521160P Johdatus Tekoälyyn

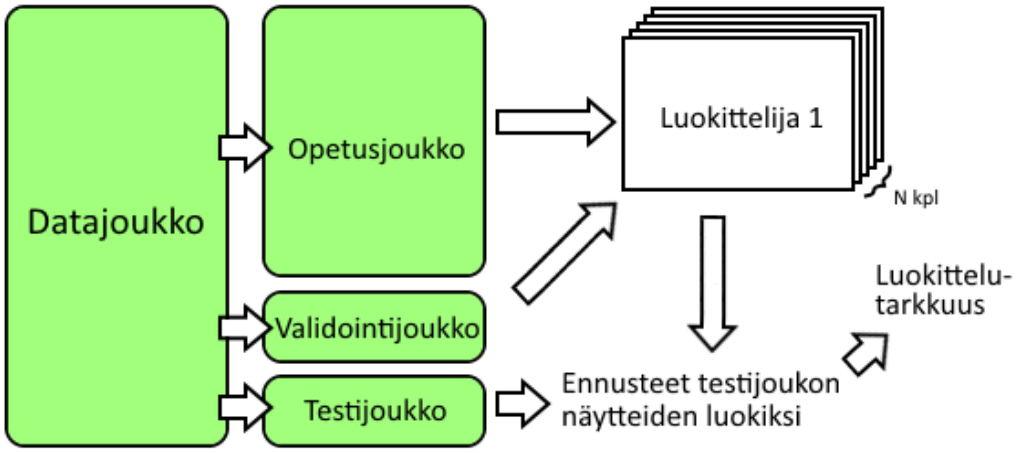
Harjoitus #3

Luokittelu

Kevät 2019

Tekoälyn tärkeä osa-alue koneoppiminen voidaan karkeasti jakaa ohjattuun oppimiseen, ohjaamattomaan oppimiseen ja vahvistusoppimiseen, mutta jako voidaan suorittaa myös muillakin tavoilla. Ohjattu oppiminen puolestaan jaetaan yleisesti regressioon ja luokitteluun. Tässä harjoituksessa käsittelemme ohjatun oppimisen osa-aluetta luokittelu ja siihen liittyvää problematiikkaa. Luokittelussa luokittelija generoidaan luokkatiedon sisältävän datan pohjalta. Tämän jälkeen luokittelija kykenee ennustamaan tuntemattomalle datanäytteelle luokan, johon se luokittelijan mukaan todennäköisimmin kuuluu.

Luokittelussa aluksi datajoukko jaetaan opetusjoukkoon, validointijoukkoon ja testijoukkoon. Opetusjoukon näytteitä käytetään nimensä mukaisesti luokittelijoiden opettamiseen. Validointijoukon näytteitä puolestaan käytetään opetettavien luokittelijoiden hyperparametrien säätämiseen. Testijoukon näytteitä käytetään luokittelijoiden suorituskyvyn testaamiseen, jotka soveltuvat tähän tarkoitukseen erittäin hyvin, sillä ne ovat luokittelijalle ennestään täysin tuntemattomia. Lisäksi testijoukon luokittelutuloksen perusteella pystytään valitsemaan opetettujen luokittelijoiden joukosta paras luokittelija tutkittavalle luokitteluongelmalle. Kuvassa 1 on esitetty alkuperäisen datajoukon jako opetus-, validointi- ja testijoukkoon, jonka jälkeen voidaan valita validointijoukkoa ja testijoukkoa hyödyntäen paras luokittelija. Ennustamalla luokittelijoilla luokat testijoukon näytteille, saadaan laskettua luokittelutuloksista luokittelutarkkuudet (engl. accuracy).



Kuva 1. Datajoukon jako opetus-, validointi- ja testijoukkoon sekä niiden käyttötarkoitukset.

Jos opetusjoukon koko on liian pieni, tapahtuu herkästi ylioppiminen, jolloin luokittelija mallintaa liian tarkasti pienen opetusnäytteistä koostuvan otoksen tunnusomaisimpia piirteitä. Jos testijoukon koko on liian pieni, on luokittelijoiden luokittelutarkkuudet epäluotettavia. Jos taas validointijoukon koko on liian pieni, ei kyetä löytämään luokittelijoille optimaalisia hyperparametrien arvoja, jolloin myöskään luokittelutulokset eivät tule olemaan parhaita mahdollisia. Ottamalla aiemmat seikat huomioon ja alkuperäisen datajoukon ollessa riittävän iso, yleinen jakosuhde datajoukolle on 80:10:10, jossa mahdollisimman suurta osaa

näytteistä (80%) käytetään luokittelijoiden opettamiseen ja pienempää osaa näytteistä (10%+10%) luokittelijan validoimiseen ja testaamiseen.

Kun testijoukon tai validointijoukon kokoa ei voida kasvattaa esimerkiksi liian pienestä alkuperäisestä datajoukosta johtuen, on käytettävä aivan erityistä menetelmää: ristiinvalidointia. Siinä opetusjoukko jaetaan S kappaleen osajoukoiksi, joista yhtä kerrallaan käytetään luokittelijan validoimiseen ja loppuja S-1 osajoukkoa luokittelijan opettamiseen. Tämä toistetaan S kertaa niin, että jokainen osajoukko on ollut kerran validointijoukkona, jonka jälkeen voidaan arvioida luokittelijan hyvyyttä ottamalla keskiarvo validointijoukkojen luokittelutarkkuuksista.

Esimerkiksi tehtävänä voi olla tunnistaa, onko digitaalisessa kuvassa lintu vai jotain muuta. Tällöin datajoukossa tulee olla kahdenlaisia näytteitä: kuvia, joissa on lintu sekä ns. negatiivia näytteitä eli kuvia, joissa ei ole lintua vaan autoja, lentokoneita, koiria jne. Ennen luokittelijoiden luomista kuvat lajitellaan luokkiin antamalla esimerkiksi lintukuville luokaksi 1 ja ei-lintukuville luokaksi 2. Nyt koko datajoukko voidaan jakaa opetusjoukkoon, validointijoukkoon ja testijoukkoon ja opettaa validointijoukon avulla optimoidut luokittelijat opetusjoukon näytteillä. Opettamisen jälkeen luokittelijat kykenevät tunnistamaan eri luokittelutarkkuuksilla, milloin testijoukon kuvassa on lintu (luokka 1) ja milloin siinä on jotain muuta (luokka 2).

