521160P Johdatus Tekoälyyn

Harjoitus #3

Luokittelu

Kevät 2018

Koneoppiminen liittyy vahvasti tekoälyyn ja onkin yksi tekoälyn osa-alue. Koneoppinen voidaan karkeasti jakaa ohjattuun oppimiseen, ohjaamattomaan oppimiseen ja vahvistusoppimiseen, mutta se voidaan jakaa myös muillakin tavoilla.

Ohjatussa oppimisessa luokittelijat luodaan täysin lajitellun datan pohjalta. Meidän tehtävänä voi olla esimerkiksi ennustaa henkilön vuositulot datan perusteella, johon on kerätty tietoa ihmisten iästä, koulutuksesta, asuinpaikkakunnasta, jne. Tämän jälkeen meidän tulee lajitella data merkaten ylös, minkä suuruinen vuositulo vastaa mitäkin parametreja datassa (ikä, koulutus, asuinpaikkakunta). Datan lajitteluun perustuen ohjatun oppimisen algoritmi laskee luokittelijalle, millä tavalla parametrit vaikuttavat henkilön vuosituloihin. Opettamisen jälkeen luokittelija pystyy ennustamaan tuntemattomalle näytteelle, mihin luokkaan näyte todennäköisesti kuuluu. Ohjattu oppiminen jaetaan usein vielä regressioon ja luokitteluun.

Kun koneoppimisen luokittelijat on luotu ilman datan lajittelua, on kyseessä ohjaamaton oppiminen. Siinä datanäytteille ei ole etukäteen määritetty luokkia, jolloin luokkien jakaminen täytyy tehdä täysin perustuen dataan. Haastavaa tästä ongelmasta tekee se, että emme tiedä tarkasti, mitä kriteerejä luokkajakoon tulisi käyttää. Lisäksi luokkien lukumääräkään ei ole aina etukäteen tiedossa ja sen tulisi olla optimaalinen luokittelun jälkeen. Klusterointi ja dimensionaalisuuden redusointi kuuluvat ohjaamattomaan oppimiseen.

Tässä harjoituksessa käsittelemme ohjatun oppimisen luokittelumenetelmiä. Luokittelussa luokittelualgoritmi erottelee datan annettuihin luokkiin ja luo luokittelijan. Luokittelija ennustaa tuntemattomalle datanäytteelle luokan, johon se todennäköisesti kuuluu. Luokittelijat on luotu opetus-setin pohjalta, joka sisältää datanäytteet etukäteen lajiteltuna. Esimerkiksi meidän tehtävänä on tunnistaa, onko kuvassa lintu vai ei. Nyt opetus-setissä on kahdenlaisia kuvia: kuvia, joissa on lintu sekä kuvia, joissa ei ole lintua (esim. autoja, lentokoneita, koiria). Seuraavaksi opetamme luokittelijan perustuen opetus-settiin ja sen näytteiden luokkiin. Opettamisen jälkeen luokittelija kykenee tunnistamaan tietyllä tarkkuudella, että onko testisetin kuvassa lintu vai ei.

Yksi luokittelun suurimmista haasteista on opetus-setin koko. Sen tulee olla riittävän suuri, jotta uusi datanäyte pystytään tunnistamaan oikein. Jos opetus-setissä on liian vähän näytteitä, algoritmi helposti ylioppii, kuten huomasimme jo aiemmin regression parissa. Tällöin luokittelija ei toimi kovin hyvin tuntemattomalle datalle mutta pystyy tunnistamaan virheettömästi opetus-setin näytteet. Tämä ongelma on erittäin yleinen koneoppimisessa ja se tulee ottaa huomioon joka kerta luodessa luokittelijaa koneoppimisen keinoin.

Luokittelussa luokittelijan suorituskykyä arvioidaan sekaannusmatriisin (engl. confusion matrix) avulla. Se on toteutettu testisetistä ja se kertoo, kuinka moni testisetin näytteistä luokitellaan oikein ja kuinka moni väärin. Sekaannusmatriisissa oikeat luokat on sijoitettu riveille ja luokittelijan ennustamat luokat sarakkeille. Mikäli tietyllä rivillä olevalla luokalla on lukuja eri sarakkeilla kuin sen oma rivinumero, luokittelija on luokitellut näytteitä vääriin luokkiin. Nimi ”sekaannusmatriisi” tuleekin siitä, kun siitä on helppo nähdä, jos luokittelija on tunnistanut luokkia väärin (sekoittanut luokkia). Esimerkki sekaannusmatriisista on esitetty kuvassa 1.

