Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Számítástudományi és Információelméleti Tanszék

Aggregált kernel alapú képi klasszifikáció

TDK dolgozat

2011

Nikházy László

Konzulensek:

Daróczy Bálint Zoltán
MTA SZTAKI

Dr. Ketskeméty László
BME SZIT

Absztrakt

Vizuális tartalmak osztályozása egyike azon képfeldolgozási és gépi tanulási feladatoknak, melyek nem mesterséges körülmények között nehéz problémának tekinthetőek. A téma jelentőségét mutatja a sok kutatási eredmény mellett, hogy az elmúlt évek során számos ipari alkalmazás készült pár éve még kísérleti algoritmusok felhasználásával (akadály detekció autókon, orvosi képi segítő diagnosztikai eszközök, arcfelismerés, infra kamerás hazugságvizsgálat stb.). Természetes fotók esetében az emberi agy általában könnyen megállapítja, hogy a képre jellemző-e valamely tulajdonság, legyen az objektum vagy egy általános fogalom. (nyár, tél, nyaraló emberek, vidám pillanat stb.) Azonban ezen objektumok, fogalmak vizuális változatossága miatt a ma ismert algoritmusok számításigényük ellenére is igen nagy szórással hibáznak. Sok százezer kategória megfelelő pontosságú felismerése rengeteg gyakorlati alkalmazásra adna lehetőséget (például pontosabb, tartalom alapú képkeresés, digitális növény/állathatározó, esemény és környezet felismerése állóképeken, hamisítványok, romlott ételek detektálása stb.), épp ezért az utóbbi években növekvő érdeklődés kíséri a képi klasszifikáció kutatást. Mivel a feladat pontosan meghatározott, így egy-egy megoldás adott körülmények között összehasonlítható más algoritmusokkal. Épp ezért több rangos képi klasszifikációs versenyt is rendeznek immár több mint 5 éve (pl. Pascal VOC [1], ImageCLEF Photo Annotation [2]).

A versenyeken elért eredmények is mutatják, hogy az elmúlt néhány év során a képi klasszifikációs eljárások terén komoly előrelépések történtek. Ennek oka az általános számítási kapacitás megnövekedése mellett a képi low-level leírók majd pedig a bag-of-words módszerek fejlődése. Dolgozatomban bemutatom a state-of-the-art rendszerekben alkalmazott technikákat, mind az alacsony szintű leírók terén, mind pedig a magas szintű, szemantikai leírók terén (például a Gaussian Mixture Model alapú Fisher-vektor [3], vagy a K-means alapú Super-Vector [4]).

Az egyes alapmódszerek a különböző kategóriákban más-más eredményeket szolgáltatnak, célszerű előnyeiket ötvözni. Dolgozatomban ehhez javaslok egy új módszert, amelyben egy egyesített kernel mátrixot készítünk a különböző szemantikai leírókból. Ez az általános megközelítés lehetőséget nyújt arra, hogy tetszőleges modalitásból származó információkat kombináljunk, az egyes kategóriákra optimális módon. A módszer értékeléséhez szükséges a képi klasszifikációban alkalmazott új technikák implementációja is. A kernel aggregáló módszeremet a Pascal VOC 2007 és a Photo Annotation 2011 adathalmazon tesztelem, és az eredményeket összehasonlítom az eddig publikált legjobb eredményekkel.

Irodalom:

1. Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. and Zisserman, A. : *"The PASCAL Visual Object Classes (VOC) Challenge"* International Journal of Computer Vision, 88(2), 303-338, 2010
2. Stefanie Nowak, Mark Huiskes: "*New Strategies for Image Annotation: Overview of the Photo Annotation Task at ImageCLEF 2010"* Conference on Multilingual and Multimodal Information Access Evaluation 20-23 September 2010, Padua
3. Csurka, G., Perronnin, F., Marchesotti, L., Clinchant, S., Ah-Pine, J.: *"Fisher Kernel Representation of Images and Some of Its Successful Applications"*, VISAPP (2010) 21-25.
4. Xi Zhou, Kai Yu, Tong Zhang and Thomas S. Huang: *„Image Classification Using Super-Vector Coding of Local Image Descriptors"*, ECCV 2010, Lecture Notes in Computer Science, 2010, Volume 6315/2010, 141-154

Tartalomjegyzék

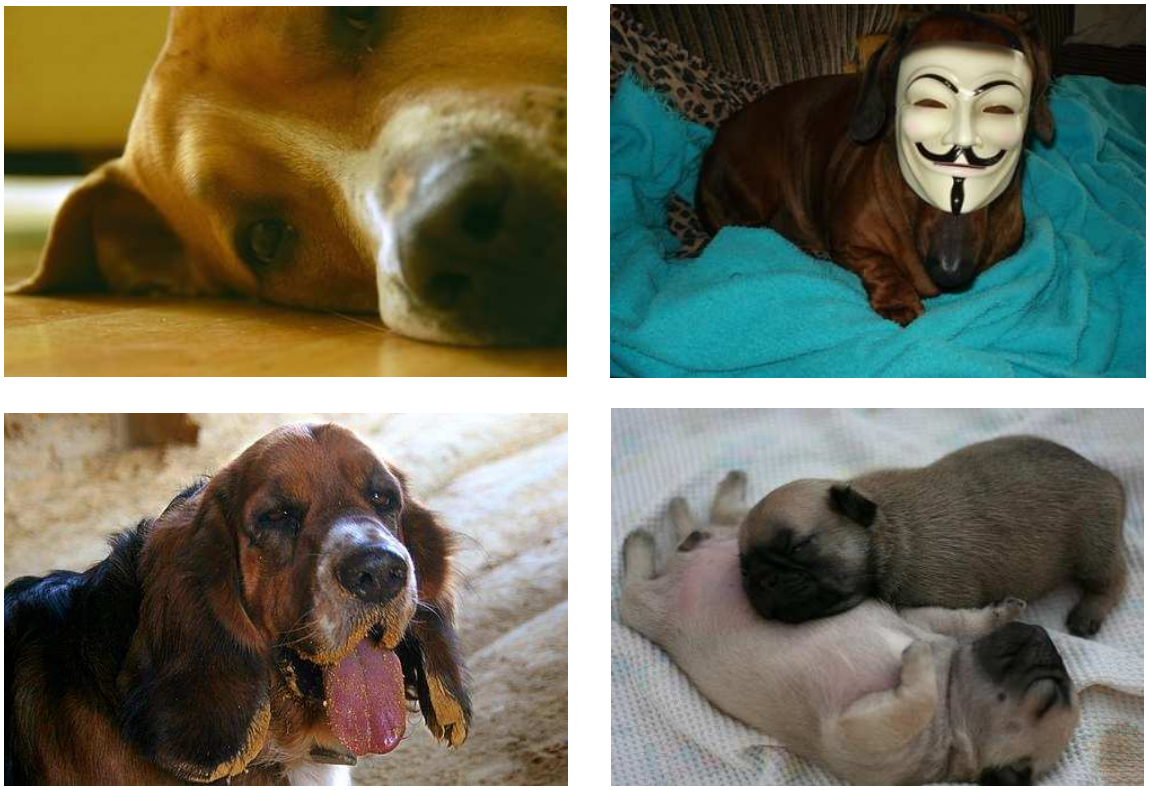
1. Bevezetés 5
1.1. A tartalom alapú képkeresés 5
1.2. A dolgozat célja, felépítése 6
2. A képi klasszifikáció 7
2.1. Az alapprobléma 7
2.2. Kiértékelés 8
2.2.1. Konfúziós mátrix 8
2.2.2. ROC görbe, AUC 9
2.2.3. Average Precision, MAP 10
3. Módszerek, szakterületi áttekintés 11
3.1. A Bag of Words alapelv 11
3.2. Alacsony szintű leírók 13
3.2.1. Szín alapú leírók 13
3.2.2. Textúra alapú leírók 14
3.3. Fontos pontok detektálása 16
3.3.1. Harris-Laplace detektor 16
3.3.2. Scale Invariant Feature Transform 18
3.4. A vizuális szótár kialakítása: PCA és klaszterezés 20
3.4.1. Principal Component Analysis 20
3.4.2. Klaszterezés 21
3.5. Magas szintű képi leírók 25
3.5.1. Bag of Words 25
3.5.2. Fisher-vektor 26
3.5.3. Szuper-vektor 27
3.5.4. Spatial pooling 28
3.6. Gépi tanulás 28
3.6.1. Lineáris klasszifikátorok 29
3.6.2. A kernel trükk 31
3.6.3. Support Vector Machine (SVM) 32
3.6.4. A Mixture Of Experts (MOE) architektúra 36
4. Aggregált kernel alapú módszer 37
4.1. Alapelv 37

4.1.1. Az előre számolt kernel mátrix 38
4.1.2. SVM előre számolt kernellel 39
4.1.3. Kernel mátrixok kombinálása 40
4.1.4. A súlyok meghatározása 41
4.2. A megvalósított rendszer 42
4.2.1. Áttekintés 42
4.2.2. A vizsgálat célja 43
4.2.3. A felhasznált képi leírók és kernelek 43
5. Implementáció és kiértékelés 48
5.1. A rendszer implementációja 48
5.2. A kiértékeléshez használt adathalmaz 49
5.3. Eredmények 49
5.3.1. A kiindulás: Fisher-vektorok 49
5.3.2. A különböző képi leírók 51
5.3.3. Kombinált módszerek 52
6. Összefoglalás és kitekintés 55
Irodalomjegyzék 56

1. Bevezetés

1.1. A tartalom alapú képkeresés

A számítógép számára rendkívül nehéz feladat egy képen szereplő vizuális tartalom alapján megállapítani, hogy mit ábrázol a kép, és megérteni mit jelent az egy ember számára. A „vizuális tartalom” ebben az értelmezésben vonatkozhat színekre, textúrákra, alakokra, illetve minden más információra, amit kizárólag a képi adatokból származtathatunk. A feladat nehézsége onnan ered, hogy egy adott objektumot vagy fogalmat ábrázoló képek vizuálisan nagyon változatosak lehetnek. Ezt szemléltetik az alábbi képek, amelyek mindegyike kutyát ábrázoló fénykép, mégis hatalmas különbségek vannak köztük.



1.1. ábra: Kutyát ábrázoló fényképek változatossága

A legtöbb on-line képkereső szolgáltatás csupán a metaadatokra (pl. a kép neve, a kép környékén lévő szöveg) és egyszerű vizuális reprezentációkra támaszkodik, ez értelemszerűen sok hibás találatot eredményez. Szükségszerű lenne a vizuális tartalom nagyobb súlyú figyelembe vétele is. Egy hagyományosnak tekinthető módszer a képek ellátása szavas annotációkkal. A keresést ezt követően már az annotációk terén hatékonyan végre tudjuk hajtani. A címkék kézi hozzáadása nagyon időigényes, fáradságos és drága munka, ezért az automatikus képannotáció, vagy más néven képi klasszifikáció témájában intenzív kutatómunka folyik, és sok előrelépés történt az elmúlt években. Az utóbbi időben teret hódító közösségi weboldalakban rejlő alkalmazási lehetőségek további inspirációt adnak a témában zajló fejlesztésekhez. Emellett még az automatikus képannotáció számos gyakorlati alkalmazásra ad lehetőséget, csak néhány példát említünk: romlott ételek detektálása, orvosi

diagnózisok segítése, bűnmegelőzés, digitális növény/állathatározó, esemény és környezet felismerése állóképeken.

A téma az informatikán belül számos szakterületet érint, egyaránt alkalmazzuk az adatbányászat, a mesterséges intelligencia, a statisztika a képfeldolgozás, és a nagyteljesítményű számítástechnika vívmányait.

1.2. A dolgozat célja, felépítése

Dolgozatomban áttekintem a képi klasszifikáció szakterületét, és bemutatom a konzulensemmel az MTA SZTAKI-ban közösen fejlesztett rendszerünket, részletesebben kitérve saját munkámra. Munkám célja az volt, hogy a rendszer kiegészítésével megjavítsam annak működését, és erre vonatkozóan teszteljem az új rendszert.

A 2. fejezetben ismertetem a képi klasszifikáció alapproblémáját, a probléma formalizálását a tudományos kutatások céljára és a megoldások kiértékelésére használt mérőszámokat.

A 3. fejezetben áttekintem a témában alkalmazott legkorszerűbb eljárásokat a közelmúlt kutatási eredményeivel. Mivel a tartalom alapú képosztályozásnak hatalmas irodalma van, számos jelenleg is folyó kutatással, ennek teljes bemutatása meghaladja a dolgozat kereteit. A mi rendszerünkben is számos összetett módszert alkalmazunk, ezekre koncentrálok a szakterületi áttekintésben, ezeket tárgyalom részletesen.

A 4. fejezetben részletesen bemutatom a képi klasszifikációs rendszerünket, különös tekintettel az előre számolt kernel alapú tanításra és a kernel aggregálás módszerére. Felvázolom, hogy a K-means klaszterezéssel és a Szuper-vektor magas szintű képi leíró számításával hogyan egészítettem ki az eddigi, Fisher-vektor alapú rendszert, és hogyan kombináltam a különböző képi jellemzőket az osztályozás során.

Az 5. fejezetben röviden szót ejtek a rendszer implementációjáról és a számításigényéről, majd ismertetem a tesztelés során kapott eredményeket. Ezzel bebizonyítom, hogy a Szuper-vektor felhasználásával a kernel aggregáló módszerünkben megjavítottam az eddigi rendszer teljesítményét.

Végül a 6. fejezetben egy rövid kitekintést adok arra vonatkozóan, hogy milyen irányban szeretnénk továbbfejleszteni a rendszerünket.

2. A képi klasszifikáció

A bevezetőben láttuk, hogy a tartalom alapú képosztályozás felhasználására rengeteg lehetőség van a hétköznapi gyakorlatban előforduló feladatokban. Ezzel együtt maga a probléma nem jól definiált, ahhoz, hogy tudományos kutatásokat lehessen végezni, előbb formalizálni kell azt. Ebben a fejezetben egy olyan keretet adunk a problémának, amelyen belül már a feladat jól meghatározott, a tudomány módszereivel lehet vizsgálni, és a különböző megoldásokat össze lehet hasonlítani.

2.1. Az alapprobléma

Elsőként meg kell határoznunk, hogy mi az, amit keresünk a képen. Nem foglalkozunk most azzal, hogy a keresett objektum hol helyezkedik el a képen belül, ez a probléma egy másik, széleskörű kutatási területhez vezet. A feladatot diszkretizáljuk azáltal, hogy a klasszifikációt előre megadott kategóriákban végezzük. Tehát a *mi van a képen?* kérdés helyett inkább a *van-e a képen ember?*, *tél van-e a képen? stb.* természetű kérdésekre keresünk választ külön-külön. A kategóriák vonatkozhatnak objektumokra és fogalmakra is. Például a Pascal Visual Object Classes 2011 adathalmazban [1] 20 darab, csak objektumokra vonatkozó kategória van: repülőgép, kerékpár, madár, hajó, palack, busz, autó, macska, szék, tehén, étkezőasztal, kutya, ló, motorkerékpár, ember, cserepes növény, birka, kanapé, vonat és képernyő. Ezzel szemben az ImageCLEF Photo Annotation 2011 adathalmazban 99 fogalom vagy objektum automatikus felismerése a feladat.

Másodszor, meg kell határoznunk, hogy milyen típusú képekre végezzük a feladatot. Ebben a dolgozatban a természetes fényképek osztályozásával foglalkozunk, de a képi klasszifikáció alaphalmaza lehet például kézzel írott számjegyeket ábrázoló képek, festmények, infrakamerás felvételek, stb.

Mivel alapvetően gépi tanulásos módszerekkel dolgozunk, minden esetben szükség van egy mintahalmazra, amelyet általában két részre bontunk, tanító és teszt képekre. A különböző módszerek összehasonlítása is kizárólag azonos tanító- és teszthalmazon nyer értelmet. Létezik olyan változata is a feladatnak, amelyben a tanításra tetszőleges képeket fel lehet használni, de mi most arra az esetre koncentrálunk, amikor adott a tanítóhalmaz.

A mintahalmaz képeire ($I = \{I_1, I_2, \dots, I_N\}$) kategóriánként adott az annotáció. Tehát minden egyes c kategória esetén adottak:

$$y_k^c = \begin{cases} 1, & ha\ az\ I_k\ képen\ szerepel\ a\ c\ kategória \\ 0, & ha\ az\ I_k\ képen\ nem\ szerepel\ a\ c\ kategória \end{cases} \qquad k = 1 \dots N$$

A tanítóhalmazon betanított osztályozó a teszthalmaz minden képére kategóriánként megad egy $f(I_k)$ jóslatot. A jóslat értékkészlete két féle lehet, vagy határozottan osztályba soroljuk a tesztképet ($f(I_k) \in \{0,1\}$), vagy egy konfidencia értéket adunk meg ($f(I_k) \in (0,1)$), amelynek jelentése az, hogy a minta milyen valószínűséggel tartozik az 1-es osztályhoz. Az alapvető követelmény, hogy a jóslat minél több képre helyes legyen, azonban jóval

kifinomultabb mutatókat is használunk a klasszifikáció értékelésére. A következő pont erről szól.

2.2. Kiértékelés

2.2.1. Konfúziós mátrix

A konfúziós mátrixot diszkrét értékkészletű jóslatokra készítjük ($f(I_k) \in \{0,1\}$), így amennyiben a kimenetünk konfidencia típusú, azt egy küszöbértékkel fel kell osztani. Az osztályozó kimenete ($f(I_k)$) és a valódi besorolás (y_k) szerint négy féle csoportba oszthatjuk a teszthalmaz elemeit, amelyek számosságait egy táblázatba foglaljuk, ezt nevezzük konfúziós mátrixnak.

- Helyes pozitívak (true positives):
$$TP = |\{I_k : f(I_k) = y_k = 1\}|$$
- Helyes negatívok (true negatives):
$$TN = |\{I_k : f(I_k) = y_k = 0\}|$$
- Fals pozitívak (false positives):
$$FP = |\{I_k : f(I_k) = 1, y_k = 0\}|$$
- Fals negatívok (false negatives):
$$FN = |\{I_k : f(I_k) = 0, y_k = 1\}|$$

		Helyes jóslat		Összesen
		1	0	
Osztályozó kimenete	1	TP	FP	P'
	0	FN	TN	N'
Összesen		P	N	

2.1. táblázat: A konfúziós mátrix

A konfúziós mátrixból több számszerű mutató is származtatható, ezek közül a képi klasszifikációban leggyakrabban az alábbiakat vesszük figyelembe:

- Accuracy:
$$Acc = \frac{TP + TN}{P + N}$$
- Precision (vagy Positive Predictive Value):
$$Prec = \frac{TP}{TP + FP}$$
- Recall (vagy True Positive Rate, Sensitivity):
$$Rec = \frac{TP}{P}$$
- False Positive Rate

$$FPR = \frac{FP}{N}$$

- Specificity (vagy True Negative Rate)

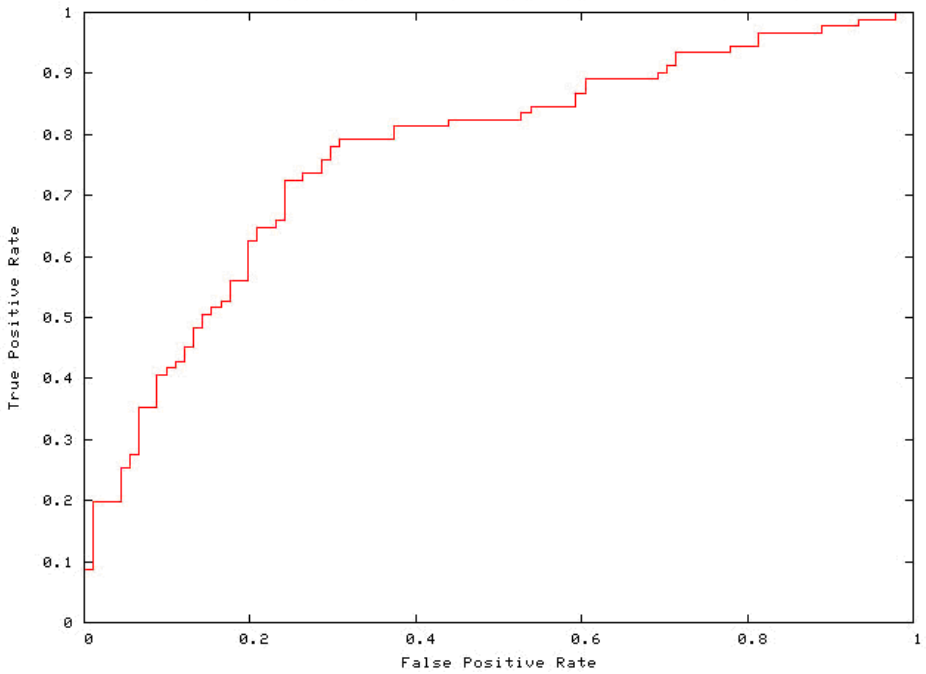
$$Spec = \frac{TN}{N}$$

- F-measure

$$F = 2 \cdot \frac{Prec \cdot Rec}{Prec + Rec}$$

2.2.2. ROC görbe, AUC

A Receiver Operating Characteristic (ROC) görbe egy grafikus ábrázolása a Recall (True Positive Rate) és a False Positive Rate mérőszámok kapcsolatának. Ha az osztályozó kimenete egy valós szám, akkor meg kell határozni egy döntési küszöböt, amellyel a két osztályt szétválasztjuk. A küszöb változtatásával különböző FPR-Recall számpárokat kapunk, ezeket ábrázoljuk egy grafikonon. Tehát a görbe azt mutatja, hogy a megengedett fals pozitív találatok arányának növelésével hogyan változik a helyesen osztályozott pozitív minták aránya. Egy példát mutat ilyen ROC görbére az alábbi ábra.



2.1. ábra: Egy ROC görbe

Annál jobb egy osztályozó, minél magasabban halad a görbe a $(0,0) - (1,1)$ átló felett, ezért szokták mérőszámként alkalmazni a görbe alatti területet, illetve a görbe és az átló közötti területet is. Ezt hívjuk Area Under Curve, röviden AUC mutatónak.