Luokittelussa luokittelijan suorituskykyä arvioidaan luokittelutarkkuuden lisäksi sekaannusmatriisin (engl. confusion matrix) avulla, joka yleensä toteutetaan testijoukosta. Sekaannusmatriisi kertoo, kuinka moni näytteistä luokitellaan oikein ja kuinka moni väärin sekä mihin luokkiin väärin luokitellut näytteet oikeasti kuuluvat. Tässä harjoitusmateriaalissa sekaannusmatriisissa oikeat luokat on sijoitettu riveille ja luokittelijan ennustamat luokat sarakkeille, mutta tämä voidaan toteuttaa myös toisinpäin. Mikäli sekaannusmatriisin tietyllä rivillä olevalla luokalla on lukuja eri sarakkeilla kuin sen oma rivinumero, on luokittelija luokitellut näytteitä vääriin luokkiin. Kuvassa 1 on esitetty esimerkki eräästä sekaannusmatriisista, johon kuuluu 9 luokkaa. Kuvaan on myös lisätty luokittelutarkkuudet jokaiselle luokalle sekä luokittelijan kokonaisluokittelutarkkuus. Paras tulos luokittelijalle saavutetaan silloin, kun sen sekaannusmatriisin päädiagonaalilla ovat kaikki luokittelijan luokittelemat testijoukon näytteet.

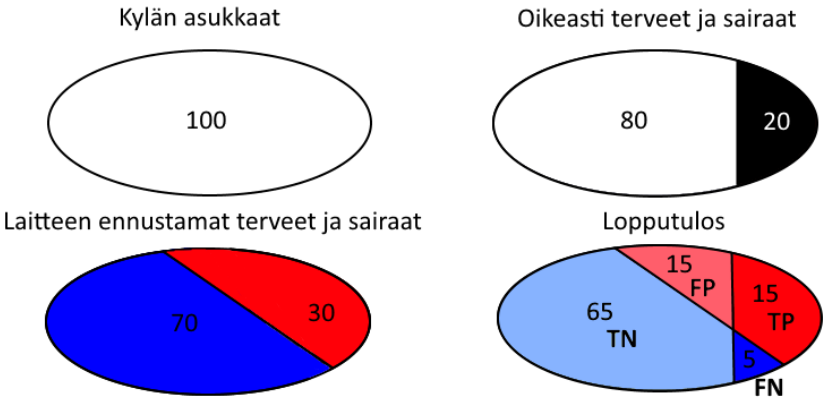
		Ennustetut luokat								Luokittelu-tarkkuus
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	
Oikeat luokat	**1**	**137**	13	3	0	0	1	1	0	0	0.89
	2	1	**55**	1	0	0	0	0	6	1	0.86
	3	2	4	**84**	0	0	0	1	1	2	0.89
	4	3	0	1	**153**	5	2	1	1	1	0.92
	5	0	0	3	0	**44**	2	2	1	2	0.82
	6	0	0	2	1	4	**35**	0	0	1	0.81
	7	0	0	0	0	0	0	**61**	2	2	0.94
	8	0	0	0	1	0	0	0	**69**	3	0.95
	9	0	0	0	0	0	0	0	2	**26**	0.93
											0.89

Kuva 2. Esimerkki sekaannusmatriisista yhdeksän luokan tapaukselle.

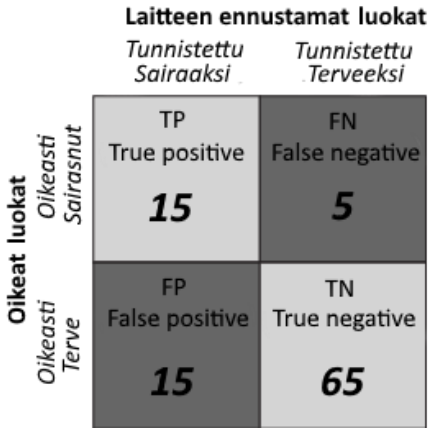
Mittasuureena luokittelutarkkuus ei kuitenkaan yksistään anna riittävän hyvää kuvaa luokittelijan hyvyydestä varsinkaan silloin, kun kyseessä on epäbalansoitu datajoukko. Epäbalansoidussa datajoukossa eri luokissa on selvästi eri määrä näytteitä. Huomaa, että kun todellisessa tilanteessa käsittelyssä on epäbalansoitu datajoukko, ensimmäinen ratkaisu on yrittää saattaa datajoukko kaikin mahdollisin keinoin balanssiin. Tämä

ei kuitenkaan aina ole mahdollista ja siksi luokittelussa käytetään luokittelutarkkuuden lisäksi mittoja sensitiivisyys/saanti (engl. recall, TPR), spesifisyys, positiivisen testin ennustearvo/tarkkuus (engl. precision) ja negatiivisen testin ennustearvo kuvaamaan luokittelijan hyvyyttä. Käydään seuraavaksi esimerkin avulla läpi näiden eri mittojen eroavaisuuksia.

100 ihmisen kylästä 20 sairastuu harvinaiseen virukseen, jonka vakavat oireet alkavat näkyä vasta kuukauden kuluttua tartunnan saamisesta. Kylän lääkärin tehtävä on selvittää kyseisen viruksen havaitsemista varten suunnitellulla laitteella tartunnan saaneet henkilöt, jotta hoito voidaan aloittaa mahdollisimman pian. Laite arvioi, että 30 ihmistä 100:sta on sairastunut virukseen. Kuukauden päästä kuitenkin selviää, että ennustetuista 30 sairaasta ihmisestä todellisuudessa 15 oli viruksen kantajia (TP) ja 15 olikin terveitä (FP). Koska sairastuneita oli yhteensä 20, niin 5 ihmistä tunnistettiin terveiksi, vaikka he olisivat olleet oikeasti sairaita (FN). Kylän ihmisistä 65 tunnistettiin oikein terveiksi (TN). Nyt voidaan laskea laitteen suorituskykyä kuvaavat tunnusluvut käyttämällä kaavoja 1-5. Kuvassa 3 on havainnollistettu tekstin esimerkki kuvasarjan avulla. Kuvassa 4 tekstin esimerkille on muodostettu sekaannusmatriisi.



