Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tadej Lokar

Generiranje psevdoumetnih podatkovnih množic

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: prof. dr. Marko Robnik-Šikonja

Ljubljana 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00254/2012
Datum: 10.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **TADEJ LOKAR**

Naslov: **GENERIRANJE PSEVDOUMETNIH PODATKOVNIH MNOŽIC**
GENERATION OF PSEUDOARTIFICIAL DATA SETS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Priprava verodostojnega testnega okolja je pomemben del razvoja novih algoritmov in reševanja številnih realnih problemov. V podatkovnem rudarjenju potrebujemo realne in zadosti velike podatkovne množice. Žal je v mnogo primerih zbiranje podatkov drago, do zadosti velike zbirke podatkov pa je včasih celo nemogoče priti. Za namene testiranja želimo zato iz že obstoječih realnih podatkov pridobiti njihove značilnosti ter nato generirati poljubno velike umetne množice z enakimi lastnostmi.
Uporabite nekaj različic učne metode RBF, ki naučeno znanje hranijo v obliki Gaussovih jeder, in se iz obstoječih podatkov naučite napovednega modela. Dobljeni model spremenite v generator ter z njim generirajte umetne množice podatkov. Kvaliteto dobljenih podatkov, njihovo podobnost z originalnimi ter primernost za uporabo v strojnem učenju preverite na več javno dostopnih podatkovnih množicah. Rešitev zasnujete v okolju R.

Mentor:
prof. dr. Marko Robnik Šikonja

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Tadej Lokar, z vpisno številko **63070274**, sem avtor diplomskega dela z naslovom:

Generiranje psevdoumetnih podatkovnih množic

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Marka Robnik-Šikonje,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 9. oktobra 2012 Podpis avtorja:

Kazalo

Povzetek

V diplomski nalogi smo zasnovali generator namenjen generiranju psevdo umetnih podatkovnih množic. Ta kot parameter sprejme neko množico podatkov, se nauči njenih lastnosti ter na tej podlagi generira nove primere. Ključni del generatorja predstavlja algoritem PRBF, saj je njegova struktura še posebej primerna za generiranje novih primerov. Generator smo testirali z več realnimi podatkovnimi množicami in analizirali njegovo uspešnost glede na značilnosti uporabljenih podatkovnih množic. V zaključku so opisane možne nadgradnje, ki bi izboljšale delovanje generatorja.

Abstract

We created a generator of pseudo artificial data sets. The generator takes a given data set as input, analyses it and creates a new data set based on learned properties. A key component of the generator is PRBF algorithm, because its structure is particularly suited for example generation. We tested the generator on multiple real data sets and measured its quality with the data sets properties. In the conclusion we propose possible enhancements to generator's abilities.

Poglavje 1

Uvod

Z razcvetom tehnologije v 20. in 21. stoletju si je človeška rasa na veliko področjih olajšala delo. Med ta področja spada tudi statistika, ki se primarno ukvarja z zbiranjem, klasificiranjem in analizo podatkov. Zaradi stroškov, ki jih terja zbiranje realnih podatkov, ter njihove omejene količine, smo zasnovali generator psevdo umetnih podatkovnih množic. Psevdo umetne podatkovne množice so sestavljene iz naključnih podatkov, ki jih kreiramo na podlagi deterministčnih procesov. Umetne množice podatkov so uporabne za testiranje novih algoritmov, v kliničnih študijah lahko z njihovo pomočjo poskrbimo za anonimnost udeležencev, ki so prispevali podatke, za izobraževalne potrebe (prvotne podatke je prepovedano distribuirati zaradi avtorskih pravic), itd..

Generator smo zasnovali na učni metodi PRBF (*ang. Probabilistic Radial Basis Function*), ki je izpeljanka metode RBF (*ang. Radial Basis Function*). Metoda PRBF je osnovana na nevronski mreži, ki vsebuje umetne nevrone. Umetni nevroni pri tej vrsti klasifikatorjev hranijo naučeno znanje v obliki Gaussovih jeder. Jedra predstavljajo lokalne distribucije podatkov in so definirana s centrom in razpršenostjo. Ta dva podatka služita za generiranje novih primerov. S pomočjo uteži smo pridobili zahtevano količino podatkov od posameznih jedr.

Diplomska naloga opisuje generator umetnih podatkovnih množic. Drugo

poglavje predstavi učno metodo PRBF. V tretjem poglavju je podana zasnova generatorja in njegova implementacija. V četrtem poglavju je opisan postopek testiranja in rezultati. Zadnje, peto poglavje, povzema delo in oceni kvaliteto in vsebino opravljenega dela ter predstavi možne izboljšave.