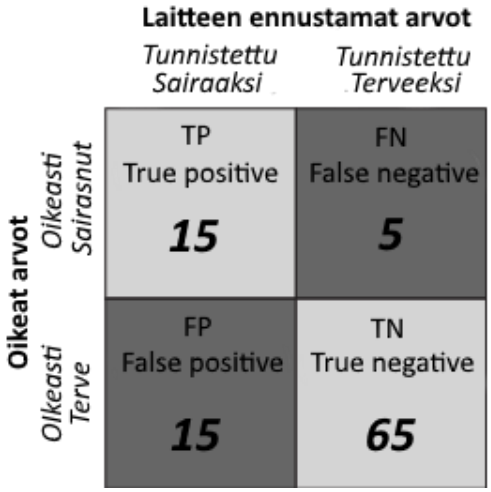
Siihen on myös lisätty sekaannusmatriisin lisäksi luokittelutarkkuudet jokaiselle luokalle sekä luokittelijan kokonaisluokittelutarkkuus. Paras tulos luokittelijalle saavutetaan silloin, kun matriisin päädiagonaalilla ovat kaikki luokitellut näytteet.

		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 1. Esimerkki sekaannusmatriisista yhdeksälle eri luokalle

Luokittelutarkkuus ei kuitenkaan yksistään anna riittävän hyvää kuvaa luokittelijan hyvyydestä. Siksi koneoppimisessa usein käytetään luokittelutarkkuuksien lisäksi esimerkiksi mittoja sensitiivisyys (recall), spesifisyys, positiivisen testin ennustearvo (precision) ja negatiivisen testin ennustearvo. Käydään esimerkin avulla läpi näiden eri mittojen eroavaisuuksia.

100 ihmisen kylässä 20 sairastuu harvinaiseen virukseen, jonka oireet alkavat näkyä vasta kuukauden päästä tartunnan saamisesta. Kylän lääkärin tehtävä on selvittää laitteen avulla tartunnan saaneet henkilöt ennen kuin on liian myöhäistä. Kyseinen laite arvioi, että 30 ihmistä 100:sta on sairastunut virukseen. Kuukauden päästä kuitenkin selviää, että havaituista 30 sairaasta ihmisestä 15 oli todellisuudessa viruksen kantajia (TP) ja 15 oli terveitä (FP). Koska sairastuneita oli yhteensä 20, niin 5 ihmistä tunnistettiin terveiksi, vaikka he olisivat olleet oikeasti sairaita (FN) ja 65 tunnistettiin oikein terveiksi (TN). Kuvassa 2 on muodostettu sekaannusmatriisi esimerkin tapaukselle.



Kuva 2. Sekaannusmatriisi tekstin esimerkille

Sensitiivisyys (Recall): $\frac{TP}{TP+FN} = \frac{15}{15+5} = 75\%$

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

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

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

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

Sensitiivisyyden (recall) arvo riippuu siitä, että kuinka moni kaikista oikeasti sairastuneista henkilöistä tunnistettiin laitteen avulla sairaaksi. Se ei kuitenkaan ota huomioon tapauksia, joissa laite tunnisti terveet henkilöt sairaiksi. Tätä varten voidaan käyttää positiivisen testin ennustearvoa (precision), joka kertoo, kuinka moni tunnistetuista sairaista oli oikeasti sairastunut. Esimerkiksi syöpää tunnistaessa on tärkeää saada havaittua kaikki oikeasti sairastuneet, jolloin sensitiivisyyden arvon tulee olla mahdollisimman iso.

Tehdessäsi harjoituksen omalla tietokoneella, voit asentaa opencv-kirjaston tietokoneellesi seuraavien asennusohjeiden avulla.