Kuva 3. Tekstin esimerkki havainnollistettuna kuvasarjan avulla.



TP, oikea hälytys (engl. true positives, hit)
FP, väärä hälytys (engl. false positives, false alarm)
FN, väärä hylkäys (engl. false negatives, miss)
TN, oikea hylkäys (engl. true negatives, correct rejection)

Kuva 4. Sekaannusmatriisi tekstin esimerkille.

Sensitiivisyys/Saanti: $\frac{TP}{P} = \frac{TP}{TP+FN} = \frac{15}{15+5} = 75\%$ (1)

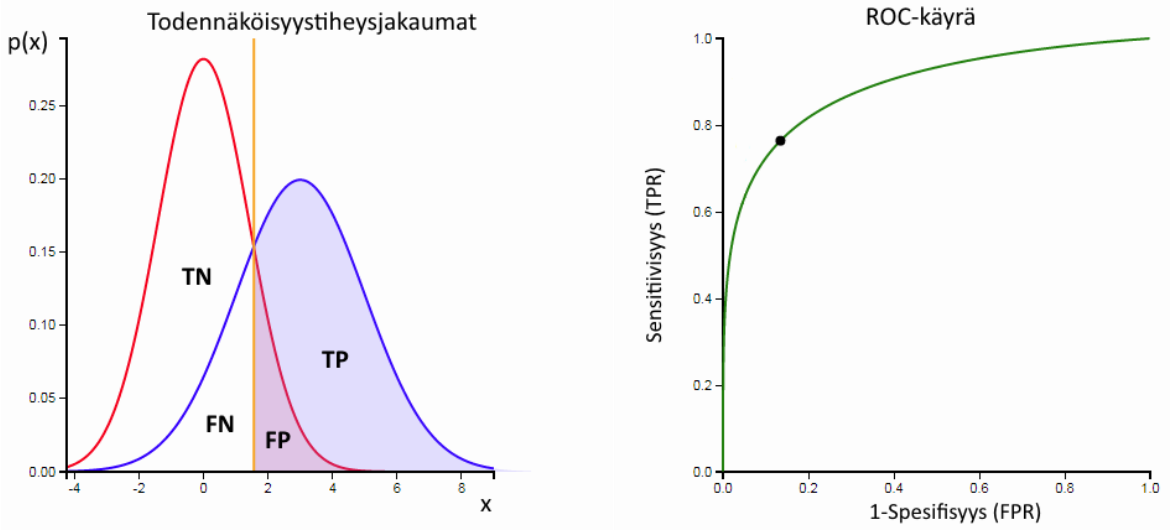
Spesifisyys: $\frac{TN}{N} = \frac{TN}{TN+FP} = \frac{65}{65+15} = 81{,}25\%$ (2)

Positiivisen testin ennustearvo/Tarkkuus: $\frac{TP}{TP+FP} = \frac{15}{15+15} = 50\%$ (3)

Negatiivisen testin ennustearvo: $\frac{TN}{TN+FN} = \frac{65}{65+5} = 92{,}9\%$ (4)

Luokittelutarkkuus: $\frac{TP+TN}{TP+FP+TN+FN} = \frac{15+65}{15+15+65+5} = 80\%$ (5)

Esimerkinkaltaista kahden luokan tilannetta voidaan tarkastella myös todennäköisyystiheysjakaumina. Oletetaan, että jakaumat ovat osittain päällekkäisiä. Kynnysarvoa säätämällä voidaan esimerkiksi pienentää väärien hylkäysten (FN) määrää mutta samalla myös väärien hälytysten määrä tulee kasvamaan (FP). Tämä sama voidaan esittää ROC-käyrän (engl. receiver operating charasterictic curve) avulla, jossa pystyakselilla on sensitiivisyys ja vaaka-akselilla 1-spesifisyys (engl. FPR) [1]. Kuvassa 5 on esitetty kahden luokan todennäköisyystiheysjakaumat kynnysarvolla sekä sitä vastaava ROC-käyrä. Esimerkiksi vakavaa sairautta tunnistaessa sensitiivisyyden arvo, joka kertoo kuinka moni kaikista sairastuneista henkilöistä (TP+FN) tunnistettiin sairaiksi (TP), on oltava mahdollisimman iso. Se voidaan saavuttaa valitsemalla kuvan 5 ROC-käyrältä piste, jossa sensitiivisyys saa mahdollisimman ison arvon, mikä näkyy myös samalla 1-spesifisyyden (FPR) arvon kasvamisena (eli spesifisyyden arvon pienenemisenä). Tämä vastaa kynnysarvon siirtämistä vasemmalle todennäköisyystiheysjakaumien kuvaajassa.



Kuva 5. Todennäköisyystiheysjakaumat kynnysarvolla ja sitä vastaava ROC-käyrä.

Tehdessäsi harjoitusta omalla tietokoneella, saat asennettua tarvittavat kirjastot tietokoneellesi seuraavan komennon avulla:

