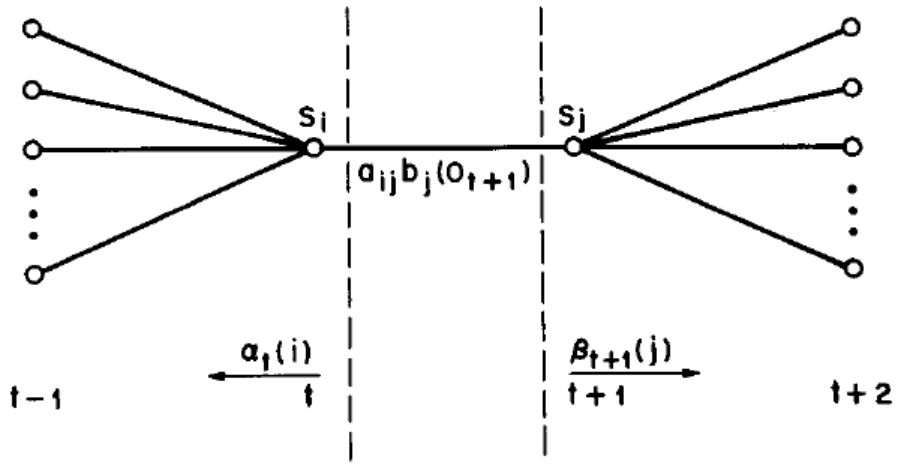
- Algoritmus Baum-Welch
- EM (Expectation-modification) algoritmus
- Gradientní techniky

Pro lepší ilustraci bude nyní rozebrán algoritmus Baum-Welch. Pro tento postup je nutné nejprve zavést proměnnou $\xi_i(i,j)$ vyjadřující sdruženou pravděpodobnost, že se automat nachází v čase t ve stavu S_i a v čase $t+1$ bude ve stavu S_j jako:

$$\xi_i(i,j) = P\big(q_t = S_i, q_{t+1} = S_j | O, \lambda\big) \qquad (52)$$

Nebo vyjádřením pomocí zpětné a dopředné proměnné:

$$\xi_i(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \tag{53}$$



Obrázek 29: Ilustrace sekvence výpočtů, potřebných k určení události, že systém bude ve stavu S_i v čase t a ve stavu S_j v čase $t+1$

Nyní si zavedeme proměnnou $\gamma_t(i)$, které vyjadřuje stav, že se pro danou posloupnost pozorování a aktuální HMM model nacházíme v čase t ve stavu S_i. To znamená, že $\gamma_t(i)$ je možné vyjádřit jako součet $\xi_i(i,j)$ přes všechna j jako:

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j) \tag{54}$$

Součtem všech $\gamma_t(i)$ od časového indexu $t=1$ do $t=T-1$, dostaneme hodnotu, která může být interpretována jako očekávaný počet průchodů stavem S_i (proto je vynechán čas $t=T$). Obdobně je možné určit očekávaný počet přechodů ze stavu S_i do stavu S_j pomocí sumace $\xi_i(i,j)$.

$$\sum_{t=1}^{T-1}\gamma_t(i) = \text{očekávaný počet přechodů ze stavu } S_i \tag{55}$$

$$\sum_{t=1}^{T-1}\xi_i(i,j) = \text{očekávaný počet přechodů ze stavu } S_i \text{ do } S_j \tag{56}$$

Pomocí výše uvedených vztahů můžeme upravit odhad parametrů zjednodušeného skrytého markovského modelu $\lambda = (A, B, \Pi)$ podle následujících kroků:

1) Pro současný model $\lambda = (A, B, \Pi)$ vypočítáme pravé strany rovnic:

   $\bar{\pi}_i =$ očekávaná frekvence průchodů stavem S_i v čase $(t = 1) = \gamma_t(i)$

$$\bar{a}_{ij} = \frac{\text{očekávaný počet přechodů ze stavu } S_i \text{ do } S_j}{\text{očekávaný počet přechodů ze stavu } S_i} = \frac{\sum_{t=1}^{T-1} \xi_i(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\bar{b}_j(k) = \frac{\text{očekávaný počet průchodů stavem j za současného pozorování symbolu } \mathrm{v_k}}{\text{očekávaný počet průchodů stavem j}} = \frac{\sum_{t=1}^{T} \gamma_t(j)}{s.t.\ O_t = v_k} = \frac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

2) Z levých stran definujeme nový model $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\Pi})$
   a. Pokud $\lambda = \bar{\lambda}$, pak byla nalezena extrémní hodnota pravděpodobnostní (*likelihood*) funkce.
   b. Jestliže pravděpodobnost generování pozorované posloupnosti nového modelu je vyšší, než modelu předchozího, tzn. $P(O|\bar{\lambda}) > P(O|\lambda)$, použijeme tento nový model a vrátíme se do kroku 1).
   c. Pokud jsme dosáhli nějaké limitující podmínky, algoritmus končí.

Výsledek tohoto algoritmu se nazývá *maximum likelihood estimate* (maximální pravděpodobnostní odhad) dané HMM a jak je z postupu patrné, vede pouze k lokálnímu maximu.

2.3.2 Naivní Bayesovský klasifikátor

Tento typ modelů je vhodný pro aplikace, kde není jednoduché nalézt přímé řešení a je nutné stanovit, s jakou pravděpodobností odpovídá zkoumaný objekt jednotlivým hypotézám [18]. Modely jsou založené na Bayesově teorému, podle kterého je podmíněná pravděpodobnost $P(y_c|\boldsymbol{x})$ třídy y_c vstupního vektoru $\boldsymbol{x} = (x_1, \dots, x_n)$ dána vztahem:

$$P(y_c|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y_c)P(y_c)}{P(\boldsymbol{x})} \qquad (57)$$

Pravděpodobnost $P(\boldsymbol{x})$ je konstantní, není tedy pro klasifikaci relevantní (různá y_c jsou porovnávána na stejném $\boldsymbol{x}$). $P(y_c)$ je apriorní pravděpodobnost

výskytu objektu v třídě y_c, kterou je možné stanovit z trénovací množiny. Zbývá tedy jenom neznámá $P(\boldsymbol{x}|y_c)$. „Naivnost" Bayesovského klasifikátoru je dána tím, že předpokládá nezávislost atributů každé třídy. Podmíněnou pravděpodobnost $P(\boldsymbol{x}|y_c)$ je za těchto podmínek možné vypočítat:

$$P(\boldsymbol{x}|y_c) = \prod_{i=1}^{n} P(x_i|y_c) \tag{58}$$

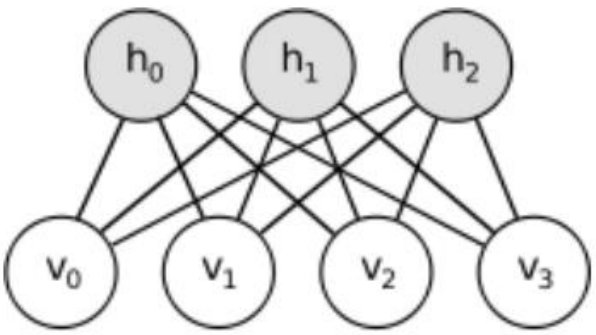
Pří trénování modelu je potřeba určit pravděpodobnost $P(y_c)$ pro všechna y_c z množiny tříd C. Určením sdružené pravděpodobnosti $P(x_i|y_c)$ pro všechny dostupné atributy $(x_1, \dots, x_n)$ je poté možné při klasifikaci testovací instance $\boldsymbol{x} = (x_1, \dots, x_n)$ vypočítat do které třídy náleží podle vztahu:

$$y = \underset{c\in\{1,\dots,C\}}{\operatorname{argmax}} P(y_c) \prod_{i=1}^{n} P(x_i|y_c) \tag{59}$$

Protože bylo při výpočtu zanedbáno $P(\boldsymbol{x})$, nedává funkce argmax konkrétní pravděpodobnost. Tu lze získat tak, když se všechny dílčí výsledky pro všechna c normují, aby jejich součet byl roven 1.

2.3.3 Omezený Boltzmannův stroj

Omezený Boltzmannův stroj (angl. RBM – *Restricted Boltzmann Machine)* patří mezi tzv. modely založené na energii (angl. EBM – *Energy-based Models)*, které přiřazují každé konfiguraci modelu skalár (energii), vyjadřující míru kompatibility mezi proměnnými (například funkce energie $E(X, Y)$, přiřazující energii vstupů X – pixelům obrázku a výstupům Y – třídy do kterých se má klasifikovat). Při učení modelu se pak hledá taková funkce energie, aby správným výsledkům přiřazovala menší energii a špatným větší energii [19]. Kromě toho, že RBM jsou postaveny na struktuře EBM, je možné s určitými změnami taktéž interpretovat jako dvouvrstvou neuronovou síť (viz. Obrázek 30).



Obrázek 30: Omezený Boltzmanův stroj. Převzato z [19]

Podle [20] je RBM možné definovat jako dvouvrstvý pravděpodobnostní model s následující distribucí:

$$p(v,h|W) = \frac{1}{Z(W)} e^{-E(v,h,W)} \tag{60}$$

Kde $E(v,h,W)$ je funkce energie, $v \in V$ jsou vstupní viditelné uzly, $h \in H$ jsou skryté náhodné proměnné a W vektor vah (interakcí) mezi vstupními a výstupními vrstvami. Pokud máme D vstupních uzlů a K výstupních, pak vektor vah má rozměr $W \in \mathbb{R}^{D \times K}$. Funkce $Z(W)$je definovaná jako:

$$Z(W) = \sum_{v' \in V} \sum_{h' \in H} p(v', h'|W) \tag{61}$$

Cílem učená RBM je maximalizování pravděpodobnosti $p(v|W)$

$$p(v|W) = \frac{1}{Z(W)} \sum_{h \in H} e^{-E(v,h,W)} \tag{62}$$

Uvažujeme-li binární případ RBM, pro který platí $V = \{0,1\}^D$ a $H = \{0,1\}^K$ může být funkce energie $E(v,h,W)$ s biasem vstupních (viditelných) uzlů c a biasem skrytých uzlů b vyjádřena jako:

$$\begin{aligned} E(v,h,W) = &- \sum_{i=1}^{D} \sum_{j=1}^{K} v_i W_{ij} h_j \\ &- \sum_{i=1}^{D} v_i c_i - \sum_{j=1}^{K} h_j b_j \end{aligned} \tag{63}$$

Obecně je ale možné použít RBM s reálnými stavy neuronů. Podle [19] toho je možné docílit tak, že se vstupní hodnoty přetransformují na interval [0,1] a tím zůstanou zachovány i pravděpodobnostní funkce. Jinou možností (lepší) je použití jiných typů neuronů (Gaussovské, usměrněné lineární). Chování takové neuronové sítě je stejné jako u perceptronové sítě, tedy až na výstup (stav) neuronu, který se definuje jako pravděpodobnost aktivace.

$$p(v_i = 1|h,W) = \sigma\left(\sum_{j=1}^{K} W_{ij} h_j\right) \tag{64}$$

$$p(h_j = 1|v, W) = \sigma\left(\sum_{j=1}^{D} W_{ij} v \right) \quad (65)$$

Přičemž pro funkci σ platí:

$$\sigma(a) = \frac{1}{1 + e^{-a}} \quad (66)$$

Pro učení RBM existuje několik algoritmů:

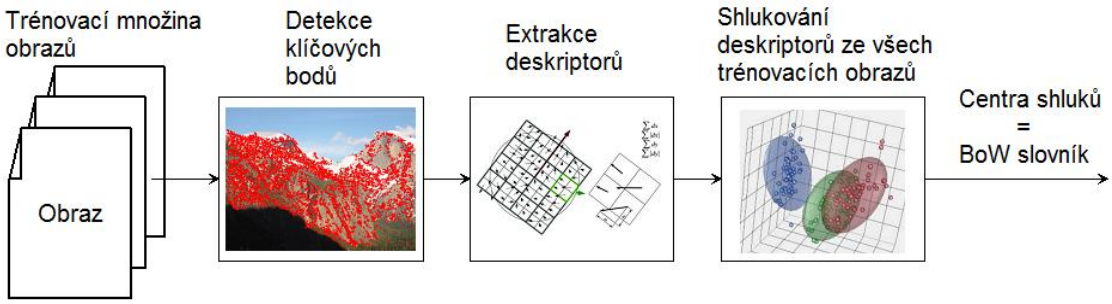
- Gibbsovo vzorkování
- Kontrastivní divergence
- Perzistentní kontrastivní divergence

2.4 Bag of Words (BoW)

BoW je metoda pro tvorbu příznakových vektorů, která byla původně určena pro reprezentaci textových dokumentů podle výskytu určitých slov bez ohledu na jejich pořadí. Díky BoW bylo urychleno vyhledávání klíčových slov v dokumentu, proto se modifikace metody (nazývané Bag of visual words nebo také Bag of Keypoints) začala využívat i v oblasti počítačového vidění, kde urychluje vyhledávání v obrazových databázích [21].