Windows:

1. Asenna Microsoft Visual Studio 2017 Community (https://www.visualstudio.com/downloads/)
2. Asenna pip (https://pypi.python.org/pypi/pip)
3. Mene komentoriville ja aja seuraava komento asentaaksesi tarvittavat paketit:
   pip install numpy scipy matplotlib sklearn scikit-image opencv-python

Ubuntu: https://www.pyimagesearch.com/2015/07/20/install-opencv-3-0-and-python-3-4-on-ubuntu/

Mac: https://www.pyimagesearch.com/2015/06/29/install-opencv-3-0-and-python-3-4-on-osx/

Huom1. Suorittaessasi tehtäviä komentoriviltä sinun tulee aluksi siirtyä oikeaan tehtäväkansioon cd komennon avulla. Esimerkiksi cd desktop\tehtävä1 (Windowsilla)

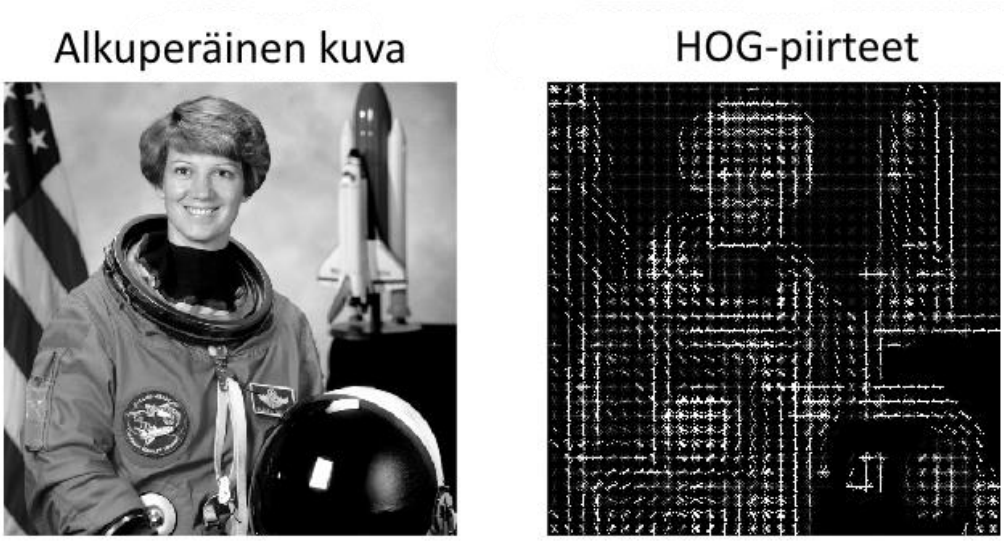
Huom2. Jotta tehtävät toimivat oikein, sinun tulee käyttää pythonin 3:tta versiota (Harjoitusta luodessa on käytetty Windowsia sekä pythonin versiota 3.5.3 ja opencv:n versiota 3.2.0).

Tehtävä 1

Sinun tehtäväsi on luoda erilaisia luokittelijoita käsinkirjoitettujen numeroiden luokitteluun. MNIST datasetti sisältää käsinkirjoitetut numerot 0:stä 9:sään. Yhteensä alkuperäisessä MNIST datasetissä on 70,000 näytettä ja ne on kerätty Yhdysvalloissa Census Bureau:n työntekijöiltä sekä lukioiden opiskelijoilta. MNIST datasetissä näytteet on normalisoitu 20x20 pikselin kuviksi sekä lopuksi keskitetty 28x28 pikselin ruudukolle. Tälle datasetille on onnistuttu luomaan yksinkertaisia lineaarisia luokittelijoita, joiden luokitteluvirhe on tyypillisesti 12% ja 0.56% välillä. Konvoluutioneuroniverkkojen avulla on saavutettu pienimmillään 0.23% luokitteluvirhe kyseisellä datasetillä opetetulle luokittelijalle.

Tässä tehtävässä opetus-setin näytteitä ei syötetä suoraan luokittelualgoritmille, sillä kuvien esikäsittelyllä saavutetaan usein parempi lopputulos luokittelijalle. Tätä erityistä ongelmaa varten datan esikäsittelytekniikaksi on valittu menetelmä nimeltä HOG (Histogram of Oriented Gradients), joka laskee kuville piirrevektorit. HOG menetelmä tuottaa nuolia, jotka kuvaavat tummien pikselien suunnan kuvassa. Nuolia kutsutaan gradienteiksi, jotka siis kertovat mihin suuntaan kuva tummenee ja millä suuruudella.