Poglavje 2

Predstavitev PRBF

V tem poglavju opišemo model PRBF [1], ki je izpeljanka RBF nevronske mreže oz. klasifikatorja.
Problem je definiran z razredi $y_k = (k = 1, ..., r)$ in atributi $x = (A_1, ..., A_a)$. Klasifikator PRBF ima a vhodov in r izhodov. Vsak izhod predstavlja posamezen razred ter ima verjetnostno porazdelitev $p(x|y_k)$.
Vsak skriti nevron izračuna verjetnostno porazdelitev $f_j(x)$ za vhod x [1].
Pogojna verjetnost vsakega razreda je definirana kot:

$$p(x|y_k) = \sum_{j=1}^{M} \pi_{jk} f_j(x), \qquad k = 1, ..., c, \tag{2.1}$$

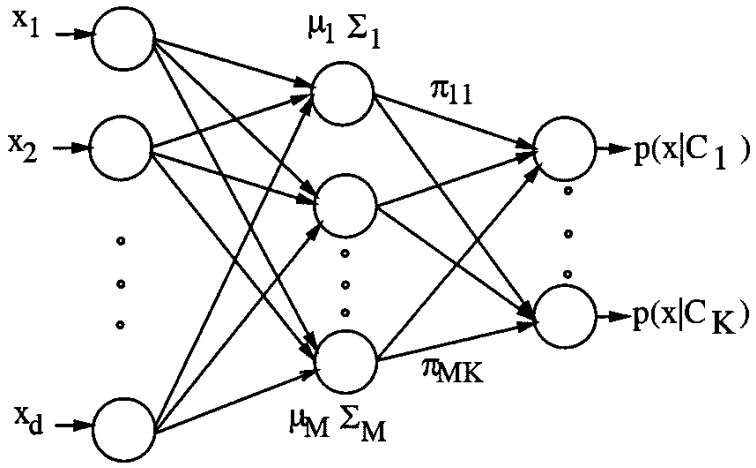
kjer π_{jk} predstavlja mešani koeficient apriorne verjetnostne porazdelitve *(ang. prior)*. Koeficienti morajo biti pozitivni z vsoto enako 1 [2].
Razred točke x se izračuna s pomočjo Bayesovega teorema, kjer izberemo razred, ki ima največjo posteriorno verjetnostno porazdelitev $p(y_k|x)$.

$$p(y_k|x) = \frac{p(x|y_k)P_k}{\sum_{l=1}^{c} p(x|y_l)P_k} \tag{2.2}$$

Verjetnostna porazdelitev za vhod x se izračuna po formuli:

$$f_j(x) = \frac{1}{(2\pi)^{a/2}|\sum_j|^{1/2}} exp\{-\frac{1}{2}(x - \mu_j)^T \sum\nolimits_j^{-1} (x - \mu_j)\} \tag{2.3}$$

kjer $\mu_j \in \Re^a$ predstavlja sredino komponente j in $\sum_j$ kovariančno matriko dimenzije $a \times a$ [3]. Komponenta predstavlja umetni nevron, ki se



Slika 2.1: Arhitektura PRBF nevronske mreže.

nahaja v skriti plasti nevronske mreže. Slika 2.1 prikazuje arhitekturo PRBF nevronske mreže.

Učenje PRBF nevronske mreže je osnovano na EM (*ang. expectation-maximization*) algoritmu, ki se pogosto uporablja za ocenjevanje mankajočih vrednosti. Predpostavimo nevronsko mrežo PRBF z M skritimi nevroni. Ko želimo zgraditi nevronsko mrežo z $M+1$ komponentami, postopek dodajanja nove komponente vsebuje globalno in lokalno iskanje v prostoru parametrov nove komponente.
EM algoritem je sestavljata dva koraka:

- E-korak - predstavlja globalno iskanje najbolj primernega območja glede na iskalne kriterije
- M-korak - predstavlja lokalno iskanje, kjer prilagodimo parametre, ki smo jih pridobili v E-koraku.

Za potrebe generatorja so bistvene komponente (vsebujejo Gaussova jedra), mešane uteži in sposobnost klasificiranja novih primerov. Nevronska mreža PRBF klasificira primere glede na posteriorno verjetnostno porazdelitev $p(x|y_k)$.

Poglavje 3

Zasnova in implementacija generatorja

3.1 Zasnova generatorja

Generator je zasnovan v programskem okolju R [4] in Octave [5], vendar se večina procesiranja izvaja v Octave. Koda v okolju Octave predstavlja jedro generatorja in se lahko uporabi tudi kot samostojni modul. Zaradi slabo podprtega dela z nizi v sistemu Octave, se vse pretvorbe diskretnih atributov v numerične izvajajo v okolju R. Uporabniška aplikacija je zato napisana v programskem jeziku R.

Uporabnik uporabi generator s klicem metode `generateData`, ki sprejme 3 vhodne parametre:

- podatkovno množico, ki je sestavljena iz označene učne množice. Podatki so v R objektu tipa `data.frame`,
- število r, ki nam pove v katerem stolpcu podatkovne množice se nahajajo oznake oz. razredi,
- število podatkov, ki jih želi uporabnik generirati.

Generirane podatke dobi uporabnik v objektu tipa `data.frame`. Funkcija,

poleg povezave z okoljem Octave, skrbi tudi za čimbolj enostavno delo z generatorjem:

- pretvori diskretne vhodne atribute v numerične, ter generirane podatke iz numeričnih nazaj v format vhodnih podatkov,
- če so vrednosti razredov zapisane v nizih znakov, aplikacija poskrbi za pretvorbo vhodnih razredov v numerične ter pretvorbo generiranih razredov, ki so izraženi v numerični obliki, nazaj v prvotno obliko,
- odstrani vrstice, ki vsebujejo mankajoče vrednosti.

Drugi del aplikacije se izvaja v okolju Octave, v katerem je razvita tudi generatorska funkcija `createData`. Funkcija sprejme tri vhodne parametre:

- numerično matriko velikosti $m \times n$, ki predstavlja učno množico oz. podatke (m je število primerov, n je število atributov),
- numerični stolpični vektor velikosti $m \times 1$ (m je število primerov). Za primer v učni množici obstaja vrednost, ki pove kateremu razredu dani primer pripada,
- število primerov p, ki jih želimo zgenerirati.