BoW slovník

Pro tvorbu modelu BoW [22] je nejprve nutné extrahovat z tréninkového datasetu vizuální vzorky, které ideálně nejsou závislé na rotaci, měřítku, intenzitě ani změně pohledu. Těmto vzorkům se také říká klíčové body (angl. keypoints). Pro detektory klíčových bodů je velmi důležité, aby splňovali podmínku opakovatelnosti (repeatability), tj. aby dokázali detekovat stejné body ve dvou nebo více obrazech se stejnou scénou, i když se liší úhlem pohledu. Nalezené klíčové body a jejich okolí jsou následně popsány prostřednictvím lokálních příznaků, k čemuž může sloužit například metody SIFT, SURF, BRISK,…

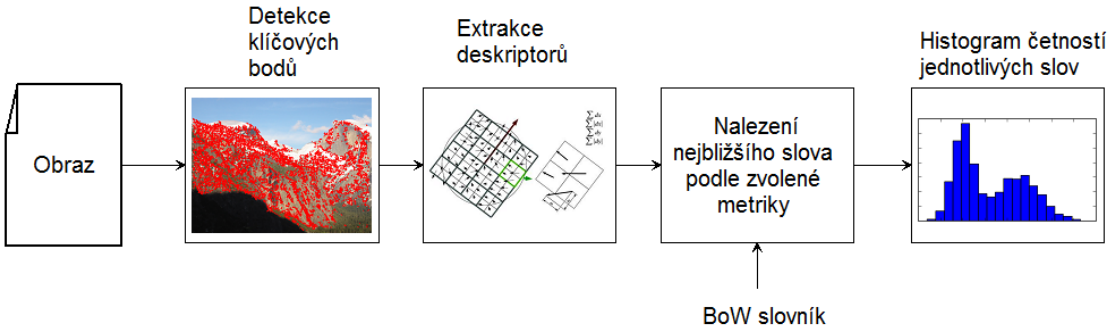


Obrázek 31: Tvorba vizuálního slovníku.

Protože je množina všech deskriptorů značně veliká, je na jejich příznaky použita shluková analýza (např metoda k-means, k-nn, apod.), která vytvoří disjunktivní podmnožiny – shluky – příznaků s podobnými vlastnostmi. Každý z těchto shluků je pak opatřen unikátním identifikačním číslem (tzv. vizuálním slovem). Postup tvorby vizuálního slovníku je znázorněn na obrázku výše (viz. Obrázek 31). Takto získaný vizuální slovník pak slouží k tvorbě příznakových vektorů z modelu BoW.

BoW histogram

Z předloženého obrazu jsou extrahovány klíčové body a jejich příznakové vektory. Pro všechny takto získané vektory je vypočítána aproximace nejbližšího souseda - slova ve slovníku a sestaven histogram četností jednotlivých slov. Délka takto vytvořeného histogramu odpovídá počtu slov v BoW slovníku (počtu shluků ze shlukové analýzy).



Obrázek 32: Získání příznakového vektoru pomocí vizuálního slovníku.

2.4.1 Varianty BoW

Jelikož je možné tvorbu a učení modelu BoW rozdělit do více dílčích kroků, může být vzhledem k dané aplikaci provedena optimalizace a hledání nových, lepších, přístupů. Obecně lze dané kroky řadit do následujících skupin [23]:

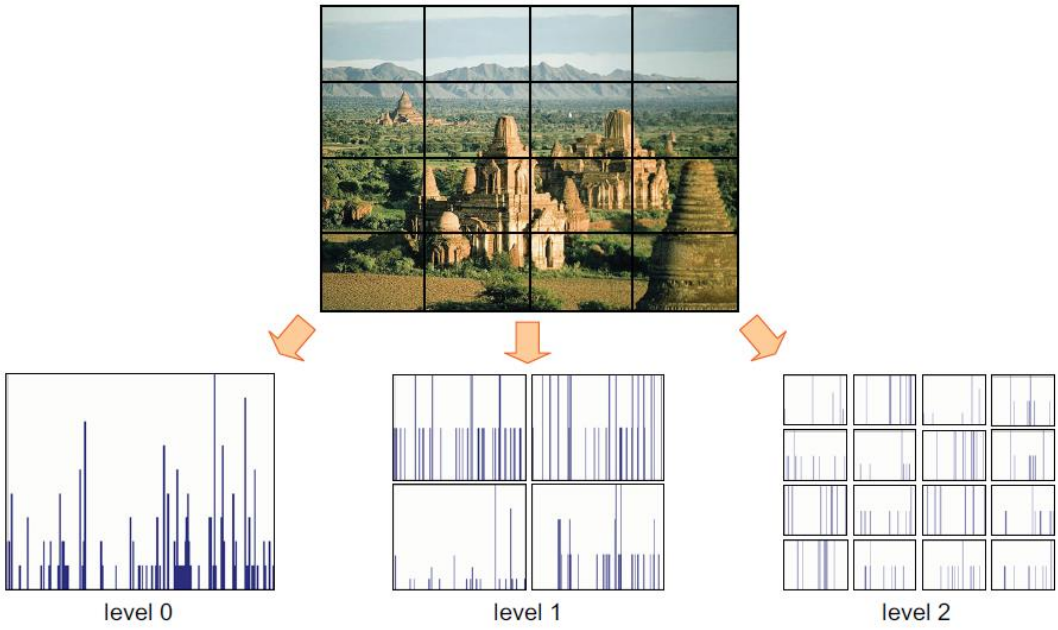
- Hledání klíčových bodů (keypoints)
- Extrakce příznaků

- Tvorba vizuálního slovníku
- Prostorová informace
- Slučování (fúze) příznaků
- Klasifikace

V následujícím textu budou uvedeny některé z těchto modifikací.

Spatial pyramid matching

Schématická ilustrace metody *spatial pyramid matching* je znázorněna na obrázku níže. Jedná se o vytváření kolekce histogramů, které se počítají nad částí obrazu. Na nejnižší úrovni (level 0) je histogram složen z celého obrazu a je ekvivalentní standardnímu *Bag-of-words* histogramu. Na další úrovni (level 1) je obraz rozdělen do čtyř kvadrantů a histogramy jsou počítány nad každým kvadrantem zvlášť. S dalšími úrovněmi se akce stále opakuje. Histogramy se poté zřetězí a jsou vstupem pro klasifikaci [24].



Obrázek 33: Ilustrace "spatial pyramid" reprezentace. Převzato z [24]

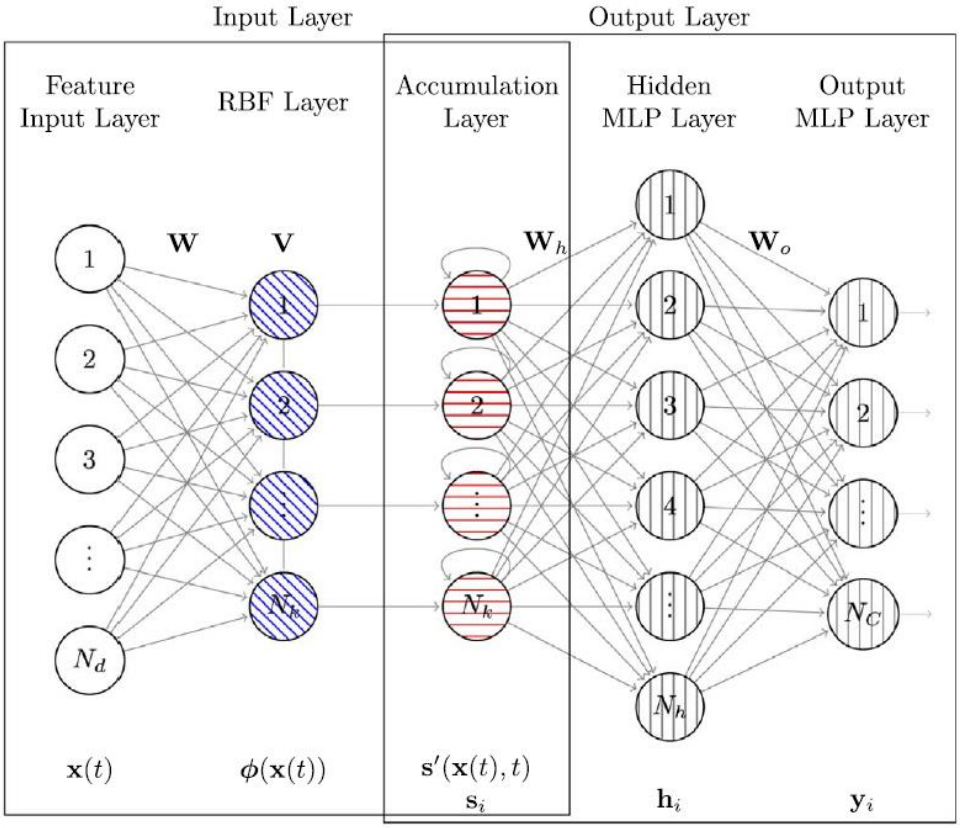
Pro klasifikaci histogramů získaných pomocí této metody je možné použít tzv. *pyramid kernel*, který příznakům ve vyšší vrstvě pyramidy přiřazuje vyšší váhu. To odráží fakt, že vyšší vrstvy lokalizují příznak mnohem přesněji.

Neural BoW

*Neural Bag-of-*Features [25] je model inspirovaný metodou BoW, ve kterém je při tvorbě slovníku využito dvou vrstev neuronové sítě:

1) RBF (Radial Basis Function) vrstva
2) Akumulační vrstva

Schopnost *Neural Bag-of-Features* modelu zlepšovat klasifikační výkon je dokázána v článku [25]. Architektura modelu je uvedena na Obrázek 34



Obrázek 34: Architektura Neural Bag-of-Features modelu. Převzato z [25]

Model se skládá ze dvou vrstev. První, RBF vrstva měří podobnost vstupu s centry RBF a druhé, rekurentní akumulační vrstvy, ve které se vytváří histogram vstupních příznaků.

Výstup k-tého RBF neuronu je definován jako:

$$[\boldsymbol{\Phi}(\boldsymbol{x})]_k = \exp(-\|(\boldsymbol{x} - \boldsymbol{v_k}) \odot \boldsymbol{w_k}\|_2) \qquad (67)$$

kde $\boldsymbol{x}$ je vstupní příznakový vektor, $\boldsymbol{v_k} \in R^D$ jsou centra a $\odot$ je operátor násobení prvku s prvkem (*element-wise multiplication*). RBF neuron představuje něco jako kódové slovo v klasickém BoW modelu, tzn. používá se k měřené podobnosti vstupního vektoru se sadou předdefinovaných vektorů. Každý z těchto RBF neuronů je vybaven vektorem vah $\boldsymbol{w_k} \in R^D$, díky kterému je upravována Gaussova funkce pro každou dimenzi.

Aby byla zajištěna omezení výstupu každého RBF neuronu, je v [25] uvedena normalizovaná RBF architektura, čehož je dosaženo upravením rovnice výše na tvar:

$$[\boldsymbol{\Phi}(\boldsymbol{x})]_k = \frac{\exp(-\|(\boldsymbol{x}-\boldsymbol{v_k})\odot\boldsymbol{w_k}\|_2)}{\sum_{m=1}^{N_K}\exp(-\|(\boldsymbol{x}-\boldsymbol{v_m})\odot\boldsymbol{w_m}\|_2)} \tag{68}$$

Výstupy všech RBF neuronů jsou akumulovány v následující vrstvě sítě (viz. Obrázek 34) pomocí časového konceptu a rekurentní smyčky (*recurrent self-loop*)

$$\begin{aligned}[\boldsymbol{s}'(\boldsymbol{x}(t),t)]_k = &\frac{1}{t}[\boldsymbol{\Phi}(\boldsymbol{x})]_k \\ &+\frac{t-1}{t}[\boldsymbol{s}'(\boldsymbol{x}(t-1),t-1)]_k\end{aligned} \tag{69}$$

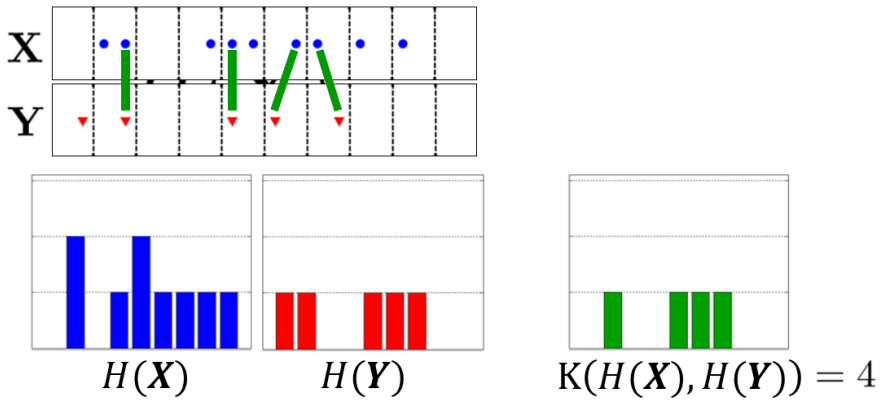
kde $\boldsymbol{x}$ je vstupní příznakový vektor a $[\boldsymbol{s}'(\boldsymbol{x}(t),t)]_k$ je výstup akumulační vrstvy v čase t. Tento výstup je na počátku (před započetím předkládání příznakových vektorů síti) vynulován.