Kyseisen menetelmän etu on siinä, että sama kuva eri kontrasteilla ei vaikuta lopputulokseen eli todella tummat ja todella vaaleat kuvat johtavat kutakuinkin samanlaisiin HOG piirteisiin. Kuvassa 2 on visualisoitu, miltä alkuperäisestä kuvasta lasketut HOG piirteet näyttävät.



Kuva 3. HOG piirteiden visualisointi

Tässä luokitteluongelmassa on käytetty seuraavia luokittelualgoritmeja: Support vector machines (SVM), k-nearest neighbor (kNN), desicion trees, satunnaismetsät (random forest), adaptiivinen tehostaminen (adaboost), gaussinen naive bayes (GaussianNB), stochastic gradient descent (SGD), linear discriminant analysis (LDA), logistinen regressio ja multi-layer perceptron (MLP). Lisäksi toteutetaan RandomQuessing-luokittelija, joka valitsee luokan satunnaisesti jokaiselle testinäytteelle ja muiden luokittelijoiden tulee toimia ainakin tätä paremmin.

Luodaksesi eri algoritmeilla toteutettuja luokittelijoita, aja python tiedosto **generateClassifiers.py** komentoriviltä ja tarkastele luotuja sekaannusmatriiseja. Tallenna komentoriville ilmestyvät sekaannusmatriisit tiedostoon alla olevalla komennolla. Luokittelijoiden luominen kestää useamman minuutin ajan ja MLP-luokittelija opetetaan iteratiivisesti, mikä tallentuu myös luokitteluraporttiin.

```
python generateClassifiers.py > confusionmatrices.txt
```

Tarkastelemalla tekstitiedostoa löydät luokittelijoille sekaannusmatriisien lisäksi luokittelutarkkuudet, jotka on ilmoitettu sensitiivisyyden (recall), positiivisen testin ennustearvon (precision) ja F1-scoren avulla. F1-score yhdistää precision ja recall arvon seuraavalla kaavalla:

$$\mathrm{F1\_score} = 2 \cdot (\mathrm{precision} \cdot \mathrm{recall}) / (\mathrm{precision} + \mathrm{recall})$$

KYSYMYS1: Arvioi kaikkien luokittelijoiden luokitteluraporttien perusteella, mitkä kolme luoduista luokittelijoista ovat parhaita tämän ongelman ratkaisemiseksi F1-scoren perusteella?

KYSYMYS2: Kun olet löytänyt parhaat luokittelijat, valitse niistä yksi ja tarkastele sen sekaannusmatriisia sekä luokitteluraporttia ja selvitä, mikä luokista oli helpoin tunnistaa ja mikä vaikein?

KYSYMYS3: Mihin käsinkirjoitettuihin numeroihin vaikeiten tunnistettava luokka oli yleisimmin sekoitettu?

Tämän jälkeen testaa komentoriviltä parasta luokittelijaa testikuville python tiedostolla **performClassifier.py** seuraavan komennon mukaisesti

```
python performClassifier.py –i=photo1.jpg –c=model11randomquessing.pkl
```

,missä -i viittaa testikuvaan ja -c viittaa luokittelijaan.

Näet mustavalkoistamisen lopputuloksen ikkunassa nimeltä *”The thresholded image”* ja luokittelun lopputuloksen kuvanäytteille ikkunassa nimeltä ”*The output image*”. Kooditiedosto piirtää suorakulmion yksittäisen käsinkirjoitetun numeron ympärille ja tunnistaa numeron luomasi luokittelijan avulla.

Kuvissa photo1.jpg, photo2.jpg ja photo3.jpg on 8 testinäytettä jokaiselle käsinkirjoitetulle numerolle (0-9). Liitä raporttiin kuva photo1.jpg:n näytteiden luokittelusta. Luo lopuksi itsetehty sekaannusmatriisi kuvien photo1.jpg, photo2.jpg ja photo3.jpg testinäytteiden luokittelun perusteella ja liitä sekaannusmatriisi raporttiin.