Vrača dve vrednosti:

- numerično matriko velikosti $p \times n$, ki vsebuje zgenerirane podatke,
- numerični stolpični vektor dolžine $p \times 1$, ki vsebuje informacijo o razredu.

Funkcijo bi v grobem lahko razdelili na tri dele. V prvem delu preverimo vhodne parametre, odstranimo primere, ki bi lahko v nadaljnem procesiranju privedli do napake (npr. stolpce, ki vsebujejo samo eno vrednost) ter spremenimo obliko podatkov, da ustreza modelu PRBF (funkcija `trainPRBF`), ki predstavlja jedro generatorja. Model PRBF je implementiran v knjižnici `marginalPRBF` [1, 2].

Ko smo podatke pripravili kličemo funkcijo `trainPRBF`, ki med drugim vrne:

- vrednost W, matrika velikosti $j \times c$, ki nosi vrednosti mešanih uteži (2.1),
- vrednost M, matrika velikosti $j \times d$, v njem so shranjeni sredinski vektorji (2.3),
- vrednost R, matrika velikosti $j \times d^2$, ki vsebuje kovariančno matriko (2.3).

Spremenljivka j nam pove koliko skritih nevronov oz. Gaussovih jeder je bilo kreirano tekom učenja PRBF, c predstavlja število razredov, d pa število atributov. S pomočjo mešanih uteži določimo število primerov, ki jih želimo zgenerirati s posameznim Gaussovim jedrom. S pomočjo funkcije `mvnrnd` za vsako jedro kreiramo podatke. Funkcija `mvnrnd` vrne matriko N naključnih podatkov, generiranih po normalni oz. Gaussovi porazdelitvi s sredinskim vektorjem μ in kovarinačno matriko $\sum$ (število N, vektor μ in matriko $\sum$ podamo kot parametre ob klicu funkcije). Ko smo z vsakim jedrom kreirali ustrezno količno podatkov, je potrebno podatkom določiti razred. To naredimo s klasifikacijsko funkcijo `outPRBF`. Na koncu spremenimo podatke v obliko, ki ustreza prvotnim podatkom.

3.2 Implementacija programske rešitve

V diplomskem delu smo uporabljali samo prosto dostopne tehnologije. Aplikacijo smo testirali v operacijskem sistemu Ubuntu 12.04 in je realizirana v programskih jezikih Octave in R. Za pisanje kode smo uporabljali program gedit ter programski okolji R verzija 2.15.1 in GNU Octave verzija 3.6.1. Za potrebe integracije okolja R in Octave je bil uporabljen R paket RcppOctave [7] verzija 0.8.5.

3.2.1 GNU Octave

Octave je odprtokoden program za reševanje numeričnih problemov. Vsebuje tudi visokonivojski programski jezik, preko katerega je sistem razširljiv ter skoraj popolnoma kompatibilen s programskim jezikom Matlab. Octave

je bil zasnovan leta 1988 kot spremljevalna programska oprema pri tečaju načrtovanja kemijskih reaktorjev. Začetek razvoja pa izhaja iz leta 1992, ko ga je vodil John W. Eaton. Poimenoval ga je po svojem profesorju Octavu Levenspielu.

3.2.2 Programsko okolje R

Jezik R izhaja iz jezika S, ki so ga razvili Rick Becker, John Chambers in Allan Wiles iz Bellovih laboratorijev [6]. Ideja je bila razviti zmogljivo programsko orodje, ki bi bilo v prvi vrsti namenjeno statistikom. Orodje S je plačljivo zato sta novo zelandska statistika Ross Ihaka in Robert Gentleman iz Univerze v Aucklandu v učne namene razvila okrnjeno verzijo programskega jezika S, ter ga poimenovala R (črka R je v abecedi pred črko S). R predstavlja odprtokodno implementacijo jezika S oz. sistema S-PLUS. Izvorna programska koda okolja R je napisana v programskih jezikih C, Fortran in R. Zaradi široke uporabnosti na področju statistike in privlačnih vizualnih izrisov je orodje široko uporabno in priljubljeno.

Poglavje 4

Testiranje

S testiranjem smo preizkusili kvaliteto delovanja generatorja. Uporabili smo navzkrižno preverjanje ujemanja klasifikacij. Manjše kot so razlike med klasifikacijskimi točnostmi boljše je delovanje generatorja. Generator smo preizkusili na 10 podatkovnih množicah, ki so predstavljene v tabeli 4.1. Postopek testiranja je naslednji.