2.4.2 Jádrové funkce pro BoW

Jako jádrová funkce, která pozitivně ovlivňuje rychlost a přesnost klasifikace, je podle literatury nejčastěji požívaný tzv. *histogram intersection.* Jádro je ve tvaru [http://acberg.com/papers/mbm08cvpr.pdf]:

$$\mathrm{K}(H(\boldsymbol{X}),H(\boldsymbol{Y})) = \sum_{i=1}^{N}\min(H(\boldsymbol{X})_i, H(\boldsymbol{Y})_i) \tag{70}$$

Kde $H(\boldsymbol{X}), H(\boldsymbol{Y})$ jsou zkoumané histogramy. Toto jádro nám v principu udává počet shod mezi danými histogramy (viz. Obrázek 35)



Obrázek 35: Princip jádra histogram intersection. Převzato z http://16720.courses.cs.cmu.edu/lec/bagwords.pdf

Pro BoW rošíření v podobě spatial pyramid autoři [24] chtěli dosáhnout úpravou jádra *histogram intersection* tak, aby shodám na vyšší úrovni pyramid byla přiřazena vyšší váha.

Pro zjednodušení si označme $\mathrm{K}\left(H(\boldsymbol{X})^{\ell}, H(\boldsymbol{Y})^{\ell}\right)$ jako K^{ℓ}, což odpovídá *histogram intersection* funkci na úrovni pyramidy $\ell = 0, \dots, \mathrm{L} - 1$. Rozdíl $\mathrm{K}^{\ell} - \mathrm{K}^{\ell+1}$ bude potom udávat počet nových shod na úrovni ℓ, přičemž každé úrovni ℓ je přiřazená váha $\frac{1}{2^{L-\ell}}$.

$$\kappa^{L}(X, Y) = K^{L} + \sum_{\ell=0}^{L-1} \frac{1}{2^{L-l}} \left(K^{\ell} - K^{\ell+1}\right) = \frac{1}{2^{L}} K^{0} + \sum_{\ell=1}^{L} \frac{1}{2^{L-\ell+1}} K^{\ell} \tag{71}$$

Intuitivně se tak dosáhne penalizace shod nalezených ve větších buňkách (na nižší úrovni), protože zahrnují více odlišné rysy. Výsledné jádro pak bude ve tvaru:

$$\mathcal{K}^{L}(X, Y) = \sum_{m=1}^{M} \kappa^{L}\left(X_m, Y_m\right) \tag{6}$$

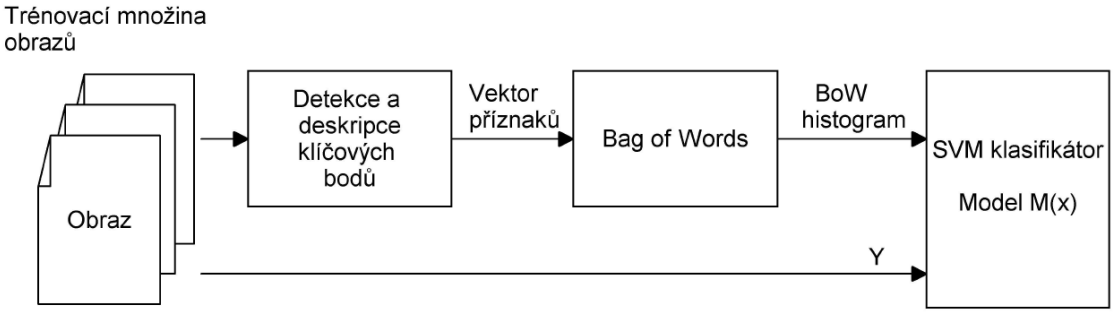
Kde X_m, Y_m odpovídají zřetězeným histogramům získaných pomocí spatial pyramid.

3 PRAKTICKÁ ČÁST

Cílem praktické části této práce je navrhnout a implementovat vybranou úlohu demonstrující princip vybrané metody strojového učení pro kategorizaci objektů v obrazu a následně ověřit funkci na dostatečně velké množině obrazů. Jako vývojové prostředí pro celý projekt byl zvolen program Matlab.

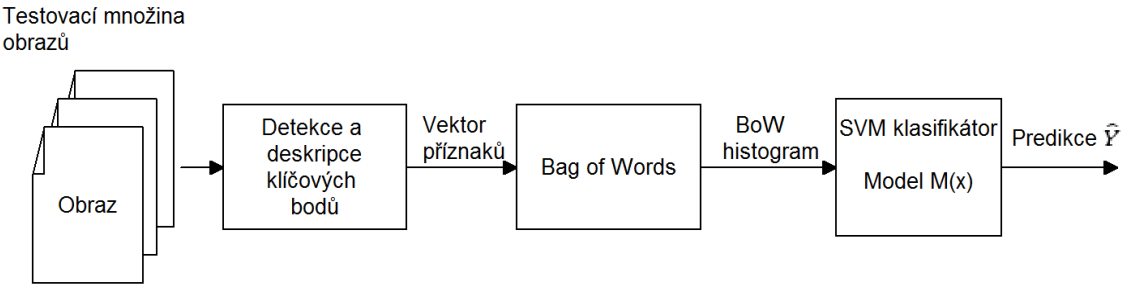
V této kapitole tedy bude nejprve rozebrána vlastní struktura navrženého programu a jeho základních funkcí. Následně bude proveden rozbor různých vlivů, které pozitivně či negativně ovlivňují výslednou úspěšnost klasifikace objektů v obraze. Na konci kapitoly budou uvedeny výsledky testování modelu na různých veřejně přístupných databázích obrazů.

Pro demonstraci zadané úlohy byla zvolena implementace metody *Bag-of-Words* (nebo také *Bag-of-features* či *Bag-of-visual-words*) s SVM klasifikátorem. Učební proces metody je možné znázornit takto:



Obrázek 36: Učení SVM klasifikátoru podle histogramů BoW

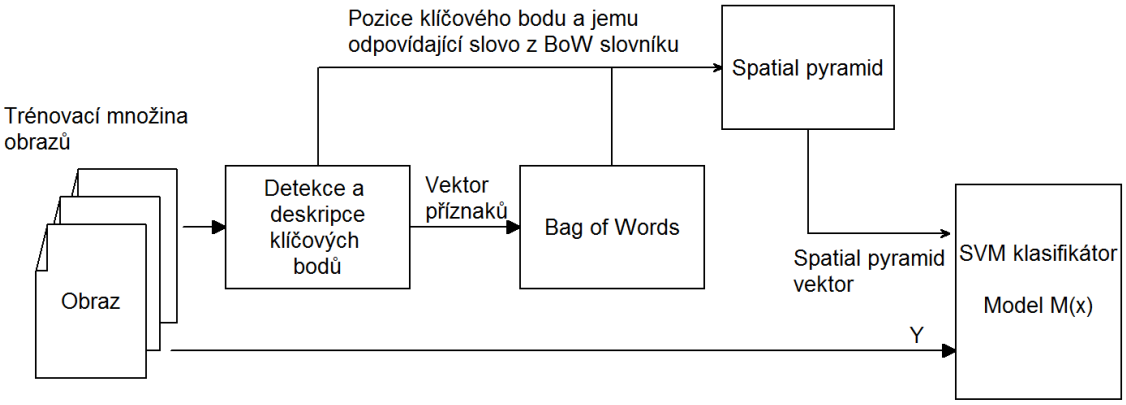
Při ověřování úspěšnosti klasifikace jsou modelu $M(x)$ předkládány obrazy z testovací množiny a jeho výstupem je potom predikce náležitosti objektu $\hat{Y}$



Obrázek 37: Predikování náležitosti obrazu do třídy pomocí SVM klasifikátoru a BoW

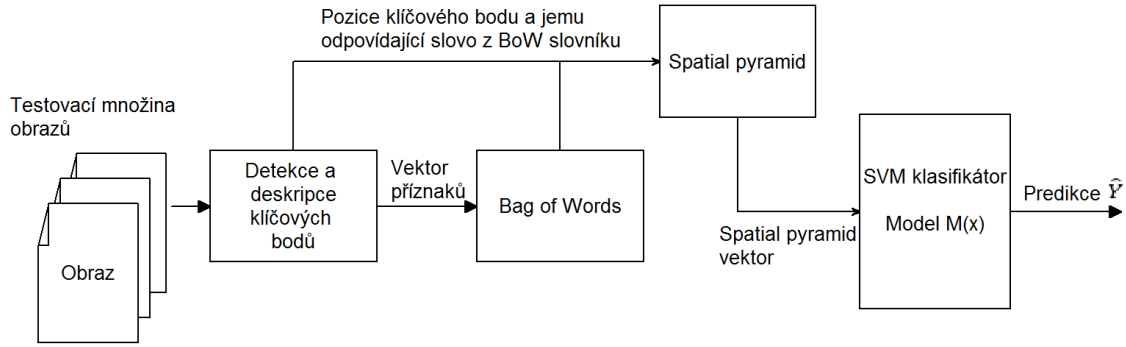
V rámci praktické části byla implementována i modifikace tradiční metody *Bag-of-Words*, která využívá tzv. „spatial pyramids“. V tomto případě jsou podle vytvořeného BoW slovníku označeny jednotlivé klíčové body BoW slovy. Poté se nad

takto označeným vstupním obrazem vytvoří pomocí algoritmu spatial pyramid rozšířený příznakový vektor (viz. Obrázek 38 - označen jako „*Spatial pyramid vektor*“). Tento vektor poté vstupuje opět do SVM klasifikátoru a v součinnosti s upravenou jádrovou funkcí je tím dosaženo zvýšení úspěšnosti klasifikace. Schéma učebního procesu je možné znázornit například takto:



Obrázek 38: Učení SVM klasifikátoru podle zřetězených BoW histogramů (spatial pyramid vektor)

Klasifikace obrazů z testovací množiny probíhá obdobným způsobem. *Spatial pyramid* vektor je předložen naučenému SVM klasifikátoru $M(x)$, který na základě trénovacích obrazů provede predikci náležitosti objektu $\hat{Y}$:



Obrázek 39: Predikování náležitosti obrazu do třídy pomocí SVM klasifikátoru a zřetězených BoW histogramů (spatial pyramid vektor)

3.1 Databáze veřejně přístupných dat

Na internetu je ke stažení mnoho veřejně dostupných databází fotografií, které je možné volně využít. Tyto sady obrazů bývají specializované pro využití v určitém oboru (lékařství, doprava, stavebnictví, bezpečnost) nebo mohou být přizpůsobeny k testování určitých úloh spojených s počítačovým viděním a strojovým učením (segmentace, porozumění scéně, sledování objektu, stereoskopie, atd.). Jako příklad nejrozšířenějších veřejně přístupných databází je možné uvést:

- ImageNet – Rozsáhlá databáze (přes 14 miliónů) sémanticky řazených kvalitních obrázků, jejichž výhodou je velká diverzita (rozdílnost ve vzhledu, směru pohledu, pozadí, pozici,…). Mimo databází obrázků je na stránkách možné získat taktéž databáze SIFT deskriptorů, „bounding boxů“ nebo atributů.

Obrázek 40: Ukázka množiny obrazů "volcanic crater" z http://image-net.org

- Caltech101/256 – menší databáze (101, resp. 256 tříd) obsahující řádově stovky obrazů pro každou třídu.

Obrázek 41: Ukázka množiny obrazů z http://www.vision.caltech.edu

- 80 million tiny images – Velká databáze obrázků v malém rozlišení (32x32). Dostupná z http://groups.csail.mit.edu/vision/TinyImages/
- LabelMe – Dataset obsahující digitální obrazy s anotacemi, jehož součástí je i „Annotation Tool“, který umožňuje vytvářet polygony okolo různých objektů v obraze a poté tento objekt pojmenovat.

Obrázek 42: Ukázka množiny obrazů z http://labelme.csail.mit.edu/Release3.0/

Pro demonstraci navržené implementace jsou využity dvě sady obrazů. První množina, nazvaná 4_object_categories, kterou je možné volně získat z http://www.micc.unifi.it/downloads/tutorial_bow/images/4_ObjectCategories.zip obsahuje 450 obrazů v každé ze 4 tříd a je složená z obrazů z databáze Caltech-101. Jedná se o letadla, auta, tváře a motocykly.