Voit myös halutessasi tehdä itse testinäytekuvia käsinkirjoitetuille numeroille. Kuvien taustan tulee olla yksivärinen ja siinä ei saa olla varjoja. Sinun täytyy myös säätää mustavalkoistamisen raja-arvo (engl. threshold) oikeaksi. Käsinkirjoitettujen numeroiden tulee olla riittävän paksuja sekä selvästi erillään toisistaan ja paperin reunoista.

Yhteenvetona tehtävän 1 palautuksista: Palauta tiedosto confusionmatrices.txt ja raportti, joka sisältää vastaukset kysymyksiin KYSYMYS1, KYSYMYS2 ja KYSYMYS3, kuvan parhaan luokittelijan luokittelusta kuvalle photo1.jpg sekä itsetehdyn sekaannusmatriisin kuvien photo1.jpg, photo2.jpg sekä photo3.jpg testinäytteiden luokittelutulokselle parhaalla luokittelijalla.

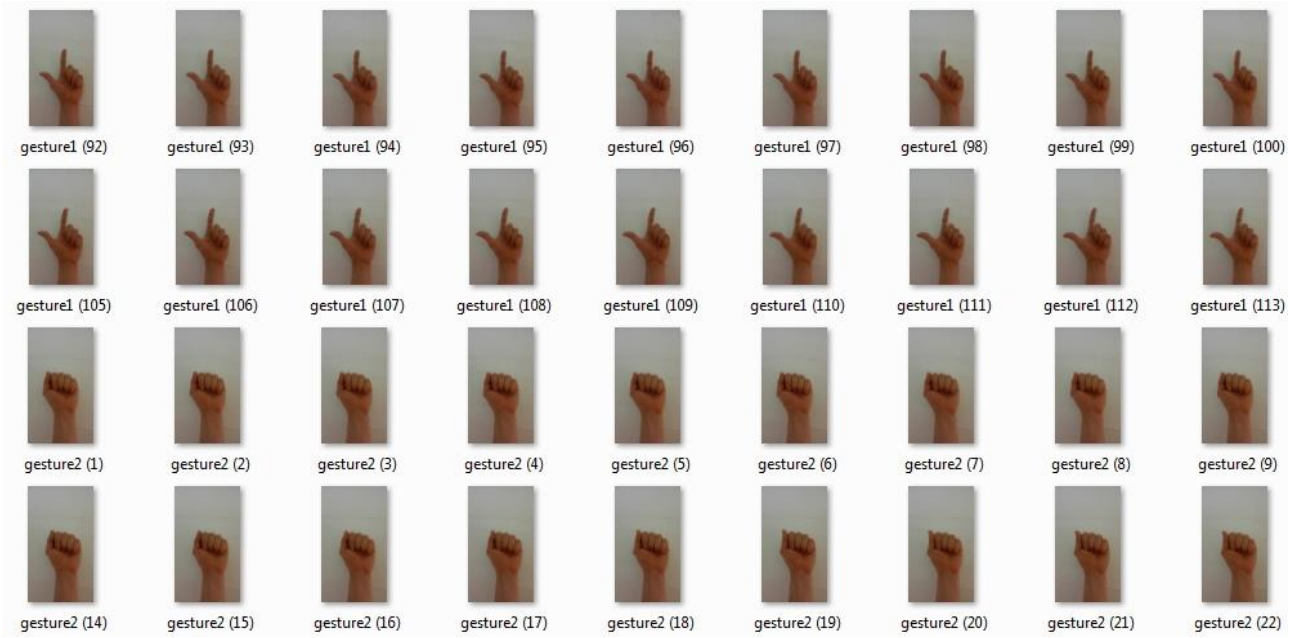
Tehtävä 2

Sinun tehtäväsi on luoda datasetti, joka sisältää erilaisia käsimerkkejä ja opettaa luokittelija luodun datasetin pohjalta, joka tunnistaa käsimerkin kuvasta. Aivan aluksi on päätettävä, montako erilaista käsimerkkiä halutaan datasetissä olevan. Mitä suurempi on luokkien määrä, sitä vaikeampi on tunnistaa luokka testisetinkuvista oikein. Lisäksi samanlaiset käsimerkit sekoitetaan hyvin helposti keskenään, joten valitse mahdollisimman erilaisia käsimerkkejä. Voit käyttää esimerkiksi alla esitettyjä kuutta käsimerkkiä datasettisi luokkina.