A görbét úgy szokás felrajzolni, hogy sorba rakjuk a teszthalmaz elemeit az osztályozó kimenete szerint, így elöl lesznek azok az elemek, amelyekre a legbiztosabban állítja az osztályozó, hogy az adott osztályba tartoznak, hátrébb a bizonytalanok, leghátul a biztosan

negatív minták. Minden elem után kiszámítjuk és ábrázoljuk az FPR-Recall számpárt, ez ekvivalens az osztályozó küszöbének változtatásával.

2.2.3. Average Precision, MAP

A ROC görbe mintájára, ha adott a tesztképeknek az osztályba tartozás valószínűsége szerinti sorrendje, a sorozat minden pontján kiszámíthatjuk az aktuális Precision és Recall mérőszámokat, és ábrázolhatjuk a Recall (r) függvényében a Precisiont ($p(r)$). Tehát ez a görbe azt adja meg, hogy az összes pozitív minta meghatározott arányú részének felsorolása mellett ($\frac{TP}{p}$) milyen arányban soroltunk fel valóban helyes pozitív mintákat ($\frac{TP}{TP+FP}$). Az Average Precision mutató az $r \in [0,1]$ intervallumon átlagolt Precisiont jelenti, vagyis a görbe alatti területet:

$$AveP = \int_0^1 p(r)dr$$

Mivel egy sorrend különböző pontjain számítjuk a Recall és Precison értékeket, ezért a fenti integrált a gyakorlatban egy diszkrét összegzéssel helyettesítjük:

$$AveP = \sum_{k=1}^{N} P(k)\Delta r(k)\,,$$

ahol N a teszthalmaz mérete, $P(k)$ a sorrend k-adik eleme után kiszámolt Precision, $\Delta r(k)$ pedig a Recall megváltozása a $k-1$-edik és k-adik elem között. Belátható, hogy a fenti szumma az alábbival egyenlő:

$$AveP = \frac{\sum_{k=1}^{N} P(k)y_k}{\sum_{k=1}^{N} y_k}$$

Több kategória esetén az Average Precision mutatókat átlagolva kapjuk a Mean Average Precision (MAP) értékét, amely a képi klasszifikáció minősítésére leggyakrabban alkalmazott mérőszám. Amennyiben C jelöli a kategóriák halmazát, a MAP kiszámítása:

$$MAP = \sum_{c \in C} \frac{AveP(c)}{|C|}$$

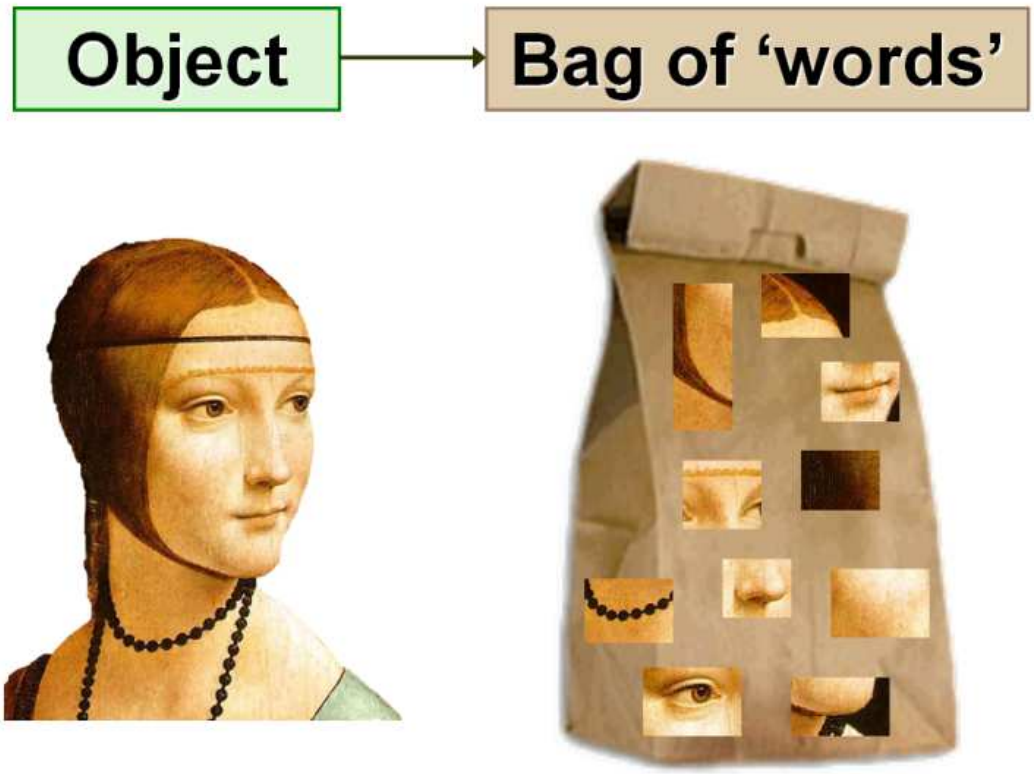
Az eredmények kiértékelése során az Average Precision és MAP mutatókat használom.

3. Módszerek, szakterületi áttekintés

3.1. A Bag of Words alapelv

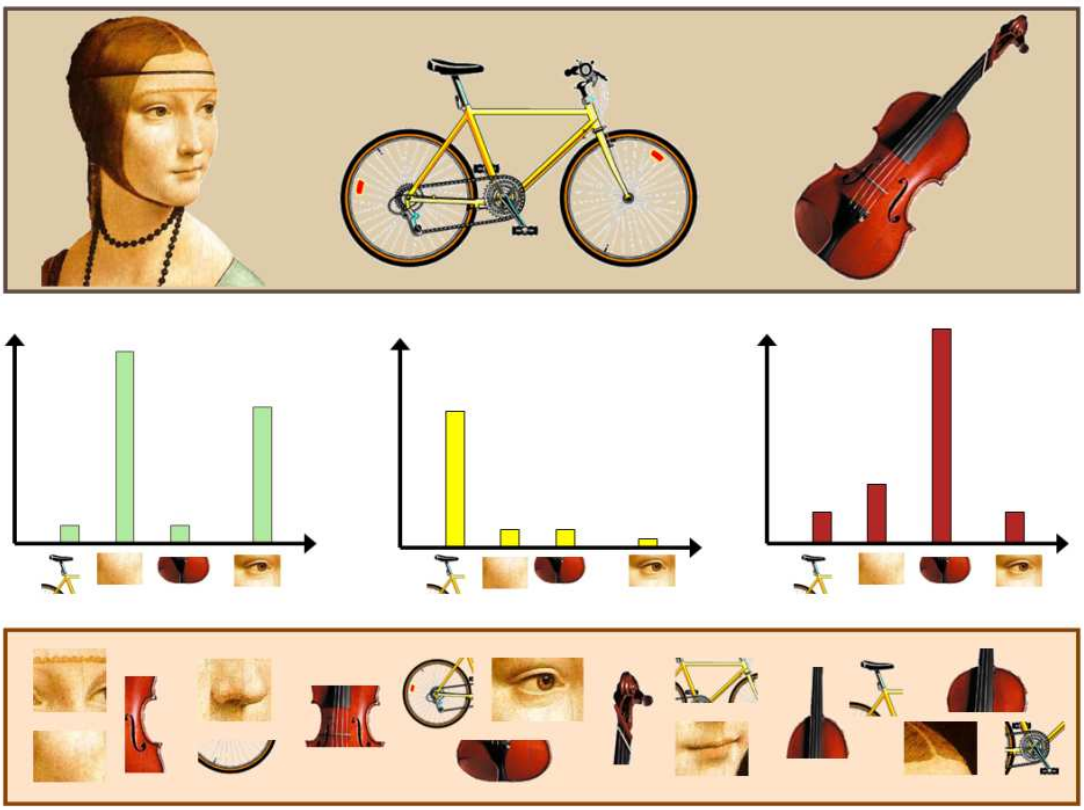
A Bag of Words (BoW) alapelvet eredetileg szöveges dokumentumok osztályozásában használták. Ebben a módszerben egy dokumentumot a benne lévő szavak összességével reprezentálnak, a szavak sorrendjét figyelmen kívül hagyva. Tehát a dokumentumot csupán a benne szereplő szavak eloszlása jellemzi, ez alapján következtetünk a témájára.

A fentivel analóg módszert lehet alkalmazni a képek reprezentálására is. Vagyis egy-egy képre meghatározzuk a rajta látható vizuális elemeket, amiket vizuális kódszavaknak nevezünk, és a képet ezek összességével jellemezzük (3.1. ábra).



3.1. ábra: A Bag of Words alapelv képekre (forrás: [1])

Tehát azt feltételezzük, hogy a kép kis részeit jellemző vizuális kódszavak eloszlásából tudunk következtetni a kép szemantikus tartalmára (3.2. ábra).



3.2. ábra: A képeket a vizuális kódszavak eloszlásával reprezentáljuk (forrás: [1])

Képek esetén ez a megközelítés viszont felvet néhány problémát, ami szöveges dokumentumoknál nem kerül elő:

- Egyáltalán mik a vizuális kódszavak? Szükség van egy vizuális szótár kialakítására.
- Hogyan határozzuk meg, hogy egy kép mely kódszavakból áll?

A vizuális kódszavak kialakításának érdekében a kép egy-egy kis részét egy alacsony szintű leíró vektorral jellemezzük, és az egymáshoz hasonló leíró vektorok jelentenek egy vizuális kódszót. Nagyon sok különböző alacsony szintű leírót használtak már erre a feladatra, a következő pontban (3.2) bemutatok néhányat. A vizuális szótár kialakítása pedig leggyakrabban egy tanítóhalmazban lévő összes képről kinyert alacsony szintű leírók klaszterezésével történik, erről részletesen a 3.4 pontban írok.

Miután eldöntöttük, hogy miket tekintünk vizuális szavaknak, definiálnunk kell, hogy hogyan határozzuk meg, hogy egy kép mely vizuális szavakból áll. Szövegek esetében ez a kérdés nem kerül elő, ott a szóhatárok miatt egyértelmű, hogy milyen szavak vannak egy szövegben. Képek esetén azt kell eldöntenünk, hogy a kép mely pontjaiban számoljuk ki a pont környezetét jellemző leíró vektort. Erre alapvetően két féle megközelítés létezik: vagy egy rács minden pontjában kiszámoljuk a leírókat, vagy próbáljuk detektálni a képen lévő számunkra érdekes pontokat, és csak azokban számolunk leíró vektorokat. Mindkét módszer esetében még egy további lehetőség a kép skálázása, és a leíró vektorok különböző skálatérben való kiszámítása. A 3.3 pontban írok olyan módszerekről, amelyeket a képen lévő fontos pontok meghatározására használnak.

A képen szereplő vizuális szavak eloszlásából kialakítunk egy magas szintű leíró vektort, amely alapján a képet osztályokba soroljuk. A magas szintű szemantikus leírók terén nagyon intenzív kutatás folyik, sok előrelépés történt a közelmúltban. A 3.5 pontban mutatok példákat ilyen módszerekre. A képeket jellemző vektorok osztályozására gépi tanulásos módszereket alkalmazunk, ezt a 3.6 pontban tárgyalom.

3.2. Alacsony szintű leírók

A kép lokális tulajdonságait a pont környezetében előforduló színek, illetve a kép textúrája alapján lehet jellemezni. Az alábbiakban mindkét fajtából olyan módszereket mutatok be, amelyeket sűrűn alkalmaznak képi klasszifikációs feladatokban.

3.2.1. Szín alapú leírók

Kézenfekvő, és gyakran alkalmazott módszer a kép részleteinek szín alapján történő jellemzése. Példaként bemutatok két különböző szín alapú leírót, a szín momentumokat és a szín hisztogramokat.

3.2.1.1. Szín momentumok

A szín momentumok módszerében [3] az adott területen lévő pixelek színcsatornánként külön vett eloszlásának első néhány momentumát vesszük. Leggyakrabban RGB, illetve HSV színrendszerekben számolják, általában az első két vagy három momentumot.

Legyen az (x, y) pixel intenzitásértéke a k-adik színcsatornában $I_k(x, y)$. Ekkor a kép egy N pixel méretű T tartományára a következő értékeket számoljuk ki:

$$E_k = \frac{1}{N} \sum_{(x,y)\in T} I_k(x, y)\,, \qquad \sigma_k = \sqrt{\frac{1}{N} \sum_{(x,y)\in T} (I_k(x, y) - E_k)^2}\,,$$

$$s_k = \sqrt[3]{\frac{1}{N} \sum_{(x,y)\in T} (I_k(x, y) - E_k)^3}$$

Az $\boldsymbol{f}$ leíró vektor pedig ezeket az értékeket tartalmazza:

$$\boldsymbol{f} = [E_0 \quad \sigma_0 \quad s_0 \quad E_1 \quad \sigma_1 \quad s_1 \quad E_2 \quad \sigma_2 \quad s_2]$$

Például RGB színrendszer esetén, ha csak a csatornánkénti első két momentumot számítjuk ki, akkor ez gyakorlatilag azt jelenti, hogy külön kiszámoljuk a pixelek piros, zöld és kék összetevőinek átlagát és szórását, és ezzel a 6 dimenziós vektorral írjuk le az adott képrészletet.

3.2.1.2. Szín hisztogram

A szín hisztogram [4] számításakor egy diszkrét színtér mellett meghatározzuk, hogy az adott régióban az egyes színekből hány pixel van. Az így kapott hisztogram vektorral jellemezzük a képrészletet. Jelölje $I(x, y)$ az (x, y) pixel színét, és legyenek $c_0, c_1, \dots, c_n$ a választott

diszkrét színtér színei. Ekkor a kép egy T tartományának leírója az alábbi módon kiszámolt $\boldsymbol{f}$ hisztogram vektor lesz.

$$h_i = |\{(x, y) \in T : I(x, y) = c_i\}|$$

$$\boldsymbol{f} = [h_0 \quad h_1 \quad \dots \quad h_n]$$

Létezik olyan megközelítés is [5], ami szerint az emberi érzékelés számára a leginkább mérvadó az adott képterületen domináns szín, vagyis az, ami a legtöbbször fordul elő, így a teljes hisztogram helyett csupán a leggyakoribb színnel jellemzi a képrészletet.

3.2.2. Textúra alapú leírók

A textúra elemzésére és jellemzésére sok módszert javasoltak már. A teljesség igénye nélkül az alábbiakban bemutatok közülük kettőt, amelyek alapelvei a legfontosabbnak tekinthetők a képi klasszifikációs alkalmazásokban. A Local Binary Pattern (LBP) módszer a lokális intenzitásminta elemzésén alapul, ezzel szemben a a Histograms of Oriented Gradients (HOG) Scale Invariant Feature Transform (SIFT) leírók a lokális gradiensek orientációjából származtathatók.

3.2.2.1. Local Binary Pattern

Az LBP leírót Ojala és társai fejlesztették ki [6]. Számítása a kép egy cellájára a következőképpen megy:

- Tekintsünk egy pixelt, és a 8 szomszédja helyett írjunk 0-t vagy 1-et, attól függően, hogy kisebb vagy nagyobb-e az intenzitása, mint a középső pixelé.
- Az így kapott 8 db 0-1 számjegy egy rögzített sorrend szerinti kiolvasásával kapunk egy 8-jegyű bináris számot.
- Ezt minden pixelre elvégezzük, majd az egyes számok (vagyis lokális bináris minták) előfordulásait megszámolva egy hisztogramot képzünk, ami a kép adott tartományát jellemzi.
- Opcionálisan normalizáljuk a hisztogramot.

3.2.2.2. Scale Invariant Feature Transform

David G. Lowe egy teljes rendszert dolgozott ki arra, hogy egy konkrét objektum előfordulásait különböző képeken megtaláljuk [7]. Az objektumokat néhány kiválasztott kulcspontban számított leíró vektorral jellemzi. Tehát a rendszer része egy fontos pontokat detektáló mechanizmus, egy skála- és orientáció független leíró vektor kiszámítása, valamint ezek alapján az objektumok összehasonlítása (gyakorlatban felmerülő problémákkal együtt, pl. indexelés). A rendszert, illetve részeit az eredetin kívül sok más célra is használják, pl. panorámaképek összeillesztésére, 3D rekonstrukcióra, agyi MRI felvételek elemzésére.

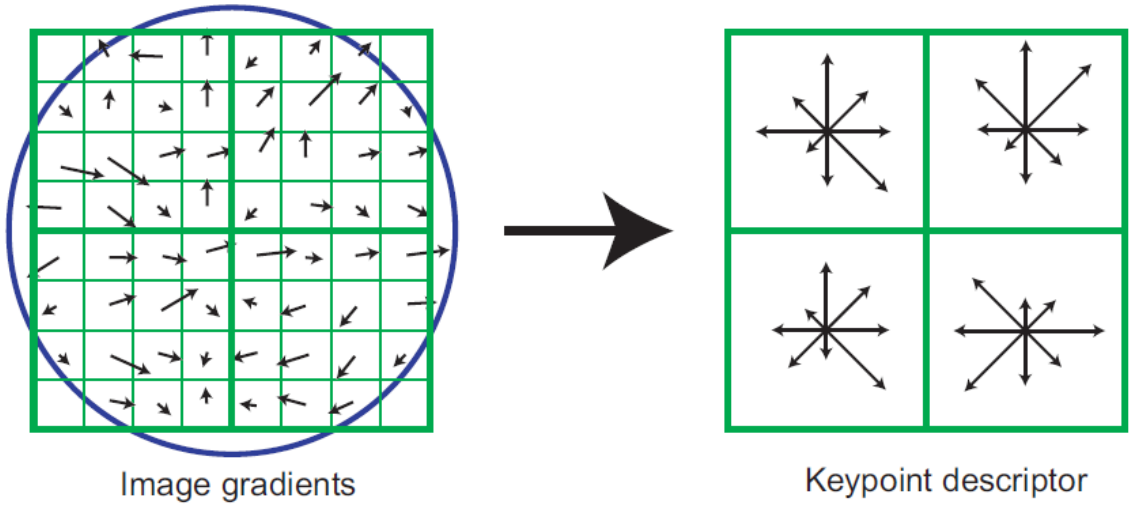
A Lowe által javasolt leírót előszeretettel alkalmazzák képi klasszifikációs feladatokban, akár csak a fontos pontokban számolva, akár egy teljes rács pontjaiban. A kulcspontok detektálásáról a 3.3.2 pontban írok.

A SIFT leíró vektor egy pont környezetében lévő élek orientációit térképezi fel. A pont környezetében képezünk $n_1 \times n_2$ cellát, mindegyikben $k_1 \times k_2$ db mintavételi ponttal, amelyekben kiszámoljuk a gradiens $\theta(x,y)$ irányát és $m(x,y)$ nagyságát:

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

$$\theta(x,y) = tan^{-1}\left(\frac{I(x,y+1) - I(x,y-1)}{I(x+1,y) - I(x-1,y)}\right)$$

A gradiensek nagyságát súlyozzuk egy Gauss-függvény szerint, aminek középpontja a vizsgált pontban van, így jobban fognak számítani a közelebbi élek. A gradiensekből cellánként egy-egy orientáció hisztogramot készítünk, amelyekbe értelemszerűen az egyes gradienseket súlyozva számítjuk bele. Ezt szemlélteti a 3.3. ábra (itt $n_1 = n_2 = 2$ és $k_1 = k_2 = 4$). A szerző $n_1 = n_2 = n = 4$ és $k_1 = k_2 = k = 4$ értékeket javasol, míg a hisztogramot $d = 8$ irányra számolja, természetesen a szomszédos irányok közé eső gradienseket megfelelő súlyozással elosztva a hisztogramban (trilineáris interpolációval).



3.3. ábra: Az orientáció hisztogramok kialakítása (forrás: [7])

A hisztogramokat konkatenálva kapunk egy $n^2 d$ dimenziós vektort (a szerző által javasolt értékekkel ez 128 dimenziós), amit normalizálunk. Végül, a nemlineáris megvilágításbeli eltérések hatásának kiküszöbölésére Lowe a leíró vektor minden 0,2-nél nagyobb elemét 0,2-re cseréli, és újra normalizálja a vektort.

3.2.2.3. Histogram of Oriented Gradients

A HOG leírót Dalal és Triggs dolgozta ki, speciálisan emberek észlelésére képeken [8]. Nagyon hasonlít a SIFT leíróra, de a szerzők szerint más a célja, más megközelítésben magyarázzák a hasznosságát, és más feladatra optimalizálták. Kifejezetten azt javasolják, hogy egy sűrű rácson számoljuk, amelynek cellái átlapolódnak, mert így megbízhatóbban lehet az emberi test felépítését detektálni.

Kimerítő módon tesztelték a leíró egyes paramétereit az ember detektálási feladat szempontjából (pl. cellák száma, mérete, körkörös, illetve négyzetes alakban; hisztogram irányainak száma; különböző szűrők; színcsatornák; normalizáció). A legjobb eredményeket a következő beállításokkal érték el:

- RGB színtér
- a gradiensek meghatározása a $[-1 \quad 0 \quad 1]$ élszűrő operátorral
- négyzet alakú cellák, a középpont körül 3× 3 darab, 6× 6 pixel méretűek
- „előjel nélküli” gradiensekkel számolás, a hisztogramban 9 irány $(0 - 180°)$, lineáris interpoláció a két szomszédos irány között
- a gradiensek nagyságának Gauss-függvénnyel súlyozása ($\sigma = 0{,}5 \cdot blokkméret$, jelen esetben 9 pixel)
- a leíró vektor normalizálása a Lowe-féle módszerrel

A szerzők megjegyzik, hogy ezek speciálisan az ember detektáláshoz használt beállítások, más feladatoknál esetleg más paraméterek jobb eredményt nyújthatnak, pl. autó vagy motorkerékpár esetén az „előjeles” gradiensek jobbak

3.3. Fontos pontok detektálása

Mint a korábbiakban említésre került, egy képet alkotó vizuális szavak meghatározásához azt kell eldöntenünk, hogy mely pontokban, és milyen skálázás mellett számoljuk ki a lokális leíró vektorokat. Az egyszerű (de sok esetben jól működő) rácspontokban való kiszámítás mellett van egy másik megközelítés, miszerint a képen először meghatározzuk azokat a pontokat, amelyek fontosak a kép reprezentációja szempontjából, és e pontok környezetében értelmezzük a leíró vektorokat. A következőkben bemutatok két módszert, amelyeket a képen lévő fontos pontok detektálására alkalmaznak.