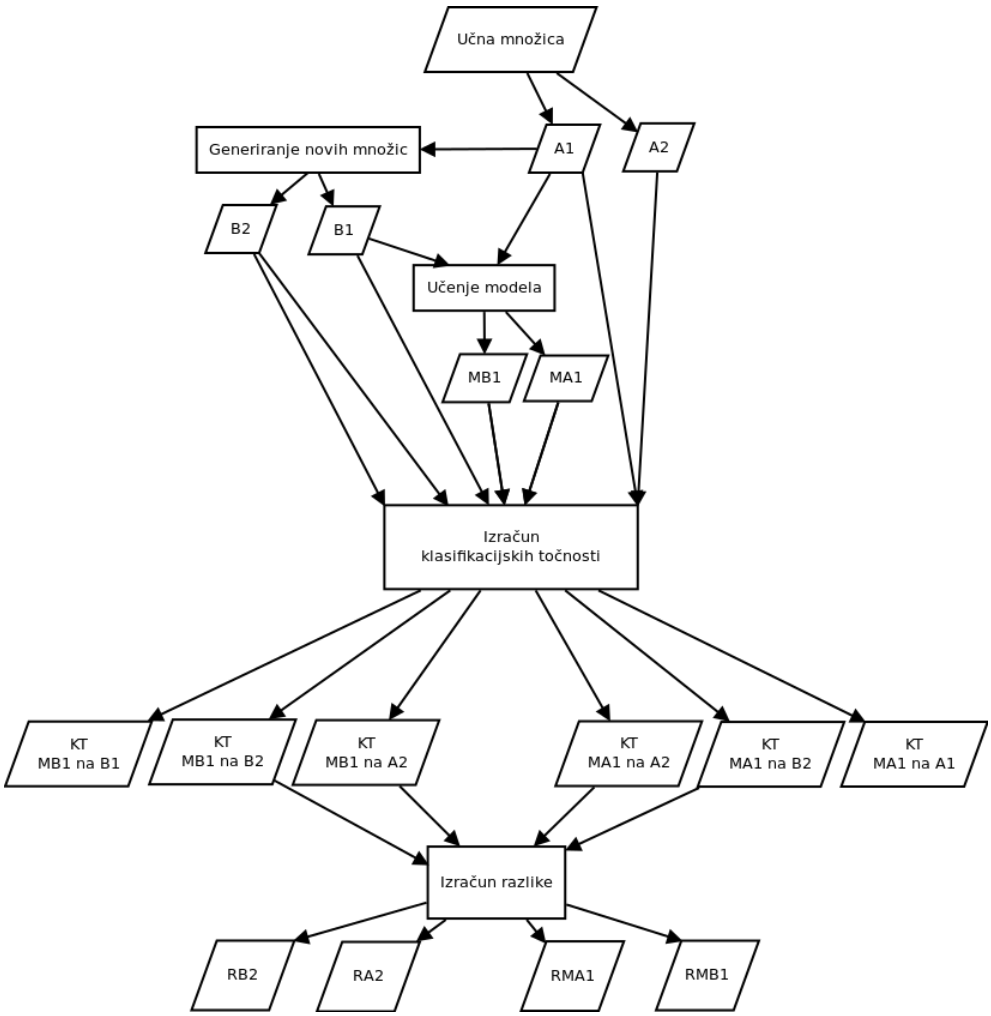
Učno množico smo razdelili na polovici, ki ju poimenujemo `A1` in `A2`. S pomočjo množice `A1` smo generirali dve novi podatkovni množici `B1` in `B2`. Nato smo nad množicama `A1` in `B1` zgradili model `MA1` in `MB1`. Za gradnjo modela smo uporabili metodo naključnih gozdov. Modela `MA1` in `MB1` smo testirali nad množicama `A2` in `B2` (izračunali smo klasifikacijske točnosti). Poleg klasifikacijskih točnosti smo izračunali tudi naslednje razlike:

- `RMA1` - razlika med klasifikacijskima točnostima `MA1 na A2` in `MA1 na B2`,
- `RMB1` - razlika med klasifikacijskima točnostima `MB1 na A2` in `MB1 na B2`,
- `RA2` - razlika med klasifikacijskima točnostima `MA1 na A2` in `MB1 na A2`,
- `RB2` - razlika med klasifikacijskima točnostima `MA1 na B2` in `MB1 na B2`.

Celoten potek testiranja prikazuje slika 4.1.

Z	Domena	Izvor	I	R	A	C	D	M
1	Iris	ML UCI	150	3	4	4	0	Ne
2	Wine	ML UCI	178	3	13	13	0	Ne
3	Glass Identification	ML UCI	214	7	10	10	0	Ne
4	Connectionist Bench	ML UCI	208	2	60	60	0	Ne
5	Ionosphere	ML UCI	351	2	34	34	0	Ne
6	German Credit Data	ML UCI	1000	2	20	7	13	Ne
7	Ecoli	ML UCI	336	8	8	7	1	Ne
8	Congressional Voting Records	ML UCI	435	2	16	0	16	Da
9	Pima Indians Diabetes	ML UCI	768	2	8	8	0	Da
10	Titanic	Delve	2201	4	3	0	3	Da

Tabela 4.1: Izbrane domene; Z označuje zaporedno številko domene, I število primerov v domeni, R število razredov v domeni, A število atributov v domeni (brez razreda), C število številskih atributov, D število diskretnih atributov, M pa pove ali domena vsebuje manjkajoče vrednosti.



Slika 4.1: Shema poteka testiranja; KT označuje klasifikacijsko točnost.

4.1 Rezultati testiranja

V tem podpoglavju so opisani in predstavljeni rezultati, ki smo jih pridobili tekom testiranja generatorja na 10 podatkovnih množicah. Tabela 4.2 prikazuje izbrane domene, število njihovih razredov, število skritih nevronov kreiranih tekom generiranja in razlike med klasifikacijskimi točnostmi.