Kuva 4. Esimerkkejä erilaisista luokista käsimerkeille valkoista taustaa vasten

Ota arviolta 100 kuvaa jokaista käsimerkkiä kohden. Huomaa, että luokissa olevien näytteiden määrä tulee olla kutakuinkin balanssissa toisiinsa nähden. Kuvan tulee myös sisältää käsimerkin lisäksi osan paljaasta ranteesta kuvan 4 tyyliin. Yritä lisäksi välttää valaistuksesta aiheutuvia varjoja. Älypuhelimien burstimage sovellukset tai vastaavat helpottavat huomattavasti kuvien ottamista. Valitse kuville yksivärinen tausta, joka poikkeaa selvästi ihonväristä (esimerkiksi valkoinen tai musta).

Kuvien ottamisen jälkeen siirrä ne kansioon nimeltä **inputimages** ja nimeä kuvat luokkien mukaan (lajittelu). 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 sille esimerkiksi nimi gesture1. Kopioi myös jokaisesta luokasta yksi kuva ja liitä kuvat kansioon **testimages**. Kuvassa 5 on esitetty havainnollistava kuva osasta kansion inputimages sisältöä.



Kuva 5. Havainnollistava kuva osasta kansion inputimages sisältöä

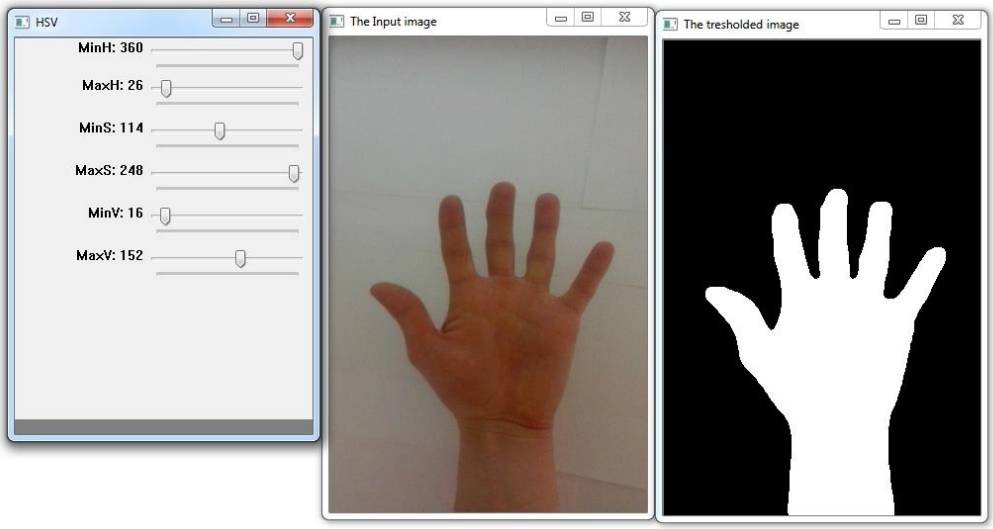
Seuraavaksi esikäsitellään oikein nimetyt käsimerkkikuvat muuntamalla ne mustavalkoisiksi .bmp kuviksi. Tämän vaiheen suorittamiseksi käytä python tiedostoa **testTransformation.py** testataksesi, miltä muunnos mustavalkoiseksi kuvaksi näyttää. Python tiedosto valitsee ensimmäisen kuvan kansiosta testimages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

```
python testTransformation.py
```

Sinun täytyy lisäksi säätää sopivalle välille HSV-värimallin arvot tunnistamaan ihonvärisi. Tämä onnistuu säätämällä säätöikkunan arvoja, kun ajat testTransformation.py tiedoston. MinH ja MaxH viittaavat värisävyn (hue) minimi- ja maksimiarvoihin, MinS ja MaxS viittaavat saturaation (saturation) minimi- ja maksimiarvoihin ja MinV ja MaxV viittaavat valoisuuden (value or brightness) minimi- ja maksimiarvoihin. Näet muunnoksen lopputuloksen ikkunassa nimeltä *”The thresholded image”*.