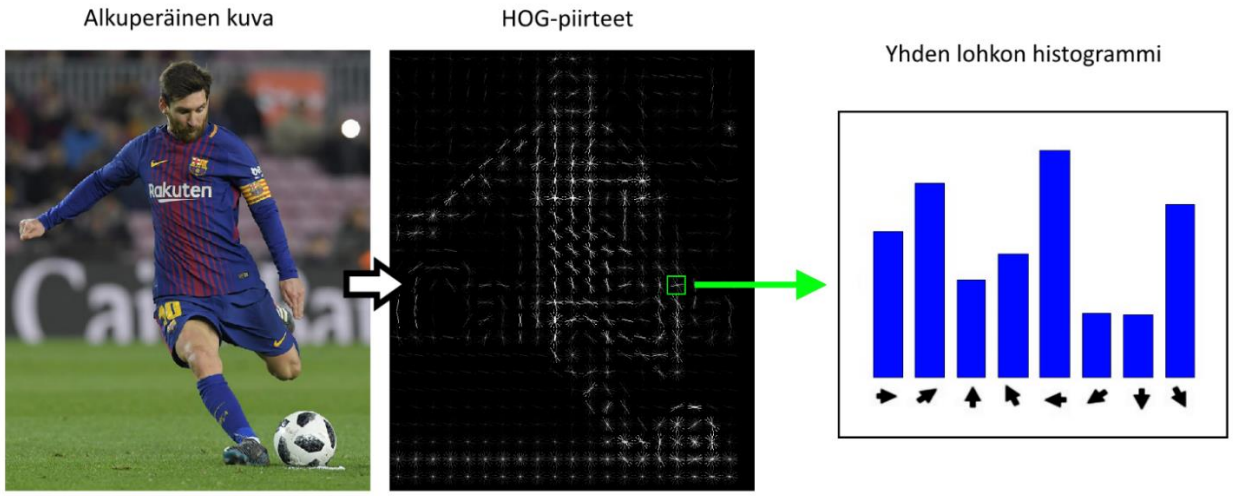
```
pip install numpy scipy matplotlib scikit-learn scikit-image glob3 imutils opencv-python==3.4.5.20
```

Huom. Jotta tehtävissä käytetyt kooditiedostot toimivat oikein, sinun tulee käyttää pythonin 3:tta versiota sekä OpenCV-kirjaston versiota 3.4.5.

Tehtävä 1 (2.5p)

Tehtäväsi on luoda erilaisia luokittelijoita käsinkirjoitettujen numeroiden luokitteluun, löytää tälle luokitteluongelmalle parhaat luokittelijat arvioimalla niiden tuottamia lopputuloksia ja raportoida havainnot. Tehtävässä käytetään MNIST-datajoukkoa, joka sisältää käsinkirjoitetut numerot 0:sta 9:sään. Yhteensä MNIST-datajoukossa on 70 000 näytettä ja ne on kerätty Yhdysvalloissa Census Bureau:n työntekijöiltä sekä toisen asteen opiskelijoilta. MNIST-datassa näytteet on normalisoitu 20x20 pikselin kuviksi ja lopuksi keskitetty 28x28 pikselin ruudukolle. Tälle datajoukolle on onnistuttu luomaan yksinkertaisia lineaarisia luokittelijoita, joiden luokittelutarkkuus on karkeasti 80 % ja 99,5 % välillä. Konvoluutioneuroniverkkojen avulla on saavutettu MNIST-datajoukolle parhaimmillaan 99,77 % luokittelutarkkuus. [2]

Tässä tehtävässä datajoukon näytteitä ei syötetä suoraan luokittelualgoritmeille, vaan kuvien esikäsittelyllä saavutetaan selvästi parempi lopputulos. Datan esikäsittelytekniikaksi on valittu menetelmä nimeltä HOG (engl. histogram of oriented gradients), joka laskee kuville piirrevektorit. HOG-menetelmä tuottaa gradientteja eli nuolia, jotka osoittavat mihin suuntaan kuva tummenee ja millä suuruudella. Kyseisen menetelmän etu on, että sama kuva eri kontrasteilla ei vaikuta tuotettuihin piirteisiin. Tämä tarkoittaa, että valaistukseltaan todella tummat ja todella vaaleat kuvat johtavat kutakuinkin samanlaisiin HOG-piirteisiin. Kuvassa 6 on esitetty, miltä alkuperäisestä kuvasta lasketut HOG-piirteet näyttävät ja miten yhden lohkon piirteet kuvautuvat histogrammissa.



Kuva 6. HOG-piirteiden muunnos RGB-kuvalle.

Tässä tehtävässä käytetään seuraavia luokittelualgoritmeja:

0. RandomGuessing-luokittelija, joka valitsee luokan sattumanvaraisesti testijoukon näytteille. Muiden luokittelijoiden tulee toimia ainakin tätä paremmin.
1. Tukivektorikone (engl. support vector machines, SVM)
2. k-lähimmän naapurin menetelmä (engl. k-nearest neighbors, kNN)
3. Päätöspuu (engl. decision tree)
4. Satunnaismetsä (engl. random forest)
5. Adaptiivinen tehostaminen (engl. adaptive boosting)
6. Normaalijakautunut naivi-Bayes-luokittelija (engl. gaussian naive bayes)
7. Stokastinen gradientinoptimointi-luokittelija (engl. stochastic gradient descent, SGD)
8. Lineaarinen diskriminanttianalyysi (engl. linear discriminant analysis, LDA)
9. Logistinen regressio (engl. logistic regression)
10. Monikerros perceptroni (engl. multi-layer perceptron, MLP)

Luodaksesi eri algoritmeilla toteutettuja luokittelijoita, aja komentoriviltä python-tiedosto **generateClassifiers.py**. Tiedostossa data jaetaan opetusjoukkoon ja testijoukkoon, opetetaan luokittelijat ja luodaan pdf-tiedosto **classification_report.pdf**, joka sisältää testijoukosta lasketut sekaannusmatriisit ja luokitteluraportit opetetuille luokittelijoille. Koska luokittelijoiden hyperparametrit on jo selvitetty etukäteen, ei validointijoukkoa tarvita tässä tehtävässä. Voit myös tallentaa komentoriville tulostuvat sekaannusmatriisit tekstitiedostoon alla olevan komennon mukaisesti. Huomaa, että luokittelijoiden luominen kestää tietokoneesta riippuen noin 5-10 minuuttia.

```
python generateClassifiers.py > classification_report.txt
```

Tarkastelemalla joko tekstitiedostoa tai pdf-tiedostoa löydät luokitteluraporteista luokittelijoiden luokittelutuloksen jokaiselle luokalle sekä koko datajoukolle, jotka on ilmoitettu sensitiivisyyden/saannin (engl. recall), positiivisen testin ennustearvon/tarkkuuden (engl. precision) ja F1-arvon (engl. F1-score) avulla. F1-arvo yhdistää sensitiivisyyden ja positiivisen testin ennustearvon kaavan 6 mukaisesti. Tiedostoista löytyy myös luokittelijoiden sekaannusmatriisit. Huomaa, että scikit-learn kirjaston sekaannusmatriisit on muodostettu niin, että riveillä on todelliset luokat (engl. ground truth) ja sarakkeilla on ennustetut luokat.

$$\mathrm{F1-score} = 2 \cdot (\mathrm{precision} \cdot \mathrm{recall}) / (\mathrm{precision} + \mathrm{recall}) \qquad (6)$$

Arvioi luokittelun lopputulosta vastaamalla raportissa seuraaviin kolmeen kysymykseen:

KYSYMYS1: Arvioi kaikkien luokittelijoiden luokitteluraportteja tarkastelemalla, mitkä kolme luoduista luokittelijoista ovat parhaita tämän ongelman ratkaisemiseksi luokittelutarkkuuden (engl. accuracy) perusteella?