Obrázek 43: Příklad obrazů z množiny 4_object_categories

Druhá sada obrazů, nazvaná 15_scene, obsahuje 15 scén (ložnice, ulice, hory les, …) a je možné ji volně získat z http://www-cvr.ai.uiuc.edu/ponce_grp/data/. Příklad obrazů z obou použitých množin je na následujících obrázcích.

Obrázek 44: Příklad obrazů z databáze 15_scene. Postupně od prvního: Ložnice, předměstí, průmyslová zóna, kuchyně, obývací pokoj, pobřeží, les, dálnice, uvnitř města, hory, otevřená krajina, ulice, výškové budovy, kancelář a obchod.

3.2 Navržená implementace

Jak již bylo napsáno výše, projekt je vytvořen v prostředí Matlab. Jednotlivé bloky programu jsou popsány v následujícím textu:

Inicializace programu

Skript *init.m* obsahuje nastavení všech potřebných proměnných, které jsou v projektu použity, včetně cesty k datasetům, nastavení detektoru a deskriptoru

klíčových bodů, počet slov v BoW slovníku, nastavení klasifikátoru atd. Všechna tato nastavení jsou uložena do struktur, které přebírají ostatní funkce.

Načtení vstupních dat

Datasety se načítají pomocí funkce *[DATA_OUT] = getDataset (DATA_IN, verbalize)*. Funkce nejprve zkontroluje, jestli cesta k souboru, generovaná do proměnné *DATA_IN.DatasetPath,* existuje a načte zvolenou množinu obrazů. Tuto množinu rozdělí na *trainingSets a validationSets* podmnožiny a uloží je do pracovního adresáře *data\local*.mat.* Dělení datasetu se provádí podle volby učiněné ve skriptu init, kde se zadává počet trénovacích obrazů pro všechny třídy a volba, jestli se jako validační obrazy mají použít všechny zbývající obrazy ve třídě nebo jestli má být počet obrazů pro všechny třídy shodný. Podle hodnoty v proměnné *DATA_IN.DataseSorting* se trénovací a testovací obrazy zvolí náhodně nebo podle abecedy.

Detektor a deskriptor klíčových bodů

Detekce a deskripce klíčových bodů je provedena pomocí funkce *findKeyPoints (DATA, verbalize).* Funkce načte trénovací i testovací dataset a podle nastavení v *init.m* provede lokalizaci klíčových bodů pomocí zvoleného detektoru. V projektu jsou požity tři rozdílné detektory:

- Dense GRID
- SURF
- MSER

Rozdíl mezi body, nalezenými jednotlivými detektory, je jak kvantitativní tak kvalitativní.

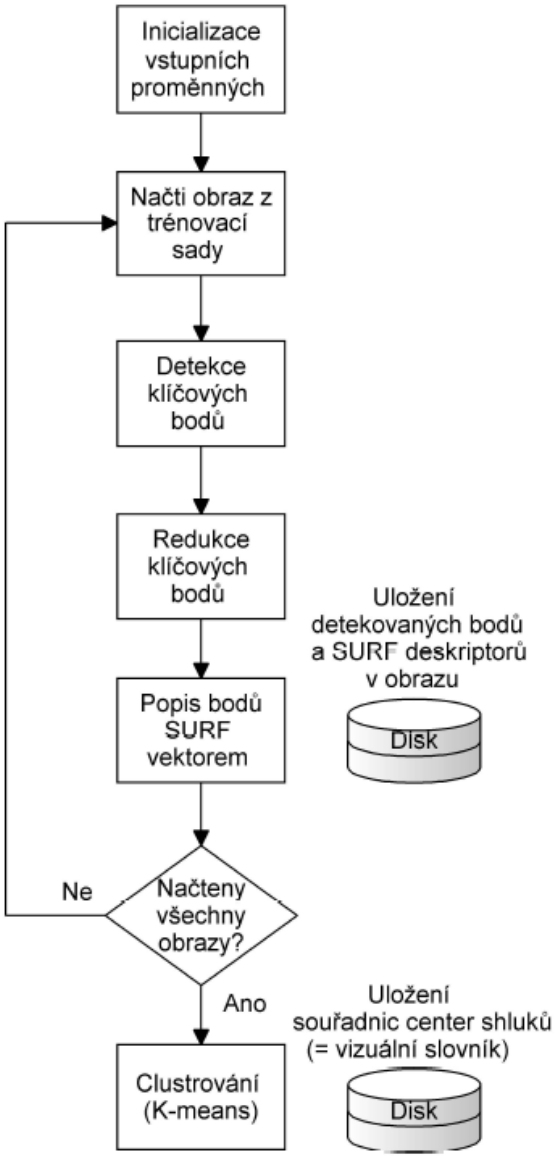
Všechny takto nalezené klíčové body jsou poté popsány pomocí SURF deskriptoru vektorem o 128 prvcích. Poloha nalezeného klíčového bodu, SURF vektor a další informace jsou potom pro každý obraz uloženy do zvláštní složky v adresáři Data\Local\<číslo obrazu>\features.mat.

V průběhu experimentování byl zjištěn pozitivní vliv redukce klíčových bodů na výslednou úspěšnost klasifikace. Z klíčových bodů pro celou třídu je proto nejprve vybráno 80% těch nejsilnějších (s největším determinandem hessianu) a následně je podle třídy s nejmenším počtem klíčových bodů oříznut i počet bodů pro třídy ostatní.

Tvorba BoW slovníku

BoW slovník je vytvořen pomocí funkce *createVocabulary (DATA, verbalize)*. Pomocí zvolené clustrovací metody a počtu slov ve slovníku jsou ze SURF vektorů

všech trénovacích obrazů ze všech tříd vypočítána centra clusterů. Souřadnice těchto center rozměrem odpovídají délce SURF vektorů a tvoří slova BoW slovníku. Slovník (souřadnice center shluků) je poté uložen do adresáře Data\Global\vocabulary.mat. Vývojový diagram, demonstrující tvorbu BoW slovníku je zobrazena na obrázku (viz. Obrázek 45)



Obrázek 45: Tvorba BoW slovníku

Popis obrazu pomocí BoW histogramu

Na základě BoW slovníku se pro každý obraz konstruuje histogram četnostní výskytu jednotlivých slov. Pro tento účel byla vytvořena funkce *getBoWHistogram (DATA, verbalize).* Funkce načte z adresáře obrazu matici s odpovídajícími klíčovými body popsanými SURF vektorem a pomocí zvolené metriky (v tomto případě prosté euklidovské vzdálenosti) nalezne pro každý bod nejpodobnější vektor v BoW slovníku. Po přiřazení BoW slova všem klíčovým bodů v obraze se z četnosti jejich výskytů sestrojí histogram. Tento histogram je normován tak, aby součet všech prvků byl roven jedné. Příklady BoW histrogramů jsou na obrázcích v příloze.

Rozšíření pomocí „Spatial pyramid“

Pro vytvořen spatial pyramid příznakového vektoru je v projektu zařazena funkce *createPyramids (DATA, verbalize).* Funkce postupně načítá všechny obrazy spolu s nalezenými klíčovými body a jejich popisem pomocí slova z BoW slovníku.

Podle zvolené výšky pyramidy je nejprve obraz rozdělen na nejmenší potřebné podoblasti (při výšce pyramidy 3 je to na 2^(3-1) * 2^(3-1) = 16 podoblasti) a v každé podoblasti je vypočítán histogram četnosti výskytu jednotlivých vizuálních slov. V dalším patře pyramidy jsou podoblasti rozšířeny tak,

aby pokrývali čtyři podoblasti z předchozího patra a opět se vypočítá jejich histogram. Poslední patro pyramidy odpovídá celému obrazu.

Jednotlivé nalezené histogramy jsou následně váhovány v závislosti na patře, ve kterém byly nalezeny, a spojeny do vektoru. Váhování

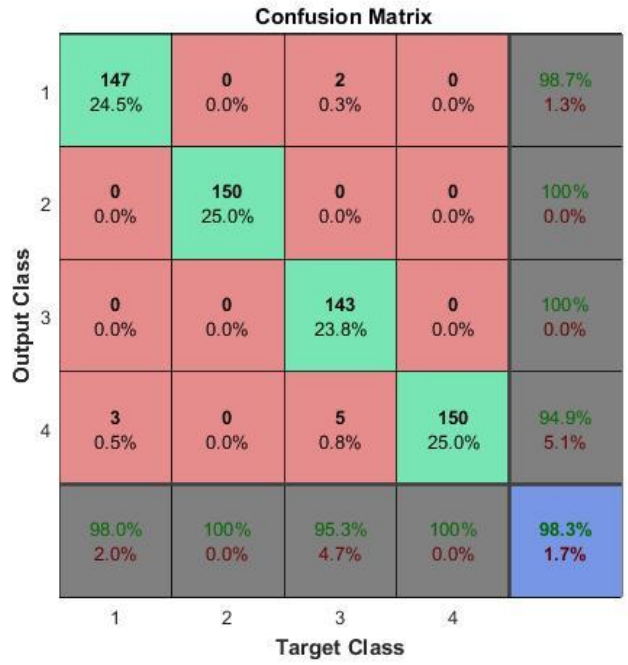
Trénování klasifikátoru

SVM klasifikátory slouží primárně pro klasifikaci do dvou tříd. Aby je bylo možné použít pro klasifikaci do více tříd, je nutné použít více SVM klasifikátorů a některou z metod uvedených v podkapitole SVM – Klasifikace do více tříd. V tomto projektu je použita metoda *One-agains-All*, pro které byl testován vliv různých jádrových funkcí (gaussian, rbf, ….) na úspěšnost výsledké klasifikace. V projektu je využito také jádro *histogram intersection*, které je popsáno v teoretické části práce.

Predikce takto natrénovanými SVM klasifikátory se pak provádí pomocí hlasovací strategie, která spočívá v přičítání hlasů jednotlivým vítězným třídám a jako výsledek klasifikace je určena třída s nejvíce hlasy.

Testování klasifikátoru

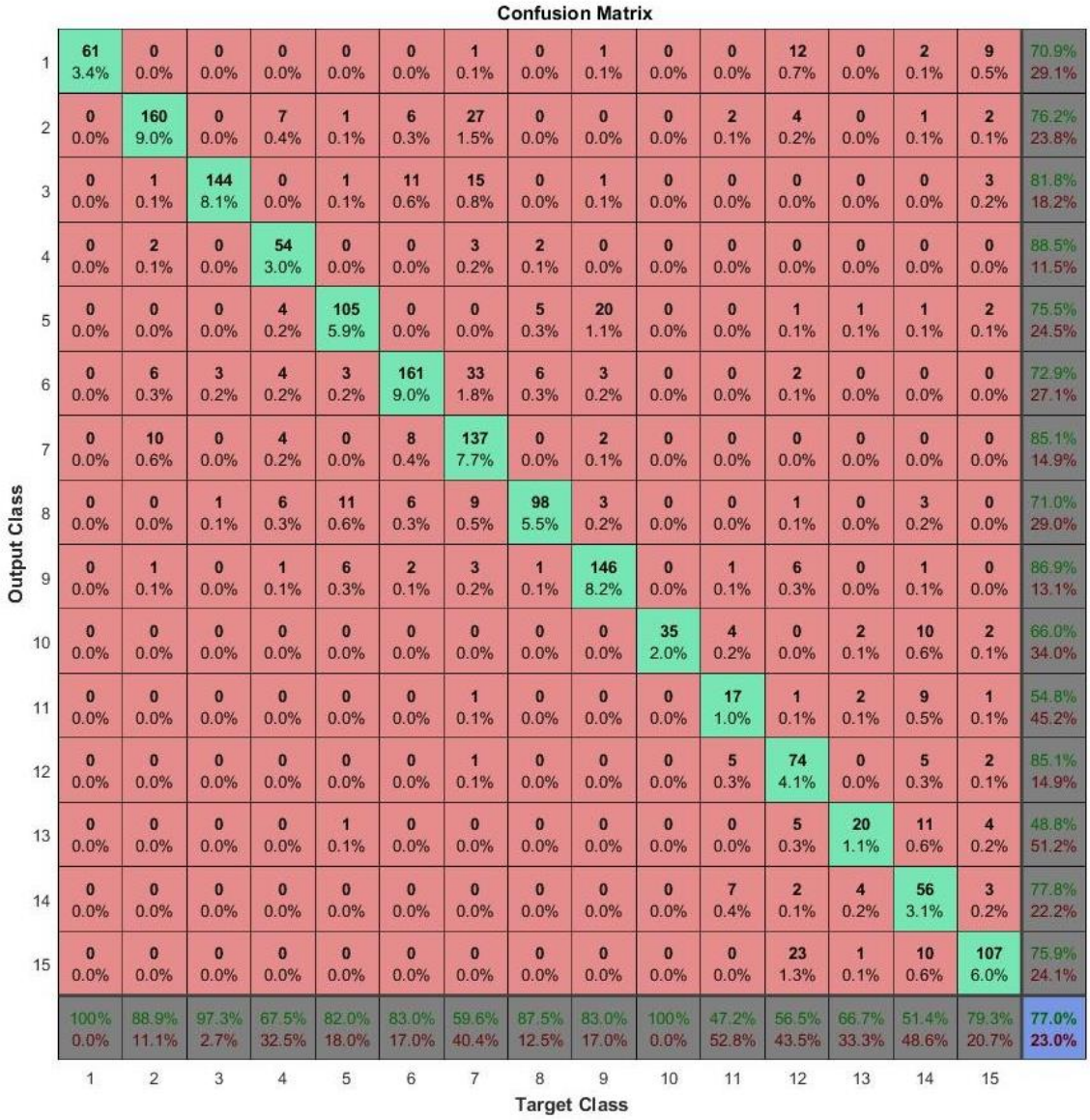
Na testování klasifikátoru je použito validační množiny, která byla vytvořena funkcí *getDataset(…)*. Výsledkem klasifikace je tzv. *comfusion matrix*, ze které je možné vyčíst jak celkovou úspěšnost klasifikace, tak i úspěšnosti klasifikací do jednotlivých tříd. Příklad *Confusion matrix* pro množinu obrazů 4_ObjectCategories je uveden na následujícím obrázku.