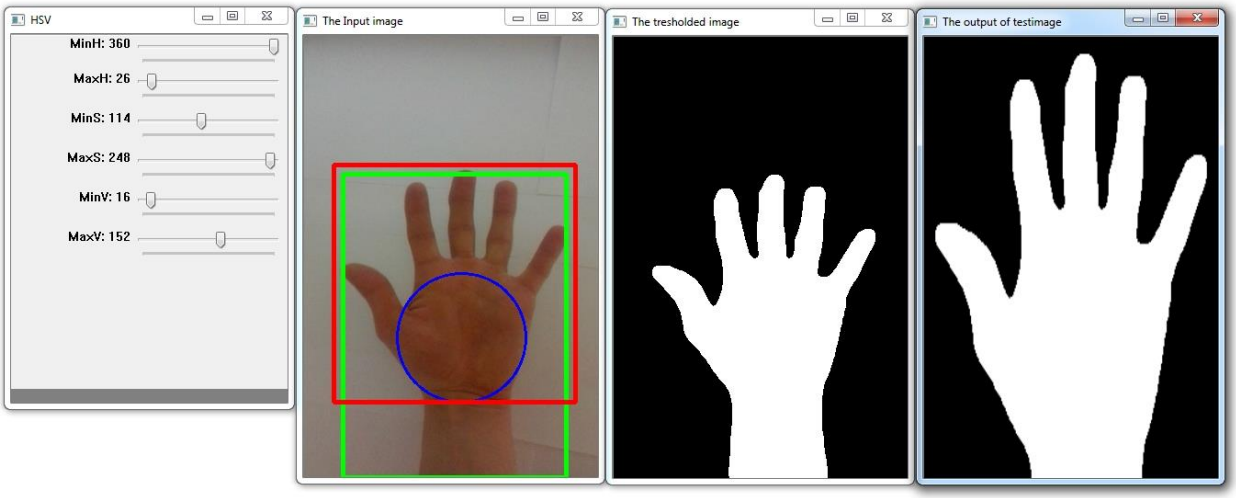
Yleisesti ottaen ihmisen ihonvärin rajojen määrittäminen mille tahansa värimallille on erittäin vaikea tehtävä muuttuvista valaistusolosuhteista johtuen. Voimme arvioida karkeasti ihonvärille, että värisävyn vaihteluväli on 330-30 (värisävyn arvo on määritelty astelukuna ja se saa arvoja väliltä 0-360), mutta saturaation ja valaisuuden vaihteluvälit riippuvat täysin kuvasta (molemmat saavat arvoja 0-255 väliltä).

Selvitä arvoja säätämällä maksimi ja minimiarvot värisävylle, saturaatiolle ja intensiteetille. Kun olet löytänyt sopivat arvot värimallille, ota ne ylös ja tallenna ne tiedostoon testTransformation.py kohtien cv2.createTrackbar() kolmansiksi argumenteiksi. Kuvassa 6 on esitetty oikein säädetyt värimallin vaihteluvälit alkuperäiselle kuvalle. Auenneet ikkunat saa suljettua valitsemalla kuva ja painamalla esc-näppäintä.



Kuva 6. Oikein säädetyt HSV-värimallin vaihteluvälit alkuperäiselle kuvalle

Kuvan binäärisoinnin lisäksi voimme leikata ranteen oikeasta kohtaa ja keskittää kuva. Tätä välivaihetta varten poista tiedostosta testTransformation.py rivit, joissa on kommenttimerkit """ (rivit 57 ja 89). Tämän jälkeen näet alkuperäisessä kuvassa rajattuja alueita, jossa vihreässä suorakulmiossa käsi on erotettu taustasta ja sininen ympyrä on sovitettu kämmenen sisään, kun ajat tiedoston uudestaan komentoriviltä. Määrittämällä ympyrän alareunan kohta lopullisen kuvan alareunaksi voimme jättää kuvasta ranteen pois oikeasta kohtaa, jota kuvaa punainen alue. Punaisen alueen yläreunaan ja sivuihin on myös lisätty muutama pikseli tyhjää tilaa. Kuvassa 7 on esitetty onnistunut muunnos alkuperäiselle kuvalle.



