SLOVENSKÁ TECHNICKÁ UNIVERZITA
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGII

Rozpoznávanie vzorov pomocou strojového učenia

Predmet	Umelá inteligencia
Vypracoval	Tomáš Babjak, 91986
Akademický rok	2018/2019

3.5.2019 Bratislava

Obsah

1. Zadanie úlohy

MNIST dataset obsahuje obrázky ručne písaných číslic a prislúchajúci label. Vytvorte model s pomocou algoritmov strojového učenia, ktorý dokáže na základe obrázku ručne písaného čísla klasifikovať aké číslo sa nachádza na obrázku.

1. Vytvorte model pomocou neuronovej siete trénovanej algoritmom **backpropagation**.
2. Vytvorte model pomocou vybraného algoritmu pre tvorbu **rozhodovacích stromov.**
3. Vytvorte model pomocou algoritmu **Random Forest**

- Vyhodnoťte kvalitu každého modelu pomocou confusion matrix a celkovej error rate.
- Do dát doplnite aspoň 3 odvodené atribúty tak, aby ste znížili error rate celkovo o minimálne 1 percentuálny bod
- Ak skombinujete všetky 3 modely do jedného modelu, o koľko sa zvýši úspešnosť klasifikácie?
- Ktorý z atribútov má najvyšší vplyv na kvalitu modelu a prečo?
- Upravte JS aplikáciu tak, aby klasifikovala ručne písané čísla. Po nakreslení čísla a prípadne stlačení tlačítka KLASIFIKUJ sa na obrazovke zobrazí výstup z klasifikátora: číslo od 0-9.

2. Prostredie

Na vytvorenie programu som požil vývojové prostredie IntelliJ IDEA, pracoval som s programovacím jazykom Java a s knižnicou Weka, určenou na prácu so strojovým učením.

Knižnicu Weka som si vybral na základe recenzii a návodov na internete a zároveň preto, že obsahovala knižnice na prácu s každým požadovaným typom modelov. Vytvorený projekt je typu Maven Project, aby sa do neho dali vložiť všetky závislosti na použitých knižniciach Weka a ostatných knižníc, potrebných k vypracovaniu projektu.

Spomínaná knižnica pracuje iba so súbormi typu .arff, preto som musel do projektu implementovať funkciu, ktorá prekladá .csv súbory do tohto formátu. Pri projekte budú priložené aj arff súbory pre možnú kontrolu alebo testovanie.

3. Modely učenia

3.1 Neurónová sieť

Neurónová sieť je implementovaná v triede NeuralNetwork a je vytvorená pomocou weka triedy MultilayerPerceptron, ktorá využíva backpropagation na učenie. A classifier that uses backpropagation to learn a multi-layer perceptron to classify instances.[1] Vo funcii trainNN() sa načítajú vstupné dáta a nastaví sa index na nulu, aby sa za vedúce prvky brali tie, ktoré vyjadrujú číslo od 0 – 9. Vytvorí sa nová inštancia MultilayerPerceptron a navolia sa parametre neurónovej siete a to learning rate, momentum, number of epochs a hlavne počet skrytých vrstiev (-H), ktorý som nastavil na 100. Pomocou funkcie buildClassifier() naučíme neurónovú sieť na tréningových dátach. Vo funkcii prepareTestInstance() sa načítajú testovacie dáta a pomocou triedy Evaluation a funckie evaluateModel vyhodnotíme tieto dáta. Vypíšeme errorRate, sumárne výsledky a confusion matrix.

3.2 Rozhodovacie stromy

Na implementáciu rozhodovacích stromov som použil weka inštanciu rozhodovacieho stromu J48 [http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html]. Vytváram ho vo funkcii trainTheTree(), nastavím mu vlastnosť unpruned, čiže neobmedzený a vytrénujem ho pomocou funckie buildClassifier(trainingData). Výsledné úspešnosti, error rate a maticu vypisujem rovnako ako pri neurónovej sieti.