KYSYMYS2: Kun olet löytänyt parhaat luokittelijat, valitse niistä yksi ja kerro valintasi. Tarkastele valitsemasi luokittelijan luokitteluraporttia ja selvitä, mikä/mitkä luokista oli helpoin tunnistaa ja mikä/mitkä vaikein F1-arvon perusteella?

KYSYMYS3: Tarkastele lopuksi valitsemasi luokittelijan sekaannusmatriisia ja kerro, mihin käsinkirjoitettuihin numeroihin vaikeinten tunnistettava luokka/luokat oli yleisimmin sekoitettu?

Tämän jälkeen testaa komentoriviltä valitsemaasi luokittelijaa eräälle toiselle testijoukolle, joka koostuu kuvista photo1.jpg, photo2.jpg ja photo3.jpg. Tämä onnistuu ajamalla python-tiedosto **performClassifier.py** seuraavan komennon mukaisesti. Testikuvat sisältävät yhteensä 8 näytettä jokaiselle käsinkirjoitetulle numerolle.

```
python performClassifier.py -i=photo1.jpg -c=model0randomguessing.pkl
```

,

missä -i viittaa kuvaan ja -c viittaa luokittelijaan.

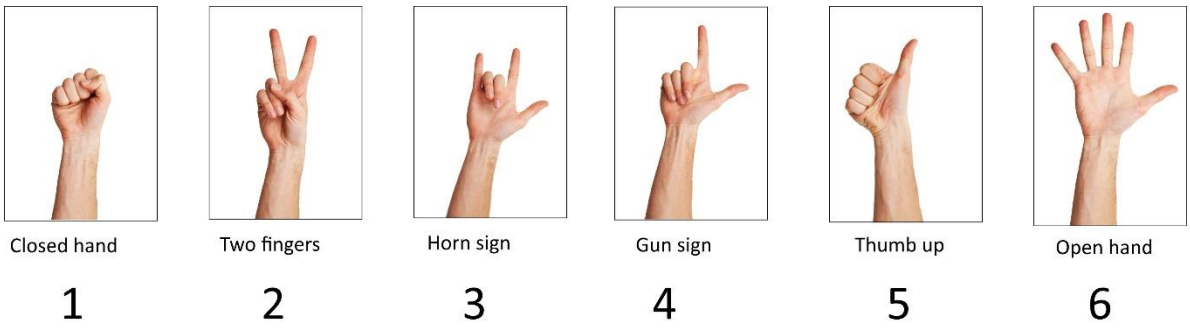
Näet mustavalkoistetun kuvan lopputuloksen ikkunassa *”The thresholded image”* ja luokittelun lopputuloksen kuvan näytteille ikkunassa ”*The output image*”. Kooditiedosto piirtää suorakulmiot yksittäisten käsinkirjoitettujen numeroiden ympärille ja tunnistaa numerot käyttämälläsi luokittelijalla. Luokittelun lopputulos tallennetaan suljettuasi ikkunat painamalla esc-näppäintä kuvaan nimeltä **output-<**photo_name**>-<**classifier_name**>.jpg**.

Liitä raporttiin valitsemasi luokittelijan luokittelun lopputulos tälle toiselle testijoukolle eli kuvat output-photo1-<classifier_name>.jpg, output-photo2-<classifier_name>.jpg, output-photo3-<classifier_name>.jpg. Luo tämän lisäksi itsetehty sekaannusmatriisi kuvista koostuvan testijoukon luokittelutuloksen perusteella.

Yhteenvetona tehtävän 1 palautuksista: Palauta tiedosto classification_report.pdf tai classification_report.txt sekä raportti, joka sisältää vastaukset kysymyksiin KYSYMYS1, KYSYMYS2 ja KYSYMYS3, kuvat output-photo1-<classifier_name>.jpg, output-photo2-<classifier_name>.jpg, output-photo3-<classifier_name>.jpg sekä itsetehty sekaannusmatriisi kuvista koostuvalle testijoukolle.

Tehtävä 2 (2.5p)

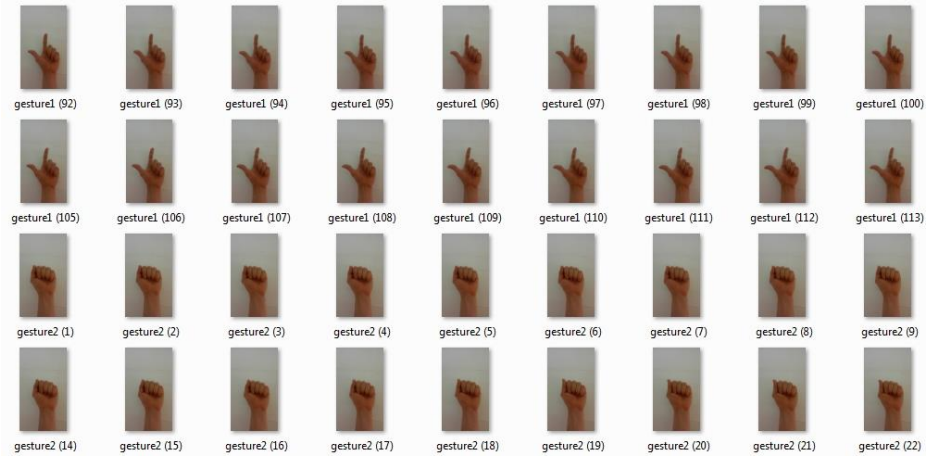
Tässä tehtävässä sinun tulee luoda itse datajoukko, joka sisältää erilaisia käsimerkkejä, opettaa luokittelija optimoiden sen hyperparametrin arvo käyttämällä ristiinvalidointia ja lopuksi arvioida luokittelijan suorituskykyä. Aivan aluksi on päätettävä, montako erilaista käsimerkkiä eli luokkaa halutaan datajoukossa olevan. Mitä suurempi on luokkien määrä, sitä vaikeampi on tunnistaa luokka oikein testijoukon kuvista. Käytä 4-6 luokkaa datajoukossasi. Voit käyttää luokkina esimerkiksi kuvan 7 käsimerkkejä.