Z	Domena	R	N	RMA1	RMB1	RA2	RB2
1	Iris	3	3	0.013333	0.026667	0.000000	0.040000
2	Wine	3	3	0.202247	0.314607	0.314607	0.202247
3	Glass Identification	7	3	0.813084	0.271028	0.588785	0.495327
4	Connectionist Bench	2	2	0.125000	0.221154	0.144231	0.201923
5	Ionosphere	2	3	0.034286	0.068571	0.028571	0.074286
6	German Credit Data	2	2	0.020000	0.076000	0.038000	0.058000
7	Ecoli	8	8	0.041667	0.154762	0.095238	0.017857
8	C. Voting Records	2	2	0.008621	0.025862	0.008621	0.008621
9	Pima-Diabetes	2	2	0.080729	0.148438	0.007812	0.059896
10	Titanic	4	22	0.458666	0.455014	0.016349	0.020000

Tabela 4.2: Izbrane domene; Z označuje zaporedno številko domene, R število razredov v domeni, N število skritih nevronov.

4.1.1 Iris

Napovedni model	Testna množica	Klasifikacijska točnost
MA1	A2	0.946670
MA1	B2	0.933333
MA1	A1	1.000000
MB1	A2	0.946667
MB1	B2	0.973333
MB1	B1	1.000000

Tabela 4.3: Klasifikacijske točnosti za podatkovno množico Iris.

Razlika	Vrednost
RMA1	0.013333
RMB1	0.026667
RA2	0.000000
RB2	0.040000

Tabela 4.4: Razlike med klasifikacijskimi točnostmi.

Pri testiranju generatorja nad podatkovno množico Iris opazimo, da so razlike med klasifikacijskimi točnostmi majhne. Število skritih nevronov, ki se kreirajo pri generiranju množic `B1` in `B2` je enako številu razredov učne množice. Sklepamo da generator za to množico deluje dobro. Tabeli 4.3 in 4.4 prikazujeta klasifikacijske točnosti ter njihove razlike za testno množico Iris.

4.1.2 Wine

Napovedni model	Testna množica	Klasifikacijska točnost
MA1	A2	0.988764
MA1	B2	0.786517
MA1	A1	1.000000
MB1	A2	0.674157
MB1	B2	0.988764
MB1	B1	1.000000

Tabela 4.5: Klasifikacijske točnosti za podatkovno množico Wine.

Razlika	Vrednost
RMA1	0.202247
RMB1	0.314607
RA2	0.314607
RB2	0.202247

Tabela 4.6: Razlike med klasifikacijskimi točnostmi.

Podatkovna množica ima 3 različne razrede, prav tako pa ima vsaka izmed novo kreiranih množic 3 skrite nevrone. Večja odstopanja (20%-30%) se pojavijo pri klasifikacijskih točnostih. Delovanje generatorja ocenjujemo kot nenatančno. Tabeli 4.5 in 4.6 prikazujeta klasifikacijske točnosti ter njihove razlike za testno množico Wine.

4.1.3 Glass Identification

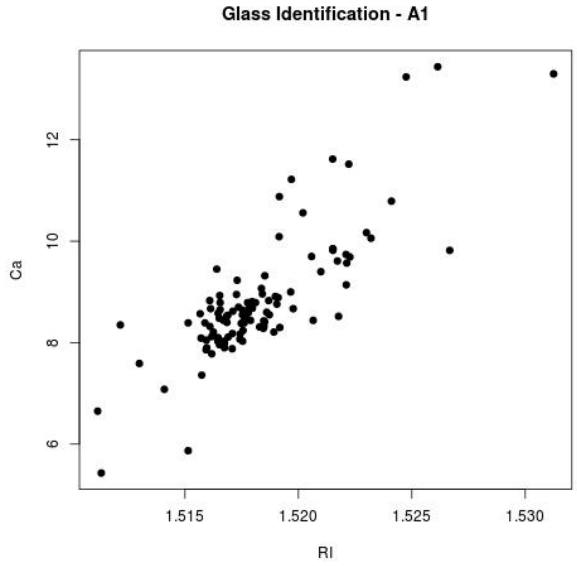
Napovedni model	Testna množica	Klasifikacijska točnost
MA1	A2	0.971963
MA1	B2	0.158879
MA1	A1	1.000000
MB1	A2	0.383178
MB1	B2	0.654206
MB1	B1	1.000000

Tabela 4.7: Klasifikacijske točnosti za podatkovno množico Glass Identification.

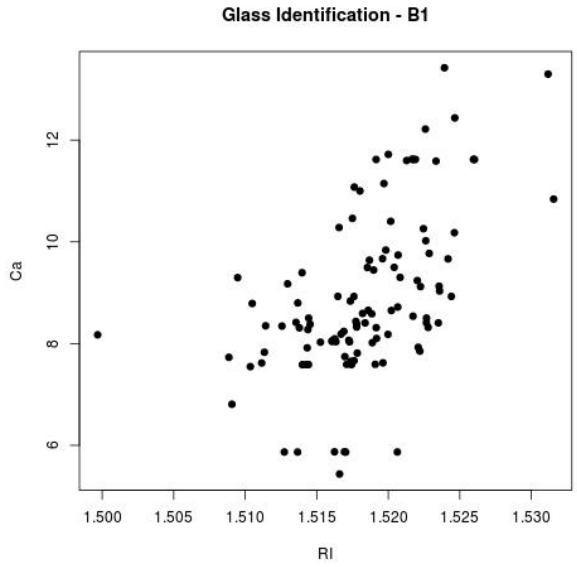
Razlika	Vrednost
RMA1	0.813084
RMB1	0.271028
RA2	0.588785
RB2	0.495327

Tabela 4.8: Razlike med klasifikacijskimi točnostmi.