3.3.1. Harris-Laplace detektor

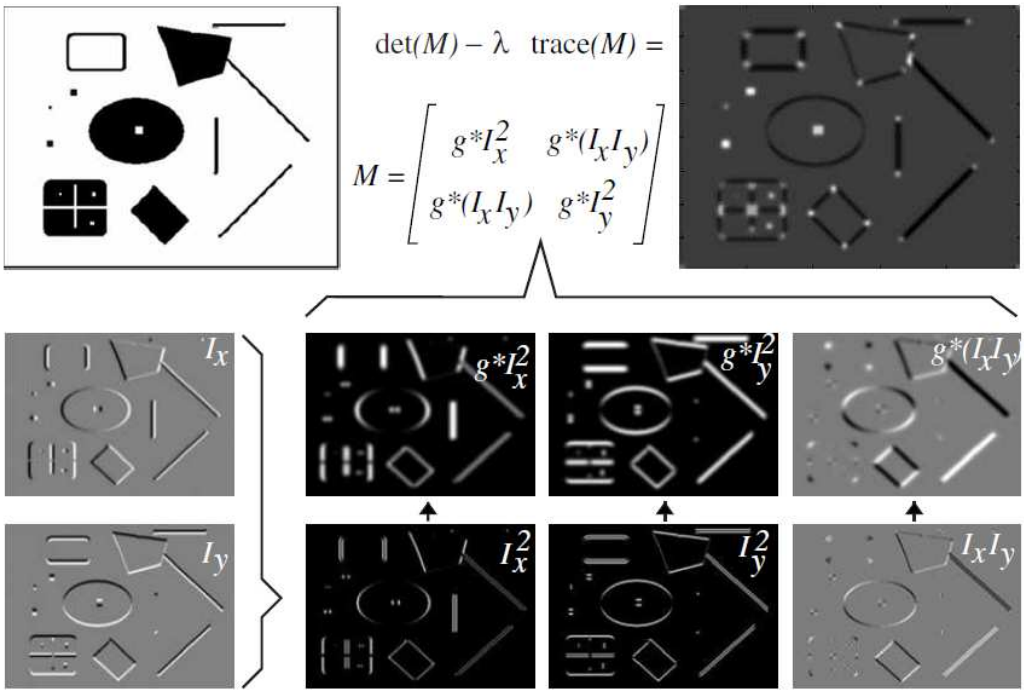
A Harris és Stephens által kifejlesztett Harris detektor [9] a képen lévő sarkokat igyekszik detektálni, vagyis ahol két, egymásra merőleges irányban is nagy a derivált. Az alapja a második momentum mátrix, más néven autokorrelációs mátrix, ami egy pont lokális környezetében a gradiensek eloszlását jellemzi:

$$M = \sum_{u,v} g(u,v,\sigma) \begin{bmatrix} I_x^2(u,v) & I_x(u,v)I_y(u,v) \\ I_x(u,v)I_y(u,v) & I_y^2(u,v) \end{bmatrix}$$

ahol I_x illetve I_y a parciális deriváltakat jelölik, g pedig egy Gauss-féle súlyozó függvény. Amennyiben a mátrixnak két nagy sajátértéke van, ez azt jelenti, hogy az adott pontban két ortogonális irányban is nagy a gradiens vektor. Harris a számítási költségek csökkentése végett azt javasolta, hogy a sajátértékek kiszámítása helyett a szorzatukból vonjuk ki az összegük α-szorosát, vagyis a következő mértékkel jellemezzük egy pont „sarkosságát”:

$$R = \lambda_1\lambda_2 - \alpha(\lambda_1+\lambda_2) = detM - \alpha\, trM$$

ahol trA a mátrix nyoma, vagyis a főátlóbeli elemek összege. A fontos pontok azok a pontok lesznek, ahol ennek a függvénynek lokális maximuma van, és ez egy adott korlátnál nagyobb. A sarokpontok detektálásának menetét egy példa képen az alábbi ábra szemlélteti.



3.4. ábra: A Harris-Laplace detektor működése egy példán (forrás: [10])

Ezt a módszert fejlesztette tovább Mikolajczyk és Schmid [11] úgy, hogy skálainvariáns legyen, ezt nevezik Harris-Laplace detektornak. Először a fenti Harris mátrixot a kép többféle skálázásában is kiszámolják:

$$g(\boldsymbol{x},\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \qquad \boldsymbol{x} = (x,y)$$

$$L(\boldsymbol{x},\sigma) = g(\boldsymbol{x},\sigma) \otimes I(\boldsymbol{x}), \qquad L_x = \frac{\partial}{\partial x} L, \qquad L_y = \frac{\partial}{\partial y} L$$

$$M(\boldsymbol{x},\sigma_I,\sigma_D) = \sigma_D^2 g(\boldsymbol{x},\sigma_I) \otimes \begin{bmatrix} L_x^2(\boldsymbol{x},\sigma_D) & L_x(\boldsymbol{x},\sigma_D)L_y(\boldsymbol{x},\sigma_D) \\ L_x(\boldsymbol{x},\sigma_D)L_y(\boldsymbol{x},\sigma_D) & L_y^2(\boldsymbol{x},\sigma_D) \end{bmatrix}$$

ahol $\otimes$ a konvolúció jele. Tehát a képet egy σ_D szórású (differenciális skála) Gauss-kernellel simítjuk, majd ennek a parciális deriváltjaiból képzett autokorrelációs mátrixot vizsgáljuk. Ezt egy σ_I szórású (integráló skála) Gauss ablakon átlagoljuk. A szerzők úgy választják meg σ_I-t és σ_D-t, hogy egymás konstans-szorosai legyenek, vagyis $\sigma_I = \sigma_n$ és $\sigma_D = s\sigma_n$, ezután értelemszerűen M csak σ_n-től függ, ez a skála paraméter. A „sarkosság” mértéke ismét

$$R = detM - \alpha\, trM$$

Első lépésben minden skálán külön detektáljuk a fontos pontokat, vagyis a fenti függvénynek egy bizonyos korlát feletti lokális maximumait. Ezután az így kiválasztott fontos pontok

skáláját és helyét egy Lindeberg által javasolt iteratív módszerrel [12] finomítjuk. Az alábbiakban $\boldsymbol{x}(k)$ és $\sigma(k)$ jelöli egy fontos pont helyét és skáláját a k-adik iterációban, az algoritmus menete:

1. Az $\boldsymbol{x}(k)$ pont $\sigma(k+1)$ skáláját a $\sigma(k+1) = t\sigma(k)$, $t \in \{0{,}7\ , \dots , 1{,}4\}$ tartományban keressük. Ezek közül azt választjuk, amelyre az alábbi Laplacian-of-Gaussians (LoG) függvény maximális:
$$|LoG(\boldsymbol{x}, \sigma)| = \sigma^2 \left| L_{xx}(\boldsymbol{x}, \sigma) + L_{yy}(\boldsymbol{x}, \sigma) \right|$$
2. A kiválasztott $\sigma(k+1)$ skálában az $\boldsymbol{x}(k)$ pont (például 8×8-as) környezetében keressük meg a fenti R Harris-mérték maximumát, és ennek a helye legyen $\boldsymbol{x}(k+1)$
3. Amennyiben $\boldsymbol{x}(k+1) \neq \boldsymbol{x}(k)$ vagy $\sigma(k+1) \neq \sigma(k)$, folytassuk újra az 1. lépéssel

3.3.2. Scale Invariant Feature Transform

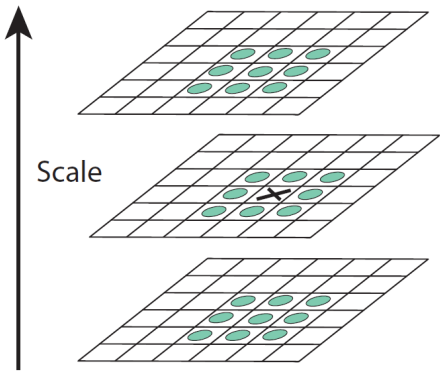
Mint korábban már említésre került, a Lowe által kidolgozott SIFT rendszerben [7] a képen lévő fontos pontokat (ő kulcspontoknak nevezi) jellemezzük egy-egy lokális leíró vektorral, úgy, hogy skálázásra és forgatásra invariáns legyen. Az alábbiakban a kulcspontok kiválasztásának mechanizmusát írom le.

A képet különböző szórású Gauss-függvényekkel konvolválva egy Gauss-féle képpiramist építünk, majd ebben a szomszédos skálájú képek különbségét vesszük, amelyeket egy konstans k faktor választ el egymástól (Difference of Gaussians – DoG):

$$L(x, y, \sigma) = g(x, y, \sigma) \otimes I(x, y)$$

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

Az így kapott DoG képeken kiválasztjuk a lokális szélsőértékeket, úgy, hogy egy pixelt a 26 szomszédjához hasonlítunk, amelyek a körülötte lévő 3×3-as régiókban helyezkednek el az adott pixel szintjén, illetve a szomszédos skálázású DoG képeken (3.5. ábra).



3.5. ábra: A DoG képekről a lokális szélsőértékek kiválasztása (forrás: [7])

A következőkben az így kiválasztott kulcspontok helyét és skáláját pontosítjuk interpolációval. Ehhez a DoG függvényt eltoljuk úgy, hogy középpontja a kulcspont-jelölt legyen, és a második Taylor-polinomjával közelítjük:

$$D(\boldsymbol{x}) = D + \frac{\partial D^T}{\partial \boldsymbol{x}}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^{\boldsymbol{T}}\frac{\partial^2 D}{\partial \boldsymbol{x}^{\boldsymbol{2}}}\boldsymbol{x}$$

Itt tehát a D függvényt és deriváltjait a vizsgált pontban számoljuk, és az $\boldsymbol{x} = (x, y, \sigma)$ az ettől vett eltolás. A fenti függvény deriválásával meghatározzuk a szélsőértékhelyét, legyen ez az $\widehat{\boldsymbol{x}}$ eltolásvektor. Amennyiben $\widehat{\boldsymbol{x}}$ bármelyik koordinátája nagyobb, mint 0,5, ez azt jelenti, hogy a szélsőérték közelebb esik egy másik mintavételezési ponthoz, így abban a pontban vizsgálódunk tovább. A $D(\widehat{\boldsymbol{x}})$ függvényérték alapján pedig kiszűrjük az esetleges hamis találatokat, minden olyan kulcspont-jelöltet elutasítunk, amire $|D(\widehat{\boldsymbol{x}})| < 0{,}03$.

Ezt követően az élek mentén lévő, de nem jól meghatározott elhelyezkedésű kulcspontokat szűrjük ki. Ugyanis a DoG függvény nagy értékeket vesz fel az élek mentén, és így sok kulcspont-jelölt keletkezik, viszont ezeknek a pozíciója az él mentén zaj hatására könnyen változhat, így nem stabilak. Ezért csak olyan kulcspontokat tartunk meg, amelyekben valamely más irányban is nagy a DoG függvény görbülete.

A DoG függvény fő görbületeit a kulcspontban számolt Hesse-féle mátrix sajátértékei adják meg:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Ha a sajátértékek aránya, $r = \lambda_1/\lambda_2$ egy küszöbértéknél nagyobb, akkor a kulcspont-jelöltet eldobjuk. A gyakorlatban a sajátértékeket nem fontos kiszámolni, mert belátható, hogy $\frac{(r+1)^2}{r} = \frac{(trH)^2}{detH}$, így elég ezt a mennyiséget vizsgálni. Ez a lépés nagyon hasonlít a Harris-féle detektor működésére, az a különbség, hogy itt a Hesse-féle mátrixszal dolgozunk a második momentum mátrix helyett.

Végül, a kiválasztott kulcspontok mindegyikéhez rendelünk egy domináns irányt. Erre azért van szükség, hogy a reprezentáció a forgatásra invariáns legyen. A 3.2.2.2 pontban leírt lokális leíró vektor számításakor az irányokat ehhez a domináns irányhoz viszonyítjuk.

A domináns irány meghatározása érdekében a kulcspont skálájának megfelelő σ szórású Gauss-függvénnyel simított $L(x, y)$ képre kiszámoljuk a gradiens $m(x, y)$ nagyságát, és $\theta(x, y)$ irányát, a pixelek különbségeinek segítségével:

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

$$\theta(x, y) = tan^{-1}\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right)$$

Egy orientáció hisztogramot készítünk minden kulcspontra, a környezetében elhelyezkedő gradiensek alapján. A hisztogramban 36 féle irány szerepel, így 10°-onként fedjük le a 360°-os értékkészletet. A gradienseket a hisztogramba számításkor súlyozzuk a nagyságukkal, és egy körkörös Gauss-féle ablakkal. Az orientáció hisztogramban lévő legnagyobb súlyú irány lesz a kulcsponthoz rendelt irány. Illetve, ha van még olyan más lokális maximum a

hisztogramban, amelynek súlya a domináns irány súlyának legalább 80%-a, akkor ezt az irányt is feljegyezzük, mégpedig úgy, hogy létrehozunk még egy kulcspontot ugyanezen a helyen, ezzel a másik iránnyal.

3.4. A vizuális szótár kialakítása: PCA és klaszterezés

A tanítóhalmazban lévő képekről kinyert alacsony szintű leírókat klaszterekbe osztjuk, és az egy klaszterbe tartozó leíró vektorok felelnek meg egy vizuális kódszónak. Két féle, a Bag of Words módszerekben is gyakran alkalmazott klaszterezést mutatok be: a gyorsan számolható K-means-t (3.4.2.2) és a kevésbé gyors, de „soft-klaszterezés" lévén több információt hordozó GMM módszert (3.4.2.3).

Klaszterezés előtt azonban hasznos lehet az alacsony szintű leírókat dimenziócsökkentéssel egy kompaktabb formában ábrázolni, amely ráadásul a számunkra fontos információkat emeli ki. Ezt a célt szolgálja a főkomponens analízis (Principal Component Analysis, PCA). Ezzel nem csak a későbbi számításigényt csökkentjük, hanem egy olyan vektortérben ábrázoljuk a leíró vektorokat, amelynek bázisai a sokaság legfontosabb komponensei.

3.4.1. Principal Component Analysis

A PCA matematikai definíciója szerint egy ortogonális, lineáris transzformáció, amely egy olyan új koordináta-rendszerbe képezi az adatokat, melyben az adatok bármely vetületének az első koordináta (úgynevezett első főkomponens) szerinti szórása lesz a legnagyobb, a második koordináta szerinti a második legnagyobb, és így tovább. Ezt szemlélteti a 3.6. ábra, egy kétdimenziós Gauss-eloszlás szerinti adathalmazon.

Legyenek az M dimenziós adatvektorok $\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_N}$. Vonjuk le az átlagukat mindegyikből ($\widehat{\boldsymbol{x}}_{\boldsymbol{i}} = \boldsymbol{x_i} - \overline{\boldsymbol{x}}$), és a kapott vektorokból képezzünk egy $N \times M$-es $\boldsymbol{X}$ mátrixot:

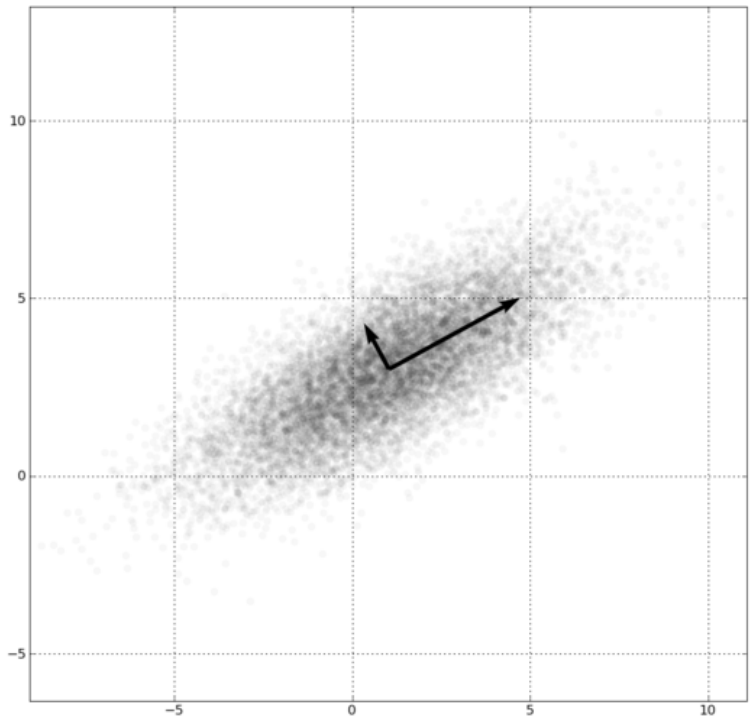
$$\boldsymbol{X} = \begin{bmatrix} \widehat{\boldsymbol{x}}_{\boldsymbol{1}} \\ \widehat{\boldsymbol{x}}_{\boldsymbol{2}} \\ \vdots \\ \widehat{\boldsymbol{x}}_{\boldsymbol{N}} \end{bmatrix}$$

Legyen $\boldsymbol{X}$ szinguláris érték felbontása (SVD) $\boldsymbol{X} = \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}}$, ahol $\boldsymbol{W}$ egy $M \times M$-es mátrix, ami $\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}$ sajátvektorait tartalmazza, $\boldsymbol{\Sigma}$ egy $M \times N$-es diagonális mátrix, a főátlóban nemnegatív valós számokkal, $\boldsymbol{V}$ pedig $N \times N$-es. Amennyiben nem akarjuk csökkenteni az adatok dimenzióját, akkor a transzformált adatok ($\boldsymbol{Y}$), a következőképpen számíthatók:

$$\boldsymbol{Y}^{\boldsymbol{T}} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{W} = \boldsymbol{V}\boldsymbol{\Sigma}^{\mathbf{T}}$$

A képi klasszifikációs feladatokban – és sok más gyakorlati alkalmazásban – az adatok dimenzióját csökkenteni szeretnénk, azzal, hogy a legkevésbé fontos komponenseket elhanyagoljuk. Ezt úgy érhetjük el, hogy az $\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}$ sajátvektorai közül csak a legnagyobb L darab sajátértékhez tartozókat vesszük, tehát a $\boldsymbol{W}$ mátrix első L sorát. Az így kialakuló $\boldsymbol{W}_{\boldsymbol{L}}$ mátrix segítségével leképezett adatok:

$$\boldsymbol{Y = W_L^T X = \Sigma_L V_L^T}$$



3.6. ábra: PCA egy kétdimenziós Gauss-eloszlásra (forrás: [13])

3.4.2. Klaszterezés

A klaszterezési algoritmusok feladata, hogy egy *D* dimenziós térben adott ponthalmaz elemeit csoportokba (klaszterekbe) ossza, úgy hogy az egyes csoportokba tartozó elemek valamilyen szempontból egymáshoz hasonlóak legyenek. Két, alapvetően különböző megközelítés létezik:

- Diszkriminatív módszerek: a pontokat szétválasztjuk klaszterekre az elhelyezkedésük alapján. Tehát egy klaszter egymásoz közel lévő pontok halmaza.
- Generatív módszerek: megpróbáljuk modellezni azokat a valószínűségi eloszlásokat, amely generálta az egyes klaszterek pontjait. Tehát a klaszterekhez egy-egy modellt rendelünk, ezeket a modelleket használjuk a pontok osztályozására.

Diszkriminatív esetben egy pont határozottan egy klaszterhez tartozik, generatív esetben ezzel szemben egy pont tartozhat több klaszterhez is bizonyos valószínűséggel. Ezért szokás a két módszert a „hard-” illetve „soft-klaszterezés” jelzővel illetni. Mindkét típusból egy-egy példát mutatok, amelyeket a képi klasszifikációs feladatokban is használnak: a diszkriminatív K-means-t, és a generatív Gaussian Mixture Model-t. De előtte egy kis kitérőt teszek, ugyanis a klaszterezés során két pont közötti távolságot nem mindig az Euklideszi mérték szerint számolunk. Számos más mérték is használatos, amelyeket a következő pontban írok le.

3.4.2.1. Távolságmértékek

Az euklideszi távolság általánosításaként egy p paraméter függvényében értelmezzük az L_p távolságot. Ebben a metrikus térben egy $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ és egy $\boldsymbol{y} = (y_1, y_2, \dots, y_n)$ vektor távolsága:

$$d_p(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_p = \sqrt[p]{|x_1 - y_1|^p + |x_2 - y_2|^p + \ \dots \ + |x_n - y_n|^p}$$

Például az L_2 távolság pont az Euklideszi teret adja vissza, míg az L_1 távolság a Manhattan távolságnak felel meg. Szokás még az L_∞ metrikát is használni, amelyet a következőképpen értelmeznek:

$$d_\infty(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_\infty = \max\{|x_1 - y_1|, |x_2 - y_2|, \dots, |x_n - y_n|\}$$

Képi leírók hasonlóságának megállapításakor általában az L_1, L_2 vagy L_∞ távolságok valamelyikét használjuk.

3.4.2.2. K-means

A K-means klaszterezéssel a klasztereket K darab középpont körül alakítjuk ki, minden pont abba a klaszterbe fog tartozni, amelyik középponthoz legközelebb van. A klasztert reprezentáló középpont pedig a klaszterbe tartozó pontok átlaga lesz. A klaszterközéppontokat egy iteratív algoritmussal határozzuk meg. Kezdetben inicializáljuk a klaszterközéppontokat, $\boldsymbol{\mu_k^{(0)}}$-t $(k = 1, 2, \dots, K)$ vagy a megadott $X = (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_N})$ ponthalmazból, vagy véletlen pontokat választunk a térben. Majd az i-edik iterációban a következő lépéseket hajtjuk végre:

1. Minden pontra meghatározzuk, hogy melyik klaszterbe tartozik:
$$c_n^{(i+1)} = arg\,\min_k \left\| \boldsymbol{x_n} - \boldsymbol{\mu_k^{(i)}} \right\| \quad n = (1, 2, \dots, N)$$
2. Új klaszterközéppontokat számolunk, az azonos klaszterbe tartozó pontok átlagaként:
$$\boldsymbol{\mu_k^{(i+1)}} = \frac{1}{s_k^{(i+1)}} \sum_{n:\, c_n^{(i+1)} = k} \boldsymbol{x_n} \quad k = (1, 2, \dots, K)$$
$$ahol \quad s_k^{(i+1)} = \left| \left\{ n: c_n^{(i+1)} = k \right\} \right|$$

Az algoritmus akkor fejeződik be, amikor konvergenciát tapasztalunk, vagyis nem változnak a klaszterközéppontok ($\boldsymbol{\mu_k^{(i+1)}} = \boldsymbol{\mu_k^{(i)}}\ \forall k$). Közben előfordulhat, hogy egy klaszter kiürül, ilyenkor szokás véletlenszerűen egy új klaszterközéppontot választani, ha mindenképpen K darab klasztert szertnénk. De folytathatjuk az algoritmust enélkül is, ilyenkor a klaszterek száma a pontok térbeli eloszlásához fog adaptálódni.

