OS2-projektidokumentaatio
Regressiomallin visualisointi

Esa Palosaari
618573, TiK, vk 2017

23.4.2019

1 Yleiskuvaus

Tehtävänannon ja suunnitelman mukaisesti olen rakentanut ohjelman, joka sovittaa regressiomallin datajoukkoon ja visualisoi tulokset. Kyseessä on vaikean tason työ. Tämä tarkoittaa annettujen kriteerien mukaisesti, että käyttäjä voi ladata datan valitsemastaan tiedostosta käyttäen kahta tiedostomuotoa (CSV ja XML). Ohjelman käsittelemät tiedostot sisältävät (x, y) -koordinaattipareja desimaalilukuina niin, että desimaalierottimena on piste. Ohjelmaa voi pyytää sovittamaan joko suoran tai paraabelin muotoisen regressiomallin. Ohjelma piirtää kuvaajan alkuperäisistä datapisteistä ja niihin sovitetusta mallista. Käyttäjä pystyy säätämään kuvaajan asetuksia. Toteutus on laajennettavissa eli uusien ominaisuuksien lisääminen ei ole vaikeaa tai vaadi koko ohjelman kirjoittamista uudelleen. Erityisesti uusia uusia tiedostomuotoja, regressiomalleja ja käyttöliittymiä on helppo lisätä. Tiedostonlukijasta ja regressiomallista on abstraktit luokat, joista voi tehdä uusia konkreettisia aliluokkia. Abstraktit luokat määrittelevät muiden rajapinnat ohjelman muille osille siten, että esimerkiksi uuden regressiomallityypin regressikäyrien piirtämistä tai kaavan kirjoittamista varten ei tarvitse muuttaa graafisia luokkia ollenkaan.

2 Käyttöohje

2.1 Ohjelman "kääntäminen"komentoriviltä ajettavaksi

Ohjelman komentorivikäyttöliittymän JAR-tiedosto voidaan tehdä esimerkiksi Eclipsessä:

- Klikkaa oikealla näppäimellä pakettia `regressionViz`
- Valitse "Export..."
- Avaa Java-kansio
- Valitse "JAR file"ja klikkaa "Next"

- Valitse JAR-tiedostolle haluamasi paikka ja nimi
- Klikkaa "Finish".

Javan versio 8 olisi oltava asennettu. Eclipsessä tulisi olla Scalan versio 2.12.3. Paketin `regressionViz` Build Path:ssä tulisi olla scala-xml_2.12-1.0.x.jar -tiedosto (se on liitetty mukaan gitiin, libs-hakemistoon). Ohjelman ajamista varten tulisi komentorivin polussa olla Scalan versio 2.12.3 (ohjeet version asentamiseksi esimerkiksi Ubuntulle: `https://gist.github.com/malzzz/15639fc36d069490b7709be2f3a4d522`). Käyttämilläni yliopiston Ubuntu-koneilla Scalan versio komentorivillä oli 2.11.8, Eclipsessä Scala oli 2.12.2. Scalan päivittäminen ei onnistu ainakaan helposti ilman root-oikeuksia eikä ohjelma näytä toimivan oikein Scalan versiolla 2.11. Eclipsessä komentorivitoimintoja voi toteuttaa seuraavalla tavalla:

- Klikkaa CLIApp.scalaa oikealla napilla ja valitse `Run As`
- Valitse `Run Configurations...`
- Kirjoita kohtaan `Main class: regressionViz.CLIApp`
- Välilehdelle `Arguments` kohtaan `Program arguments:` voi kirjoittaa ohjelman optioita alla olevien ohjeiden mukaan.
- Kohtaan `Program arguments:` voi myös kirjoittaa `${string_prompt}`, jolloin Eclipse antaa ikkunan ohjelman optioiden kirjoittamiseen joka kerta kun ohjelman käynnistää.