[1] http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html

3.3 Random Forest

Random Forest implementujem v triede RF ako inštanciu weka triedy Random Forest. Vytváram a trénujem ho podobne ako predošlé prípady a navyše mu ešte nastavujem pomocou funkcie setNumIterations() počet stromov, ktoré sa v lese majú nachádzať. Testovanie a výpis výsledkov sa vykonáva rovnako ako pri neurónových sieťach.

3.4 Porovnanie výsledkov modelov

Neurónová sieť s počtom skrytých vrstiev 100 ponúkla nasledujúce výsledky:

```
Correctly Classified Instances        9506               95.06   %
Incorrectly Classified Instances       494                4.94   %
Relative absolute error                11.5331 %
Root relative squared error            29.6581 %
Total Number of Instances          10000

=== Confusion Matrix ===
   a    b    c    d    e    f    g    h    i    j   <-- classified as
 962    0    3    1    0    3    8    2    1    0 |    a = 0
   0 1119    7    2    0    2    1    1    3    0 |    b = 1
  10    1  987    1    7    3    7    9    6    1 |    c = 2
   0    0   17  925    1   45    0   12    9    1 |    d = 3
   1    4    4    0  954    1    8    1    2    7 |    e = 4
   7    1    1    3    3  858    8    2    5    4 |    f = 5
   7    3    2    1    7   11  925    0    2    0 |    g = 6
   2   19   23    4   10    0    0  960    2    8 |    h = 7
   4    4    9    7    8   19    7    4  911    1 |    i = 8
   6    7    2    6   48   17    1   10    7  905 |    j = 9
```

Rozhodovací strom:

```
Correctly Classified Instances        8933               89.33   %
Incorrectly Classified Instances      1067               10.67   %
Relative absolute error               13.1979 %
Root relative squared error           46.9359 %
Total Number of Instances          10000

=== Confusion Matrix ===
   a    b    c    d    e    f    g    h    i    j   <-- classified as
 939    1    9    2    2    3   11    5    5    3 |    a = 0
   3 1088   14    7    1    4    2    1   14    1 |    b = 1
  19    7  913   32    5    7    6   20   19    4 |    c = 2
   4   10   37  856    5   46    2   21   20    9 |    d = 3
   8    6   13    7  866    7    8    2   23   42 |    e = 4
  14    9    8   46   11  735   20    9   20   20 |    f = 5
   9    4   15    3    7   15  898    0    5    2 |    g = 6
   3    4   24   14   12    9    0  925   17   20 |    h = 7
  17    9   16   25   23   24   17   11  813   19 |    i = 8
```

```
   8    6    0   15   30   16    2   12   20  900 |    j = 9
```

Random Forest s počtom stromov 10:

```
Correctly Classified Instances        9393               93.93   %
Incorrectly Classified Instances       607                6.07   %
Relative absolute error              23.6213 %
Root relative squared error          39.3744 %
Total Number of Instances            10000

=== Confusion Matrix ===
   a    b    c    d    e    f    g    h    i    j   <-- classified as
 963    1    4    0    0    2    7    1    0    2 |    a = 0
   0 1124    2    3    1    1    2    1    1    0 |    b = 1
   7    0  986    3    5    1    6   11   12    1 |    c = 2
   5    4   24  928    1   17    2   10   15    4 |    d = 3
   0    0    5    1  930    2    5    4    7   28 |    e = 4
  11    1    5   37    7  807    9    4    8    3 |    f = 5
  10    4    4    2    6   12  915    1    3    1 |    g = 6
   3    7   23    6    7    1    0  960    7   14 |    h = 7
  12    2   16   25   10   18    4   11  865   11 |    i = 8
  10    6    3   12   27    7    1   14   14  915 |    j = 9
```

Z toho malého prieskumu vyšli víťazne neurónové siete, ktoré mali najvyššie percento správne určených písmen. Naopak najmenej sa osvedčili rozhodovacie stromy, ktoré mali toto percento najnižšie. Samozrejme, všetky tieto výsledky by sa mohli ešte výrazne líšiť ak by sa zvolili iné vstupné parametre ako počet stromov v Random Forest-e alebo počet skrytých vrstiev v neurónovej sieti.