Obrázek 46: Confusion matrix získaná testování klasifikátoru na množině obrazů 4_ObjectCategories

Z *confusion matrix* je patrné, ve kterých případech dochází k chybné klasfikaci a díky tomu je možné upravit parametry modelu. Na obrázku výše je například vidět, že klasifikátor chybně klasifikoval 10 objektů z celkových 600 obrazů, přičemž nejčastěji dělal chybu při klasifikování třídy 3 (tváře), kterou chybně přiřadil třídě motocykly (5x) a třídě letadla (2x).

Na následujícím obrázku je *confusion matrix* vypočítaná v rámci praktické části práce na datasetu *15 scene*. Jak je vidět, nejčastěji docházelo k chybné klasifikaci třídy 7 (les), kterou v 27 případech klasifikátor přiřadil do třídy 2 (předměstí). Jednu z nejmenších úspěšností má klasifikace třídy 11 (otevřená krajina), která je ale způsobena zejména malým počtem testovacích obrazů.



Obrázek 47: Confusion matrix získaná testování klasifikátoru na množině obrazů 15_scene

Naopak, původní obavy z klasifikace dvojice tříd „obývací pokoj“ a „ložnice“ se ukázali jako chybné a klasifikátor zde dosahuje dobré úspěšnosti.

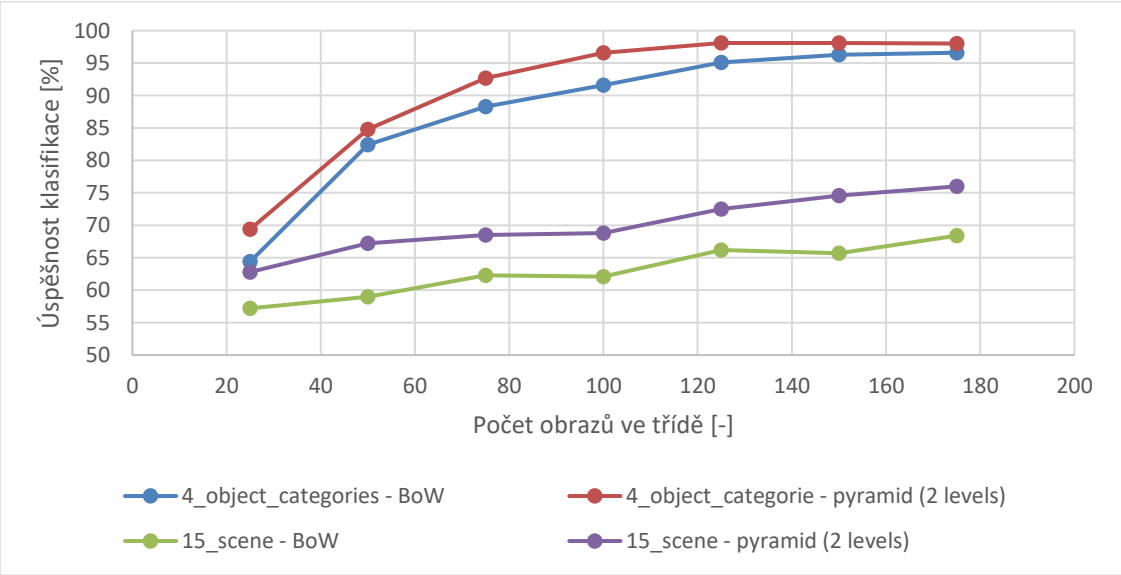
3.3 Výsledky

Při testování navrženého algoritmu a snaze dosáhnout co nejlepší úspěšnosti klasifikace, byla provedena řada experimentů, jejichž výsledky jsou uvedeny v této kapitole. Protože algoritmus byl navržen tak, aby byl schopný provést klasifikaci jakékoliv množiny obrazů obsahující jakékoliv objekty, jsou testy provedeny na dvou rozdílných množinách obrazů:

- Dataset „4 Object Categories“ obsahuje 4 třídy se 450 obrazy pro každou kategorii. Pro trénování modelu bylo použito 300 obrazů z každé třídy a pro testování bylo použito zbývajících 150.
- Dataset „15 Scenes“ obsahuje různý počet obrazů pro každou kategorii. K trénování modelu bylo použito 180 obrazů a pro testování zbývající počet obrazů ze všech tříd. Úspěšnost klasifikace výsledného modelu značně ovlivňují třídy s menším počtem obrazů, kde měla chybná klasifikace objektu v obrazu značný vliv na celkovou úspěšnost.

Vliv kvantity vstupních dat

Množství dostupných dat je jedním z nejvýznamnějších předpokladů pro tvorbu úspěšného klasifikátoru. Proto byl proveden experiment, který toto tvrzení dokazuje a výsledky jsou zobrazeny v grafu níže (viz. Obrázek 48). Z grafu je mimo jiné na první pohled patrný pozitivní přínos rozšíření příznakového vektoru pomocí *spatial pyramid.*



Obrázek 48: Vliv počtu vstupních dat na úspěšnost klasifikace

Pro testovací sadu 4_object_categories je obstojné úspěšnosti dosaženo již při 120 trénovacích obrazech v jednotlivých třídách. Je ovšem nutné podotknout, že množina obsahuje pouze 4 třídy objektů a nutný počet trénovacích obrazů proto není nutný příliš vysoký.

Pro komplexnější scény z testovací sady 15_scene je tomu jinak. Z naměřených údajů je patrný trend zvyšování úspěšnosti klasifikace s přibývajícím počtem trénovacích obrazů, který by patrně při dalším navyšování obrazů dále rostl.

Vliv použitého detektoru

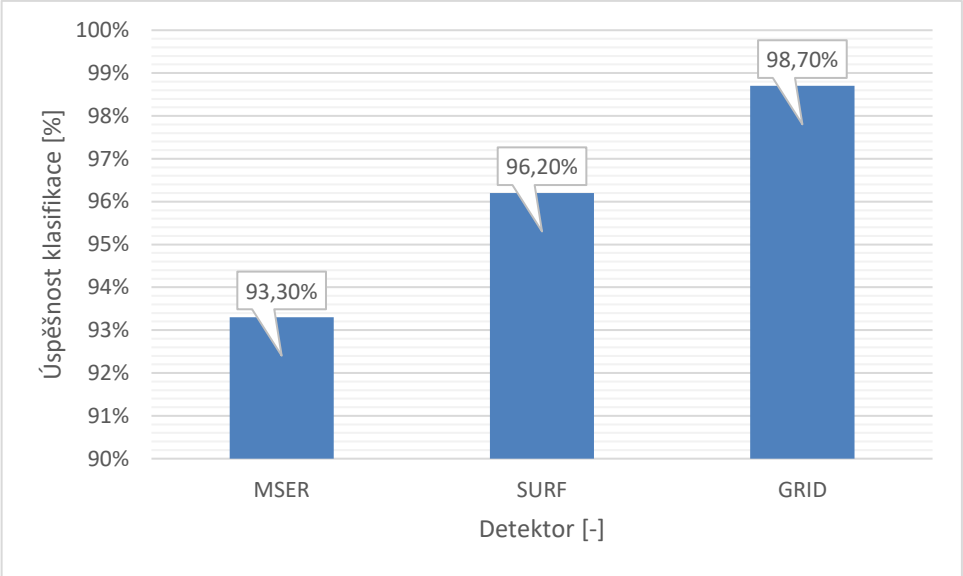
Dalším parametrem, který silně ovlivňoval výslednou úspěšnost klasifikace, byl typ použitého detektoru klíčových bodů. V projektu byly použity tyto tři:

- Dense GRID – Jedná se o prosté odebrání vzorků podle stanovené mřížky [26]. Výhodou je možnost odebrání vysokého počtu vzorků z obrazu, což může být vhodné zejména při klasifikaci scén. Pro každý vzorek je navíc možné vypočítat ještě
- SURF – viz. kapitola SURF. Detektor vyhledává klíčové body v integrálním obraze a bod označí, pokud je determinant hessianu vyšší než zvolený práh.
- MSER – (angl. *Maximally Stable Extremal Regions*) je algoritmus, který prahuje obraz se stále se zvyšujícím prahem. Oblasti, které jsou patrné na více úrovních prahu algoritmus označí jako stabilní [27].

Jak z grafů níže vyplývá, výsledná úspěšnost klasifikace rostla společně s množstvím bodů detekovaných v jednotlivých obrazech.

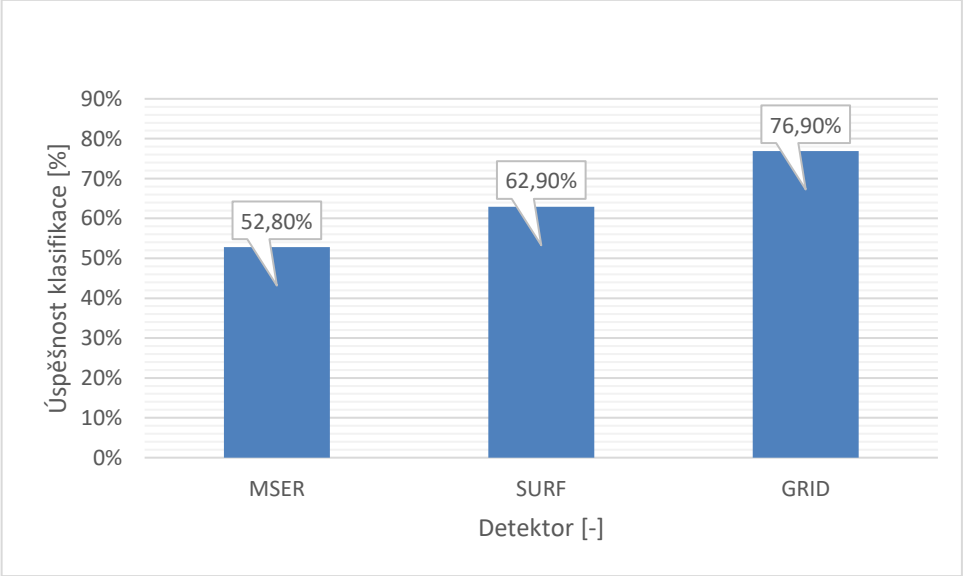
Pro množinu *4_object_*categories je graf uveden na Obrázek 49. Při použití MSER detektoru, bylo při tvorbě BoW slovníku použito celkově cca. 50477 bodů. SURF detektor s prahem 300 získal na stejné množině obrazů asi 101877 bodů. Nejvíce bodů z obrazu je možné obdržet pomocí GRID detektoru, jehož mřížka (gridStep) byla zvolena 8x8 pixelů, až kolem 1154904 klíčových bodů (10x více oproti SURF detektoru). Přestože je pomocí tohoto detektoru získáno mnoho irelevantních bodů, klasifikátor založený na bodech z tohoto detektoru dává nejlepší výsledky.

Ukázka bodů, detekovaných jednotlivými detektory jsou na obrázcích v příloze této práce.



Obrázek 49: Vliv použitého detektoru klíčových bodů na úspěšnost klasifikace množiny obrazů 4_object_categories

Na následujícím grafu (viz. Obrázek 50) je znázorněn vliv použitého detektoru na úspěšnost klasifikace u druhé testované množiny obrazů – *15_scene*. I v tomto případě vycházel jako nejlepší GRID detektor, který (stejně jako u předchozí množiny obrazů) našel nejvíce klíčových bodů.