2.2 Ohjelman käyttäminen komentoriviltä

Komentorivin kautta ohjelma käynnistetään komennolla:

```
scala -cp "${CLASSPATH}:${SCALA_HOME}/lib/scala-library.jar:
NameOfTheJARExported.jar" regressionViz.CLIApp [options]
```

`NameOfTheJARExported.jar` on aiemmin valitsemasi nimi JAR-tiedostolle. Ohjelman ohjeet saa näkyviin kun sen käynnistää ilman mitään optioita:

```
scala -cp "${CLASSPATH}:${SCALA_HOME}/lib/scala-library.jar:
NameOfTheJARExported.jar" regressionViz.CLIApp
```

Ohjeet kertovat mitä ja miten parametreja ohjelmalle voi käynnistysoptioiden avulla antaa. Tässä käynnistysoptiot ja niiden selitykset:

- `--d` Datatiedoston polku ja tiedostonnimi. Vaaditaan ohjelman toimimiseksi. Nimen on päätyttävä joko .csv tai .xml. Kohdassa 6 kuvataan datatiedostojen hyväksyttävät muodot.
- `--o` Polku ja nimi tallennettavalle PNG-kuvatiedostolle. Vaaditaan.

- `--varx` Ensimmäisen, selittävän muuttujan nimi. Tämä optio vaaditaan vain XML-tiedoston käytön yhteydessä, jolloin sen on oltava sama kuin XML-tiedostossa olevan muuttujan nimi. CSV-tiedostojen kanssa se on valinnainen. Nimeä käytetään visualisaatiossa.
- `--vary` Toisen, selitettävän muttujan nimi. Vaaditaan XML-tiedoston käytön yhteydessä, jolloin se on oltava sama kuin XML-tiedostossa olevan selitettävän muuttujan nimi.
- `--sizex` Kuvan vaakasuora koko pikseleinä. Jos optiota ei anneta, käyttää ohjelma oletusarvoa on 500.
- `--sizey` Kuvan pystysuora koko pikseleinä. Jos optiota ei anneta, käyttää ohjelma oletusarvoa on 500.
- `--xmax` Vaakasuoran akselin suurin arvo. Oletuksena on suurin ensimmäisen muuttujan arvo + yksi akselilla käytettävän suuruusluokan yksikkö.
- `--xmin` Vaakasuoran akselin pienin arvo. Oletuksena on pienin ensimmäisen muuttujan arvo - yksi akselilla käytettävän suuruusluokan yksikkö.
- `--ymax` Pystysuoran akselin suurin arvo. Oletuksena on suurin toisen muuttujan tai regressiokäyrän arvo suurimmalla ensimmäisen muuttujan arvolla + yksi akselilla käytettävän suuruusluokan yksikkö.
- `--ymin` Pystysuoran akselin pienin arvo. Oletuksena on pienin toisen muuttujan tai regressiokäyrän arvo pienimmällä ensimmäisen muuttujan arvolla - yksi akselilla käytettävän suuruusluokan yksikkö.
- `--pR` Datapisteiden RBG-värin Red-arvo (0-255). Oletuksena 255.
- `--pB` Datapisteiden RBG-värin Blue-arvo (0-255). Oletuksena 0.
- `--pG` Datapisteiden RBG-värin Green-arvo (0-255). Oletuksena 0.
- `--cR` Regressiokäyrän RBG-värin Red-arvo (0-255). Oletuksena 0.
- `--cB` Regressiokäyrän RBG-värin Blue-arvo (0-255). Oletuksena 255.
- `--cG` Regressiokäyrän RBG-värin Green-arvo (0-255). Oletuksena 0.

Ohjelma osaa tällä hetkellä skaalata vaaka- ja pystyakselin asteikkoja ylöspäin kertaluokasta 10^0, mutta ei alaspäin. Jos kaikki muuttujan arvot ovat kertaluokkaa 10^{-1} tai pienempiä, datatiedostossa olevat arvot tulee korottaa muualla siten että muuttujan kertaluokka on vähintään 10^0.