Učna množica vsebuje 7 razredov, vendar eden izmed njih ni nikoli uporabljen. Razlike med klasifikacijskimi točnostmi so velike. Tekom generiranja množic `B1` in `B2` so bili inicializirani 3 skriti nevroni. Slabo delovanje generatorja bi lahko pripisali neenakomerni distribuciji razredov (najbolj množična razreda skupaj predstavljata 68 odstotkov celotne populacije). Slabše delovanje generatorja lahko opazimo pri sliki 4.2 (učna množica) in sliki 4.3 (novo generirana množica), ki prikazujeta razmerje med kalcijem in lomnim količnikom. Tabeli 4.7 in 4.8 prikazujeta klasifikacijske točnosti ter njihove razlike za testno množico Glass Identification.



Slika 4.2: Razpršeni prikaz za učno množico `A1`.



Slika 4.3: Razpršeni prikaz za novo kreirano množico `B1`.

4.1.4 Connectionist Bench (Sonar, Mines vs. Rocks)

Napovedni model	Testna množica	Klasifikacijska točnost
MA1	A2	0.846154
MA1	B2	0.721154
MA1	A1	1.000000
MB1	A2	0.701923
MB1	B2	0.923077
MB1	B1	1.000000

Tabela 4.9: Klasifikacijske točnosti za podatkovno množico Connectionist Bench.

Razlika	Vrednost
RMA1	0.125000
RMB1	0.221154
RA2	0.144231
RB2	0.201923

Tabela 4.10: Razlike med klasifikacijskimi točnostmi.

Pri kreiranju novih podatkovnih množic `B1` in `B2` sta bila kreirana po 2 skrita nevrona. Učna množica vsebuje 2 razreda. Razlike med klasifikacijskimi točnostmi so od 10 do 20 odstotne. Tabeli 4.9 in 4.10 prikazujeta klasifikacijske točnosti ter njihove razlike za testno množico Connectionist Bench.

4.1.5 Ionosphere

Napovedni model	Testna množica	Klasifikacijska točnost
MA1	A2	0.920000
MA1	B2	0.885714
MA1	A1	1.000000
MB1	A2	0.891429
MB1	B2	0.960000
MB1	B1	1.000000

Tabela 4.11: Klasifikacijske točnosti za podatkovno množico Ionosphere.

Razlika	Vrednost
RMA1	0.034286
RMB1	0.068571
RA2	0.028571
RB2	0.074286

Tabela 4.12: Razlike med klasifikacijskimi točnostmi.

Razlike med klasifikacijskimi točnostmi so relativno majhne. Število skritih nevronov, ki se kreirajo pri generiranju množic `B1` in `B2` je 3. Kot zanimivost bi lahko izpostavili, da je število skritih nevronov za posamezno množico večje kot je število razredov. Tabeli 4.11 in 4.12 prikazujeta klasifikacijske točnosti ter njihove razlike za testno množico Ionosphere.

4.1.6 German Credit Data

Napovedni model	Testna množica	Klasifikacijska točnost
MA1	A2	0.744000
MA1	B2	0.724000
MA1	A1	1.000000
MB1	A2	0.706000
MB1	B2	0.782000
MB1	B1	1.000000

Tabela 4.13: Klasifikacijske točnosti za podatkovno množico German Credit Data.

Razlika	Vrednost
RMA1	0.020000
RMB1	0.076000
RA2	0.038000
RB2	0.058000

Tabela 4.14: Razlike med klasifikacijskimi točnostmi.

Učna množica ima 2 razreda, tekom kreiranja novih podatkov sta bila za vsako množico inicializirana po 2 skrita nevrona. Razlike med klasifikacijskimi točnostmi so majhne. Tabeli 4.13 in 4.14 prikazujeta klasifikacijske točnosti ter njihove razlike za testno množico German Credit Data.

4.1.7 Ecoli

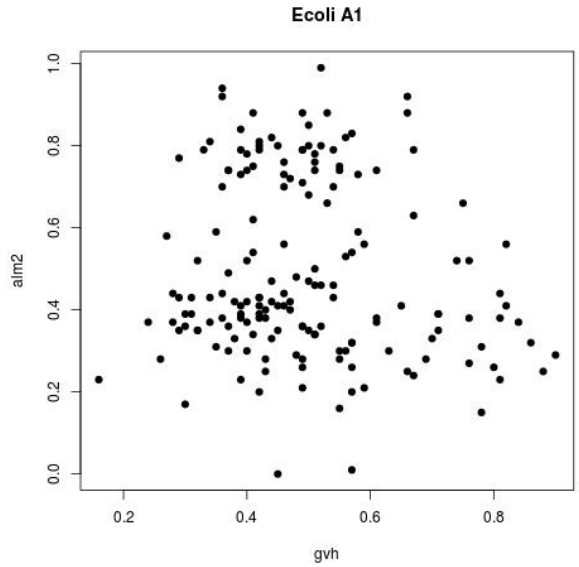
Napovedni model	Testna množica	Klasifikacijska točnost
MA1	A2	0.886905
MA1	B2	0.928571
MA1	A1	0.994048
MB1	A2	0.791667
MB1	B2	0.946429
MB1	B1	1.000000

Tabela 4.15: Klasifikacijske točnosti za podatkovno množico Ecoli.