3.4.2.3. Gaussian Mixture Model [14]

A Gaussian Mixture Model-ben (GMM) a klaszterekre úgy tekintünk, mintha a pontjaikat egy-egy ($\boldsymbol{\mu_k}$ várható értékű és σ_k szórású) normális eloszlású valószínűségi változó generálta volna. Az elnevezés tehát innen ered, a pontok valószínűségi sűrűségfüggvényét K darab Gauss-féle függvény súlyozott összegével próbáljuk leírni:

$$f(\boldsymbol{x};\theta) = \sum_{k=1}^{K} p_k g(\boldsymbol{x};\boldsymbol{\mu_k},\sigma_k)\ ,$$

$$ahol \quad g(\boldsymbol{x};\boldsymbol{\mu_k},\sigma_k) = \frac{1}{\left(\sqrt{2\pi}\sigma_k\right)^D} e^{-\frac{1}{2}\left(\frac{\|\boldsymbol{x}-\boldsymbol{\mu_k}\|}{\sigma_k}\right)^2}, \quad \sum_{k=1}^{K} p_k = 1$$

$$és \quad \theta = (\theta_1, \dots, \theta_K) = ((p_1, \boldsymbol{\mu_1}, \sigma_1), \dots, (p_K, \boldsymbol{\mu_K}, \sigma_K))$$

Fent D jelöli a dimenziók számát, K pedig a Gauss-féle függvények számát. Tehát θ egy $K(D+2)$ dimenziós paraméter vektor, amely a p_k súlyozó együtthatókat, és az egyes Gauss-függvények $\boldsymbol{\mu_k}$ átlagát és σ_k szórását tartalmazza.

A modellezés során azt a θ-t keressük, amelyre a legnagyobb a valószínűsége annak, hogy a megadott $X = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_N}\}$ ponthalmaz keletkezett. Az $f(\boldsymbol{x};\theta)$ sűrűségfüggvény mellett annak a valószínűsége, hogy az $\boldsymbol{x}$ pont egy $d\boldsymbol{x}$ méretű környezetéből választunk egy pontot $f(\boldsymbol{x};\theta)d\boldsymbol{x}$. Mivel $d\boldsymbol{x}$ egy konstans, ezért figyelmen kívül hagyható, vagyis a következő függvény maximumát keressük:

$$\Lambda(X;\theta) = \prod_{n=1}^{N} f(\boldsymbol{x_n};\theta) = \prod_{n=1}^{N}\sum_{k=1}^{K} p_k g(\boldsymbol{x_n};\boldsymbol{\mu_k},\sigma_k)$$

$\Lambda(X;\theta)$ helyett kereshetjük a logaritmusának a maximumát is:

$$\lambda(X;\theta) = log\ \Lambda(X;\theta) = \sum_{n=1}^{N} log \sum_{k=1}^{K} p_k g(\boldsymbol{x_n};\boldsymbol{\mu_k},\sigma_k)$$

Így már Expectation Maximization (EM) algoritmussal iteratívan meg tudjuk határozni azon $\theta = ((p_1, \boldsymbol{\mu_1}, \sigma_1), \dots, (p_K, \boldsymbol{\mu_K}, \sigma_K))$ paramétereket, amelyek mellett a fenti függvény maximális.. Jelölje $p(k|\boldsymbol{x_n})$ azt a feltételes valószínűséget, hogy az adott ponthalmazból az $\boldsymbol{x_n}$ pontot a k-adik Gauss-féle függvény generálta. Vagyis

$$p(k|\boldsymbol{x_n}) = \frac{p_k g(\boldsymbol{x_n};\boldsymbol{\mu_k},\sigma_k)}{\sum_{m=1}^{K} p_m g(\boldsymbol{x_n};\boldsymbol{\mu_m},\sigma_m)}$$

Az EM algoritmus lépéseinek részletes levezetését most mellőzzük, ez megtalálható [14]-ben. Jelölje az i-edik iterációban a paramétereket $p_k^{(i)}$, $\boldsymbol{\mu_k^{(i)}}$, $\sigma_k^{(i)}$ $(k = 1, 2, \dots, K)$. Az algoritmus elején kezdőértéket adunk a $p_k^{(0)}$, $\boldsymbol{\mu_k^{(0)}}$, $\sigma_k^{(0)}$ paramétereknek $\forall k$-ra. Ezt követően két lépést ismétlünk egymás után. Az Expectation (E) lépésben meghatározzuk a $p_k^{(i)}$, $\boldsymbol{\mu_k^{(i)}}$, $\sigma_k^{(i)}$ paramétereket feltételezve az egyes pontoknak a klaszterekbe tartozási valószínűségeit, $p^{(i)}(k|\boldsymbol{x_n})$-t ($\forall n, k$-ra). A Maximization (M) lépésben pedig olyan $p_k^{(i+1)}$, $\boldsymbol{\mu_k^{(i+1)}}$, $\sigma_k^{(i+1)}$ paramétereket keressünk, amelyek jobban modellezik a ponthalmazt, vagyis a $\lambda(X;\theta)$ függvényérték nagyobb. Ez utóbbit a $\lambda(X;\theta)$ függvény egy alsó becslésének szélsőérték keresésével érjük el. Egész pontosan, az $(i+1)$-edik iteráció lépései:

1. E lépés:

$$p^{(i)}(k|\boldsymbol{x_n}) = \frac{p_k^{(i)} g(\boldsymbol{x_n}; \boldsymbol{\mu_k^{(i)}}, \sigma_k^{(i)})}{\sum_{m=1}^{K} p_m^{(i)} g(\boldsymbol{x_n}; \boldsymbol{\mu_m^{(i)}}, \sigma_m^{(i)})} \quad \forall n, k$$

2. M lépés:

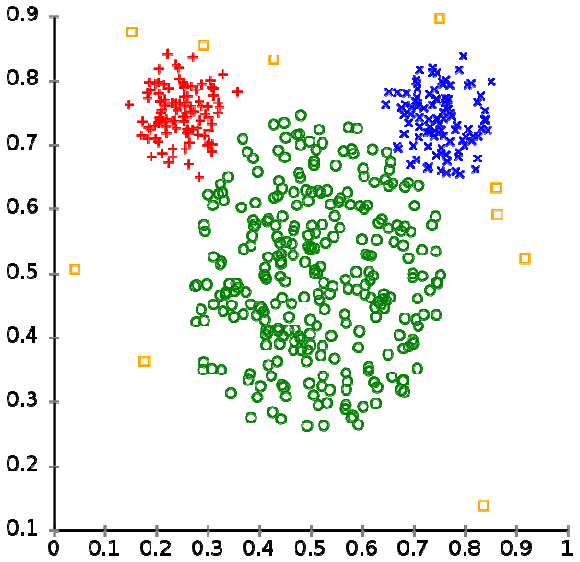
$$\boldsymbol{\mu_k^{(i+1)}} = \frac{\sum_{n=1}^{N} p^{(i)}(k|\boldsymbol{x_n})\, \boldsymbol{x_n}}{\sum_{n=1}^{N} p^{(i)}(k|\boldsymbol{x_n})} \quad \forall k$$

$$\sigma_k^{(i+1)} = \sqrt{\frac{1}{D} \frac{\sum_{n=1}^{N} p^{(i)}(k|\boldsymbol{x_n}) \left\| \boldsymbol{x_n} - \boldsymbol{\mu_k^{(i+1)}} \right\|^2}{\sum_{n=1}^{N} p^{(i)}(k|\boldsymbol{x_n})}} \quad \forall k$$

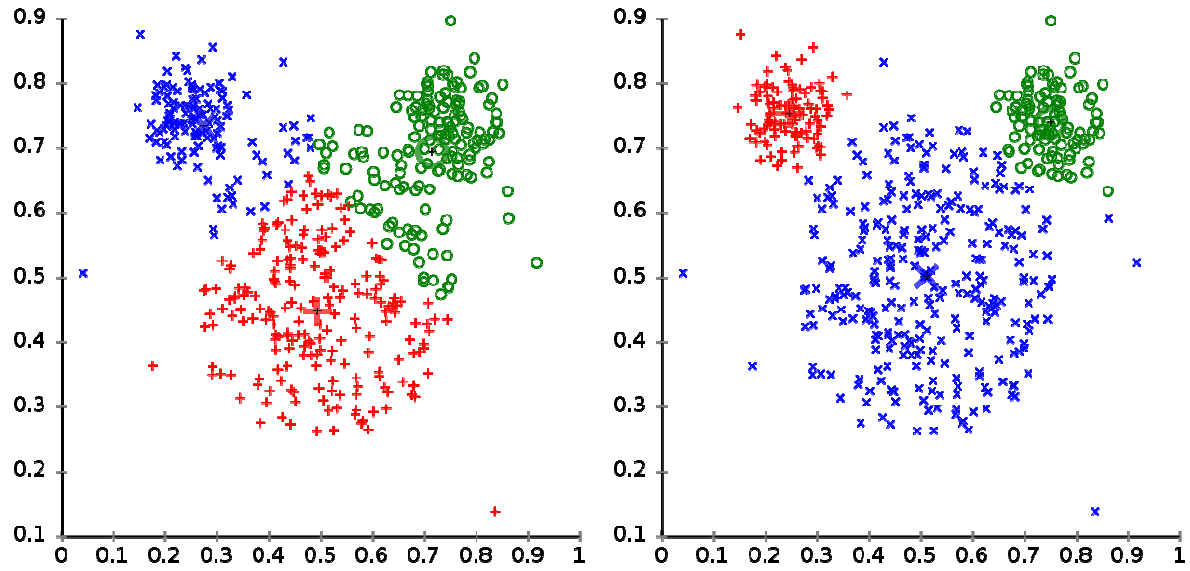
$$p_k^{(i+1)} = \frac{1}{N} \sum_{n=1}^{N} p^{(i)}(k|\boldsymbol{x_n}) \quad \forall k$$

Ugyan ezek az eredmények egy matematikai bizonyítás eredményeképp adódtak, intuitív jelentés is társítható hozzájuk, hiszen $\boldsymbol{\mu_k^{(i+1)}}$ éppen a pontoknak a k-adik klaszterbe tartozás valószínűsége szerint súlyozott átlaga, $\sigma_k^{(i+1)}$ az ehhez tartozó szórás, $p_k^{(i+1)}$ pedig az átlagos klaszterbe tartozás valószínűségével arányos.

Az alábbi képek a GMM előnyeit mutatják a K-means klaszterezéssel szemben, a mesterséges „egér” adathalmazon. Feljebb látható az eredeti adathalmaz, alább bal oldalon a K-meansszel kialakított klaszterek, jobb oldalon pedig a GMM klaszterek (a GMM színezésénél egy pont itt egyértelműen be van sorolva egy klaszterbe, a maximális valószínűségűbe).



3.7. ábra: Az egér adathalmaz (forrás: [15])



3.8. ábra: A K-means és a GMM klaszterezés (forrás: [15])

3.5. Magas szintű képi leírók

Egy-egy képet egy magas szintű leíró vektorral kívánunk jellemezni, amely alapján osztályba soroljuk a képet. A Bag of Words alapelvből adódó logikus módszer a képen az egyes vizuális szavak előfordulási gyakoriságaiból egy hisztogram vektor készítése (3.5.1). Ezen kívül bemutatok még két további, fejlettebb reprezentációs eljárást: a GMM alapú Fisher-vektort (3.5.2) és a K-meansre épülő Szuper-vektort (**Hiba! A hivatkozási forrás nem található.**).

3.5.1. Bag of Words

A kép egyes pontjaiban kiszámolt alacsony szintű leírókat ($X = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_N}\}$) klaszterekbe soroljuk, ezzel meghatározva, hogy mely vizuális szónak tekintjük az adott lokális régiót. K-means vagy más hard-klaszterezés esetén ez azt jelenti, hogy egy $\boldsymbol{x}$ vektort a hozzá legközelebb eső klaszterközépponttal approximálunk:

$$\boldsymbol{\mu}_*(\boldsymbol{x}) = \arg\min_{\boldsymbol{\mu} \in C} \|\boldsymbol{x} - \boldsymbol{\mu}\|,$$

ahol $C = \{\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \dots, \boldsymbol{\mu_K}\}$ a klaszterközéppontok halmaza. Ezzel gyakorlatilag kvantáljuk a vektorteret, ezért ezt a módszert vektor kvantálás néven szokták említeni (Vector Quantization, VQ). Ezt követően megszámoljuk, hogy az egyes klaszterekbe mennyi vektor esett és az így kapott hisztogram vektort esetleg normáljuk. A képet jellemző magas szintű leíró vektor:

$$\boldsymbol{v}(X) = \frac{1}{N} \left[|X_1| \quad |X_2| \quad \dots \quad |X_K| \right],$$

ahol N a képen kiszámolt alacsony szintű leíró vektorok száma, K a klaszterek száma, és X_i jelöli az i-edik klaszterbe eső vektorok halmazát.

GMM, vagy általában generatív klaszterezés esetén, amikor egy-egy vektor több klaszterhez is tartozhat bizonyos valószínűségekkel, ezeket a valószínűségeket összegezzük, így kapjuk a hisztogram vektort. A korábbi jelölésekkel:

$$\boldsymbol{v}(X) = \frac{1}{N}\,[p_1 \quad p_2 \quad \dots \quad p_K]\,, \quad p_k = \sum_{n=1}^{N} p(k|\boldsymbol{x_n})$$

Itt tehát $p(k|n)$ azt jelenti, hogy az n-edik vektor milyen valószínűséggel tartozik a k-adik klaszterhez.

3.5.2. Fisher-vektor

A Fisher kernelt Jaakkola és Haussler dolgozta ki [16] a generatív statisztikai modellezés és diszkriminatív osztályozás előnyeinek kombinálására. A GMM alapú Fisher kernel használatát képi klasszifikációban Perronin és Dance javasolta 2007-ben [17], de korábban már sikeresen alkalmazták audio fájlok indexelésére [18] is. Az alapötlet az, hogy egy jelet az azt generáló valószínűségi sűrűségfüggvény gradiensével jellemzünk. Az előző, hisztogram reprezentációval szemben előnye, hogy több információt tartalmazó, magasabb dimenziós vektorokat kapunk, amelyeket egy diszkriminatív osztályozó eredményesebben tud kategóriákba osztani.

Legyen p egy valószínűségi sűrűségfüggvény, amelynek paramétereit λ-val jelöljük. Ekkor az $X = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_N}\}$ mintákat a következő gradiens vektorral jellemezzük:

$$\nabla_\lambda \log p(X|\lambda)$$

Ennek intuitív jelentést is tulajdoníthatunk: a sűrűségfüggvény logaritmusának deriváltja azt jelenti, hogy milyen irányba kellene módosítani a paramétereket, hogy a legjobban illeszkedjen a mintahalmazra.

Bevezetve az $L(X|\lambda) = \log p(X|\lambda)$ jelölést, és feltételezve a minták függetlenségét, kapjuk:

$$L(X|\lambda) = \sum_{\mathrm{n}=1}^{\mathrm{N}} \log p(\boldsymbol{x_n}|\lambda)$$

Ahol $p(\boldsymbol{x_n}|\lambda)$ annak a valószínűsége, hogy az x_n mintát a GMM generálta:

$$p(\boldsymbol{x_n}|\lambda) = \sum_{k=1}^{K} p_k g(\boldsymbol{x_n}; \boldsymbol{\mu_k}, \boldsymbol{\sigma_k})$$

A GMM paraméterei az egyes Gauss-függvények súlya, középpontja és szórása, $\lambda = \{p_k, \boldsymbol{\mu_k}, \boldsymbol{\sigma_k} | k = 1 \dots K\}$. Feltételezzük, hogy a Gauss-függvények szórása, $\boldsymbol{\sigma_k}$ diagonális mátrix. A Fisher-vektor tagjait az $L(X|\lambda)$ függvény deriválásával kapjuk:

$$\boldsymbol{v}(X) = \left[\frac{\partial L(X|\lambda)}{\partial p_2} \quad \dots \quad \frac{\partial L(X|\lambda)}{\partial p_K} \quad \frac{\partial L(X|\lambda)}{\partial \mu_1^1} \quad \dots \quad \frac{\partial L(X|\lambda)}{\partial \mu_K^D} \quad \frac{\partial L(X|\lambda)}{\partial \sigma_1^1} \quad \dots \quad \frac{\partial L(X|\lambda)}{\partial \sigma_K^D}\right]$$

Felhasználva a GMM tárgyalásakor bevezetett $p(k|\boldsymbol{x_n})$ jelölést:

$$\frac{\partial L(X|\lambda)}{\partial p_k} = \sum_{n=1}^{N} \left(\frac{p(k|\boldsymbol{x_n})}{p_k} - \frac{p(1|\boldsymbol{x_n})}{p_1} \right), \quad k = 2 \dots K$$

$$\frac{\partial L(X|\lambda)}{\partial \mu_k^d} = \sum_{n=1}^{N} p(k|\boldsymbol{x_n}) \left(\frac{x_n^d - \mu_k^d}{\left(\sigma_k^d\right)^2} \right), \quad k = 1 \dots K, d = 1 \dots D$$

$$\frac{\partial L(X|\lambda)}{\partial \sigma_k^d} = \sum_{n=1}^{N} p(k|\boldsymbol{x_n}) \left(\frac{\left(x_n^d - \mu_k^d\right)^2}{\left(\sigma_k^d\right)^3} - \frac{1}{\sigma_k^d} \right), \quad k = 1 \dots K, d = 1 \dots D$$

Fent D egy minta dimenzióinak számát jelöli. A Gauss-függvények súlya szerinti deriváltakból csak $K - 1$ darab van, mert $\sum_{k=1}^{K} p_k = 1$, így például p_1 kiszámítható a többi ismeretében, és ezért csak $K - 1$ szabad paraméter van. Tehát a Fisher-vektor $K(2D + 1) - 1$ dimenziós.

A gradiens vektor az egyes dimenzióiban eltérő nagyságrendű lehet, ezért fontos a normalizálása mielőtt egy osztályozó bemenetére adjuk. Perronnin és Dance ezt az eredeti cikkben a Fisher információs mátrix approximációjával végzik. Egy későbbi munkájukban Perronninék ezen kívül még egy úgynevezett power normalizációt és L2 normalizációt is javasolnak [19].

3.5.3. Szuper-vektor

A Szuper-vektor a VQ módszer egy kiterjesztése, amit 2010-ben alkalmaztak először a képklasszifikációs feladatra [20]. A nem-ellenőrzött tanítással kialakított vizuális kódszótárból indul ki, amelyben a kódszavak $(\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \dots, \boldsymbol{\mu_K})$ lineáris kombinációjával közelítünk egy $\boldsymbol{x}$ alacsony szintű leíró vektort:

$$\boldsymbol{x} \approx \sum_{k=1}^{K} \gamma_k(\boldsymbol{x}) \boldsymbol{\mu_k}$$

Tehát $\gamma_k(\boldsymbol{x})$ az egyes kódszavak, mint bázisvektorok együtthatói. Vegyük észre, hogy ez magában foglalja a VQ módszert is, hiszen ekkor egyetlen együttható értéke 1, a többi 0. Soft-klaszterezés esetén tetszőleges 0 és 1 közötti értékeket engedünk meg, úgy, hogy $\sum_{k=1}^{K} \gamma_k(\boldsymbol{x}) = 1$.

Egy $\boldsymbol{x}$ vektor Szuper-vektor kódolását a következőképpen értelmezzük:

$$\varphi(\boldsymbol{x}) = [s\gamma_1(\boldsymbol{x}) \quad \gamma_1(\boldsymbol{x})(\boldsymbol{x} - \boldsymbol{\mu_k}) \quad \dots \quad s\gamma_K(\boldsymbol{x}) \quad \gamma_K(\boldsymbol{x})(\boldsymbol{x} - \boldsymbol{\mu_k})],$$

ahol s egy konstans. Amennyiben a VQ módszert alkalmazzuk, ez egy nagyon ritka vektor, csupán $D + 1$ nem 0 elem van benne, miközben a dimenziója $K(D + 1)$, ahol D az alacsony szintű leíró vektorok dimenziója. $\varphi(\boldsymbol{x})$ ekkor nem mást tartalmaz, mint az s konstanst egy adott koordinátán, majd a legközelebbi klaszterközépponttól vett távolságvektort. Ezekből a

képre vonatkozó magas szintű leíró vektort a következőképpen alkotjuk (a korábbi jelöléseket alkalmazva):

$$\boldsymbol{v}(X) = \frac{1}{N}\sum_{k=1}^{K}\frac{1}{\sqrt{p_k}}\sum_{\boldsymbol{x}\in X_k}\varphi(\boldsymbol{x}),\quad p_k = \frac{|X_k|}{N}$$

A szerzők a fentieket K-means klaszterezés felhasználásával végzik. Emellett soft K-means alkalmazását is javasolják, úgy, hogy a legközelebbi 20 klaszterközéppont lineáris kombinációjával approximálnak egy vektort. A klaszterközéppontok együtthatói ebben az esetben az egyes klaszterekhez tartozás valószínűségei, ($\gamma_k(\boldsymbol{x}) = p(k|\boldsymbol{x})$), és így a klaszterközépponttól vett távolságtól függnek. A cikkben nem térnek ki arra szerzők, hogy konkrétan milyen módon számítják a klaszterekhez tartozás valószínűségeit. A fenti képletben $p_k = \frac{1}{N}\sum_{n=1}^{N}\gamma_k(\boldsymbol{x_n})$-re módosul. Ezt a változatot soft Szuper-vektornak nevezik a szerzők.