Tällä hetkellä ohjelma käyttää sekä pysty- että vaaka-akselin mitta-asteikkona samaa pikselimäärää. Eli jos vaaka-akselilla etäisyyteen 1.0 mahtuu 10 pikseliä, myös pysty- akselilla etäisyyteen 1.0 mahtuu 10 pikseliä. Tämä auttanee tulkitsemaan kuvaajia monissa kohdissa, mutta toisissa taas vaikeuttaa. Se myös rajoittaa, miten paljon akseleilta voi jättää piirtämättä, mikä voi johtaa rumiin kuviin.

3 Ohjelman rakenne

3.1 Ohjelman luokat ja niiden tehtävät

Ohjelman luokat jakautuvat tehtävien mukaan datan lukemiseen tiedostosta (`DataReader, CSVReader, XMLReader`), datan liittyvään informaation säilyttämiseen (`Data`), regressiomallin sovittamiseen (`Model, OLSModel, QuadModel`) ja sovittamisessa käytettäviin matriisioperaatiohin (`ArrayMatrix`), visualisoinnin piirtä-miseen (`Drawing`, ja ohjelman tehtävien suorittamiseen kaikkien luokkien avulla ja käyttäjän käskyjen mukaisesti (`Engine, CLIApp`). Yksikkötestejä varten on oma luokkansa.

Luokkarakenteessa on ollut keskeistä jatkokehityksen ja lisäominaisuuksien lisäämisen helppous. Sen vuoksi datan lukemisesta ja regressiomallin sovittamisesta on abstraktit luokat, joiden rajapinta-ominaisuuksia toteuttavia uusia luokkia on helppo lisätä ohjelmaan. Samoin käyttöliittymiä voi lisätä ja muuttaa ilman että koko ohjelmaa pitää kirjoittaa uusiksi, koska usea käyttöliittymä voi säilyttää olioita ja käyttää niiden toimintoja `Engine`-luokassa. Se voisi mahdollistaa myös useamman samanaikaisen näkymän ja käyttäjän ohjelmalle. Jos ei olisi ajatusta muuttaa tai lisätä ominaisuuksia ohjelmaan, voisi päästä vähemmällä kirjoittamisella poistamalla abstraktit luokat ja ottamalla toimintalogiikka ja olioiden säilytys osaksi käyttöliittymäluokkaa.

3.2 Luokkien keskeiset metodit

- `DataReader readFile(filename: String): Option[Data]` Kaikki DataReaderin perivät luokat (`CSVReader, XMLReader`) toteuttavat tämän metodin, jolle annetaan tiedoston nimi, ja joka palauttaa Data-luokan olion Optioniin käärittynä. Metodi siis lukee annetusta tiedostosta numeroarvot ja tallentaa ne Data-olioon.
- `Data initializeDataset(newPoints, newName, newVarNames)` Metodin avulla voi vaihtaa Data-olion datapisteitä, nimeä ja muuttujien nimiä. Näillä Data-olion muuttujilla on oletusarvot, eikä kaikkia muuttujia tarvitse muuttaa samalla kertaa.
- `Data getPoints, getVarNames` palauttavat Optioniin käärittynä Double-muotoiset datapisteet kaksiulotteisessa taulukossa ja Optioniin kääritty ArrayBufferin muttujien nimiä.
- `Model fitData: Unit` sovittaa Model-luokan perivissä luokissa (`OLSModel, QuadModel`) regressiomallin konstruktorissa annetun Data-olion datapisteisiin. Sovitetun mallin, yhtälön ja yhtälön arvon voi saada omilla getterimetodeilla.
- `ArrayMatrix multiplyMatrixAndVector` kertoo sopivan ulotteisen matriisin vektorilla vasemmalta ja palauttaa Double-tyyppisen vektorin.
- `ArrayMatrix multiplyMatrices` kertoo kaksi sopivan ulotteista matriisia keskenään ja palauttaa Double-tyyppisen matriisin.