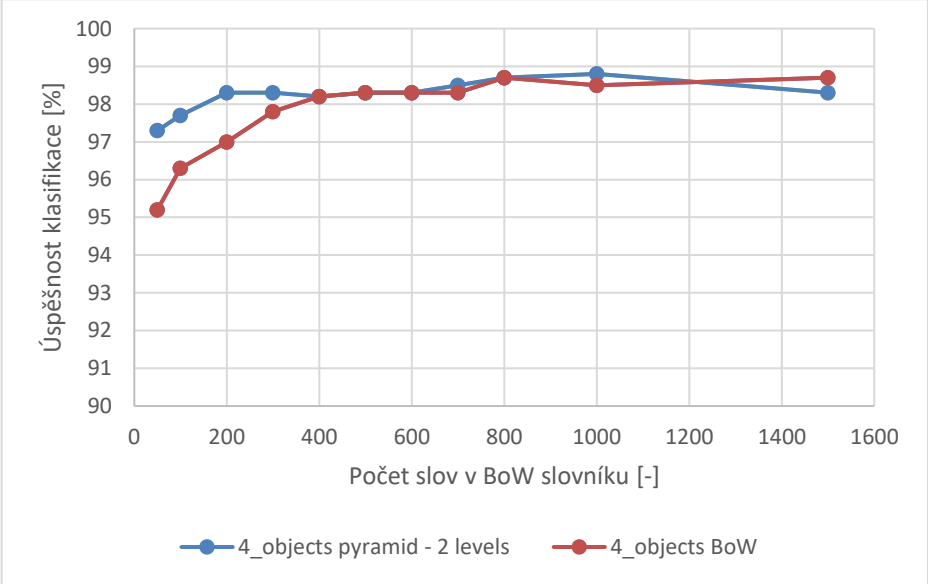
Obrázek 50: Vliv použitého detektoru klíčových bodů na úspěšnost klasifikace množiny obrazů 15_scene

Příklady detekovaných klíčových bodů v obraze jsou přiloženy v příloze.

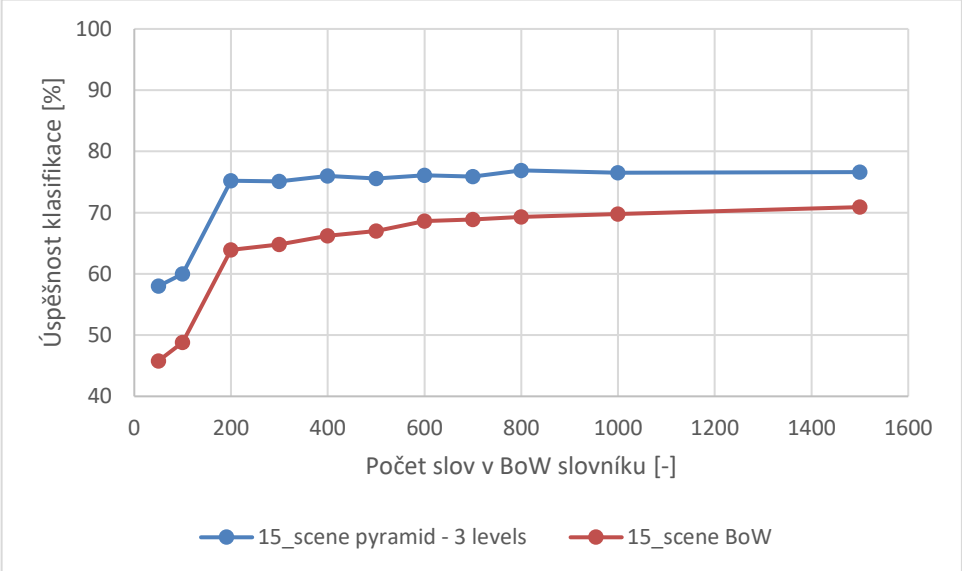
Vliv velikosti BoW slovníku

Po získání dostatečného počtu klíčových bodů bylo nutné stanovit, jakou velikost bude mít *Bag-of-words* slovník, tzn. kolik vizuálních slov bude obsahovat. Výsledky v grafech uvedených níže ukazují, že rostoucí velikost slovníku má pozitivní vliv na úspěšnost klasifikace až do určitého počtu, kde již přesnost zůstává téměř konstantní. Velikost slovníku je tedy nejvhodnější zvolit na konci tohoto zlomu, jelikož s dalším zvětšováním slovníku již nedochází k znatelnému zlepšení přesnosti, ale výpočetní čas se stále prodlužuje.

Dále byl zjištěn pozitivní vliv redukce klíčových bodů před tvorbou slovníku (řádově jednotky procent).



Obrázek 51: Vliv velikosti BoW slovníku pro množinu obrazů 4_object_categories

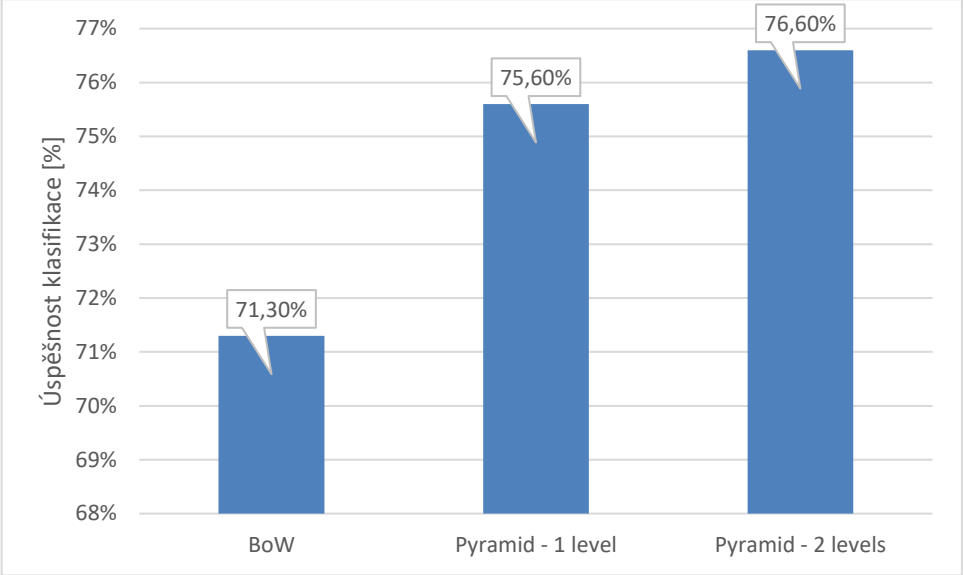


Obrázek 52: Vliv velikosti BoW slovníku pro množinu obrazů 15_scene

Jak je patrné z grafů, pro obě testované množiny obrazů se jako nejvhodnější velikost slovníku jeví 500 slov. V této hodnotě je dosaženo nejlepšího kompromisu mezi přesností a délkou výpočetního času, který je na shlukování potřebný.

Vliv výšky spatial pyramid

Metoda spatial pyramid v principu rozšiřuje příznakový prostor pro trénování SVM klasifikátoru. Výhodou této metody je tedy zachování výpočetního času při tvorbě BoW slovníku, ale patrné je prodloužení výpočetního času při trénování SVM klasifikátoru.

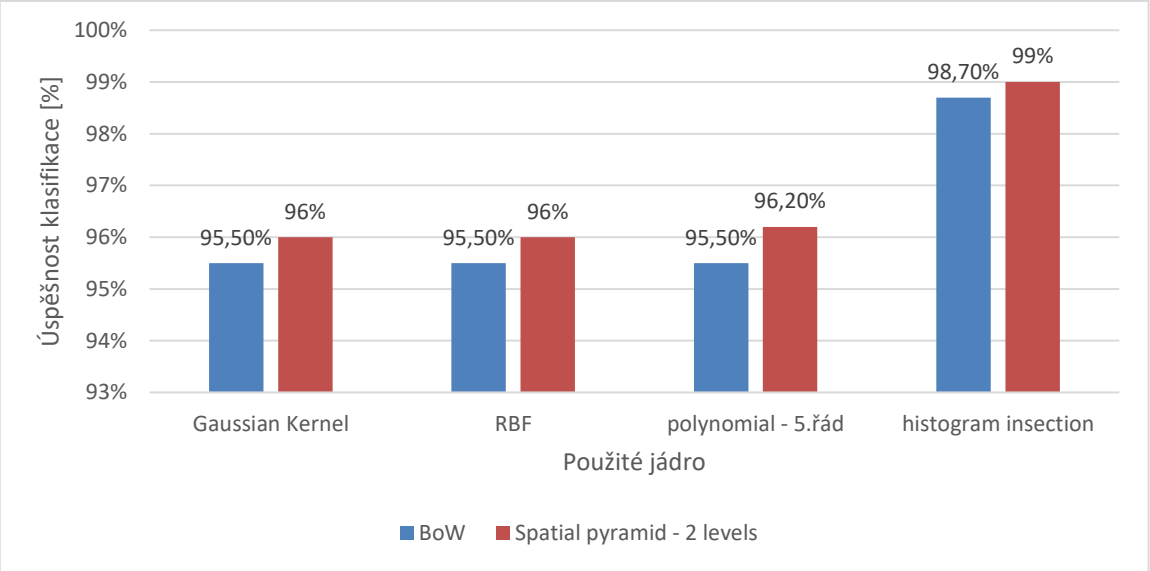


Obrázek 53: Vliv výšky pyramidy na úspěšnost klasifikace množiny obrazů 15_scene

Zlepšení úspěšnosti klasifikace s rostoucí výškou pyramid je jasně vidět z obrázku výše. Rozdíl mezi klasickou metodou BoW a vektorem příznaků, získaného pomocí spatial pyramid, činí 5,3 % - podmínkou je ale využití váhovaní a histogram intersection jádra SVM.

Vliv jádrové funkce SVM klasifikátoru

Posledním testovaným parametrem byl vliv jádrové funkce SVM klasifikátoru. Na následujících grafech je znázorněn vliv gaussian, rbf a histogram intersection (v případě spatial pyramid jde o upravenou verzi s váhováním) jádra na úspěšnost klasifikace.



Obrázek 54: Vliv jádra na úspěnost klasifikace

Kladný přínos spatial pyramid je opět více než patrný. V kombinaci s histogram intersection bylo dosaženo až 99 % úspěšnosti klasifikace objektů z množiny 4_object_categories. Oproti tomu klasické BoW s gaussian jádrem dosahovalo úspěšnosti pouhých 95,5 %.

Nejlepší dosažené výsledky

Dataset „4 Object Categories“ obsahuje 4 třídy se 450 obrazy pro každou kategorii. Pro trénování modelu bylo použito 300 obrazů z každé třídy a pro testování bylo použito zbývajících 150. Dataset „15 Scenes“ obsahuje různý počet obrazů pro každou kategorii. K trénování modelu bylo použito 180 obrazů a pro testování zbývající počet obrazů ze všech tříd. Pro tyto dvě množiny obrazů bylo nalezeno shodné nejlepší nastavení, které je popsané níže.

Výsledky testování klasifikátoru na zvolených datasetech jsou zobrazen v tabulce (viz: Tabulka 1). Po nalezení nejvhodnějšího nastavení modelu bylo pro každou testovanou množinu obrazů provedeno 5 trénovacích a validačních cyklů na náhodně zvolených obrazech. Z těchto pěti hodnot úspěšností klasifikace byl vypočítán průměr, který je uveden v tabulce.

TESTOVANÝ MODEL	4_OBJECT_CATEGORIES	15_SCENE
BOW + SVM	97,8 %	67 %
SPATIAL PYRAMID + SVM	98,54 %	76,65 %

Tabulka 1: Dosažená úspěšnost klasifikace

Po detekci klíčových bodů pomocí GRID detektoru a jejich popsání pomocí SURF vektoru byl počet bodů pro každou třídu redukován 80% nejsilnějších a poté

byly množiny SURF vektorů jednotlivých tříd zmenšeny podle nejmenšího počtu (tzn. každá trénovací třída přispěla pro tvorbu BoW slovníku stejným počtem SURF vektorů). Slovník byl vytvořen pomocí K-Means shlukovacího algoritmu tak, aby nalezl 800 center shluků = vizuálních slov, která posloužila pro popis všech nalezených klíčových bodů v testovací sadě pomocí indexu, který označuje nejpodobnější slovo ve slovníku (k měření podobnosti byla použita euklidovská vzdálenost).

Příznakový vektor pro trénování SVM klasifikátoru metodou *one-against-all* byl vytvořen pomocí algoritmu spatial pyramid, který v obrazu s vyznačenými vizuálními slovy rozšíří původní příznakový vektor, tvořený histogramem BoW, pomocí váhovaných histogramů.

ZÁVĚR

V teoretické části této práce byl proveden rozbor konceptů strojového učení v počítačovém vidění a ukázáno, jak se tyto obory prolínají. Velká pozornost byla přitom věnována statistickým modelům a jejich rozdělení do dvou skupin – diskriminativních a generativních.
V kapitole 2 byly nejprve rozebrány základní fáze konceptu strojového učení, jmenovitě: popis objektu, učení a rozpoznávání. Poté bylo v této kapitole podrobněji popsáno několik typů statistických modelů.