3.5.4. Spatial pooling

Az előzőekben egy-egy magas szintű leíró vektort mindig egy kép jellemzésére értelmeztünk. Lehetséges azonban a magas szintű leíró vektorokat a kép egy régiójára is számítani, ha csak az adott régióban kiszámolt alacsony szintű leíró vektorokat vesszük figyelembe. A kép régiókra bontása történhet egy egyszerű rács ráillesztésével (a leggyakrabban $3x1$-es, illetve $2x2$-es rácsot alkalmaznak), vagy szegmentálással. Ezáltal egy kicsit figyelembe vesszük a képen lévő vizuális kódszavak térbeli elrendeződését is. Gyakran alkalmazott módszer az, hogy egy kép több régiójára kiszámolt magas szintű leírókat egymás után fűzzük egy vektorba. Például [20]-ban a szerzők a Szuper-vektoros leírást kiszámolják az egész képre, és még a $3x1$-es és $2x2$-es felosztásban a kép minden régiójára, majd ezek konkatenációjával jellemeznek egy képet:

$$\boldsymbol{v_s}(X) = [\boldsymbol{v}(X)\quad \boldsymbol{v}(X_{11}^{2x2})\quad \boldsymbol{v}(X_{12}^{2x2})\quad \boldsymbol{v}(X_{21}^{2x2})\quad \boldsymbol{v}(X_{22}^{2x2})\quad \boldsymbol{v}(X_{11}^{3x1})\quad \boldsymbol{v}(X_{21}^{3x1})\quad \boldsymbol{v}(X_{31}^{3x1})],$$

ahol X_{ij}^{axb} jelöli a kép axb-s felosztásában az i-edik sor j-edik partíciójába eső alacsony szintű vektorok halmazát.

Kernel módszerek (3.6.2) esetén lehetséges a régiókra kiszámolt leíró vektoroknak lineáris kombinálása kernel formában (4.1.3). Kernelek egy másik érdekes alkalmazása az úgynevezett Spatial Pyramid Matching [21].

3.6. Gépi tanulás

Az előző pontban bemutatott magas szintű leíró vektorok alapján osztályozzuk a képeket, azaz minden kategóriában adunk egy jóslatot arra vonatkozóan, hogy szerepel-e az adott objektum a képen, illetve jellemző-e az adott fogalom a képre. Ehhez adottak a tanítóhalmaz képein kiszámított magas szintű leírók ($\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_T}$) és a hozzájuk tartozó annotációk kategóriánként ($y_1, y_2, \dots, y_T$). Ezek felhasználásával, gépi tanulásos módszerekkel hozunk

létre egy osztályozót, amely egy tesztképre kiszámított $\boldsymbol{x}$ leíró vektorra ad egy $f(\boldsymbol{x})$ jóslatot (minden kategóriában).

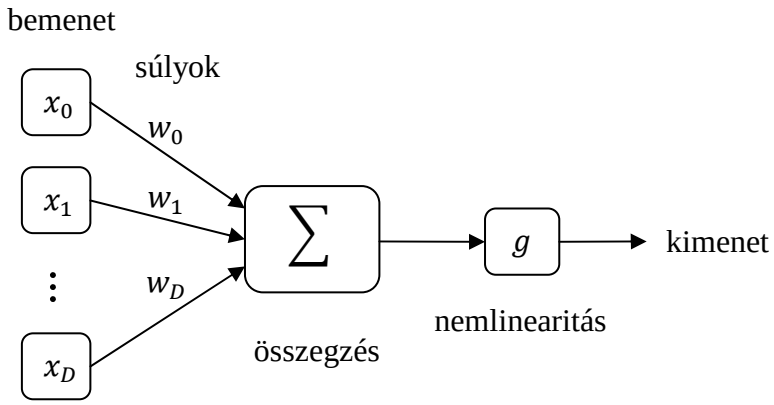
Több kategória esetén választási lehetőségünk van, hogy a tanítást kategóriánként külön végezzük, vagy együtt, ezzel megpróbálva kihasználni a kategóriák közötti összefüggéseket. Az előbbi megközelítést „n-ből egy” módszernek nevezik, mert minden tanításnál az egyik osztályba tartozó pontok a pozitív minták, a többit pedig negatív mintaként használjuk fel. Ilyenkor tehát n darab egymástól független, bináris osztályozót hozunk létre. Egyszerűsége ellenére ezt a módszert használják a leggyakrabban, sokszor jobb eredményt ad, mint más, összetettebb módszerek. Ebben a dolgozatban is a bináris osztályozással foglalkozunk, a többosztályos klasszifikáció részletesebb tárgyalásával nem.

3.6.1. Lineáris klasszifikátorok

Lineáris klasszifikátorról akkor beszélünk, amikor a két osztály pontjait egy hipersíkkal próbáljuk szeparálni. Mivel a hipersíktól vett előjeles távolságot a normálvektorral vett skaláris szorzattal számíthatjuk, egy $\boldsymbol{x}$ vektorra adott $f(\boldsymbol{x})$ jóslat ennek függvényeként kapható:

$$f(\boldsymbol{x}) = g(\boldsymbol{w} \cdot \boldsymbol{x}) = g\left(\sum_{i=1}^{D} w_i x_i\right),$$

ahol $\boldsymbol{w}$ a hipersík normálvektora, D a dimenziók száma, a g függvény szerepe pedig az, hogy az osztályozó kimenetét a megfelelő értékkészletbe képezze. g egy egyszerű küszöbfüggvény is lehet, vagy ha az osztályba sorolás megbízhatóságát is érzékeltetni szeretnénk, akkor egy szigmoid típusú függvényt használunk. A fenti képletben nincs eltolásvektor, ehelyett minden $\boldsymbol{x}$ vektort kiegészítünk egy $x_0 = 1$ bias változóval. Mivel a $\boldsymbol{w}$ vektorral gyakorlatilag súlyozzuk az $\boldsymbol{x}$ vektor koordinátáit, szokás súlyvektornak is nevezni. Az alábbi ábra szemlélteti a lineáris klasszifikátorok általános felépítését.

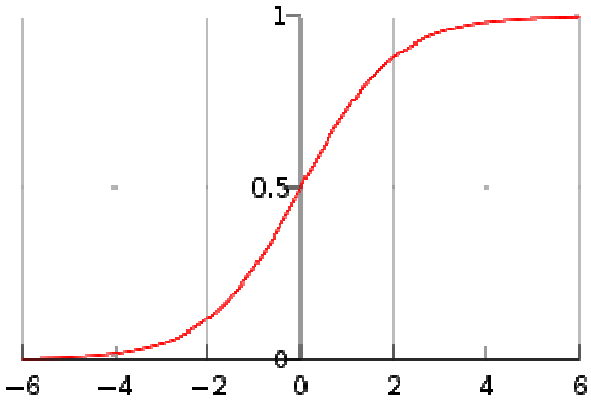


3.9. ábra: Egy lineáris klasszifikátor

3.6.1.1. Logisztikus regresszió

Logisztikus regresszió esetén az osztályozó kimenetén a logisztikus függvényt alkalmazzuk:

$$g(z) = \frac{1}{1 + e^{-z}}$$



3.10. ábra: A logisztikus függvény

Tehát $\boldsymbol{w}$ súlyvektor esetén az $\boldsymbol{x}$ vektorra vonatkozó jóslat:

$$f(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}\cdot\boldsymbol{x}}}$$

Mivel 0 és 1 közötti értéket kapunk, tekinthetünk rá úgy, mint az 1-es osztályhoz tartozás valószínűsége:

$$P(1|\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{1 + e^{-\boldsymbol{w}\cdot\boldsymbol{x}}}$$

A 0 osztályhoz tartozás valószínűsége pedig:

$$P(0|\boldsymbol{x}, \boldsymbol{w}) = 1 - \frac{1}{1 + e^{-\boldsymbol{w}\cdot\boldsymbol{x}}} = \frac{e^{-\boldsymbol{w}\cdot\boldsymbol{x}}}{1 + e^{-\boldsymbol{w}\cdot\boldsymbol{x}}}$$

A tanítás során azt a $\boldsymbol{w}^*$ súlyvektort keressük, amelyre az $\{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \dots, (\boldsymbol{x_T}, y_T)\}$ tanítóhalmazon a legnagyobb annak a valószínűsége, hogy minden tanítópontot helyesen osztályozunk. A valószínűség helyett annak logaritmusát is maximalizálhatjuk:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, L(\boldsymbol{w}),$$

$$L(\boldsymbol{w}) = \ln \prod_{i=1}^{T} P(y_i|\boldsymbol{x_i}, \boldsymbol{w}) = \sum_{i=1}^{T} \ln P(y_i|\boldsymbol{x_i}, \boldsymbol{w}) =$$

$$= \sum_{i=1}^{T} y_i \ln P(1|\boldsymbol{x_i}, \boldsymbol{w}) + (1 - y_i) \ln P(0|\boldsymbol{x_i}, \boldsymbol{w})$$

A $\boldsymbol{w}^*$ súlyvektort iteratív módszerrel határozzuk meg, kiindulunk egy tetszőlegesen választott $\boldsymbol{w}^{(0)}$-ból, majd ezt módosítjuk minden lépésben a $\nabla L(\boldsymbol{w})$ gradiens vektor λ-szorosával (λ a bátorsági tényező):

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \lambda \cdot \nabla L(\boldsymbol{w}) \ ,$$

$$\nabla L(\boldsymbol{w}) = \left[\frac{\partial L(\boldsymbol{w})}{\partial w_1} \quad \frac{\partial L(\boldsymbol{w})}{\partial w_2} \quad \dots \quad \frac{\partial L(\boldsymbol{w})}{\partial w_D}\right],$$

$$\frac{\partial L(\boldsymbol{w})}{\partial w_k} = \sum_{i=1}^{T} x_{ik}(y_i - P(1|\boldsymbol{x_i}, \boldsymbol{w})) \ , \quad i = 1 \dots D$$

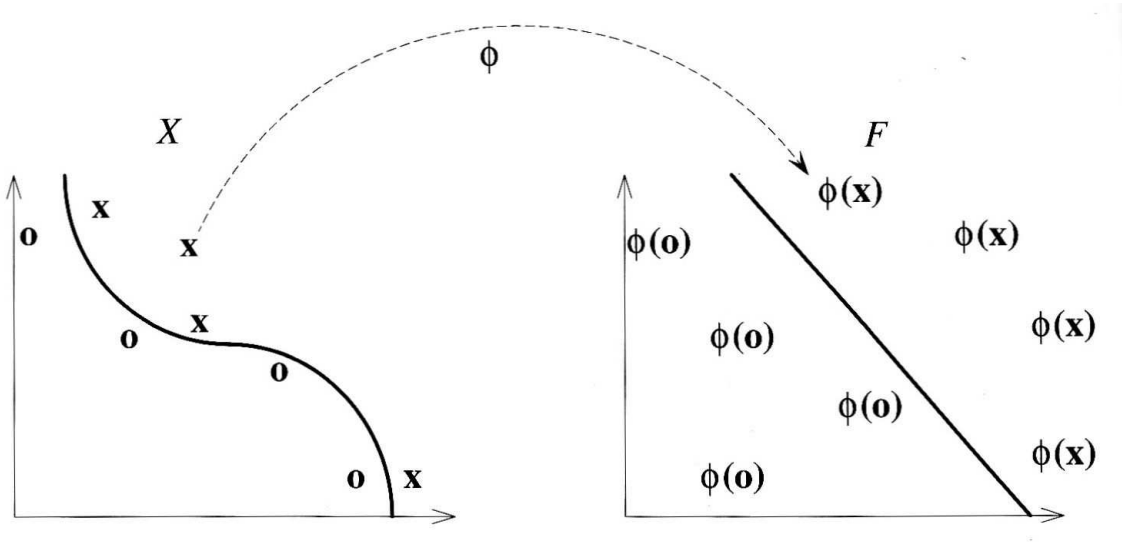
Fent D jelöli a minták dimenzióinak számát, x_{ik} pedig az $\boldsymbol{x_i}$ vektor k-adik koordinátáját jelenti.

3.6.2. A kernel trükk

Lehetséges, hogy a minták nem jól szeparálhatók abban a térben, ahol elhelyezkednek (az úgynevezett bemeneti térben). Egy gyakran alkalmazott előfeldolgozási lépés az adatok leképezése egy másik térbe (a jellemző térbe), ahol már jobban szét lehet őket választani:

$$\boldsymbol{x} = [x_1 \quad x_2 \quad \dots \quad x_D] \rightarrow \boldsymbol{\varphi}(\boldsymbol{x}) = [\varphi_1(\boldsymbol{x}) \quad \varphi_2(\boldsymbol{x}) \quad \dots \quad \varphi_S(\boldsymbol{x})]$$

Az alábbi ábra szemlélteti ezt az alapötletet, az ábrázolt mintapontok lineárisan nem szeparálhatók a bemeneti térben, de a jellemző térben igen.



3.11. ábra: Leképezés a jellemző térbe (forrás: [23])

Egy $(\boldsymbol{w}, b)$ hipersíkkal jellemzett lineáris klasszifikátorral a jóslatokat a jellemző térbe leképezett vektorokon a következőképp számíthatjuk:

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{\varphi}(\boldsymbol{x}) + b$$

Ha az osztályozó olyan, hogy a hipersík normálvektora ($\boldsymbol{w}$) kifejezhető a tanítópontok lineáris kombinációjaként (például a következő pontban bemutatott SVM), akkor a döntési függvényt számíthatjuk csupán a tanítópontok és a tesztpont (jellemzőtérbeli) skaláris szorzatának felhasználásával:

$$f(\boldsymbol{x}) = \sum_{i=1}^{t} y_i \alpha_i \boldsymbol{\varphi}(\boldsymbol{x_i}) \cdot \boldsymbol{\varphi}(\boldsymbol{x}) + b$$

Ha ki tudjuk számítani a $\boldsymbol{\varphi}(\boldsymbol{x_i}) \cdot \boldsymbol{\varphi}(\boldsymbol{x})$ jellemzőtérbeli skaláris szorzatot az eredeti pontok ($\boldsymbol{x_i}$ és $\boldsymbol{x}$) függvényeként, vagyis létezik olyan K függvény, melyre:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{\varphi}(\boldsymbol{x}) \cdot \boldsymbol{\varphi}(\boldsymbol{z}),$$

akkor a jellemzőtérbe képezést kiküszöbölhetjük, és közben egy nemlineáris osztályozót kapunk. A K függvényt kernel függvénynek hívjuk. Tehát a kernel függvénnyel így írhatjuk fel a döntési függvényt:

$$f(\boldsymbol{x}) = \sum_{i=1}^{t} y_i \alpha_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b$$

Ezt követően általában a jellemző térbe képező $\boldsymbol{\varphi}$ függvényt nem is határozzuk meg explicit módon, csak a kernelekben gondolkodunk. Azt, hogy milyen függvényt választhatunk kernelnek, Mercer tétele írja le, itt ezzel az elméleti résszel nem foglalkozunk (megtalálható például [23]-ben). Az alábbi táblázatban (a teljesség igénye nélkül) összefoglalom a leggyakrabban alkalmazott kernel függvényeket:

$Lineáris$	$K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} \cdot \boldsymbol{y}$
$Polinomiális\ (d\ fokszámmal)$	$K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y} + 1)^d$
$Gauss - féle\ (\sigma\ szórással)$	$K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\frac{1}{2}\left(\frac{\|\boldsymbol{x}-\boldsymbol{y}\|}{\sigma}\right)^2}$

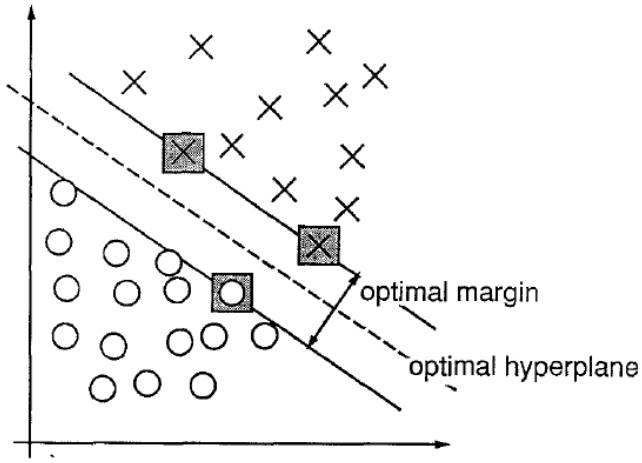
3.1. táblázat: Kernel függvények

3.6.3. Support Vector Machine (SVM)

Az SVM megalkotása Vladimir Vapnik nevéhez fűzhető, később egy nagyon fontos módosítását, a szoft margójú változatot Corinna Cortesszel együtt dolgozta ki [22]. Ez talán jelenleg a leggyakrabban használt klasszifikátor, rengeteg irodalma van. Az eredeti cikk mellett még két könyv ([23], [24]) segítségével foglaltam össze a számunkra leglényegesebb tudnivalókat az alábbiakban.

3.6.3.1. Maximális margójú lineáris szeparálás

Tételezzük fel, hogy a tanítóhalmaz mintapontjai lineárisan szeparálhatók. Ekkor végtelen sok olyan hipersík van, ami helyesen választja szét a tanítópontok két osztályát, ezek közül azt szeretnénk választani, amely esetén a legjobb az osztályozó általánosító képessége. Feltételezve, hogy a tanítópontok jól reprezentálják a valódi eloszlást, ezt az „optimális" hipersíkot úgy választjuk meg a jó hipersíkok közül, hogy a lehető legtávolabb helyezkedjen el a tanítópontoktól. Pontosabban a hipersík és a hozzá legközelebb eső tanítópontok távolságát maximalizáljuk. A hipersík két oldalán tehát „üres sávok" helyezkednek el, ezek együttes szélességét hívjuk margónak, amit maximalizálni szeretnénk (3.12. ábra).



3.12. ábra: A maximális margójú hipersík (forrás:[22])

Legyen a tanítóminták halmaza $S = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \dots, (\boldsymbol{x_T}, y_T)\}$. Ezek lineárisan szeparálhatók, ha létezik olyan $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ egyenletű hipersík, melyre:

$$\boldsymbol{w} \cdot \boldsymbol{x_i} + b \geq 1, \quad ha \;\; y_i = 1,$$

$$\boldsymbol{w} \cdot \boldsymbol{x_i} + b \leq -1, \quad ha \;\; y_i = -1.$$

$\boldsymbol{w}$ és b tetszőleges skalárral megszorozható, ez nem változtatja a hipersíkot, ezért írhattunk az egyenlőtlenségek jobb oldalára 1-et és -1-et, és feltételezhetjük, hogy egyenlőség áll fenn a margó határán elhelyezkedő tanítópontokra. Ekkor az osztályozás margójának nagysága:

$$\rho(\boldsymbol{w}, b) = \min_{\{x_i : y_i = 1\}} \frac{\boldsymbol{w} \cdot \boldsymbol{x_i}}{\|\boldsymbol{w}\|} - \max_{\{x_i : y_i = -1\}} \frac{\boldsymbol{w} \cdot \boldsymbol{x_i}}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|} = \frac{2}{\sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}}$$

Vagyis az optimális hipersík az, amelyre $\boldsymbol{w} \cdot \boldsymbol{w}$ minimális, $y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \geq 1 \;\; (i = 1 \dots T)$ feltételek mellett. Ez egy kvadratikus programozási feladat, amelynek megoldásával megkapjuk az optimális $(\boldsymbol{w}^*, b^*)$ hipersíkot. Ennek részletezését mellőzzük (megtalálható [22]-ban), csak a számunkra fontos lépéseket emeljük ki. Az α_i Lagrange-multiplikátorokkal felírt duális alak a következőképpen alakul:

$$maximalizálandó: \quad L(\boldsymbol{\alpha}) = \sum_{i=1}^{T} \alpha_i - \frac{1}{2} \sum_{i=1}^{T} \sum_{j=1}^{T} y_i y_j \alpha_i \alpha_j \, \boldsymbol{x_i} \cdot \boldsymbol{x_j} \,,$$

$$ahol: \quad \sum_{i=1}^{T} y_i \alpha_i = 0 \,, \qquad \alpha_i \geq 0 \,, \quad i = 1 \ldots T$$

A fenti probléma megoldásaként meghatározott α_i^* multiplikátorok segítségével $\boldsymbol{w}^*$ felírható a tanítópontok lineáris kombinációjaként:

$$\boldsymbol{w}^* = \sum_{i=1}^{T} y_i \alpha_i^* \boldsymbol{x_i}$$

Általában sok tanítópontra $\alpha_i^* = 0$ lesz. Azokat a tanítópontokat, amelyekre $\alpha_i^* > 0$, szupport vektoroknak nevezzük, mivel ezek vesznek részt a megoldás előállításában. Belátható az is, hogy ezek a szupport vektorok épp azok a tanítópontok lesznek, amelyek a margó határán helyezkednek el (a 3.12. ábra bekeretezett pontjai), vagyis ezekre a pontokra $y_i(\boldsymbol{w}^* \cdot \boldsymbol{x_i} + b^*) = 1$. Az előzőek értelmében $\boldsymbol{w}^*$-ot felírhatjuk csupán a szupport vektorok lineáris kombinációjaként:

$$\boldsymbol{w}^* = \sum_{i \in SV} y_i \alpha_i^* \, \boldsymbol{x_i} \,,$$

ahol SV jelöli a szupport vektorok sorszámainak halmazát. Egy $\boldsymbol{x}$ tesztbemenetre adott jóslat tehát a következőképpen adódik:

$$f(\boldsymbol{x}) = \sum_{i \in SV} y_i \alpha_i^* \, \boldsymbol{x_i} \cdot \boldsymbol{x} + b^*$$

3.6.3.2. Szoft margójú lineáris szeparálás

Ha a tanítópontok lineárisan nem szeparálhatók hiba nélkül, a célunk az, hogy minimális számú hibával válasszuk szét őket. Vapnik és Cortes a probléma megoldására az úgynevezett gyengítő változók bevezetését javasolja. Mivel az $y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \geq 1$ feltétel nem állhat fenn minden pontra, ehelyett a következő feltételeket fogalmazzuk meg:

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \geq 1 - \xi_i \,, \qquad \xi_i \geq 0 \,, \qquad i = 1 \ldots T$$

Ha $\xi_i = 0$, akkor $\boldsymbol{x_i}$ a hipersík megfelelő oldalán, a biztonsági sávon kívül helyezkedik el, $0 < \xi_i < 1$ esetén a jó oldalon van, de a biztonsági sávon belül, ha pedig $\xi_i > 1$, akkor a pont a hipersík rossz oldalára esik. A minimalizálandó kifejezésbe a gyengítő változók összegét is belevesszük, ezáltal érjük el, hogy minél kevesebb hiba legyen:

$$\mathcal{L}(\boldsymbol{w}, \xi) = \frac{1}{2} \boldsymbol{w^2} + C \left(\sum_{i=1}^{t} \xi_i \right)$$

Az optimális $(\boldsymbol{w}^*, b^*)$ hipersík ismét kvadratikus programozással határozható meg. A duális felírásban csak a feltételek változnak meg:

$$maximalizálandó: \quad L(\boldsymbol{\alpha}) = \sum_{i=1}^{T} \alpha_i - \frac{1}{2} \sum_{i=1}^{T} \sum_{j=1}^{T} y_i y_j \alpha_i \alpha_j \, \boldsymbol{x_i} \cdot \boldsymbol{x_j} \,,$$

$$ahol: \quad \sum_{i=1}^{T} y_i \alpha_i = 0 \,, \qquad 0 \leq \alpha_i \leq C \,, \;\; i = 1 \ldots T$$

Most a szupport vektorok azok a tanítópontok lesznek, melyekre $y_i(\boldsymbol{w}^* \cdot \boldsymbol{x_i} + b^*) = 1 - \xi_i$. Igaz marad az a tény, hogy $\boldsymbol{w}^*$ felírható a szupport vektorok lineáris kombinációjaként. Így egy $\boldsymbol{x}$ bemenetre vonatkozó jóslat ugyanúgy számítható ki, mint az előző pontban:

$$f(\boldsymbol{x}) = \sum_{i \in SV} y_i \alpha_i^* \, \boldsymbol{x_i} \cdot \boldsymbol{x} + b^*$$

Az $\mathcal{L}(\boldsymbol{w}, \xi)$ minimalizálandó kifejezésben szereplő C konstanst szokás hiperparaméternek is nevezni. Azt határozza meg, hogy milyen arányban vesszük figyelembe a $\boldsymbol{w}$ súlyvektor hosszát, illetve a tanítópontokra számított osztályozási hibát. Megválasztása nehéz feladat, például cross-validation segítségével történhet.