Na základe confusion matrix modelu neurónovej siete vidíme, že najčastejšie si model pomýlil číslo 4 s číslicou 9 a číslo 5 si v 45 prípadoch pomýlil za číslo 3. Rozhodovací strom si najčastejšie mýlil číslicu 3 za číslicu 5 a naopak, a taktiež aj číslice 3 a 2. Aj random forest mal najviac problémov s vyhodnocovaním číslic 3 a 5, ale aj 4 a 9.

4. Skombinovanie modelov do jedného

Na skombinovanie modelov spomínaných v predchádzajúcej kapitole som použil funkcionalitu Weka nazývanú Vote. Je to trieda, ktorá kombinuje predikcie jednotlivých podmodelov na základe počtu hlasov pre jednotlivé čísla. Trieda, v ktorej som túto kombináciu modelov využil sa nazýva Combination.

Implementácia je veľmi podobná predošlým, kde je však potrebné buď si načítať už vytvorené – serializované modely alebo vytvoriť a naučiť nové modely stromu, lesu a neurónovej siete a následne vytvoriť pole klasifikátrov a vložiť do nej tieto modely. Následne sa vytvorí nová inštancia typu Vote, nastavia sa jej Classifiers, otestuje sa a vypíšu sa výsledky.

Pri skombinovaní modelov, ktorým som sa venoval v predošlej kapitole za pomoci Vote som získal nasledujúce výsledky:

Correctly Classified Instances	9541	95.41 %
Incorrectly Classified Instances	459	4.59 %
Relative absolute error	16.0485 %	
Root relative squared error	31.4514 %	
Total Number of Instances	10000	

Na základe týchto údajov vieme povedať že skombinovanie modelov prinieslo zníženie error rate o 6 % v porovnaní s najhorším a 0,35 % v porovnaní s najlepším modelom. Vidíme teda, že kombináciou rôznych typov modelov dokážeme ešte o niečo znížiť finálnu error rate.

5. Atribút s najvyšším vplyvom na kvalitu

Hľadanie najdôležitejšieho bodu vo vzorke je náročná úloha čo sa času týka, keďže by bolo potrebné overiť nárast error rate vzorky pre každý odstránený bod obrázka rozmerov 28x28 čo je 784 bodov. Preto je potrebné túto úlohu zjednodušiť a to tak, že predpokladáme, že na okrajoch obrázka sa nenachádzajú dôležité body a väčšina políčok na okrajoch má hodnotu 0, čiže nie je zafarbená.

Experimentálne to overíme tak, že z trénovacej a testovacej vzorky odrežeme 4 riadky zo všetkých štyroch strán, vykonáme učenie a testovanie a porovnáme error rate s error rate pôvodnej vzorky. Vytvoril som tak súbor mnist_test_reduced.csv a mnist_train_reduced.csv Po vytrénovaní neurónovej siete pomocou nových testovacích a trénovacích vzoriek a rovnakých parametrov ako predtým mi vyšiel nasledujúci výsledok:

Correctly Classified Instances	9623	96.23 %
Incorrectly Classified Instances	377	3.77 %
Relative absolute error	10.5225 %	
Root relative squared error	26.8807 %	
Total Number of Instances	10000	

Tento výsledok som považoval za pozitívny, keďže sa znížila error rate vzhľadom k pôvodným neurónovým sieťam, avšak treba dodať, že sa taktiež znížil počet parametrov, ktoré musela neurónová sieť vyhodnocovať, ale počet skrytých vrstiev ostal rovnaký. Keďže