- `ArrayMatrix invertMatrix` kääntää ja palauttaa neliömatriisin käänteismatriisin. Metodi olettaa että matriisi on kääntyvä.
- `Engine readData` lukee parametrina annetun datatiedoston tiedot käyttäen tiedoston päätteen mukaista lukijaa ja tallentaa osoittimen Data-olioon Engine-olion kokoelmamuuttujaan. Metodin parametreina annetut muuttujien nimet tallennetaan Data-olioihin.
- `Engine fitModel` sovittaa parametrina annettuun Data-olioon parametrina annetun tyyppisen regressiomallin. Viittaus Model-olioon lisätään Engine-olion kokoelmamuuttujaan.
- `Engine drawImage` luo annettujen parametrien mukaisen Drawing-olion ja lisää viittauksen siihen Engine-olion kokelmamuuttujaan. Ainoa vaadittu parametri on viittaus Model-olioon. Muut parametrit voivat olla None-tyyppejä.
- `Engine saveImage` tallentaa parametrina annetun Drawing-olion mukaisen kuvan PNG-formaatissa parametrina annetun nimiseen tiedostoon.
- `CLIApp main` parsii ohjelman kutsussa annetut optiot ja kutsuu Engine-olion metodeja annettujen parametrien ja arvojen mukaan.

4 Algoritmit

Pienimmmän neliösumman menetelmässä tavallisen regressioyhtälön ratkaisuun käytettiin normaaliyhtälön kaavaa (Ordinary Least Squares, Wikipedia). Kaavalla laskemiseen kuului matriisien ja vektorien kertolaskuja ja transpooseja. Vaikein laskentatehtävä oli käänteismatriisin laskeminen, joka tapahtui Gauss-Jordan -menetelmällä (ks. Viitteet).

Regressioyhtälön ja käänteismatriisin olisi voinut laskea muilla menetelmillä. Käytin tässä yksinkertaisimpia oppimisen ja ajankäytön vuoksi.

5 Tietorakenteet

Ohjelmassa käytettävät tietorakenteet ovat joko yksi- tai kaksiulotteisia Arrayta ja ArrayBuffereita. Yksi- ja kaksiulotteiset Arrayt ovat datapisteiden varastointia ja käsittelyä varten silloin kun niiden koko tiedetään. Arrayt ovat nopeita ja yleisesti käytettyjä rakenteita varsinkin vektoreiden ja matriisien esittämiseen ja manipulointiin. ArrayBuffereita käytän silloin kun kokoelman koko muuttuu tai saattaa muuttua. Arrayn kokoa on hankalampi ja tehottomampi muuttaa kuin ArrayBufferin. Käytän lähinnä muuttuvatilaisia rakenteita, koska en vielä hallitse funktionaalista tyyliä tarpeeksi hyvin. Javamaisempi tyyli (muuttuvatilaisia, mutta mahdollisesti yksityisiä rakenteita, joihin pääsee gettereillä ja settereillä) tuntuu vielä helpommalta eikä liian ongelmalliselta.

Olisi varmaan voinut käyttää funktionaalisempaa tapaa muuttumattomine rakenteineen. Olisi myös voinut pärjätä yksiulotteisilla Arraylla, joiden rivit olisi voinut laskea moduloa käyttäen, mutta kaksiulotteisten taulukoiden hieman korkeampi abstraktitaso auttoi nyt ajattelemaan paremmin. Yksiulotteisilla taulukoilla voinee tiristää jotain tehokkuutta jatkossa jos haluaa.

6 Tiedostot ja verkossa oleva tieto

Ohjelmalla käyttäjä voi sovittaa kahden muuttujan regressiomallin ja tallentaa datan ja regressiomallin visualisaation PNG-tiedostoon. Muuttujien arvot annetaan tiedostoissa, jotka ovat joko CSV (Comma Separated Variables) tai XML (Extensible Markup Language) -muotoisia. CSV-tiedostoissa numeroiden desimaalierottimena on piste ja numerot erotetaan toisistaan pilkulla. Eri muuttujien arvot ovat omissa sarakkeissa. Numerot voivat olla tiedostossa esimerkiksi seuraavalla tavalla:

```
1.2, -3.43
3.123, 2.3423
4, 3.0
```