3.6.3.3. Nemlineáris szeparálás

Amint azt előrevetítettük a 3.6.2 pontban, a fent bemutatott elveket fel tudjuk használni egy nemlineáris osztályozó megalkotására. Az adatokat nem a bemeneti térben próbáljuk meg szeparálni, hanem egy nemlineáris $\boldsymbol{\varphi}$ leképezéssel kapott jellemző térben. Viszont jelen esetben a jellemző térben csak a tanítópontokkal vett skaláris szorzatokat számolnánk, így alkalmazhatjuk a kernel trükköt, vagyis nem végezzük el a $\boldsymbol{\varphi}$ leképezést explicit módon, hanem a jellemzőtérbeli skaláris szorzást a bemeneti vektorokon számolt K kernel függvénnyel helyettesítjük.

Ebben az esetben tehát a duális kvadratikus programozási feladat:

$$maximalizálandó: \quad L(\boldsymbol{\alpha}) = \sum_{i=1}^{T} \alpha_i - \frac{1}{2} \sum_{i=1}^{T} \sum_{j=1}^{T} y_i y_j \alpha_i \alpha_j \, K(\boldsymbol{x_i}, \boldsymbol{x_j}) \,,$$

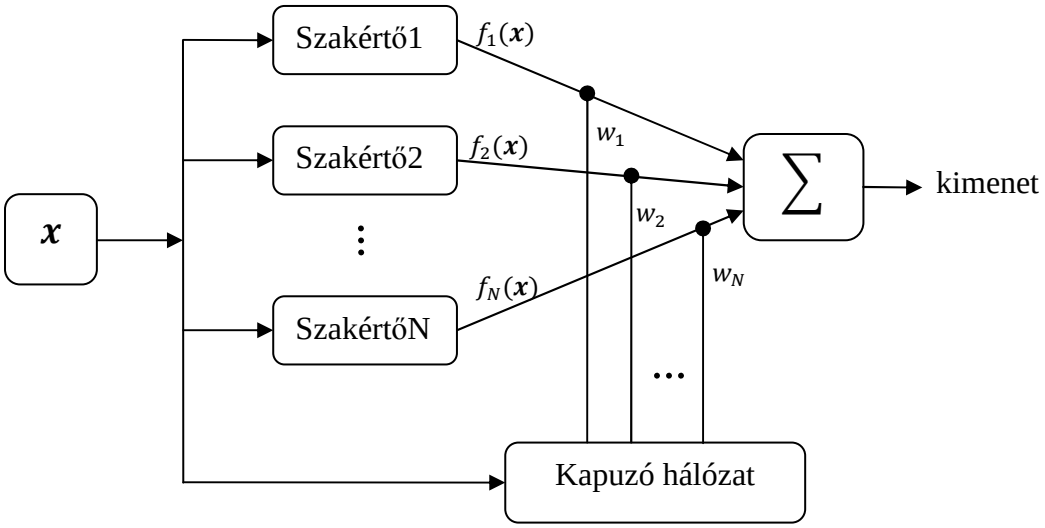
$$ahol: \quad \sum_{i=1}^{T} y_i \alpha_i = 0 \,, \qquad 0 \leq \alpha_i \leq C \,, \;\; i = 1 \ldots T$$

Amikor egy $\boldsymbol{x}$ tesztpontra a jóslatot számítjuk, a skaláris szorzás helyett ismét a kernel függvényt használjuk:

$$f(\boldsymbol{x}) = \sum_{i \in SV} y_i \alpha_i^* \, K(\boldsymbol{x_i}, \boldsymbol{x}) + b^*$$

3.6.4. A Mixture Of Experts (MOE) architektúra

Amikor egy osztályozási feladatot gépi tanulással oldunk meg, több különböző osztályozót is betaníthatunk, ezek más-más eredményeket adnak. Bevett módszer a különböző osztályozók, más néven szakértők kimeneteinek kombinálása, ezzel megpróbálva kihasználni mindegyik előnyét. Az így kapott rendszert Mixture Of Experts (MOE) architektúrának hívjuk, és az alábbi ábra szemlélteti a felépítését.



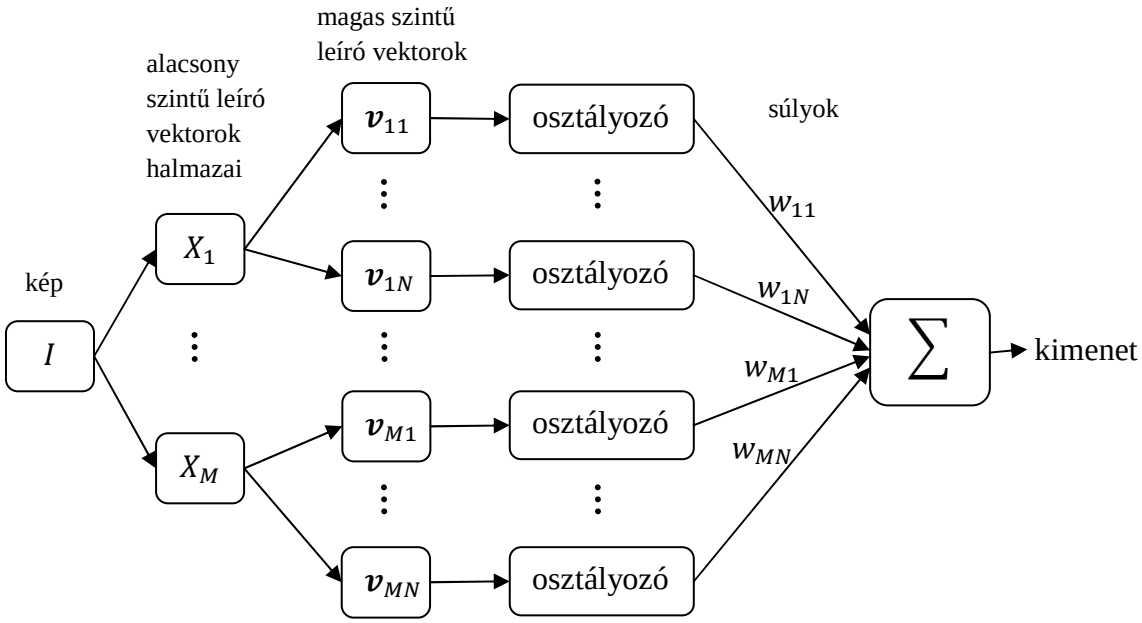
3.13. ábra: A MOE architektúra ([24] szerint)

A fenti ábrán az látható, hogy a szakértőkhöz tartozó súlyokat a bemenet függvényében egy kapuzó hálózattal számítjuk. Lehet alkalmazni azonban a bemenettől független, konstans súlyokat is, ez egy kicsit egyszerűsített változata a fenti általános megoldásnak.

4. Aggregált kernel alapú módszer

4.1. Alapelv

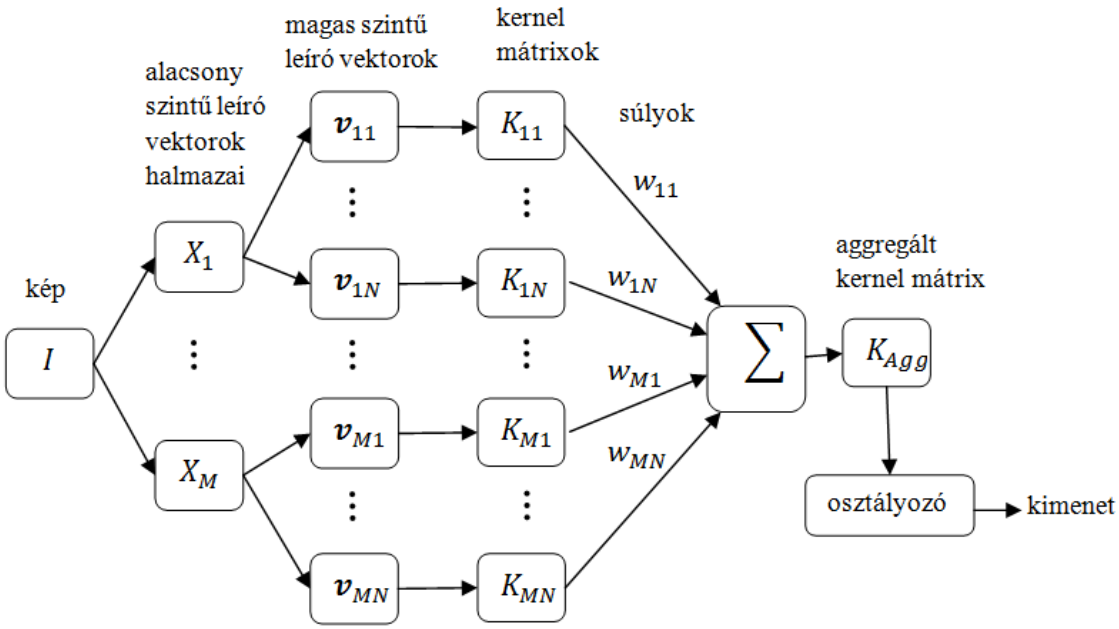
Képi klasszifikáció esetén több féle alacsony szintű leírót is számolhatunk, majd ezekből megint több féle magas szintű leíró vektort képezhetünk egy kép reprezentálása. Mindegyik típusú leíró vektor szerint betaníthatunk egy-egy osztályozót, így különböző szakértőket kapunk a feladat megoldására. Ezek között nyilván lesz jobb és rosszabb is, de igazán jó eredményeket akkor tudunk elérni, ha a szakértők tudását kombináljuk, mivel ez egyes leíró vektorok más-más információkat tartalmazhatnak egy képről. Tehát kézenfekvően adódik egy MOE rendszer, amit a 4.1. ábra mutat.



4.1. ábra: MOE rendszer képi klasszifikációra

Mivel a különböző képi leíró vektorokból származó információkat későn, csak a tanítás után egyesítjük, ezért ennek a módszernek a neve Late Fusion. A fenti ábra nem hangsúlyozza ki ezt, de a módszer ereje abban rejlik, hogy a kimeneti súlyokat kategóriánként határozzuk meg. Tehát bizonyos képi jellemzők dominánsabbak lehetnek egy személy felismerésében, mint egy repülőgép észlelésében, ehhez tudjuk megfelelően testre szabni a rendszert a kimeneti súlyokkal. A súlyok meghatározásáról később, a 4.1.4 pontban írok részletesen.

Mi az előzőekben leírt MOE rendszer mellett egy más megközelítést is javaslunk a magas szintű leírók kombinálására. A különböző képi leírókból származó információkat még a tanítás előtt, a kernel számítás során kombináljuk, majd az így létrejött, aggregált kernel szerint végezzük a tanítást. Ennek a módszernek az alapelvét mutatom be a következőkben. Az aggregált kernel alapú rendszer felépítését mutatja az alábbi ábra, egyben látható a különbség is az előbbi MOE struktúrához képest.



4.2. ábra: Aggregált kernel alapú rendszer

4.1.1. Az előre számolt kernel mátrix

Az osztályozás során egy-egy mintának a tanítópontokkal számolt kernel függvényét használjuk. Ezeket kiszámíthatjuk előre is, így létrejön egy kernel mátrix. Legyenek a tanítóhalmaz pontjai $\{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_T}\}$, a teszthalmaz pontjai pedig $\{\boldsymbol{z_1}, \boldsymbol{z_2}, \dots, \boldsymbol{z_S}\}$. A tanítóhalmazon számolt kernel mátrix, ami alapján a tanítást végezzük:

$$\begin{bmatrix} K(\boldsymbol{x_1}, \boldsymbol{x_1}) & K(\boldsymbol{x_1}, \boldsymbol{x_2}) & \dots & K(\boldsymbol{x_1}, \boldsymbol{x_T}) \\ K(\boldsymbol{x_2}, \boldsymbol{x_1}) & K(\boldsymbol{x_2}, \boldsymbol{x_2}) & \dots & K(\boldsymbol{x_2}, \boldsymbol{x_T}) \\ \vdots & & & \vdots \\ K(\boldsymbol{x_T}, \boldsymbol{x_1}) & K(\boldsymbol{x_T}, \boldsymbol{x_2}) & \dots & K(\boldsymbol{x_T}, \boldsymbol{x_T}) \end{bmatrix}$$

Hasonlóképpen a teszthalmazra is kiszámíthatunk egy kernel mátrixot, ezt a kiértékeléskor használjuk fel:

$$\begin{bmatrix} K(\boldsymbol{z_1}, \boldsymbol{x_1}) & K(\boldsymbol{z_1}, \boldsymbol{x_2}) & \dots & K(\boldsymbol{z_1}, \boldsymbol{x_T}) \\ K(\boldsymbol{z_2}, \boldsymbol{x_1}) & K(\boldsymbol{z_2}, \boldsymbol{x_2}) & \dots & K(\boldsymbol{z_2}, \boldsymbol{x_T}) \\ \vdots & & & \vdots \\ K(\boldsymbol{z_T}, \boldsymbol{x_1}) & K(\boldsymbol{z_T}, \boldsymbol{x_2}) & \dots & K(\boldsymbol{z_T}, \boldsymbol{x_T}) \end{bmatrix}$$

A fenti mátrixokban a K függvény bármilyen kernel függvény lehet, mi távolságfüggvényeket alkalmazunk, egész pontosan az alábbiakat:

$L1$	$K(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{D} \lvert x_i - y_i \rvert$
$L2$	$K(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{D} (x_i - y_i)^2}$
Cos	$K(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{\boldsymbol{x}\cdot\boldsymbol{y}}{\lVert\boldsymbol{x}\rVert\lVert\boldsymbol{y}\rVert}$
χ^2	$K(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{D} \frac{(x_i - y_i)^2}{\frac{1}{2}(x_i + y_i)}$

4.1. táblázat: Az alkalmazott távolságfüggvények

Ezáltal a kernel mátrix szemléletes jelentést is nyer, valójában azt tartalmazza, hogy egy-egy kép mennyire hasonló a tanítóhalmaz egyes képeihez. Ez adott esetben lehetséges, hogy jobb jellemző, mint maga a képet leíró vektor.

Mivel a különböző magas szintű leíró vektorokból kiszámolt kernel mátrixok méretei megegyeznek, és ugyanolyan típusú információkat tartalmaznak, így közvetlenül adódik a lehetőség különböző, előre kiszámolt kernel mátrixok kombinálására. A kombinált kernel mátrix alapján pedig már egy lineáris klasszifikátorral hatékonyan lehet osztályozni a mintákat, és ennek a tanítása sokkal gyorsabb, mint egy nemlineáris klasszifikátoré.

4.1.2. SVM előre számolt kernellel

Ha az előre számolt kernel mátrix sorait tekintjük egy lineáris SVM bemeneti vektorainak, akkor a 3.6.3.3 pontban leírtakhoz képest kissé megváltozik a tanítás során optimalizált kifejezés:

$$maximalizálandó\text{:}\quad L(\boldsymbol{\alpha}) = \sum_{i=1}^{T} \alpha_i - \frac{1}{2}\sum_{i=1}^{T}\sum_{j=1}^{T} y_i y_j \alpha_i \alpha_j \sum_{t=1}^{T} K(\boldsymbol{x_i},\boldsymbol{x_t}) K\left(\boldsymbol{x_j},\boldsymbol{x_t}\right),$$

$$ahol\text{:}\quad \sum_{i=1}^{T} y_i \alpha_i = 0\,, \qquad 0 \le \alpha_i \le C\,,\ \ i = 1 \ldots T$$

Valamint $\boldsymbol{x}$ tesztpontra a jóslat számítása a következőnek felel meg:

$$f(\boldsymbol{x}) = \sum_{i \in SV} y_i \alpha_i^* \sum_{t=1}^{T} K(\boldsymbol{x_i},\boldsymbol{x_t}) K(\boldsymbol{x},\boldsymbol{x_t}) + b^*$$

Tekinthetünk erre úgy is, hogy a távolság kernelekből egy új kernel függvényt származtattunk (mivel kernel függvények összege és szorzata is kielégíti a kernel függvény kritériumait):

$$K'(\boldsymbol{y}, \boldsymbol{z}) = \sum_{t=1}^{T} K(\boldsymbol{y}, \boldsymbol{x_t}) K(\boldsymbol{z}, \boldsymbol{x_t})$$

4.1.3. Kernel mátrixok kombinálása

A különböző magas szintű leíró vektorokból kiszámított kernel mátrixokat súlyozva aggregáljuk egyetlen kernel mátrixba, vagyis egy lineáris kombinációjukat vesszük. A súlyokat a klasszifikációs feladat minden kategóriájára külön határozzuk meg, hiszen különböző típusú objektumok esetén más-más leíró vektorok jellemzik jobban a képet (például ha növényt keresünk a képen, akkor valószínűleg a szín alapú leíró dominánsabb lehet).

Legyenek adottak a $K_1, K_2, \dots, K_N$ különböző kernelek, amelyek lineáris kombinációját egy lineáris SVM-mel osztályozzuk. Minden c kategória esetén keressük a kerneleknek egy olyan lineáris kombinációját ($\sum_{n=1}^{N} \beta_n^c K_n$), amelyre az SVM osztályozás a lehető legjobb eredményt adja. Az előző pontban felírt optimalizálási feladatot a következőképpen módosíthatjuk:

$$maximalizálandó\text{:}$$
$$L(\boldsymbol{\alpha^c}, \boldsymbol{\beta^c}) = \sum_{i=1}^{T} \alpha_i^c - \frac{1}{2} \sum_{i=1}^{T} \sum_{j=1}^{T} y_i^c y_j^c \alpha_i^c \alpha_j^c \sum_{n=1}^{N} \beta_n^c \sum_{t=1}^{T} K_n(\boldsymbol{x_i}, \boldsymbol{x_t}) K_n(\boldsymbol{x_j}, \boldsymbol{x_t}) ,$$

$$ahol\text{:} \quad \sum_{i=1}^{T} y_i^c \alpha_i^c = 0 , \qquad 0 \le \alpha_i^c \le C , \quad i = 1 \dots T$$

$$és \quad \sum_{n=1}^{N} \beta_n^c = 1$$

Ennek a feladatnak az optimális megoldását nem adjuk meg, csak közelíteni próbáljuk, logikus módszerrel. Egyrészt kézenfekvő a kerneleket egyszerűen átlagolni (ilyenkor $\forall\, \beta_n = \frac{1}{N}$). Ez a megoldás nem kielégítő abból a szempontból, hogy szeretnénk kategóriánként különböző módon súlyozni a kerneleket, ezzel nagyobb részben figyelembe venni a kategóriát jobban jellemző képi leírókat. Egy Late Fusion rendszerben meghatározott súlyok viszont pont azt az információt hordozzák, hogy az egyes képi leírók mennyire tudják jól jellemezni az adott kategóriát. Tehát egy közelítő megoldásként felhasználhatjuk a Late Fusionből származó súlyokat a kernel mátrixok kombinálására.

A több kernel alapján tanítás ötlete nem új, ez Multiple Kernel Learning néven ismert probléma. Publikáltak már több algoritmust a megoldására, mi alapvetően Rakotomamonjy és társai megoldásából indultunk ki [25]. Megjegyezzük, hogy ez a megoldás számunkra túl sok SVM tanítást igényelne, gyorsabb megoldásra van szükségünk, ezért a következő három féle módszert fogjuk alkalmazni és kiértékelni:

- Late Fusion
- Átlagolt kernel
- Súlyozott kernel (a Late Fusion súlyaival számolva)

Late Fusion esetén is az előre számolt kernelek segítségével végezzük az osztályozást, így ez annyiban különbözik a súlyozott kerneltől, hogy először kernelenként külön-külön tanítunk egy osztályozót, és ezek kimenetét kombináljuk. Az aggregált kernel esetén kategóriánként más-más az osztályozók bemenete, míg az átlagolt kernellel az osztályozók bemenetére ugyanazt az információt adjuk, csupán az osztályozók vannak testre szabva az egyes kategóriákhoz. Megjegyezzük, hogy az átlagolt kernel alapú módszer (konstans szorzótól eltekintve) ekvivalens azzal, hogy a különböző magas szintű leírókat konkatenáljuk, és utána számítjuk a távolságfüggvényt, mely utóbbi egy gyakran alkalmazott módszer.