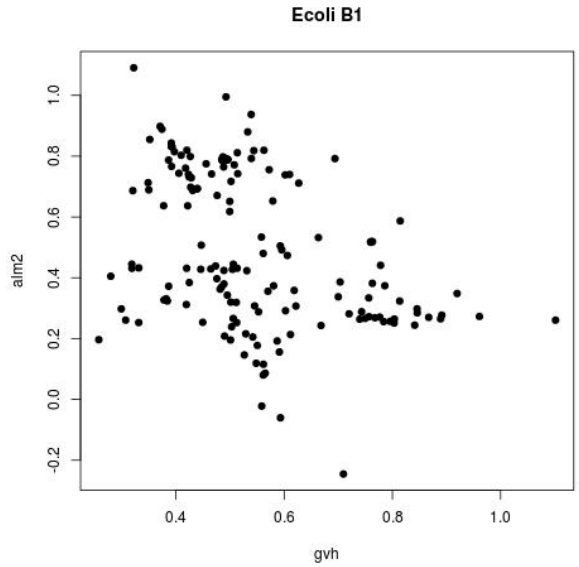
Razlika	Vrednost
RMA1	0.041667
RMB1	0.154762
RA2	0.095238
RB2	0.017857

Tabela 4.16: Razlike med klasifikacijskimi točnostmi.

Tekom generiranja novih množic `B1` in `B2` je bilo kreirano 8 skritih nevronov, prav takšno število razredov pa vsebuje učna množica. Razlike med klasifikacijskimi točnostmi se gibljejo med 4 in 15 odstotki. Na sliki 4.4 je razpršeni prikaz za učno množico, na sliki 4.5 pa za novo generirano množico. Tabeli 4.15 in 4.16 prikazujeta klasifikacijske točnosti ter njihove razlike za testno množico Ecoli.



Slika 4.4: Razpršeni prikaz učne množice Ecoli.



Slika 4.5: Razpršeni prikaz novo kreirane množice, ki smo jo generirali na podlagi množice Ecoli.

4.1.8 Congressional Voting Records

Napovedni model	Testna množica	Klasifikacijska točnost
MA1	A2	0.965517
MA1	B2	0.974138
MA1	A1	1.000000
MB1	A2	0.956897
MB1	B2	0.982759
MB1	B1	1.000000

Tabela 4.17: Klasifikacijske točnosti za podatkovno množico Congressional Voting Records.

Razlika	Vrednost
RMA1	0.008621
RMB1	0.025862
RA2	0.008621
RB2	0.008621

Tabela 4.18: Razlike med klasifikacijskimi točnostmi.

Učna množica vsebuje 2 razreda. Generator kreira po 2 skrita nevrona za množici `B1` in `B2`. Razlike med klasifikacijskimi točnostmi so minimalne, zato lahko delovanje generatorja za ta primer ocenimo kot dobro. Tabeli 4.17 in 4.18 prikazujeta klasifikacijske točnosti ter njihove razlike za testno množico Congressional Voting Records.

4.1.9 Pima Indians Diabetes

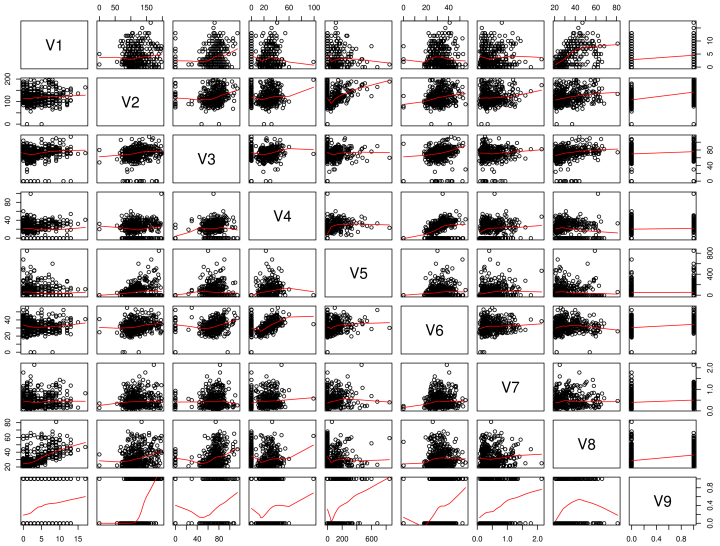
Napovedni model	Testna množica	Klasifikacijska točnost
MA1	A2	0.739583
MA1	B2	0.820312
MA1	A1	1.000000
MB1	A2	0.731771
MB1	B2	0.880208
MB1	B1	1.000000

Tabela 4.19: Klasifikacijske točnosti za podatkovno množico Pima Indians Diabetes.

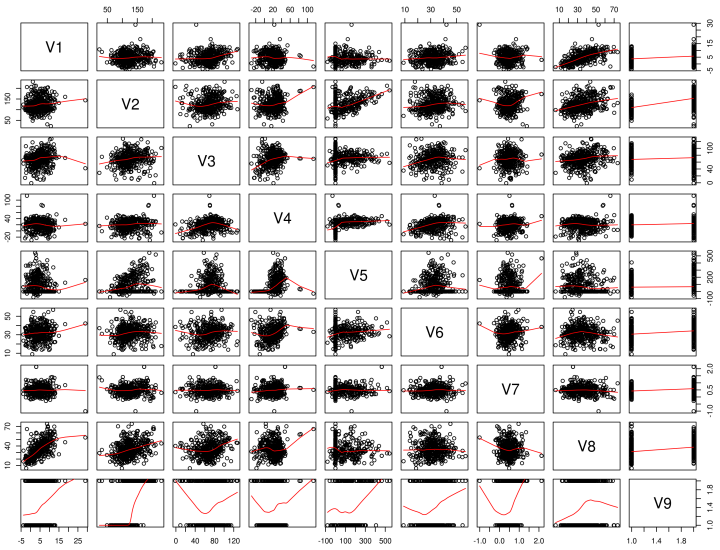
Razlika	Vrednost
RMA1	0.080729
RMB1	0.148438
RA2	0.007812
RB2	0.059896

Tabela 4.20: Razlike med klasifikacijskimi točnostmi.