Kuva 7. Esimerkkejä erilaisista luokista käsimerkeille valkoista taustaa vasten.

Ota noin 100 kuvaa jokaista käsimerkkiä kohden. Huomaa, että luokissa olevien näytteiden lukumäärä tulee olla kutakuinkin sama toisiinsa nähden saavuttaaksesi balansoidun datajoukon. Kuvien tulee myös sisältää käsimerkin lisäksi osan paljaasta ranteesta kuvan 7 tyyliin. Yritä lisäksi välttää valaistuksesta aiheutuvia varjoja. Älypuhelimien burstimage-sovellukset tai vastaavat helpottavat huomattavasti kuvien ottamista. Valitse kuville yksivärinen tausta, joka poikkeaa selvästi ihonväristä kuten valkoinen tai musta. Kannattaa myös vaihdella käsimerkin asentoa kuvien ottamisen aikana, niin että käsimerkkeihin tulee variaatiota ja myöhemmin luomasi luokittelija joutuu todelliseen testiin. Lisäksi käsimerkin ympärillä on oltava riittävästi taustaa, jotta ohjelma onnistuu sovittamaan suorakulmion käsimerkin ympärille ongelmitta. Tarkasta ennen seuraavaan vaiheeseen siirtymistä, että ottamasi kuvat täyttävät ylläkuvatut ehdot ja tarvittaessa poista virheelliset kuvat.

Kuvien ottamisen jälkeen siirrä ne kansioon nimeltä **input_images** ja nimeä kuvat luokkien mukaan. Käytä kuvien nimeämisessä seuraavaa muotoa: **gesture3 (80).jpg**, joka viittaa kolmannen luokan 80:teen näytteeseen. Windowsilla tämä onnistuu automaattisesti mustamaalaamalla kaikki yhden käsimerkin näytteet ja antamalla niille nimeksi esimerkiksi gesture3. Kuvassa 8 on havainnollistettu, miltä näyttää otos oikein nimetyistä kuvista kansion input_images sisällä.



Kuva 8. Havainnollistava kuva osasta kansion input_images sisältöä.

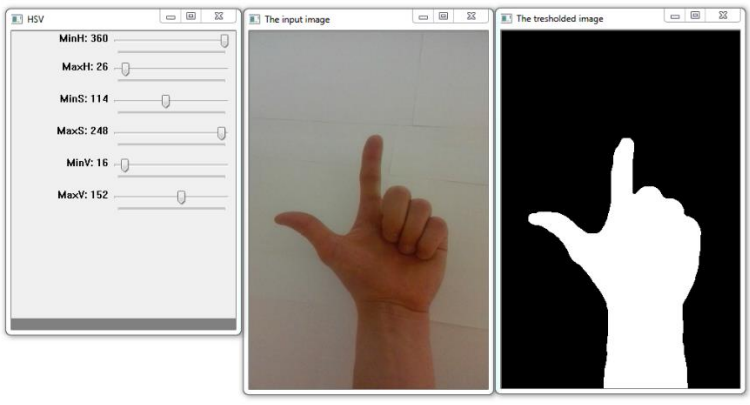
Seuraavaksi esikäsitellään oikein nimetyt käsimerkkikuvat muuntamalla ne mustavalkoisiksi .bmp kuviksi. Tämän vaiheen suorittamiseksi käytä python-tiedostoa **testTransformation.py** testataksesi, miltä muunnos mustavalkoiseksi kuvaksi näyttää. Python-tiedosto valitsee testattavaksi kuvaksi kansion input_images ensimmäisen kuvan. Suorita tämä välivaihe komennolla:

```
python testTransformation.py
```

Ajaessasi testTransformation.py tiedostoa, sinulle avautuu säätöikkuna, jonka HSV-värimallin arvoja säätämällä pyritään erottamaan ihonvärisi rajaama alue alkuperäisestä kuvasta. MinH ja MaxH viittaavat värisävyn (engl. hue) minimi- ja maksimiarvoihin, MinS ja MaxS viittaavat värikylläisyyden (engl. saturation) minimi- ja maksimiarvoihin sekä MinV ja MaxV viittaavat kirkkauden (engl. value or lightness) minimi- ja maksimiarvoihin. Näet muunnoksen lopputuloksen ikkunassa *”The thresholded image”* ja alkuperäisen kuvan ikkunassa *”The input image”*.

Yleisesti ottaen ihmisen ihonvärin rajojen määrittäminen mille tahansa värimallille on erittäin vaikea tehtävä jo pelkästään muuttuvista valaistusolosuhteista johtuen. Karkeasti arvioituna värisävyn vaihteluväli ihonvärille on 330-30°, kun värisävyn arvo on määritelty astelukuna väliltä 0-360°. Värikylläisyyden ja kirkkauden vaihteluvälit ihonvärille riippuvat täysin kuvasta ja molemmat saavat arvoja 0-255 väliltä.

Selvitä värimallin arvoja säätämällä maksimi ja minimiarvot värisävylle, värikylläisyydelle ja kirkkaudelle. Kun olet löytänyt sopivat arvot värimallille, sulje tiedoston testTransformation.py auenneet ikkunat painamalla esc-näppäintä, jonka jälkeen säätämäsi arvot tallentuvat tiedostoon **hsvvalues.txt**. Kuvassa 9 on esitetty oikein säädetyt värimallin vaihteluvälit eräälle RGB-kuvalle.