4.1.4. A súlyok meghatározása

Célunk, hogy egy Late Fusion rendszerben a különböző osztályozók kimenetéhez tartozó súlyokat meghatározzuk. Ehhez a tanítóhalmaz mellett szükségünk van egy validáló halmazra (ami nem egyezik meg a teszthalmazzal). Ezt elérhetjük úgy is, hogy az eredeti tanítóhalmazt kettébontjuk. Először a tanítóhalmazon betanítjuk az N darab különböző kernel mátrix alapján az N darab osztályozót. Majd meghatározzuk a validációs halmaz minden pontjára az osztályozók által adott N darab jóslatot. Ezután olyan súlyokat keresünk az egyes osztályozók kimenetéhez, amivel a validációs halmazon javul az osztályozást minősítő mutató, esetünkben az Average Precision.

A $\boldsymbol{w} = (w_1, w_2, \dots, w_N)$ súlyokat egy egyszerű iteratív algoritmussal keressük egy (a, b) intervallumban. Jelölje $AP(\boldsymbol{w})$ a $\boldsymbol{w}$ súlyvektor esetén kialakuló Average Precisiont. Az algoritmus váza:

$$\boldsymbol{w} \leftarrow (1, 1, \dots, 1)$$
$$Ciklus\ 1 - től\ az\ iterációk\ számáig$$
$$Ciklus\ i = 1 - től\ N - ig$$
$$w_i = \underset{w_i \in (a,b)}{\operatorname{argmax}}\, AP(\boldsymbol{w})$$

Természetesen az (a, b) intervallum elemeit csak egy bizonyos lépésközzel tudjuk megvizsgálni. Ezt a lépésközt az iterációk folyamán csökkentjük, miközben a keresési intervallumokat is szűkítjük.

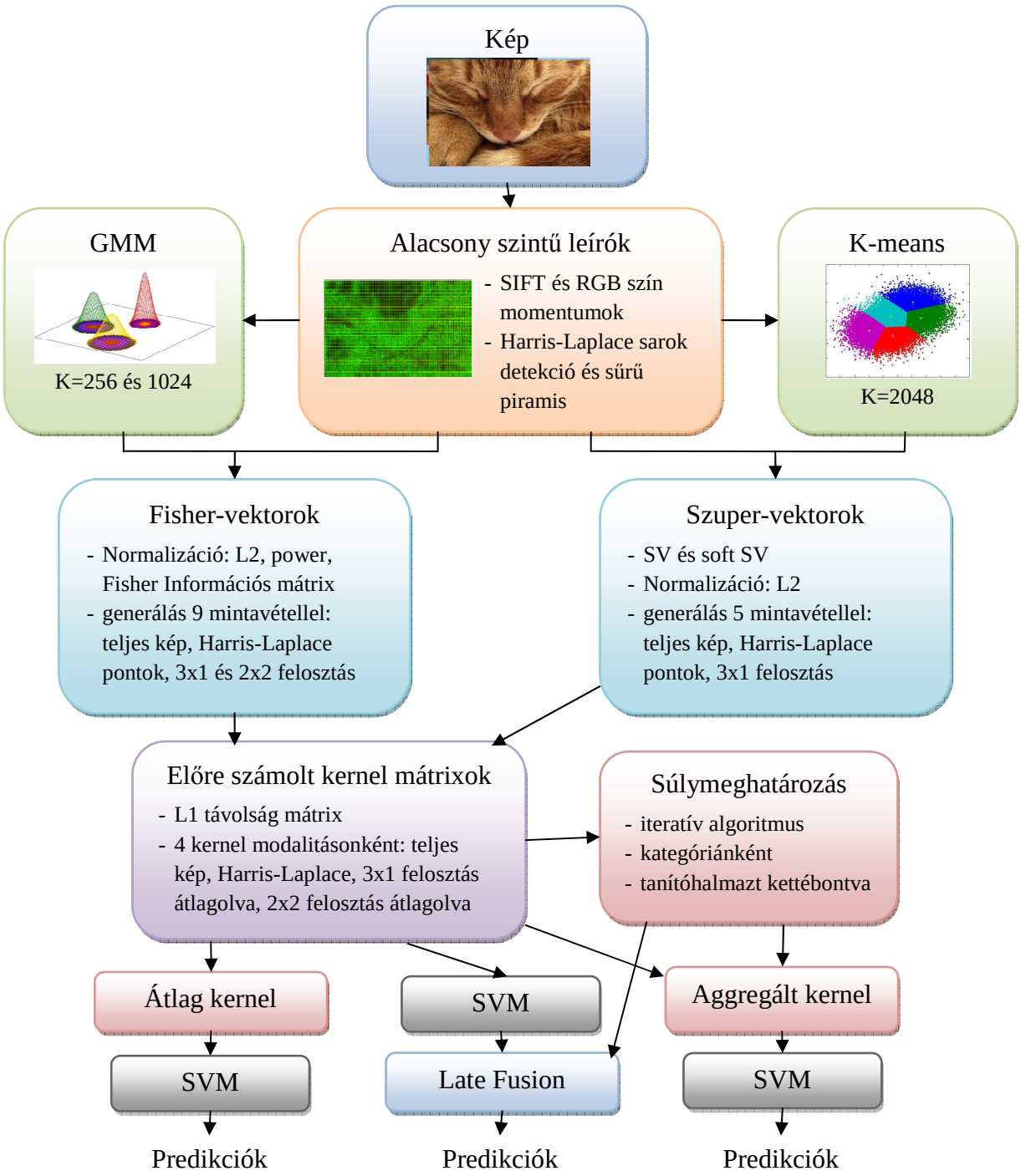
Könnyen előfordulhat a súlyok túltanulása, mely esetben a teszthalmazon végül rosszabb eredményt kapunk, mint az átlagolt kernellel. Bizonyosan jobb eredményt adna egy SVM alapú iteratív MKL [25], mely a teljes tanulóhalmaz felhasználásával lényegesen több kép (esetünkben 2500 helyett 5717) segítségével dönti el a súlyozást, de mivel mi nem ritka kernel mátrixokkal dolgozunk, ezért számunkra a számításigénye nem elfogadható. Így kénytelenek voltunk túltanulás elkerülése érdekében kis intervallumokat használni, de a későbbiekben mindenképpen szeretnénk kifinomultabb súlymeghatározást is alkalmazni.

4.2. A megvalósított rendszer

Ebben a pontban bemutatom, hogy a 3. fejezetben részletesen tárgyalt, korszerű módszerek közül melyeket és hogyan használtunk fel arra, hogy a képi klasszifikációs rendszerünket létrehozzuk. Kiemelem, hogy ezen belül mi volt az én munkám, és milyen céllal.

4.2.1. Áttekintés

Az alábbi, áttekintő ábra szemlélteti a rendszerünk működését. Az ábra egyes elemeit a 4.2.3 pontban részletezem.



4.2.2. A vizsgálat célja

A fenti rendszert a konzulensemmel együtt fejlesztjük. Az én munkám a K-means klasszifikáció és a Szuper-vektor leíró implementálása, beillesztése a rendszerbe az aggregált kernel alapú módszer segítségével, és a rendszer kiértékelése volt. A Fisher-vektorok alkalmazásával készített rendszer már korábban remekül szerepelt az ImageCLEF Photo Annotation 2011 versenyen [26]. A munkám célja ennek a megoldásnak a javítása volt, és az 5. fejezetben megmutatom, hogy a Szuper-vektor használata a kernel aggregáló módszerünkkel együtt javítja a teljesítményt az eredeti, csak Fisher-vektor alapú rendszerhez képest.

4.2.3. A felhasznált képi leírók és kernelek

4.2.3.1. Alacsony szintű leírók

A képekre kiszámoltunk mind szín alapú, mind textúra alapú leíró vektorokat. A szín alapúak közül RGB szín momentumokat (3.2.1.1) számoltunk, az első és második momentumokat, tehát az egyes színcsatornák átlagát és szórását vettük, 4×4 pixel méretű cellákban. Így egy cellát 6 számmal jellemzünk. A kép egy pontja körül 4×4 darab cellában számoljuk a szín momentumokat, így egy $16 \cdot 6 = 96$ dimenziós vektorral jellemezzük egy pont lokális környezetét.

Korábban a Pascal VOC 2007 [27] adathalmazon leteszteltük a szürkeskálás alacsony szintű leírókra számolt Fisher-vektorokat:

Alacsony szintű leíró	Klaszterezés	Magas szintű leíró	Kernel	Osztályozó	MAP
LBP	GMM, K=256	Fisher-vektor	L1	SVM	**0,3912**
HOG	GMM, K=256	Fisher-vektor	L1	SVM	**0,4215**
SIFT	GMM, K=256	Fisher-vektor	L1	SVM	**0,4611**

4.2. táblázat: A textúra alapú alacsony szintű leírók összehasonlítása (a teljes képre illesztett sűrű rácson számítva)

A jelenlegi rendszerben textúra alapú leírók közül csak a SIFT leírót (3.2.2.2) számoljuk. A paramétereket a Lowe által javasolt módon állítottuk be, vagyis az orientáció-hisztogramot 8 irány szerint készítettük, 4×4 pixel méretű cellákban, és egy pont körül 4×4 cellából álló blokkban számoltuk ezeket. Így $16 \cdot 8 = 128$ dimenziós vektorokat kapunk. Ezekre a vektorokra egy PCA transzformációt végzünk, ezzel egy 80 dimenziós térbe képezzük le őket.

Az alacsony szintű leírókat egyrészt kiszámoltuk egy képre illesztett rács pontjaiban, amelynek lépésközét 4 pixel nagyságúra állítottuk, másrészt pedig Harris-Laplace pontdetektálással (3.3.1) meghatározott kulcspontokban. Így képenként átlagosan 15000 alacsony szintű leíró vektort számítottunk.

4.2.3.2. Magas szintű leírók

Két féle fejlett, magas szintű szemantikus leírót számítunk ki az alacsony szintű leírókból, a korábban tárgyalt Fisher-vektort (3.5.2) és Szuper-vektort (3.5.3**Hiba! A hivatkozási forrás nem található.**). Mindkét leíró már bizonyított, eredményesen szerepeltek több, képi klasszifikációs versenyre készített rendszerben is [20], [26]. A 4.3. táblázatban összehasonlító eredményeket láthatunk a K-means Bag of Words (3.5.1) megoldással, a saját vizsgálataink alapján, a Pascal VOC 2007 adathalmazon. A K-means alapú Bag of Words egy nagyon sokak által alkalmazott, sztenderdnek tekinthető megoldás [29], [30]. Vizsgálatainkhoz RGB szín alapú leírókat használtunk, amelyeket a teljes képre illesztett sűrű rácson számítottunk ki, ez azt jelenti, hogy átlagosan 5000 alacsony szintű leíró vektort használtunk képenként. Ez a táblázat is mutatja, hogy a Fisher- és Szuper-vektoros reprezentációk jóval erősebbek egy alap BoW megoldásnál, amennyiben csak egyfajta mintavételezést, ebben az esetben a sűrű gridet alkalmazzuk.

Alacsony szintű leíró	Klaszterezés	Magas szintű leíró	Kernel	Osztályozó	MAP
RGB szín	K-means, K=2048	Bag of Words	L1	SVM	**0,3477**
RGB szín	K-means, K=2048	Bag of Words	χ^2	SVM	**0,3517**
RGB szín	K-means, K=2048	Szuper-vektor	L1	SVM	**0,4279**
RGB szín	GMM, K=256	Fisher-vektor	L1	SVM	**0,4501**

4.3. táblázat: A magas szintű leírók összehasonlítása a VOC 2007 adathalmazon (a teljes képre illesztett sűrű rácson számított alacsony szintű leírókkal)

Fisher-vektor esetén a vizuális szótárat GMM klaszterezéssel (3.4.2.3) alakítjuk ki. Mindkét típusú alacsony szintű leíró vektorra (szín momentumok és SIFT PCA-val) készítettünk két GMM modellt, melyekben a Gauss-függvények száma 256 és 1024. A GMM tanítására a bemutatott EM algoritmust használjuk. A Fisher-vektorokon három féle normalizációt is végrehajtunk, elsőként a Fisher információs mátrixszal normáljuk, majd végrehajtjuk az úgynevezett power-normalizációt, végül egy L2 normálással a vektor hosszát egységnyire állítjuk be.

Perronnin és társai [19] a Fisher-vektor implementációjukban a nagy számításigény miatt csak ritka Fisher-vektorokat számítanak, a GMM klaszterei közül csak a legnagyobb valószínűségű néhány klasztert veszik figyelembe. Másrészt Perronninék magát a GMM-et egy hierarchikus, Maximum-a-Posteriori módszerrel tanítják, és alkalmazzák az előbbi egyszerűsítést. Ezzel szemben a mi algoritmusunk, miután GPGPU architektúrára implementált [27], nem hierarchikus EM-et használ közelítés nélkül. Ezen kívül a Fisher-vektorokat is a teljes GMM modell szerint számítjuk.

Szuper-vektor esetén K-means klaszterezést használunk. A Pascal VOC 2007 adathalmazon leteszteltem kisebb (256) és nagyobb (2048) klaszterszámmal is a Szuper-vektorokat. Ahogy várható volt, a nagyobb klaszterszám jobb közelítését adja a térnek, ezért javít főként a soft Szuper-vektor működésén, ezt mutatja a 4.4. táblázat is. A soft Szuper-vektor esetén az eredeti cikkben [20] javasolt 20 legközelebbi klaszterközéppontot vettem figyelembe, a

klaszterközéppontoktól vett távolsággal fordítottan arányos súlyokkal. A Szuper-vektor s paraméterének változtatását kipróbáltam a kisebb adatokon (256-os klaszterszám mellett), de úgy tapasztaltam, hogy ez nem sokat számít, a cikkben is javasolt $s = 0{,}1$ beállításnál maradtunk. A kiszámított Szuper-vektorokon még végrehajtunk egy L2-normalizálást, hogy a kernel számításnál ne legyenek nagy numerikus eltérések a különböző képekre, illetve különböző térbeli felosztásban kiszámolt Szuper-vektorok között.

Alacsony szintű leíró	Klaszterezés	Magas szintű leíró	Kernel	Osztályozó	MAP
RGB szín	K-means, K=256	SV s=0,1	nincs	SVM	**0,3156**
RGB szín	K-means, K=256	SV s=0,1	L1	SVM	**0,4081**
RGB szín	K-means, K=256	SV s=0	L1	SVM	**0,4077**
RGB szín	K-means, K=256	SV s=10	L1	SVM	**0,4087**
RGB szín	K-means, K=256	soft SV s=0,1	L1	SVM	**0,3214**
RGB szín	K-means, K=2048	SV s=0,1	L1	SVM	**0,4279**
RGB szín	K-means, K=2048	soft SV s=0,1	L1	SVM	**0,3761**
SIFT	K-means, K=256	SV s=0,1	nincs	SVM	**0,3905**
SIFT	K-means, K=256	SV s=0,1	L1	SVM	**0,4688**
SIFT	K-means, K=256	soft SV s=0,1	L1	SVM	**0,4112**
SIFT	K-means, K=2048	SV s=0,1	L1	SVM	**0,4661**
SIFT	K-means, K=2048	soft SV s=0,1	L1	SVM	**0,4752**

4.4. táblázat: Szuper-vektoros eredmények a VOC 2007 adathalmazon (spatial pooling nélkül, teljes képre számítva)

A fenti táblázat alapján láthatjuk, hogy a 2048-as klaszterszám mellett SIFT esetén a soft Szuper-vektor, az RGB szín alapú leíróra pedig a Szuper-vektor ad jobb eredményt. Azt is megjegyezzük, hogy az előre számolt kernel alapú osztályozás sokkal jobb eredményeket ad, mintha kernel számítás nélkül, a nyers Szuper-vektorokat próbálnánk lineárisan szeparálni.

Mind a Fisher-vektorokat, mind a Szuper-vektorokat kiszámoljuk a képre illesztett rácson számolt alacsony szintű leíró vektorok halmazán és a Harris-Laplace pontokban számolt alacsony szintű leírók halmazán is. Ezek együtteséből számítunk egy magas szintű leírót, amit a továbbiakban a teljes képre számolt magas szintű leírónak hívunk, valamint külön a Harris-Laplace detektált pontokon is számítunk egy magas szintű leírót. Emellett Spatial Poolingot is végzünk, a kép 3×1-es és 2×2-es felosztásában az egyes részekre külön-külön számolunk magas szintű leíró vektorokat. A 2×2-es felosztást csak a 256-os klaszterszámú Fisher-vektorok esetén alkalmazzuk a számításigények miatt. A 3×1-es felosztást részesítjük előnyben, mert jobb eredményeket ad, és ezt intuitíven magyarázni is lehet: a 2×2-es felosztás nagy eséllyel pont szétdarabolja a kép fókuszában található objektumokat.

Az alábbi táblázatban összefoglaltam, hogy milyen leírókat használunk egy kép reprezentálására:

Magas szintű leíró típusa	Alacsony szintű leíró	Térbeli felosztás	Leíró vektorok darabszáma
Fisher-vektor, K=256	RGB szín momentumok	teljes kép	1
		Harris-Laplace pontok	1
		3 × 1-es felosztás	3
		2 × 2-es felosztás	4
	SIFT	teljes kép	1
		Harris-Laplace pontok	1
		3 × 1-es felosztás	3
		2 × 2-es felosztás	4
Fisher-vektor, K=1024	RGB szín momentumok	teljes kép	1
		3 × 1-es felosztás	3
	SIFT	teljes kép	1
		3 × 1-es felosztás	3
Szuper-vektor, K=2048	RGB szín momentumok	teljes kép	1
		Harris-Laplace pontok	1
		3 × 1-es felosztás	3
soft Szuper-vektor, K=2048	SIFT	teljes kép	1
		Harris-Laplace pontok	1
		3 × 1-es felosztás	3
Összesen:			36

4.5. táblázat: A felhasznált képi leírók

4.2.3.3. Kernelek

A kernel számítást távolságfüggvények segítségével végezzük, amelyeket a 4.1. táblázatban megadtunk. Ezek közül Fisher-vektorokra és Szuper-vektorokra is tapasztalataink szerint az L1 távolság kernellel lehet a legjobb eredményeket elérni. A fenti táblázatban összefoglalt leíró vektorokra először külön-külön számítunk kernel mátrixokat, de a 3×1-es és a 2×2-es felosztásból számolt 3, illetve 4 kernel mátrixot rögtön átlagoljuk. Ez a lépés ekvivalens azzal, mintha a kép egyes részeire kiszámolt leírókat konkatenáltuk volna, és utána számolunk távolságokat (konstans szorzótól eltekintve). Így a fenti táblázat minden sorához fog tartozni egy kernel mátrix, vagyis összesen 18 darab kernel mátrixunk lesz.

4.2.3.4. Tanítás

Az előre számolt kernel mátrixokra alkalmazott lineáris SVM-et használtunk az osztályozási feladatra, amint erről már a 4.1.2 pontban volt szó. A tanítást és kiértékelést elvégeztem külön kernelenként is, de az elsődleges célom ezeknek a kombinálása volt. Három féle módszert próbáltam ki erre:

- A különböző kerneleken tanított SVM-ek kimenetét Late Fusion módszerrel kombináltam, kategóriánként különböző súlyozással. A súlyokat a 4.1.4 pontban leírt iteratív algoritmussal határoztam meg.
- A normalizált kernel mátrixokat átlagoltam egy kernel mátrixba, és ez alapján végeztem a tanítást, hasonlóan a kiértékelést
- A Late Fusion során meghatározott súlyokat felhasználva az eredeti kernel mátrixokat aggregáltam egy kernel mátrixba, (kategóriánként eltérő súlyozással) majd ezekkel tanítottam az SVM-eket.

A kernel mátrixok kombinálása során ügyelni kell arra is, hogy a különböző leírókból származó kernelek eltérő nagyságrendűek lehetnek az eltérő dimenziószámok, normalizációk miatt. Ezért mielőtt a mátrixok lineáris kombinációját vesszük, minden mátrixot leosztunk a megfelelő magas szintű leíró dimenziójának négyzetgyökével.

Az alábbi táblázatban összefoglalom, hogy a három különböző kombinálási módszer esetén hány SVM tanítást végzünk, külön csak a tanítóhalmazon, a súlyok meghatározása során, és a tanító- és validáló halmazon a végső megoldás elkészítésekor.

Módszer	**SVM tanítások száma (kategóriánként)**		
	Tanítóhalmazon	Tanító- és validáló halmazon	Összesen
Late Fusion	18	18	36
Átlagolt kernel	0	1	1
Aggregált kernel	18	1	19

4.6. táblázat: Az SVM tanítások száma a három használt módszer esetén

Láthatjuk, hogy az átlagolt kernel igényli a legkevesebb SVM tanítását. Az aggregált kernel módszer a súlyok meghatározása miatt a tanítóhalmazon igényli a Late Fusion során elvégzett SVM tanításokat, de megjegyzendő, hogy ezek kisebb adatok. A nagy adaton jóval kevesebb SVM tanítást végzünk az aggregált kernel alapú módszerben, mint a Late Fusionben, így jelentősen kevesebb a számításigénye.

5. Implementáció és kiértékelés

5.1. A rendszer implementációja

A rendszer a hatékonysága érdekében a C++ nyelvet használtuk. A legtöbb számítás igénylő részeket, mint a GMM modellezés és Fisher-vektor számítás CUDA platformon, GPGPU architektúrára készítettük. Az SVM osztályozáshoz a libsvm [31] függvénykönyvtárat használtuk, a rendszer többi része saját fejlesztés.