XML-tiedostoissa muttujien `PE` ja `MarketCap` arvot voidaan esittää esimerkiksi seuraavalla tavalla:

```
    <?xml version="1.0"?>
<symbols>
 <symbol ticker="Cisco" >
    <PE>21.65</PE>
    <MarketCap>271.18</MarketCap>
</symbol>
<symbol ticker="Sandisk" >
    <PE>23.71</PE>
    <MarketCap>15.53873</MarketCap>
</symbol>
<symbol ticker="Oracle" >
    <PE>17.87</PE>
    <MarketCap>225.61</MarketCap>
</symbol>
<symbol ticker="Red Hat" >
    <PE>65.41</PE>
    <MarketCap>32.24</MarketCap>
</symbol>
</symbols>
```

Datatiedostoissa ei saa olla puuttuvia tietoja (eli tyhjiä kohtia) tai datana muuta kuin numeroita. CSV-tiedoston ensimmäisen rivin mahdolliset muuttu-

jien nimet on poistettava ennen tiedoston lukemista ohjelmalla. Ohjelma tunnistaa datatiedoston muodon sen päätteestä (.csv tai .xml). Jos tiedoston lukemisessa on ongelmia, kannattaa tarkistaa ettei datatiedostoissa ole mukana ylimääräisiä "näkymättömiä"merkkejä kuten tabulaattoreita. Tämä vaikutti aiheuttavan ongelmia CSV-muotoisessa testitiedostossa yliopiston koneella.

7 Testaus

Ohjelmaa testattiin heti alusta yksikkötestein. Yksikkötestit kirjoitettiin ennen varsinaista ohjelmakoodia, ja testien tuli ensin ilmoittaa virheestä. `Drawing`-luokan joitain metodeita testattiin yksikkötestein, mutta pääasiallinen testaustapa oli vain katsoa kuvia joita luokka tuotti ja verrata niitä haluttuihin kuvaajiin. Komentorivityökalua ja Engine- luokkaa testattiin myös lähinnä kokeilemalla niitä.

Suunnitelmassa mainituista testeistä ohjelma läpäisee sen, että regressioyhtälö palauttaa oikeat arvot virherajojen puitteissa kun syötteenä on ennalta tunnettuja datoja. En ole keskittynyt testeissä algoritmin virhealttiuteen enkä siihen, miten täydellinen lineaarinen riippuvuus vaikuttaa. Datan lukeminen näyttää toimivan suunnitellusti, mutta yliopiston koneessa oli jokin yllättävä ongelma, minkä syy ei täysin selvinnyt. Puuttuvien tietojen käsittely ei näyttänyt toimivan niin kuin ajattelin, minkä takia jätin sen ajanpuutteen vuoksi lopulta vain pois.

Komentorivikäyttöliittymän omilla silmillä testaamiset ohjelma näyttää läpäisevän.

8 Ohjelman tunnetut puutteet ja viat

- Akseliyksiköt ovat samat mikä voi olla ongelma, kun kertaluokat ovat erit tai pitää saada
- pystyakselin yksiköt eivät lähde nollasta isoilla numeroilla
- pysty- ja vaaka-akselin automaattinen sovitus tuottaa välillä liialta näyttävää ylimääräistä tilaa
- puuttuvan tiedon käsittely eli sellaisten rivien poisto, joista toisesta muuttujasta puuttui tieto, ei toiminut
- skaalaus alaspäin ei toimi, pienin numero akseleilla on yksi. Tämä liittynee kokonaislukuihin.
- numeroiden mahtuminen akseleille, kun mennään suureen määrään numeroita per akseli

9 3 parasta ja 3 heikointa kohtaa

9.1 3 parasta kohtaa

- Regressiomallin sovittamisen toimivuus. Regressiomallin estimointi normaaliyhtälöllä vaikuttaa toimivan täsmälleen kuten pitääkin. Tuntui melkein taikuudelta, että ohjelma osasi sovittaa oikean toisen asteen käyrän vain viiden datapisteen perusteella. Normaaliyhtälöä on myös helppo laajentaa korkeamman asteen polynomeihin. Ohjelmaan ei olisi kovin vaikeata lisätä ominaisuutta, jossa sille annettaisiin vain luku nollasta n:ään regressioyhtälön asteesta ja ohjelma sovittaisi halutun asteen polynomin.