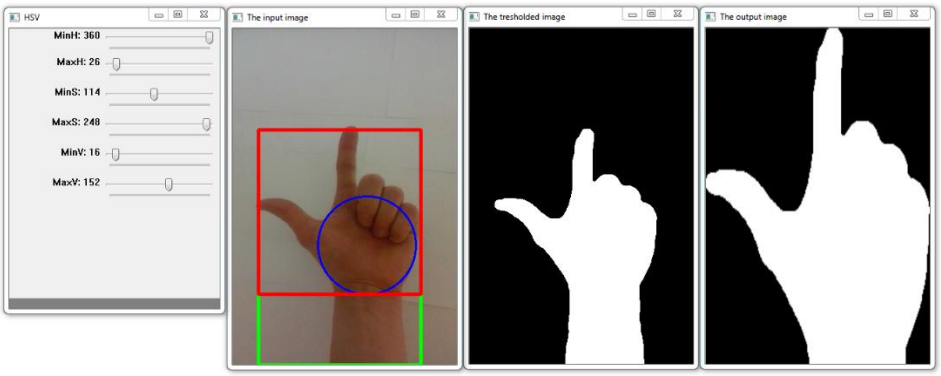


Kuva 9. Oikein säädetyt HSV-värimallin vaihteluvälit eräälle RGB-kuvalle.

Kuvan binäärisoinnin lisäksi rajataan kuvaa ranteen kohdalta sekä keskitetään kuva. Tätä välivaihetta varten aja python tiedosto testTransformation.py argumentilla *cutWrist*:

```
python testTransformation.py --cutWrist
```

Nyt näet ikkunassa *”The input image”* rajattuja alueita, jossa vihreässä suorakulmiossa koko käsi on erotettu taustasta, sininen ympyrä on suurin sovitettavissa oleva ympyrä, joka mahtuu kämmenen sisään ja punainen alue muodostuu määrittämällä ympyrän alareunan kohta lopullisen kuvan alareunaksi. Ohjelma luo myös uuden ikkunan ”The output image”, jossa näkyy koko muunnoksen lopputulos. Kuvassa 10 on esitetty onnistunut muunnos eräälle RGB-kuvalle.



Kuva 10. Onnistunut muunnos eräälle RGB-kuvalle.

Kun olet tyytyväinen lopputulokseen, aja python-tiedosto **runTransformation.py**, joka käsittelee kaikki kansiossa input_images olevat kuvat. Muunnetut kuvat tallentuvat kansioon nimeltä **output_images**. Kyseinen välivaihe kestää muutaman minuutin ajan.

```
python runTransformation.py
```

Lopputuloksena saat mustavalkoisia käsimerkkiin keskitettyjä .bmp kuvia. Ennen seuraavaan vaiheeseen siirtymistä, silmäile vielä output_images kansion sisältö läpi tarkastaen, että kuvat näyttävät oikeilta ja tarvittaessa poista epäonnistuneet muunnokset.

Seuraavaksi erotetaan opetusjoukon ja testijoukon kuvat alkuperäisestä datajoukosta. Tämä tapahtuu ajamalla python-tiedosto **makeDatasets.py**, joka erottaa datajoukosta opetusjoukon ja testijoukon 90:10 jakosuhteella luoden tiedostot **testdataset.pkl** ja **traindataset.pkl**. Testijoukossa ja opetusjoukossa käytettävät kuvat tallennetaan myös kansioihin **output_test_images** ja **output_train_images**.

```
python makeDatasets.py
```

Parhaan mahdollisen luokittelutuloksen saavuttamiseksi laskemme ennen luokittelijan luomista piirteet mustavalkoisista kuvista. Eräs tehokas menetelmä tämän tyyppisen ongelman piirteiden laskemiseksi on muodon kuvailijoihin perustuvat menetelmät (engl. shape descriptor methods). Ne muuntavat kuvassa esiintyvän muodon esimerkiksi histogrammiksi tai listaksi kertoimia, jotka kuvaavat kuvassa esiintyvää muotoa.

Käytetään Hu-momentteja invariantteina muodon kuvailijoina, sillä ne ovat riippumattomia kuvan kääntämisestä, koon muuttamisesta tai muodon sijainnista kuvassa (engl. invariant of rotation, scaling and translation). OpenCV:n funktio *cv2.HuMoments()* käsittelee kuvissa esiintyviä muotoja tilastollisesti ja laskee kuville seitsemän momenttia. Näistä seitsemästä numeroarvosta koostuvat piirrevektorit kuvaavat kuvissa esiintyviä muotoja ja niitä käytetään käsimerkkien tunnistamiseen mustavalkoisten 15 000 pikseliä sisältävien kuvien sijaan.

Alkuperäisen datajoukon näytteiden vähyyden takia tässä tehtävässä ei käytetä ollenkaan validointijoukkoa, vaan luokittelijan hyperparametrien optimointi suoritetaan ristiinvalidoinnilla. Tätä luokitteluongelmaa varten on valittu lineaarinen tukivektorikone-luokittelija, jonka hyperparametrin C arvo tulee optimoida.

Aja komentoriviltä tiedosto **optimizeClassifier.py** löytääksesi optimaalisin hyperparametrin C arvo. Tiedostossa aluksi mustavalkoisista kuvista lasketaan piirteinä Hu-momentit. Tämän jälkeen opetusjoukko jaetaan sattumanvaraisesti viiteen yhtä suureen osaan ja yhtä osaa kerrallaan käytetään validointijoukkona luokittelutarkkuuden laskemiseen ja neljästä jäljelle jäävästä osasta koostuvaa joukkoa luokittelijan opettamiseen. Lopuksi viiden eri luokittelijan luokittelutarkkuuksista voidaan laskea keskiarvo ja vaihteluväli. Kyseinen ristiinvalidointi-menetelmä toteutetaan eri hyperparametrin C:n arvoille parhaan arvon löytämiseksi. Hyperparametrin C arvoina käytetään seuraavia numeroarvoja: 10^{-10}, 10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1. Lopuksi esitetään graafisesti kuvaajassa ristiinvalidoinnilla lasketut luokittelutarkkuuksien keskiarvot ja vaihteluvälit eri C:n arvoilla. Kuvaaja tallennetaan myös tehtäväkansioon nimellä **crossvalidation.png**.

```
python optimizeClassifier.py
```

KYSYMYS1: Arvioi karkeasti tälle luokitteluongelmalle tuotetun kuvaajan perusteella, mikä on käytetyn lineaarisen tukivektorikone-luokittelijan optimaalisin hyperparametrin C arvo?