Én implementáltam a K-means modell építést, egy K-meansre épülő Bag of Words leíró számítást, valamint a Szuper-vektor számítást. Emellett én végeztem a Szuper-vektorokra a kernelek kiszámítását, az SVM-ek tanítását, és a vizsgálatokat azon a téren, hogy a Szuper-vektor használatával a különböző kernel kombinálási módszerekben lehet-e javítani a korábbi rendszer teljesítményét. A vizsgálatok eredményeit az 5.3 pontban mutatom be.

A számításokhoz az MTA SZTAKI számítógépeit használtuk. Így elegendő számítási kapacitás és tárhely állt a rendelkezésünkre. Több gépet is használhattunk, ezek mindegyike 8 processzormaggal rendelkezett, így számításainkat tudtuk párhuzamosan végezni. Az alábbi táblázatban a rendszer szempontjából kritikus futási időket foglaltuk össze, az 5823 képet tartalmazó Pascal VOC 2011 validációs halmazon, azzal a céllal, hogy elemezzük, legalább mennyi időre van szükség egy új kép kiértékeléséhez. A futtatást egy 2,4 GHz órajelű Xeon E5620 számítógépen végeztük.

	RGB szín leíró	**SIFT leíró**
Alacsony szintű leíró vektorok kiszámítása	~250 perc	~1200 perc
Szuper-vektor számítás (a teljes képre)	~200 perc	~160 perc
Kernel számítás (5717 képből álló tanítóhalmaztól vett távolságok)	~350 perc	~250 perc
SVM jóslatok számítása (egy kategória esetén)	~20 perc	~20 perc
Összesen	~820 perc	~1630 perc

5.1. táblázat: Futási idők

A fenti táblázatban csak azokat a futási időket tüntettem fel, amelyek nem elhanyagolhatóak, és a jelenlegi implementációval tovább nem párhuzamosíthatóak. Egy képre a jelenlegi rendszerben könnyű párhuzamosítás mellett a kritikus futási idő körülbelül $\frac{1630\ perc}{5823} \approx 16{,}8\ másodperc$. Ennyi idő szükséges legalább egy kép kiértékelésére. Amennyiben egy olyan ipari alkalmazásban szeretnénk használni a rendszerünket, amiben ennél gyorsabb kiértékelésre van szükség, további párhuzamosítással, pl. GPGPU implementációval lehet

csökkenteni ezt az időt, de mivel alapvetően kutatási céllal dolgozunk, jelenleg nem ez az elsődleges szempont.

5.2. A kiértékeléshez használt adathalmaz

A kiértékeléshez a Pascal VOC 2011 verseny adathalmazát [1] használtam, mivel az idei verseny kapcsán erre az adathalmazra végeztük a képi leírók és kernel mátrixok kiszámítását, és ezen az adathalmazon kapott eredmények a legérdekesebbek most számunkra. A versenyen 20, objektumokra vonatkozó kategóriában kell osztályozni a képeket, ezeket már a 2.1 pontban említettem. Összesen 22524 képet tartalmaz az adatbázis, melyek a Flickr weboldalról származnak. Ezek közül 10994 kép alkotja a teszthalmazt, amire nyilvánosan nem elérhetők az annotációk, 11530 képből áll a tanító- és validáló halmaz. Ennek következtében a vizsgálataimhoz az 5823 képből álló, eredetileg validáló halmazt használtam tesztelésre. Az 5717 képből álló tanítóhalmazt kettébontottam egy 2500 képből álló tanítóhalmazra (a továbbiakban tanítóhalmaz1), és egy 3217 képből álló validáló halmazra (a továbbiakban tanítóhalmaz2), amit a Late Fusion súlyainak meghatározásához használtam.

Tehát először a 4.2.3.3 pontban leírt 18 kernel mátrixban a tanítóhalmaz képeitől vett távolságokat számítjuk ki a tanítóhalmaz1 és tanítóhalmaz2-re, és ez alapján végzünk SVM tanításokat. Meghatározzuk a tanítóhalmaz2-n a jóslatokat, és ezek felhasználásával kiszámítjuk a Late Fusion súlyait. Ezt követően két tanítóhalmazt együtt használjuk a tanításra (a kernel mátrixokban az 5717 képtől vett távolságok vannak), és a validáló halmazon értékeljük a rendszert.

5.3. Eredmények

5.3.1. A kiindulás: Fisher-vektorok

A rendszer még a Szuper-vektorok nélkül, csak Fisher-vektorok felhasználásával az idei versenyeken jól szerepelt. Ezt két táblázattal szemléltetem, ami az ImageCLEF Photo Annotation 2011 és a Pascal VOC 2011 versenyek eredményeit tartalmazza.

Az ImageCLEF Photo Annotation versenyen több modalitás felhasználásával lehet annotálni a képeket, a nyers képek mellett rendelkezésre állnak a képekhez jó esetben kapcsolódó emberek által készített képaláírások (Flickr tagek) [26]. A harmadik helyen végzett rendszer csak 256-os GMM alapú Fisher-vektorok alapú kernelekre és egy a képaláírásokon értelmezett Jensen-Shannon divergencia alapú kernelre épült. A legfelső sor egy újabb eredményt mutat ugyanezen az adathalmazon, ebben az átlagolt kernel és az aggregált kernel módszerek vannak kombinálva.

Helyezés	Fejlesztő csapat	Modalitások	MAP	AUC
	A dolgozatban tárgyalt kombináció 256-os GMM alapú Fisher vektorok használatával	**IMG+TXT**	**0,448274**	**0,837789**
1.	TUBFI - Technische Universität Berlin + Frauenhofer	IMG+TXT	0,443449	0,835753
2.	LIRIS - Universite de Lyon, CNRS, Ecole Centrale de Lyon	IMG+TXT	0,436968	0,836570
3.	**BPACAD - SZTAKI**	**IMG+TXT**	**0,436294**	**0,827747**
4.	ISIS - University of Amsterdam	IMG+TXT	0,432758	0,821462
5.	MLKD - Department of Informatics, Aristotle University of Thessaloniki	IMG+TXT	0,401642	0,817084
6.	CEALIST - Laboratoire d' Intégration des Systemes et des Technologies	IMG+TXT	0,383528	0,819797
7.	CAEN – University of Caen	IMG	0,382403	0,805420
8.	MRIM - Laboratoire d'Informatique de Grenoble	IMG+TXT	0,377179	0,809472
9.	IDMT – Fraunhofer	IMG+TXT	0,370975	0,751788
…				

5.2. táblázat: Eredmények az ImageCLEF Photo Annotation 2011 versenyen

A Pascal VOC 2011 Challenge eredményei még nem hivatalosak, csupán a résztvevőknek kiadott előzetes eredményeket mutatom be. Az eredményeket két oszlopba osztottam aszerint, hogy a megoldásban használják-e a tanító képeken lévő releváns objektumok előre körberajzolt pozícióját, mert ez nagy különbségeket okozott. Látható, hogy a rendszerünk (amiben ekkor még szintén nem volt benne a Szuper-vektor) a legjobban szerepelt a körberajzolásokat nem használó megoldások között. A jövőben meg fogjuk vizsgálni, hogy milyen eredményt tudunk elérni, ha figyelembe vesszük a körberajzolásokat. Illetve amint elérhető lesz az on-line kiértékelő rendszer, tesztelni fogjuk a Szuper-vektorral kiegészített megoldásunkat ezen a teljes Pascal VOC 2011 adathalmazon is.

Összesített helyezés	Fejlesztő csapat	MAP (Nem használ előre körberajzolt tanító területeket)	MAP (Használ előre körberajzolt tanító területeket)
1.	NUS - National University of Singapore	-	**0,7856**
2.	NLPR - National Laboratory of Pattern Recognition, Institute of Automation Chinese Academy of Sciences	0,6108	0,7823
3.	ISIS - University of Amsterdam, University of Trento	-	0,7335
4.	MSR - Microsoft Research Asia & University of Science and Technology of China	-	0,7049
5.	LIRIS - LIRIS, Ecole Centrale de Lyon, CNRS, UMR5205, France	0,6094	0,6681
6.	**DMWS, MTA SZTAKI**	**0,6139**	-
7.	SJT - Shanghai Jiao Tong Univeristy	0,5485	0,604
8.	JDL - Institute of Computing Technology, Chinese Academy of Sciences	0,5478	-
9.	KUL - University of Leuven	0,4512	-
…			

5.3. táblázat: Előzetes eredmények a Pascal VOC 2011 versenyen

A fentiek alapján kijelenthetjük, hogy a kiindulási Fisher-vektor alapú rendszer hatékonyan működik. Ezt sikerült megjavítanom a Szuper-vektoros kiegészítéssel, az 5.3.3 pontban szereplő eredmények alapján.

5.3.2. A különböző képi leírók

Az alábbi táblázatban azt foglaltam össze, hogy a 18 féle képi leíróból származó kernel mátrixokat egyenként felhasználva a tanításhoz, milyen eredményeket kapunk. A Late Fusion során ezeknek az osztályozóknak a kimenetét kombináljuk. A táblázatban a 20 kategóriára kapott Average Precision mutatók átlagát a Mean Average Precision (MAP) értékeket tüntettük fel, mivel a képi klasszifikációban általában ez a mérvadó mérőszám.

Magas szintű leíró	Alacsony szintű leíró	Spatial pooling	MAP
Fisher-vektor, K=256	RGB szín	teljes kép	**0,4328**
Fisher-vektor, K=256	RGB szín	H-L pontok	**0,3828**
Fisher-vektor, K=256	RGB szín	1x3	**0,4519**
Fisher-vektor, K=256	RGB szín	2x2	**0,4427**
Fisher-vektor, K=256	SIFT	teljes kép	**0,4756**
Fisher-vektor, K=256	SIFT	H-L pontok	**0,4333**
Fisher-vektor, K=256	SIFT	1x3	**0,4966**
Fisher-vektor, K=256	SIFT	2x2	**0,4771**
Fisher-vektor, K=1024	RGB szín	teljes kép	**0,4428**
Fisher-vektor, K=1024	RGB szín	1x3	**0,4509**
Fisher-vektor, K=1024	SIFT	teljes kép	**0,4796**
Fisher-vektor, K=1024	SIFT	1x3	**0,4895**
Szuper-vektor, K=2048	RGB szín	teljes kép	**0,4245**
Szuper-vektor, K=2048	RGB szín	H-L pontok	**0,3598**
Szuper-vektor, K=2048	RGB szín	1x3	**0,4270**
soft Szuper-vektor, K=2048	SIFT	teljes kép	**0,4535**
soft Szuper-vektor, K=2048	SIFT	H-L pontok	**0,4674**
soft Szuper-vektor, K=2048	SIFT	1x3	**0,4858**

5.4. táblázat: Eredmények a képi leírók egyenkénti felhasználásával

Az eredmények alapján azt tapasztaltuk, hogy a szín momentum és a SIFT hasonlóan viselkedik, mindkét esetben az 3x1-es pooling jobb teljesítményt nyújt, mint a többi. Emellett érdekes tapasztalat, hogy van olyan eset ahol a kevesebb mintavételi pontból készült, Harris-Laplace detektált pontokon számolt leíró jobb eredményt ért el, mint a sűrű griden és a detektált pontokon egybe számolt leíró.

5.3.3. Kombinált módszerek

Külön vizsgálatokat végeztem úgy, hogy csak a Fisher-vektorokat, csak a Szuper-vektorokat, végül mindkettőt kombinálom, mind a Late Fusion, a kernel átlagolás és a kernel aggregálás módszerével. Tehát az alábbi táblázatban „Fisher” címkével ellátott sorok a 12 féle Fisher-vektorból származó kernelek kombinálását jelenti, a „Szuper” címke a 6 féle Szuper-vektor kombinálását, a „Fisher+Szuper” pedig mind a 18 féle kernel kombinálását.

Képi leírók	Kombinálási módszer	MAP
Fisher	Late Fusion	**0,5498**
Fisher	Kernel átlagolás	**0,5479**
Fisher	Kernel aggregálás	**0,5493**
Fisher	LF+átlag+aggregált	**0,5569**
Szuper	Late Fusion	**0,5449**
Szuper	Kernel átlagolás	**0,5493**
Szuper	Kernel aggregálás	**0,5411**
Szuper	LF+átlag+aggregált	**0,5532**
Fisher+Szuper	Late Fusion	**0,5553**
Fisher+Szuper	Kernel átlagolás	**0,5548**
Fisher+Szuper	Kernel aggregálás	**0,5562**
Fisher+Szuper	LF+átlag+aggregált	**0,5639**

5.5. táblázat: Eredmények a képi leírók kombinálásával

Az elvárásainknak megfelelően a Szuper-vektor javított ez eredeti, csak Fisher-vektorokra épülő rendszer működésén, mindhárom módszer esetén, 0,005-0,007-tel emelve a MAP értékét. A legjobb eredményt minden esetben a három módszer predikcióinak átlaga adja.

A csak Szuper-vektorokra épülő kernel aggregálás és Late Fusion viszont meglepő módon rosszabb eredményeket ad, mint az átlagolt kernel. Ennek indoka a súlyok túltanulása lehet.

A három kombinálási módszer végül nagyjából azonos, 0,555 – 0,556 MAP-ot ad. Viszont korántsem igaz, hogy azonos módon működnek, ezt a következő táblázat mutatja, amely a kategóriánkénti eredményeket adja meg a három módszerre. Jó láthatóan kategóriánként változó, hogy az előzetes súlyozás segít-e a rendszer működésében, illetve melyik kombináció a leghatékonyabb a validációs halmazon. Más és más találatokat preferálnak, épp ezért adódik a három rendszer kimenetének kombinálása, ami mint egy többségi döntés a három módszer által preferált találatokat részesíteni előnyben. Végeredményben a legjobb Fisher-vektoros kombináláshoz képest a Szuper-vektorral képes voltam 0,007-tel javítani a MAP értékét.

Kategória	Kernel átlagolás	Kernel aggregálás	Late Fusion	LF+átlag+aggregált
aeroplane	0,8359	0,8437	0,8411	**0,8456**
bicycle	0,5129	0,5219	0,5198	**0,5323**
bird	**0,5886**	0,5631	0,5638	0,5864
boat	0,5876	0,6032	**0,6230**	0,6115
bottle	0,2316	0,2268	0,2251	**0,2371**
bus	0,7927	**0,8062**	0,7951	0,8056
car	0,5650	0,5689	0,5568	**0,5764**
cat	0,6214	**0,6337**	0,6229	0,6333
chair	0,5210	0,5240	0,5157	**0,5273**
cow	0,3334	0,3399	0,3136	**0,3457**
diningtable	0,4312	0,4459	0,4342	**0,4510**
dog	0,5056	0,5137	0,5113	**0,5190**
horse	**0,4521**	0,4348	0,4380	0,4480
motorbike	0,6191	0,6155	0,6150	**0,6220**
person	0,8383	0,8374	0,8346	**0,8391**
pottedplant	0,3694	0,3739	0,3585	**0,3742**
sheep	0,4989	0,5042	**0,5289**	0,5179
sofa	0,4322	0,4200	0,4209	**0,4390**
train	0,7443	0,7447	**0,7516**	0,7505
tvmonitor	0,6153	0,6027	0,5989	**0,6168**
MAP	0,5548	0,5562	0,5534	**0,5639**

5.6. táblázat: Kategóriánkénti eredmények a Fisher- és Szuper-vektorok kombinálásával

6. Összefoglalás és kitekintés

Dolgozatomban megmutattam, hogy az általam implementált Szuper-vektor leírók segítségével, kernelek felhasználásával javítani lehet egy Fisher-vektor alapú rendszer működését.

A jövőben mindenképpen szeretnénk az aggregált kernel súlyainak meghatározását optimálisan megoldani, a kényszerű közelítések helyett. Ehhez a 4.1.3 pontban felírt optimalizálási problémát kell megoldanunk hatékony módon, hogy nagyméretű adathalmazra és nagy mennyiségű kategóriára is alkalmazható legyen.

Az egyszerű partícionálással végzett Spatial poolingot továbbfejlesztenénk szegmentálás alapú felosztással, és lokális, tömörebb leírók segítségével, amelyek egy-egy objektumot jobban jellemeznek.

Tervezzük nagy adatmennyiségre is alkalmazni a rendszerünket. Ennek érdekében kompaktabb szemantikai leírókat szeretnénk készíteni. Amennyiben több ezer, akár több mint tízezer kategóriában lennénk képesek hatékonyan automatikus képi annotációt végezni, a jelenlegi módszereknél finomabb szöveg alapú képkeresésre adódna lehetőség.

Irodalomjegyzék

[1] Everingham, M., Van~Gool, L., Williams, C. K. I., Winn, J., Zisserman, A.: *„The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results”*, http://www.pascal-network.org/challenges/VOC/voc2011/

[2] Li Fei-Fei, Rob Fergus, Antonio Torralba: *„Recognizing and Learning Object Categories”*, ICCV 2005 Beijing, Short Course

[3] M. Stricker, M. Orengo: *„Similarity of color images”*, Proc. SPIE Storage and Retrieval for Image and Video Databases, 1995

[4] M. Swain, D. Ballard: *„Color indexing”*, International Journal in Computer Vision, vol. 7, no. 1, pp. 11–32, 1991

[5] Y. Liu, D.S. Zhang, G. Lu, W.-Y. Ma: *„Region-based image retrieval with perceptual colors”*, Proceedings of the Pacific-Rim Multimedia Conference (PCM), December 2004, pp. 931–938.

[6] T. Ojala, M. Pietikäinen, D. Harwood: *„Performance evaluation of texture measures with classification based on Kullback discrimination of distributions”*, Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR 1994), vol. 1, pp. 582 - 585.

[7] Lowe, D. G.: *„Distinctive Image Features from Scale-Invariant Keypoints”*, International Journal of Computer Vision, 60, 2, pp. 91-110, 2004.

[8] N. Dalal, B. Triggs: *„Histograms of oriented gradients for human detection”*, Computer Vision and Pattern Recognition, San Diego, CA, June 20–25, 2005.

[9] C. Harris, M. Stephens: *„A combined corner and edge detector”*, in Alvey Vision Conference, pp. 147–151, 1988.

[10] T. Tuytelaars, K. Mikolajczyk: *„Local Invariant Feature Detectors: A Survey”*, Foundations and Trends in Computer Graphics and Vision, Vol. 3, No. 3 (2007) pp. 177–280.

[11] Mikolajczyk, K., Schmid, C.: *„Scale & affine invariant interest point detectors”*, International Journal on Computer Vision 60(1): pp. 63-86, 2004.

[12] T. Lindeberg: *„Feature detection with automatic scale selection”*, International Journal of Computer Vision 30 (2): pp. 77-116, 1998.

[13] *„Principal Component Analysis”*, Wikipedia, http://en.wikipedia.org/wiki/Principal_component_analysis (letöltés dátuma: 2011.05.11.)

[14] Carlo Tomasi: *„Estimating Gaussian Mixture Densities with EM – A Tutorial”* http://www.cs.duke.edu/courses/spring04/cps196.1/handouts/EM/tomasiEM.pdf (letöltés dátuma: 2011.05.05.)

[15] *„K-means clustering”*. Wikipedia. http://en.wikipedia.org/wiki/K-means_clustering (letöltés dátuma: 2011.10.15.)

[16] Tommi Jaakkola, David Haussler: *„Exploiting Generative Models in Discriminative Classifier”*, Advances in Neural Information Processing Systems 11, pp. 487–493. MIT Press, 1998.

[17] Perronnin, F., Dance, C.: *„Fisher Kernels on Visual Vocabularies for Image Categorization”* Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference

[18] P. Moreno and R. Rifkin. *„Using the Fisher kernel method for web audio classification”*, ICASSP, volume 4, pp. 2417-2420, 2000.

[19] Jorge Sánchez, Florent Perronnin, Thomas Mensink: *„Improved Fisher Vector for Large Scale Image Classification”*, COMPUTER VISION – ECCV 2010. Lecture Notes in Computer Science, 2010, Volume 6314/2010, pp. 143-156.

[20] Xi Zhou, Kai Yu, Tong Zhang, Thomas S. Huang: *„Image Classification Using Super-Vector Coding of Local Image Descriptors”*, COMPUTER VISION – ECCV 2010, Lecture Notes in Computer Science, 2010, Volume 6315/2010, pp. 141-154.

[21] Lazebnik, S., Schmid, C., Ponce, J.: *„Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories”*, Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference. pp. 2169 - 2178.

[22] Corinna Cortes, Vladimir Vapnik: *„Support-vector networks”*, Machine Learning, Volume 20 (1995), Number 3, pp. 273-297.

[23] Cristianini, N., Shawe-Taylor, J.: *„An introduction to Support Vector Machines and other kernel-based learning methods”*, Cambridge University Press, 2000.

[24] Altrichter Márta, Horváth Gábor, Pataki Béla, Strausz György, Takács Gábor, Valyon József: *„Neurális hálózatok”*, PANEM, 2006.

[25] A. Rakotomamonjy, F. Bach, S. Canu, Y. Grandvalet: *„SimpleMKL”*, Journal of Machine Learning Research, 9, 2491-2521, 2008

[26] Bálint Daróczy, Róbert Pethes, András A. Benczúr: *SZTAKI @ ImageCLEF 2011*, Working Notes of the ImageCLEF 2011 Workshop at CLEF 2011 Conference, Amsterdam, 2011.

[27] Erik Bodzsár, Bálint Daróczy, István Petrás, András A. Benczúr: *„GMM based Fisher vector calculation on GPGPU”* http://datamining.sztaki.hu/?q=en/GPU-GMM

[28] Everingham, M. and Van~Gool, L. and Williams, C. K. I. and Winn, J. and Zisserman, A. *„The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results”*, http://www.pascal-network.org/challenges/VOC/voc2007/workshop/

[29] Chris Dance, Jutta Willamowski, Lixin Fan, Cedric Bray, Gabriela Csurka: *„Visual categorization with bags of keypoints”*, ECCV International Workshop on Statistical Learning in Computer Vision (2004)

[30] Koen E.A. van de Sande, Theo Gevers, Cees G.M. Snoek *"Evaluating Color Descriptors for Object and Scene Recognition"*, Pattern Analysis and Machine Intelligence, September 2010 (vol. 32 no. 9), pp. 1582-1596

[31] Chih-Chung Chang and Chih-Jen Lin. *„LIBSVM - A Library for Support Vector Machines”*, http://www.csie.ntu.edu.tw/~cjlin/libsvm/ (letöltés dátuma: 2011.05.11.)