- Kuvaajan oikeellisuus. Vaikka kuvaajan visuaalisten apukeinojen, erityisesti akseleiden lukujen automatiikassa on parannettavaa, kuvaajat näyttävät olevan "matemaattisesti"tosia. Toisin sanoen, pisteet, origosta lähtevät apuakselit ja regressiokäyrä ovat teknisesti oikeissa suhteissa toisiinsa (mittausvirheen ja pikseleiden rajoissa).

- Ohjelman rakenne ja laajennettavuus. Ohjelman rakenne on mielestäni selkeä: Luokat vastaavat kukin yhdestä pääasiasta ja ne sisältävät tehtävään tarvittavan datan ja menetelmät. Luokkien keskittyminen omaan asiaansa ja abstraktien luokkien käyttö tekevät ohjelman laajentamisesta helppoa. Uuden käyttöliittymän lisääminen ei vaadi koskemista muuhun ohjelmaan. Erilaisia regressiomalleja tai luettavia tiedostomuotoja voi lisätä mielivaltaisen määrän kirjoittamalla niille abstraktin luokan mukaiset toteutukset ja niitä vastaavat kutsut Engine-luokkaan.

9.2 3 heikointa kohtaa

- Vaaka- ja pystyakselin koon automaattinen määritys voi tuottaa epäesteettisiä eli rumia lopputuloksia. Automatiikkaa voisi hienosäätää paremmaksi näpräilemällä lisää pikselien sekä akseleiden ja yksiköiden skaalojen kanssa. Suuren osan epäesteettisyyksistä voi korjata ohjelmalle annetuilla parametreilla, mutta parametreilla ei voi tällä hetkellä muuttaa kaikkia kohtia. Esimerkiksi kummallekin akselille saman suuruinen numeerinen etäisyys tarkoittaa saman suuruista pikselimäärää. Omat yksiköt eri akseleille tuottavat uusia ongelmia, mutta niitä voisi ratkoa ja antaa käyttäjälle mahdollisuuksia säätää akseleiden yksiköiden pikselimäärää.

- Käyttöliittymä. Tätä kirjoittaessa komentorivityökalukäyttöliittymä ei ole kovin kummoinen. Se sopii parhaiten tilanteisiin tai työ, joissa käytetään muita vastaavia komentorivityökaluja.

- Käynnistämisen hankaluus. Komentorivityökalun käynnistäminen eri koneilla, joilla on erilaisia Scalan versioita on epävarmaa ja hankalaa.

10 Poikkeamat suunnitelmasta, toteutunut työjärjestys ja aikataulu

Viikkojen 7 ja 8 aikana tein ja demosin suunnitelmat kuten olin suunnitellutkin. Gitin olin saanut toimimaan 27.2. eli viikolla yhdeksän kuten olin suunnitellut. Aloitin Data- luokan kirjoittamisesta, jota tein yksikkötestien kanssa 27.2.-7.3. En ollut laittanut suunnitelmaan yhtään aikaa Data-luokan tekemiselle, mutta siinä menikin viikko. Model ja OLSModel-luokkia tein 7.3.-15.3. Tässä vaiheessa olin viikon myöhässä aikataulusta. Tein jossain välissä projektidokumentaation pohjan kuten olin suunnitellut. Suunnitelmastani puuttui kokonaan Drawing-luokka, joka osoittautui yhdestä tärkeimmistä ja vaikeimmista ohjelman osista. Tein sitä 18.3.-21.3, 4.4., 11.-12.4., 15.-16.4. Drawing-luokka jäi suunnitelmasta pois varmaankin sen vuoksi, etten ollut ratkaissut suunnitelmaa tehdessäni miten toteuttaisin varsinaisen piirtämisen. Aloitin datan lukemisen testaamisen ja kirjoittamisen 21.-26.3. Tällöin oli siis viikko 13, kun alkuperäisen suunnitelman mukaan olisi pitänyt olla viikko 11. Komentorivin (`CLIApp`) ja Engine-luokan ohjelmointi tapahtui noin 26.3.-11.4. (viikot 13-15). Alkuperäisen suunnitelman mukaan tällöin olisi pitänyt tehdä graafinen käyttöliittymä ja viimeistellä koodia. Lisäsin toisen asteen regressiopolynomin ohjelmaan 13.4. (`QuadModel`). XMLReader-luokan tein 14.4.