Metodě *Bag-of-words* byla vyhrazena kapitola 2.4. kde je uveden princip této metody, používané názvosloví a popis modifikací, které zlepšují výsledky při použití této metody.

V praktické části této práce byla implementována metoda *Bag-of-words* a její modifikace využívající spatial pyramid. Pro klasifikaci byl využit diskriminativní model SVM. Na volně dostupných množinách obrazů byla provedena řada testů. Pro množinu obrazů 4_object_categories, který obsahuje 4 třídy objektů, bylo dosaženo úspěšnosti klasifikace 98,54 %. Další testovanou množinou, která pro změnu obsahoval 15 scén (15_scene) bylo dosaženo úspěšnosti 76,65 %. Rozborem vlivů parametrů bylo dokázáno, jaký efekt má na výslednou úspěšnost klasifikace počet vstupních obrazů v jednotlivých třídách, velikost BoW slovníku, zvolený detektor klíčových bodů, výška spatial pyramid a jádrová funkce SVM klasifikátoru.

Dostatek trénovacích dat je důležitý zejména při klasifikaci komplexních scén z množiny 15_scene, kde byl trend křivky, určující úspěšnost klasifikace, s rostoucím počtem trénovacích dat stále rostoucí. Lze tedy předpokládat, že při větší množině trénovacích dat by bylo dosaženo větší úspěšnosti. Pro druhou z testovacích sad (4_object_categories) bylo dosaženo obstojné úspěšnosti (96,6 %) již při 100 obrazech v jednotlivých třídách.

Dalším prověřovaným parametrem byl počet slov v BoW slovníku. Jak bylo znázorněno ve výsledných grafech (viz. Obrázek 51 a Obrázek 52), ve všech případech byl zaznamenán prudký růst úspěšnosti klasifikace do cca. 500-ti slov ve slovníku a poté se již úspěšnost zlepšovala jen velmi pozvolna. Pro tvorbu BoW slovníku bylo ovšem důležité získat dostatek relevantních příznakových vektorů z jednotlivých obrazů. K tomuto účelu byla napsána funkce pro redukování klíčových bodů, která na ně byla aplikována až po získání všech klíčových bodů ze všech trénovacích obrazů a zlepšovala úspěšnost klasifikace řádově o jednotky procent.

Vliv výšky spatial pyramid byl nejpatrnější pro slovníky s malým počtem slov. Díky této modifikaci docházelo k rozšíření příznakového prostoru pro trénování SVM klasifikátoru při nezměněné velikosti BoW slovníku.

Posledním testovaným parametrem byla jádrová funkce SVM klasifikátoru, kde se podle očekávání nejlépe uplatnilo jádro histogram intersection a jeho modifikace pro váhování sdružených histogramů získaných pomocí spatial pyramid.

Literatura

[1] HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J. H. *The elements of statistical learning: data mining, inference, and prediction*. 2nd ed. New York, NY: Springer, c2009. ISBN 978-0-387-84858-7.

[2] ŠVEC, J. *Diskriminativní model pro porozumnění mluvené řeči*. Plzeň: Západočeská univerzita v Plzni, Fakulta aplikovaných věd, 2013.

[3] HORÁL, K., KALOVÁ, I., PETYOVSKÝ. P., RICHTER, M. *Počítačové vidění*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008.

[4] EGMONT-PETERSEN, M., DE RIDDER, D., HANDELS, H. *Image processing with neural networks - a review.* Pattern Recognition, Vol. 35, No. 10, pp. 2279-2301, 2002.

[5] HORÁK, K. *Popis objektů* [online]. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií [cit. 2017-05-14]. Dostupné z: http://midas.uamt.feec.vutbr.cz/POV/Lectures/08_Popis_objektu.pdf. Prezentace k předmětu LPOV.

[6] HORÁK, K., KLEČKA, J. *Deskriptory oblastí* [online]. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií [cit. 2017-05-14]. Dostupné z: http://midas.uamt.feec.vutbr.cz/ROZ/Lectures/05_Deskriptory_oblasti.pdf. Prezentace k předmětu LPOV.

[7] LOWE, D.G. Object recognition from local scale-invariant features. *International Journal of Computer Vision.* 2004, Vol. 60, stránky 91-110.

[8] BAY, H., ESS, A., TUYTELAARS, T., aj. *Speeded-Up Robust Features (SURF)*. Comput. Vis. Image Underst., ročník 110, č. 3, Červen 2008: s. 346–359, ISSN 1077-3142.

[9] TUČKOVÁ, J. *Vybrané aplikace umělých neuronových sítí při zpracování signálu.* Praha : České vysoké učení technické v Praze, 2009. ISBN 978-80-01-04229-8.

[10] SAMMUT, C., WEBB, G.I. *Encyclopedia of machine learning.* London : London: Springer, 2010. ISBN 978-0-387-30768-8.

[11] HLAVÁČ, V., ŠONKA, M. *Počítačové vidění*. Praha: Grada, 1992. ISBN 80-85424-67-3.

[12] HORÁK, K. *Porozumění obsahu obrazu* [online]. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií [cit. 2017-05-14]. Dostupné z: http://midas.uamt.feec.vutbr.cz/ROZ/roz_cz.php. Prezentace k předmětu LPOV.

[13] ŠŤASTNÝ, J. *Netradiční metody a algoritmy pro rozpoznávání objektů technologické scény: Nontraditional methods and algorithms for object recognition of technilogical scene : Zkrácená verze habilitační práce.* Brno : VUTIUM, 2006. ISBN 80-214-3117-2.

[14] JIRSÍK, V. *Vícevrstvá neuronová síť s algoritmem učení backpropagation* [online]. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií [cit. 2017-05-14]. Prezentace k předmětu LUIN.

[15] JIRSÍK, V. *Hopfieldova síť* [online]. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií [cit. 2017-05-14]. Prezentace k předmětu LUIN.

[16] JIRSÍK, V. *Kohonenova samoorganizační mapa* [online]. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií [cit. 2017-05-14]. Prezentace k předmětu LUIN.

[17] RABINER, L. R. *A tutorial on hidden Markov models and selected applications in speech recognition. IEEE.* February, 1989, Vol. vol. 77, no. 2.

[18] TRKAL, O**.** *Rozpoznávání SPZ.* Brno : Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. str. 88. Vedoucí diplomové práce Ing. Karel Horák, Ph.D..

[19] LISA Lab**.** Deep Learning Tutorial: Release 0.1. *Deeplearning.net.* [Online] University of Montreal, 2015. Dostupné z: http://deeplearning.net/tutorial/deeplearning.pdf.

[20] SWERSKY K., CHEN B., MARLIN B., DE FREITAS N**.** *A Tutorial on Stochastic Approximation Algorithms for Training Restricted Boltzmann Machines and Deep Belief Nets.* BC, Canada : University of British Columbia.

[21] BARTOŠ, P**.** *Automatický výběr reprezentativních fotografií, diplomová práce.* Brno : FIT VUT v Brně, 2011.

[22] CSURKA, G., et al. *Vusual Categorization with Bags of Keypoints.* s.l. : ECCV 1, 2004.

[23] GAJOVÁ, V. *Automatické třídění fotografií podle obsahu*, diplomová práce, Brno, FIT VUT v Brně, 2012

[24] LAZEBNIK, S., SCHMID, C., PONCE, J. *Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories.* 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Volume 2 CVPR06, ročník 2, č. 2169-2178, 2006: s. 2169–2178.

[25] PASSALIS, N., TEFAS, A. *Neural Bag-of-Features learning.* Pattern Recognition*.* April 2017, 2017, Vol. 64, pp. 27

[26] TUYTELAARS, T. *Dense Interest Points* [online]. K.U.Leuven, ESAT - PSI [cit. 2017-05-14]. Dostupné z:

https://lirias.kuleuven.be/bitstream/123456789/263727/2/Tuytelaars_dip_CVPR2010.pdf

[27] NISTÉR, D., STEWÉNIUS, H. (2008) Linear Time Maximally Stable Extremal Regions. In: Forsyth D., Torr P., Zisserman A. (eds) Computer Vision – ECCV 2008. ECCV 2008. Lecture Notes in Computer Science, vol 5303. Springer, Berlin, Heidelberg

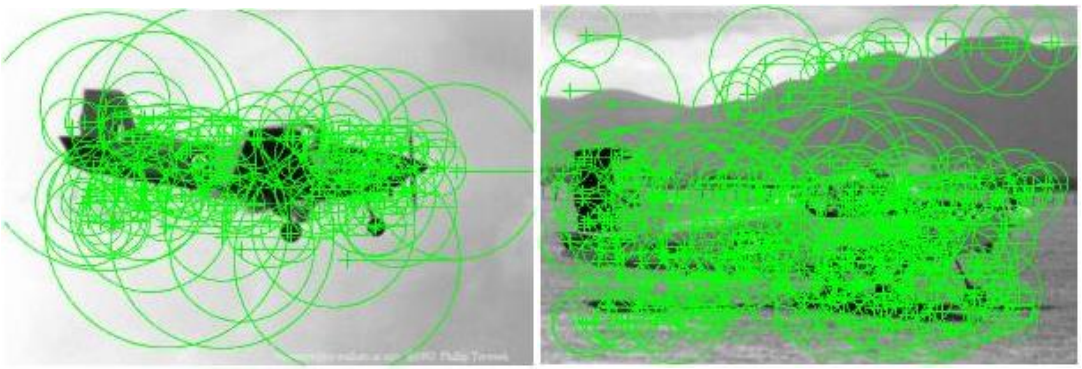
Seznam příloh

A OBSAH CD

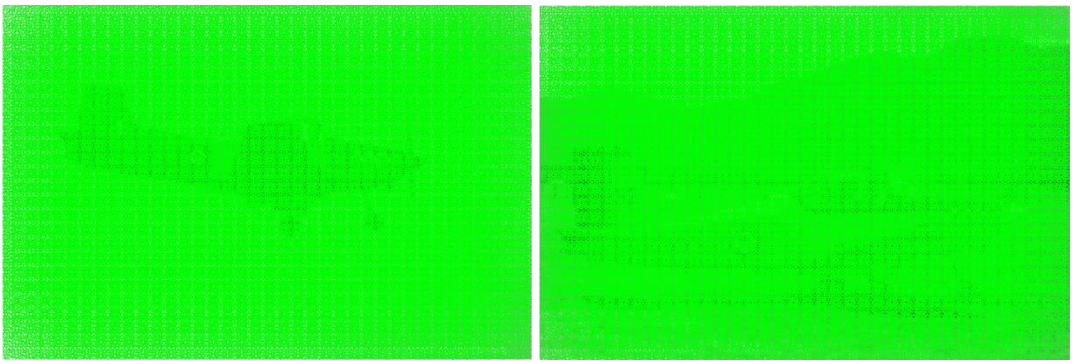
1) Diplomová práce v elektronické podobě
2) Projekt v prostředí Matlab R2015b
3) Množina obrazů „4_object_categories“
4) Množina obrazů „15_scene“

B DETEKTORY KLÍČOVÝCH BODŮ

Příklad klíčových bodů získaných pomocí MSER detektoru (popis pomocí SURF deskriptoru)



Příklad klíčových bodů získaných pomocí SURF detektoru (popis pomocí SURF deskriptoru)

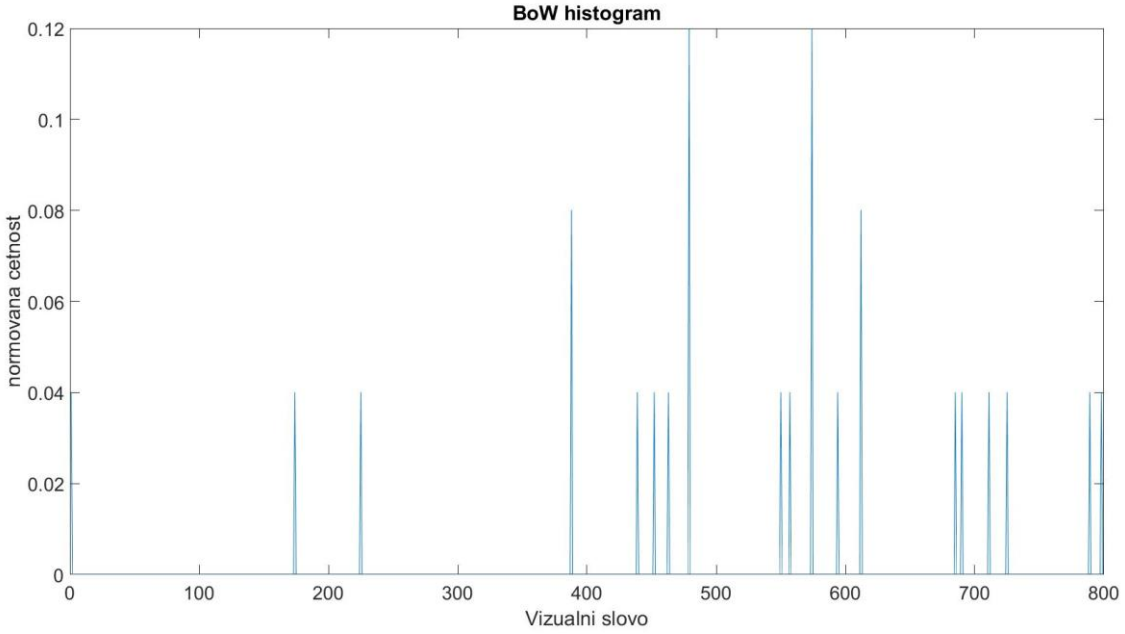


Příklad klíčových bodů získaných pomocí GRID detektoru (popis pomocí SURF deskriptoru)