Kuva 7. Onnistunut muunnos testikuvalle

Kun olet tyytyväinen lopputulokseen aja python tiedosto **runTransformation.py**, joka käsittelee kaikki inputimages kansion kuvat. Muista muuttaa oikeat HSV-väriarvojen vaihteluvälit myös runTransformation.py tiedostoon ennen komennon suorittamista. Muunnetut kuvat tallentuvat kansioon nimeltä **outputimages**. Kyseinen välivaihe kestää muutaman minuutin ajan.

```
python runTransformation.py
```

Lopputuloksena saat mustavalkoisia käsimerkkiin keskitettyjä .bmp kuvia. Silmäile outputfolder kansion kuvat läpi ja tarkasta, että kuvat näyttävät oikeilta ja poista tarvittaessa epäonnistuneet muunnokset.

Nyt voimme luoda datasetin muunnetuille kuville ajamalla python tiedosto makeDataset.py ja lopputuloksena saamme datasetin handdetectiondataset.pkl , joka sisältää muunnetut kuvat ja niiden luokat.

```
python makeDataset.py
```

Datasetin tekemisen jälkeen on vuorossa luokittelijan luominen. Ennen tätä laskemme vielä piirteet mustavalkoisista kuvista. Eräs tehokas menetelmä tämän tyyppisen ongelman ratkaisemiseksi 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 moment invariantteja muodon kuvailijoina, sillä ne ovat riippumattomia kuvan kääntämisestä, koon muuttamisesta tai muodon sijainnista kuvassa. Opencv:n funktio cv2.HuMoments() käsittelee kuvissa esiintyviä muotoja tilastollisesti ja laskee kuville 7 momenttia. Näistä seitsemästä numeroarvosta koostuvat piirrevektorit kuvaavat kuvissa esiintyviä muotoja ja niitä käytetään käsimerkkien tunnistamiseen mustavalkoisten kuvien sijasta. Kuvien luokittelemiseen piirrevektorien avulla voimme käyttää esimerkiksi sklearn-kirjaston SVM algoritmia. Aja python tiedosto **generateClassifier.py** toteuttaaksesi luokittelija eri käsimerkkien tunnistamiseen.

```
python generateClassifier.py > confusionmatrix.txt
```

Lopputuloksena saat luokittelijan nimeltä handdetectionmodel1svm.pkl ja tiedostossa confusionmatrix.txt sekaannusmatriisin testisetille, joka on erotettu alkuperäisestä datasetistä.

KYSYMYS1: Arvioi sekaannusmatriisin perusteella, mitkä käsimerkeistä ovat vaikein tunnistaa ja mitkä helpoin?

KYSYMYS2: Mihin käsimerkkeihin vaikeiten tunnistettava luokka oli yleisimmin sekoitettu?

Nyt voit testata luokittelijaa ottamillesi testikuville ajamalla python tiedosto **classifyImage.py**. Kooditiedosto piirtää suorakulmion käsimerkin ympärille ja tunnistaa käsimerkin luomasi luokittelijan avulla. Sinun täytyy säätää myös tähän koodiin aiemmin selvittämäsi HSV-väriarvojen vaihteluvälit. Liitä raporttiin kuvat luokittelun lopputuloksesta jokaiselle testikuvalle.

```
python classifyImage.py –c=handdetectionmodel1svm.pkl
```

Tiedosto ottaa argumenttina –c, joka viittaa luokittelijaan.

Yhteenvetona tehtävän 2 palautuksista: Palauta tiedostot confusionmatrix.txt, handdetectiondataset.pkl, handdetectionmodel1svm.pkl ja raportti, joka sisältää vastaukset kysymyksiin KYSYMYS1 ja KYSYMYS2 sekä yhden testikuvan jokaisesta luokasta luokittelun lopputuloksesta. Kerro myös raportissa käyttämäsi HSV-värimallin vaihteluvälit sekä mikä luokan nimi viittaa mihinkin käsimerkkiin (esim. gesture1 viittaa käsimerkkiin open hand jne.).

Palauta

Palauta pyydetyt tiedostot ja raportit pakattuna tiedostona (.zip tai .rar), jossa on kaksi erillistä kansiota (esim. tehtävä1 ja tehtävä2) molemmille tehtäville Optiman palautuslaatikkoon Harjoitus 3 **16.4.2018 klo 23:59** mennessä. Tästä harjoituksesta on mahdollisuus tienata 5 pistettä (2.5p + 2.5p).