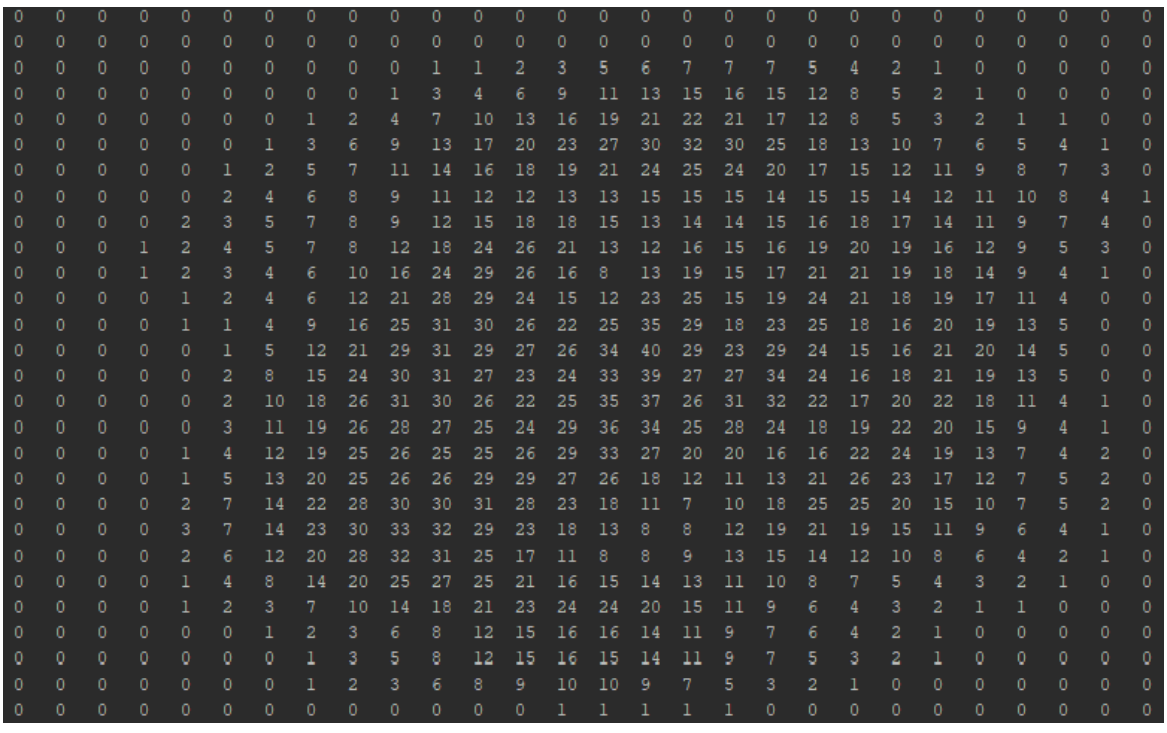
som odrezaním rámov zmenšil obrázok na 400 pixelov, plánoval som ďalej pracovať s týmito bodmi, čo by bolo však ešte stále dosť pracné a tak som sa ešte raz obhliadol po internete.

Na istom webovom portáli som našiel informáciu o funkcii knižnice Weka, ktorá by toto rozhodovanie o najdôležitejšom atribúte dokázala vykonať za mňa. Celá táto funkcionalita sa vykonáva v triede BestAttribute. Weka trieda na evaluáciu atribútu sa nazýva InfoGainAttributeEval. InfoGainAttributeEval evaluates the worth of an attribute by measuring the information gain with respect to the class[2].

Vytvoril som si teda inštanciu triedy InfoGainAttributeEval a pomocou funkcie buildEvaluator(testingData) som ho naplnil dátami a vyhodnotil. Jednotlivé atribúty vyhodnocujem v cykle a vkladám do hash mapy, v ktorej si ich na základe hodnoty usporiadam a vypíšem túto mapu a jej prvý prvok.