Toteuttamisjärjestys toteutui pääpiirteissään, mutta eritysesti Data- ja Drawing-luokkiin meni huomattavan paljon aikaa eivätkä ne olleet alkuperäisessä suunnitelmassa mukana. Myös ohjelman saaminen toimimaan komentoriviltä vei enemmän aikaa kuin mitä odotin aiempien C ja C++ -kokemusteni perusteella.

Alussa tein yleensä noin 10 tuntia viikossa, tenttiviikon aikoihin vähemmän ja lopussa enemmän. Tein yhteensä vähemmän kuin suunnitelin, koska muut opinnot ja työt söivät aikaa. Varasin työlle puskuriaikaa, koska arvelin että yllättäviä asioita tapahtuu. Näin kävikin. Jälkikäteen arvioituna käytin alussa liikaa aikaa toisarvoisten ominaisuuksien kuten puuttuvien tietojen käsittelyn hiomiseen.

11 Kokonaisarvio lopputuloksesta

Ohjelma täyttää vaatimukset: se lukee kahdenlaisia datatiedostoja, sovittaa niihin kahdenlaisia regressiomalleja, ja lopuksi tuottaa malleista visualisoinnin, jonka yksityiskohtiin käyttäjä voi vaikuttaa ja jonka voi tallentaa tiedostoon. Ohjelmaan voi helposti lisätä uusia ominaisuuksia tai muuttaa luokkien toteutuksia ilman että koko ohjelmaa pitäisi kirjoittaa uudelleen. Uusien regressiomallien ja tiedostomuotojen lisäksi erilaisten käyttöliittymien lisääminen pitäisi olla kivutonta. Käyttölogiikkaa (`Engine`) olen rakentanut niin, että käyttöliittymissä on esimerkiksi mahdollista käyttää undo- ja redo-toimintoja.

Huonot puolet liittyvät tällä hetkellä visualisoinnin yksityiskohtiin ja niin sanottuun käyttöliittymään. Eri skaalat pysty- ja vaaka-akselilla johtavat helposti rumiin kuviin vaikka kuvat olisivatkin sinänsä paikkansa pitäviä. Asiaan

voisi vaikuttaa mahdollistamalla eri pikselimäärät samoille yksiköille pysty- ja vaaka-akselilla.

Akseleiden skaalaus kokonaisluvuista alaspäin eli numeron kymmenen negatiiviisiin potensseihin ei myöskään tällä hetkellä toimi, minkä kiertämiseksi hyvin pieniä lukuja sisältäviä datatiedostoja tulisi käsitellä etukäteen ennen niiden syöttämistä ohjelmalle. Asiaa voisi lähteä korjaamaan esimerkiksi vaihtamalla akseleiden yksiköt kokonaisluvuista liukuluvuiksi.

Akseleiden pituudet voisivat vastata paremmin annettuja päätepisteitä regressiokäyrälle. Ohjelmalle voisi tehdä graafisen käyttöliittymän. Minua kiinnostaisi lisätä ohjelmaan mahdollisuus saada datapisteiden tiedot näkyviin kun niiden päälle vie hiiren kursorin. Eniten minua kuitenkin kiinnostaa taustalla olevan matematiikan parantaminen. Käytetty normaaliyhtälö on yksinkertaisin menetelmä regressikertoimien laskemiseen, mutta se on myös virhealtis. Aion ohjelmoida muita algoritmeja. Aion myös ohjelmoida p-arvojen laskemisen regressioyhtälön kertoimille, sekä epävarmuuden kuten luottamusvälien visualisointeja.