Kun olet selvittänyt optimaalisimman C:n arvon, opeta koko opetusjoukkoa käyttäen lineaarinen tukivektorikone-luokittelija testijoukon näytteiden tunnistamista varten. Tämä onnistuu käyttämällä komentoa:

```
python generateClassifier.py -C=0.0001 > classification_report.txt,
```

missä -C viittaa lineaarisen tukivektorikone-luokittelijan optimoituun hyperparametrin C arvoon

Lopputuloksena saat luokittelijan nimeltä **model_linearsvm.pkl** sekä tiedostossa **classification_report.txt** ja **classification_report.pdf** sekaannusmatriisin ja luokitteluraportin luokittelijalle toteutettuna testijoukosta.

KYSYMYS2: Minkä takia testijoukko erotettiin datajoukosta aivan prosessin alussa eikä vasta ristiinvalidoinnin jälkeen?

Testaa vielä lopuksi luokittelijaa testijoukon kuvilla ajamalla python-tiedosto **classifyImages.py**. Komento ottaa argumenttina luokittelijan nimen ja luo mosaiikkikuvan testijoukon kuville ja liittää kuviin luokan, johon se luokittelijan mukaan kuuluu. Luokittelun lopputulos tallennetaan myös kuvaan nimeltä **output_model_linearsvm.jpg** suljettuasi ikkunan painamalla esc-näppäintä.

```
python classifyImages.py –c=model_linearsvm.pkl
```

,

missä -c viittaa luokittelijaan

Tehtävän 2 palautukset: Palauta tiedostot classification_report.txt tai classification_report.pdf, hsvvalues.txt, traindataset.pkl, testdataset.pkl ja raportti, joka sisältää vastaukset kysymyksiin KYSYMYS1 ja KYSYMYS2 sekä kuvat output_model_linearsvm.jpg ja crossvalidation.png. Kerro myös raportissa mikä luokan nimi vastaa mitäkin käsimerkkiä (esim. gesture1 vastaa käsimerkkiä open hand jne.).

Lähteet

[1] Duda R., Hart P. & Stork D. (2012) Pattern classification. John Wiley & Sons.

[2] LeCun Y., Cortes C. & Burges C. (1998) The MNIST database of handwritten digits. URL: http://yann.lecun.com/exdb/mnist/. Accessed 22.8.2018.

Yleisiä ongelmakohtia ja kysymyksiä harjoitukseen 3 liittyen

”-c ja –i argumentit antavat virheen vaikka luokittelijan ja kuvan tiedostojen nimet ovat kirjoitettu oikein”

- Jos kopioit komennot suoraan pdf-dokumentista niin välimerkki ”viiva” muuntuu välimerkiksi ”miinus” ja aiheutuu ylläkuvattu virhe. Varmin keino on kirjoittaa komennot näppäimistöllä merkki merkiltä.

”Tehtävää 2 tehdessä ilmenee virhe: error: (-215) ssize.width > 0 && ssize.height > 0 in function cv::resize.”

- Tämä virhe voi aiheutua monesta eri syystä. Virhe voi aiheutua, jos käsimerkki on liian lähellä kuvan reunaa eli sen ympärillä ei ole tarpeeksi taustaa tai jos kuvat eri luokista on otettu erilaisissa ympäristöissä valaistusolosuhteiden muuttuessa radikaalisti tai jos HSV-värimallin värialueet on säädetty väärin. Mikäli kyseinen virhe toistuu kaikesta huolimatta, voit ladata alapuolella esitetystä linkistä valmiiksi luodun datajoukon tehtävää 2 varten.

https://unioulu-my.sharepoint.com/:u:/r/personal/tholmber_univ_yo_oulu_fi/Documents/inputfolder.rar?csf=1&e=lb3gow

”OpenCV:n asentaminen ei onnistu pip komennon avulla Windowsilla”

- Joskus OpenCV:n asentaminen vaatii myös Microsoft Visual Studion asentamisen.

Harjoituksen 3 oppimistavoitteet:

- Ymmärtää, kuinka sekaannusmatriisia luetaan ja kuinka kahden luokan tapauksessa siitä lasketaan yksinkertaisia tunnuslukuja, kuten luokittelutarkkuus, sensitiivisyys ja positiivisen testin ennustearvo.
- Ymmärtää balansoidun ja epäbalansoidun datajoukon eron ja mitä vaikutuksia epäbalansoidulla datajoukolla voi olla luokittelutuloksiin.
- Hahmottaa, miten ROC-käyrää luetaan ja miten kynnysarvon muuttaminen vaikuttaa väärien hälytysten tai väärien hylkäysten määrään.
- Osaa luoda oman datajoukon ja käyttää sitä luokittelijan luomiseen ja sen suorituskyvyn testaamiseen.

Palauta

Palauta pyydetyt tiedostot ja raportit pakattuna tiedostona (.zip tai .rar), jossa on kaksi erillistä kansiota molemmille tehtäville (esim. tehtävä1 ja tehtävä2) Optiman palautuslaatikkoon Harjoitus 3 **12.4.2019 klo 23:59** mennessä. Tästä harjoituksesta on mahdollisuus tienata 5 pistettä (2.5p + 2.5p). Voit antaa palautetta tästä harjoituksesta liittämällä raporttiin erillisen osion palautetta varten. Jos keksit tehtävään 2 jonkin toisen luokittelukohteen kuin käsimerkit, kerro myös siitä palautteessa. Harjoituksia tullaan kehittämään palautteen pohjalta tuleville vuosille vastaamaan paremmin oppimistavoitteita.