Podľa výsledkov InfoGainAttributeEval triedy najdôležitejším atribútom je ten na pozícii 14x15 v obrázku nasledovaný pozíciou 15x15, a 16x15. Celkové výsledky nájdete v súbore results.txt. Na chvoste rebríčka sa umiestnili atribúty, ktoré som pri redukovanej vzorke odstránil, čiže tie na okraji obrázka. Ich hodnota je 0 alebo veľmi blízka nule, čo znamená že nikdy sa na základe daného políčka nerozhoduje o aké číslo sa jedná.

Trochu som sa ešte pohral s výpisom nájdených hodnôt významnosti atribútov a vypísal som si ich na pozície kde sa nachádzajú reálne atribúty na obrázku. Výsledok je takýto:



[2] http://weka.sourceforge.net/doc.stable-3-8/weka/attributeSelection/InfoGainAttributeEval.html

Na základe obrázka je veľmi zreteľne vidieť, ktoré hodnoty sa na vyhodnotenie používajú najčastejšie a ktoré vôbec. Najmä ľavý okraj obrázka obsahuje na troch stĺpcoch samé čísla 0, horný okraj dva riady s nulami, pravý a spodný jeden riadok, ktorý by sa mohol odstrániť. Ak by sme tieto málo vážené atribúty odstránili mohli by sme tým zvýšiť rýchlosť rozoznávania obrázku.

6. Realtime klasifikácia čísla

Na realtime klasifikáciu čísla som nevyužil ponúkaný spôsob pomocou Javascriptu, ale našiel som spôsob ako ho môžem vyhodnocovať priamo v Jave pomocou Canvas-u. Vykonávanie klasifikácie čísla sa vykonáva v trieda MnistClassifierUI. Funkcie na vytvorenie plátna a prevod toho, čo sa na plátne nachádza na obrázok som našiel na Githube[3] a využil som ho v mojom projekte (vytváranie plátna a spracovanie na obrázok nebolo náplňou projektu).

Samostatne som vypracoval funkciu predictImage(), v ktorej si načítavam obrázok do ByteArrayOutputStream a odtiaľ ho prevádzam na pole typu byte a na pole int. Hodnoty byte prevádzam na int tak, že ak je v poli byte -1 premením ho na 0 (minimálnu hodnotu) a ak je číslo 0 premením ho na 255 (maximálnu hodnotu), k ostatným číslam pripočítam 127 (maximálne teda bude mať hodnotu 254). Po vytvorení nového .csv súboru doňho vložím hodnoty z obrázka oddelené čiarkou a na začiatok vložím názvy všetkých atribútov. Následne si načítam serializovanú neurónovú sieť (uloženú ako NeuralNetwork v adresári) do triedy Classifier a vyhodnotím pomocou nej o aké číslo sa jedná vo funkcii classifyInstance(i).

Vyhodnocovanie nie je veľmi presné, pretože predsa je to plátno trochu rozdielne od papiera a písať myšou číslice na počítač nie je to isté ako písať ich ručne na papier. Napriek tomu, ak sa človek posnaží pekne písať dokáže čísla relatívne správne rozoznávať až na niektoré prípady. Ako v ňom pracovať nájdete v používateľskej príručke.

7. Zhodnotenie

Výsledný projekt považujem za úspešný z hľadiska toho, že som sa naučil ako pracujú modely strojového učenia a typy aké poznáme a že sa mi podarilo úspešne implementovať tieto vzory pomocou dostupných knižníc Weka. Relatívne dobre vyhodnocuje dáta neurónová sieť, ale aj Random forest, o niečo menej presné výsledky ponúka rozhodovací strom.