Olen tyytyväinen tietorakenteisiin ja luokkajakoon. Ohjelma on helposti laajennettavissa ja muutettavissa.

Jos aloittaisin projektin nyt uudelleen alusta, en käyttäisi niin paljoa yksityiskohtien tai ei-tarpeellisten ominaisuuksien hiomiseen kuin mitä nyt tein alussa. Yrittäisin sen sijaan toteuttaa ensin toimivan rungon, johon voisi lisätä ominaisuuksia käytettävissä olevan ajan mukaan. Siihen suuntaan aloin projekin loppua kohti mennä.

12 Viitteet

Algorithm to round to the next order of magnitude in R. Stack Overflow. [Viitattu 23.4.2019].
https://stackoverflow.com/questions/7906996/algorithm-to-round-to-the-next-order-of-magnitu

Best way to parse command-line parameters? Stack Overflow. [Viitattu 23.4.2019].
https://stackoverflow.com/questions/2315912/best-way-to-parse-command-line-parameters

Drawing images. Otfried Cheong. [Viitattu 23.4.2019].
http://otfried.org/scala/drawing.html

How to process a CSV file in Scala. Alvin Alexander. [Viitattu 23.4.2019].
https://alvinalexander.com/scala/csv-file-how-to-process-open-read-parse-in-scala

How to rotate text with Graphics2D in Java? Stack Overflow. [Viitattu 23.4.2019].
https://stackoverflow.com/questions/10083913/how-to-rotate-text-with-graphics2d-in-java

Inverse Matrix by Gauss Jordan Elimination. Java Programming Lab Code. [Viitattu 23.4.2019].
http://cljavacode.blogspot.com/2017/06/inverse-matrix-by-gauss-jordan.

html

Ohjelmointistudio 2. Aalto-yliopisto. [Viitattu 23.4.2019].
https://plus.cs.hut.fi/studio_2/k2019/

Ordinary Least Squares. Wikipedia. [Viitattu 23.4.2019].
https://en.wikipedia.org/wiki/Ordinary_least_squares

Working With XML in Scala. Mahesh Chand. [Viitattu 23.4.2019].
https://dzone.com/articles/working-with-xml-in-scala

13 Liitteet

13.1 Liite 1: Lähdekoodi

Lähdekoodi on saatavilla kokonaisuudessaan osoitteesta:
https://version.aalto.fi/gitlab/palosae2/regression-visualization

13.2 Liite 2: Ajoesimerkkejä

Ohjelma lukee CSV-tiedoston ja käyttää oletusregressiomallia (ensimmäisen asteen yhtälö) ja visualisoinnin oletuasarvoja:

```
scala -cp "${CLASSPATH}:${SCALA_HOME}/lib/scala-library.jar:
NameOfTheJARExported.jar" regressionViz.CLIApp --d datatiedosto.csv
--o kuvatiedosto.png
```

Ohjelma lukee XML-tiedostosta muuttujat varName1 ja varName2 ja sovittaa toisen asteen yhtälön datapisteisiin:

```
scala -cp "${CLASSPATH}:${SCALA_HOME}/lib/scala-library.jar:
NameOfTheJARExported.jar" regressionViz.CLIApp --d datatiedosto.xml
--o kuvatiedosto.png --modeltype quad --varx varName1 --vary varName2
```

Ohjelma lukee CSV-tiedoston; sovittaa toisen asteen yhtälön; antaa nimet muuttujille; tekee kuvaajan, jonka koko on 600x600 pikseliä; rajoittaa kuvaajan vaaka-akselin välille (-1, 5); ja värjää datapisteet keltaisiksi. Regressiokäyrän väri ja y-akselin väli ovat oletuksen mukaisia:

```
scala -cp "${CLASSPATH}:${SCALA_HOME}/lib/scala-library.jar:
NameOfTheJARExported.jar" regressionViz.CLIApp --d datatiedosto.csv
--o kuvatiedosto.png --modeltype quad --varx X --vary Y --sizex 600
--sizey 600 --xmin -1 --xmax 5 --pR 255 --pB 215 --pG 20
```