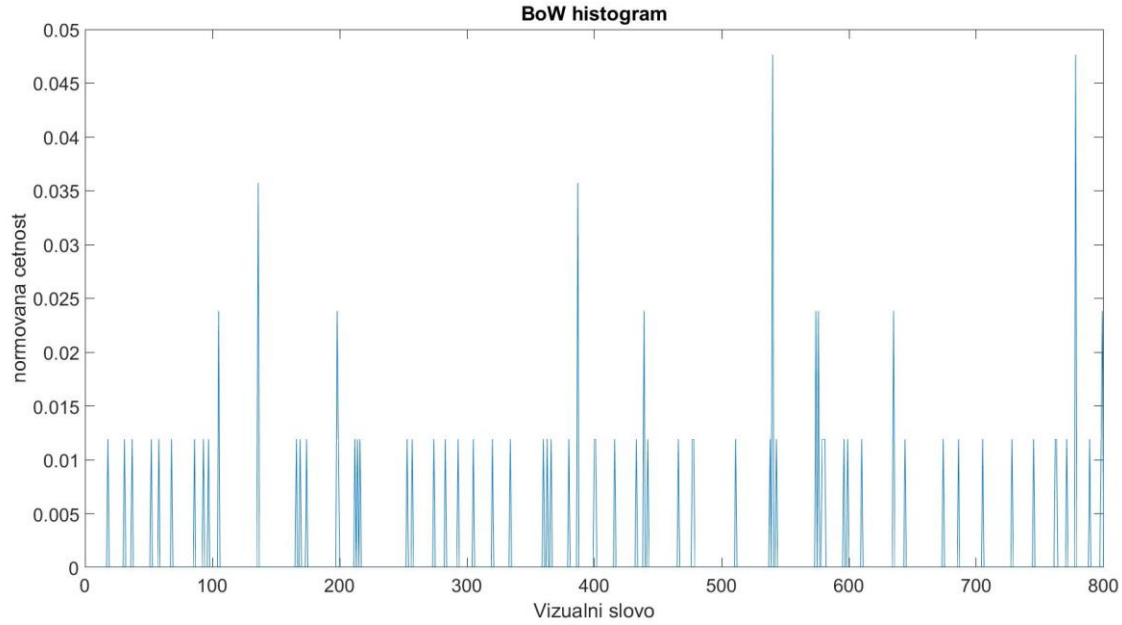
C BOW HISTOGRAMY

Obrázek „Letadlo 1“

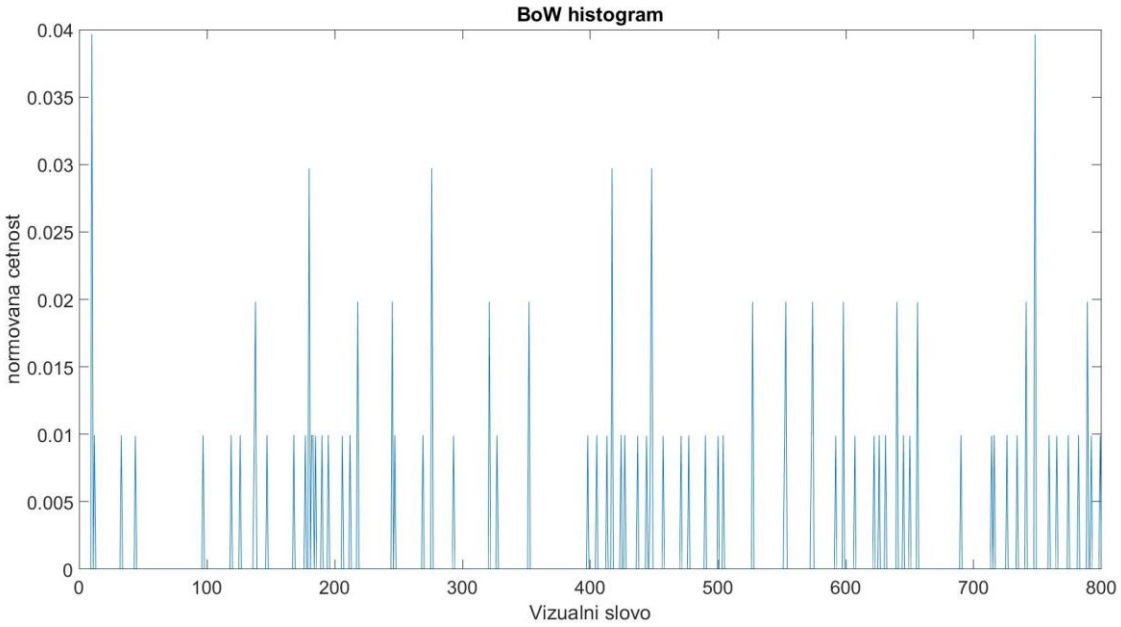
Obrázek „Letadlo 2“



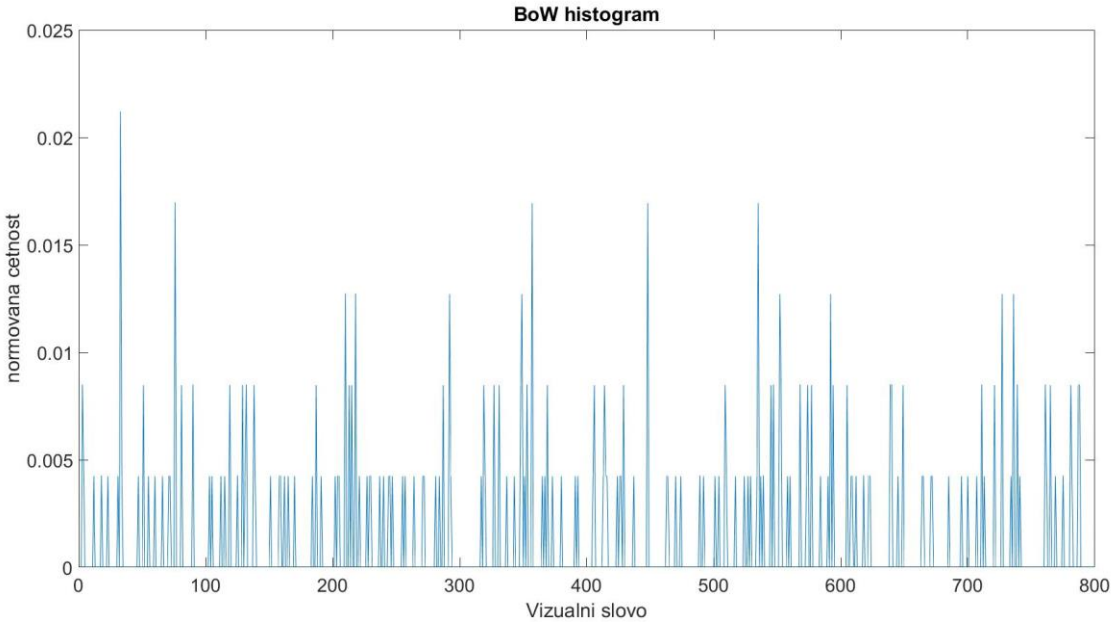
BoW histogram obrázku „Letadlo 1“ při použití MSER detektoru



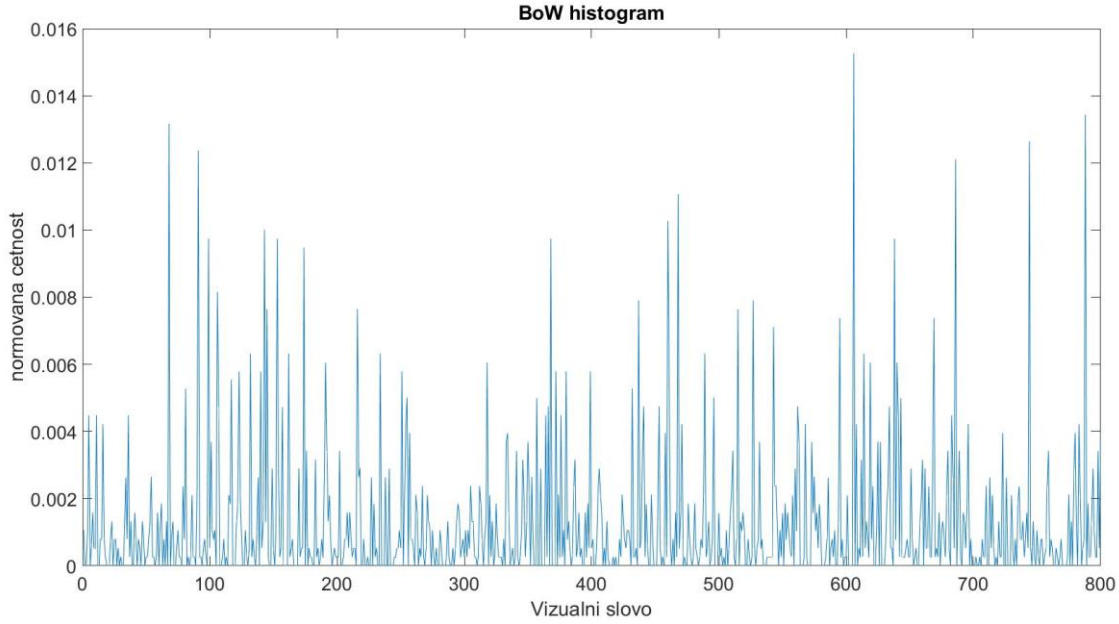
BoW histogram obrázku „Letadlo 2“ při použití MSER detektoru



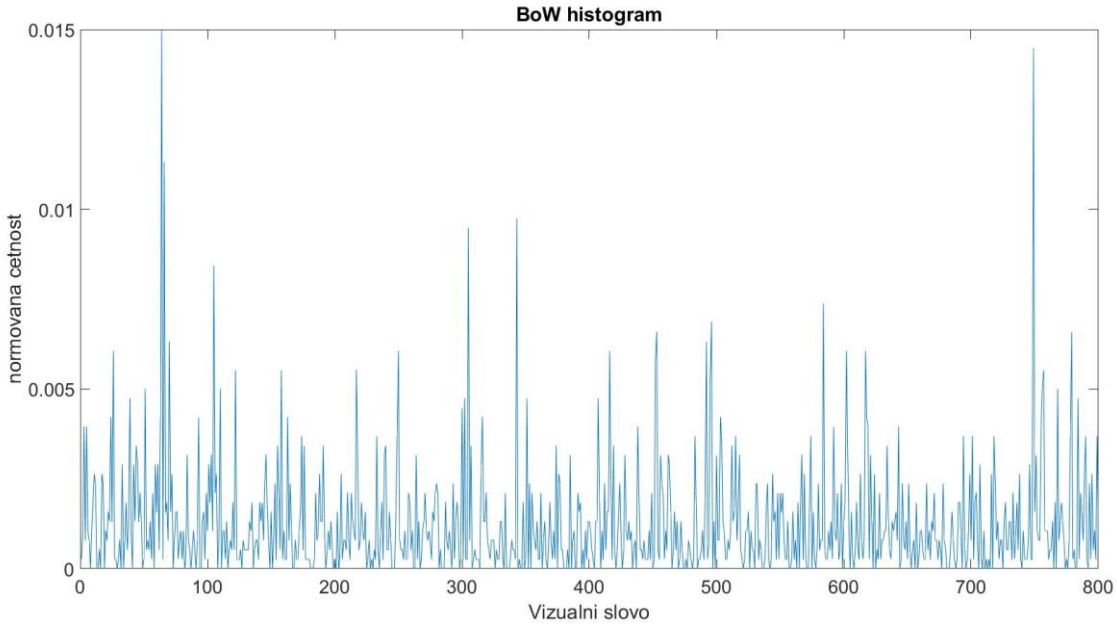
BoW histogram obrázku „Letadlo 1“ při použití SURF detektoru



BoW histogram obrázku „Letadlo 2“ při použití SURF detektoru



BoW histogram obrázku „Letadlo 1“ při použití GRID detektoru



BoW histogram obrázku „Letadlo 2“ při použití GRID detektoru