[3] https://github.com/deeplearning4j/dl4j-examples/blob/master/dl4j-examples/src/main/java/org/deeplearning4j/examples/convolution/mnist/MnistClassifierUI.java

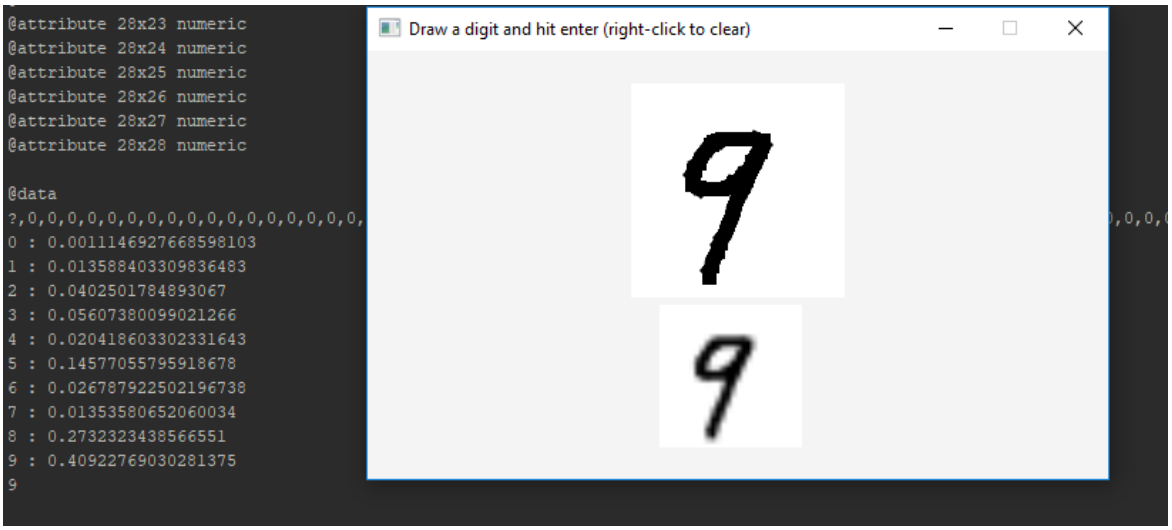
Naimplementoval som taktiež real-time rozpoznávanie písaných čísel na počítač pomocou kurzora, ktorý síce nie je najpresnejší, ale funguje ako príklad toho, čo všetko sa dá pomocou strojového učenia dokázať. Kombinovaním jednotlivých modelov, ako som ukázal, by sa dali dosiahnuť ešte lepšie výsledky s nižším error rate. Prípadné pridanie odvodených atribútov, ktoré sa mne však nepodarilo naimplementovať, by tiež túto konečnú chybovosť dokázalo ovplyvniť.

8. Používateľská príručka

Projekt je naprogramovaný v jazyku Java a je typu Maven Project, aby sa dali správne vložiť Dependencies na Weka knižnice. Je teda pravdepodobné, že ak si budete chcieť vyskúšať na vlastnom stroji budete potrebovať aj použité knižnice a závislosti na nich[4]. Tieto závislosti sú v súbore pom.xml. Pred začatím práce je taktiež potrebné skonvertovať .csv súbory do .arff alebo použiť mnou vytvorené súbory mnist_train.arff a mnist_test.arff.

Pre vyskúšanie trénovania a testovania neurónových sietí, rozhodovacích stromov a random forest je potrebné spustiť main() v triedach NeuralNetwork, DecisionTree a RF. Pre vyskúšanie skombinovania spustite main() v triede Combination (nemusia byť dopredu vytvorené modely, spravia sa nanovo).

Pre vyskúšanie realtime rozoznávania čísel spustite main() v triede MnistClassifierUI. Otvorí sa vám plátno na ktoré môžete písať pomocou kurzora myši a po kliknutí na enter sa vám v konzole zobrazia pravdepodobnosti a predpovie číslo, ktoré bolo nakreslené. Plátno vyčistíte kliknutím pravým tlačidlom myši. Je potrebné už mať natrénovanú neurónovú sieť pred tým ako spustíte toto okno. Dôležitosť jednotlivých atribútov zistíte po spustení main() v triede BestAttribute.



[4] https://mvnrepository.com/artifact/nz.ac.waikato.cms.weka/weka-stable/3.8.0