Učna množica Pima Indians vsebuje 2 razreda. Tekom generiranja množic `B1` in `B2` sta bila kreirana po 2 skrita nevrona. Razlike med klasifikacijskimi točnostmi so od 5 do 15 odstotne. Razmerja med atributi so prikazana na sliki 4.6 za učno množico in sliki 4.7 za novo generirano množico. Tabeli 4.19 in 4.20 prikazujeta klasifikacijske točnosti ter njihove razlike za testno množico Pima Indians Diabetes.



Slika 4.6: Razpršeni prikazi učne množice Pima Indians Diabetes.



Slika 4.7: Razpršeni prikazi novo kreirane množice, ki smo jo generirali na podlagi množice Pima Indians Diabetes.

4.1.10 Titanic

Napovedni model	Testna množica	Klasifikacijska točnost
MA1	A2	0.532243
MA1	B2	0.990909
MA1	A1	0.530909
MB1	A2	0.515895
MB1	B2	0.970909
MB1	B1	0.971818

Tabela 4.21: Klasifikacijske točnosti za podatkovno množico Titanic.

Razlika	Vrednost
RMA1	0.458666
RMB1	0.455014
RA2	0.016349
RB2	0.020000

Tabela 4.22: Razlike med klasifikacijskimi točnostmi.

V primeru množice Titanic lahko opazimo slabo delovanje generatorja. Odstopanja med klasifikacijskimi točnostmi so velika. Generiranih je bilo 22 skritih nevronov. Prvotna množica vsebuje 4 razrede. Tabeli 4.21 in 4.22 prikazujeta klasifikacijske točnosti ter njihove razlike za testno množico Titanic.

Poglavje 5

Zaključek

V delu smo predstavili glavne značilnosti napovedenega modela PRBF, ki smo ga uporabili za generiranje novih podatkovnih množic temelječih na obstoječih realnih podatkih. Predstavili smo zasnovo generatorja, tehnologijo in orodja za implementacijo. Delo generatorja smo testirali na 10 različnih množicah.

Klasifikacijska točnost novih primerov je solidna. Za izboljšanje rezultatov bi bilo potrebno implementirati še spodnje rešitve.

Največja slabost aplikacije je pretvorba diskretnih atributov v numerične. Trenutna pretvorba diskretnih atributov v numerične predpostavlja, da imamo stolpični vektor, ki vsebuje m zapisov iz množice U. Množica U vsebuje od 1 do K unikatnih neštevilskih vrednosti. Vsak zapis v množici m pretvorimo v numerično vrednost, ki predstavlja točno določeno vrednost v množici U. Ta pretvorba je težavna, ko računamo razdalje med sredino Gaussovega jedra in točkami, zaradi napačnega ovrednotenja točk. Pretvorbo bi bilo potrebno spremeniti, da se nominalni diskretni atributi spremenijo v toliko binarnih atributov, kolikor imajo nominalni atributi različnih vrednosti.

V strojnem učenju je neodvisnost atributov zaželena lastnost, ki pa v praksi navadno ni vedno izpolnjena. Generator bi moral znati poiskati odvisnosti med atributi ter jih preprečiti. Algoritem, bi poiskal podprostore celotnega učnega prostora, ki bi vsebovali samo neodvisne atribute in jih združil v

novo učno množico, katera bi bila podana kot vhodni parameter v generator. Po končanem generiranju novih primerov bi algoritem na podlagi odvisnosti, ki jih je našel v prvotnih podatkih na novo kreiral odstranjene atribute.

Literatura

[1] (2011) Marko Robnik-Sikonja, Aristidis Likas, Constantinos Constantinopoulos, Igor Kononenko, Erik Strumbelj, *Efficiently Explaining Decisions of Probabilistic RBF Classification Networks*. Dostopno na: http://www.cs.uoi.gr/%7early/full/conferences/C69.pdf

[2] (2001) Michalis K. Titsias, Aristidis C. Likas, *Shared Kernel Models for Class Conditional Density Estimation*. Dostopno na: http://www.cs.uoi.gr/%7early/papers/TNN2001.pdf

[3] (2006) Constantinos Constantinopoulos, Aristidis Likas, *An Incremental Training Method for the Probabilistic RBF Network*. Dostopno na: http://www.cs.uoi.gr/%7early/papers/TNN2006.pdf

[4] (2012) Spletna stran okolja GNU Octave. Dostopno na: http://www.gnu.org/software/octave/

[5] (2012) Spletna stran okolja R. Dostopno na: http://www.r-project.org

[6] (2011) Michael J. Crawley. *Statistics: An Introduction using R*. Založba John Wiley & Sons, Ltd.

[7] (2012) Spletna stran paketa RcppOctave. Dostopno na: http://cran.r-project.org/web/packages/RcppOctave/index.html