Juraj Holas

Domáca úloha č. 1

**1.a) Maticový počet**

Označme si počet príkladov trénovacej množiny , a počet atribútov . Potom a . Vo všeobecnosti nemôžeme rátať s tým, že , a teda matice a nie sú vždy štvorcové. Pre neštvorcové matice sa však nedá vytvoriť inverzná matica (Lema 1). Chyba v postupe prof. Premúdrelého sa teda vyskytla v kroku:

kde počíta s maticami a , ktoré však môžu existovať iba v prípade, keď , rozhodne však nie vo všeobecnosti.

Lema 1: Ku štvorcovej matici neexistuje inverzná

Pre každú maticu platí, že ak k nej existuje inverzná matica , tak platí vzťah:

kde je jednotková matica veľkosti .

Majme maticu a predpokladajme, že k nej existuje inverzná matica . Musí teda existovať také aby platilo . Notáciou v nasledujúcom postupe označujem veľkosti násobených matíc a ich súčinu. Využitím pravidiel násobenia matíc dostaneme:

čiže platí:

čiže platí:

Neexistuje teda také , aby platil vzťah , čím sme dospeli k sporu. □

**1.b) Bonus**

Odvodenie prof. Premúdrelého (v danej formulácii) platí iba ak existuje , a teda , čo sme ukázali v predchádzajúcej časti. (Presnejšie by sme mali uvádzať podmienku aby sa zaručila regulárnosť dizajnovej matice, tú však vieme v prípade potreby docieliť aj pridaním vhodného mierneho šumu k vstupným dátam. Budeme teda uvažovať že . )

Po malej úprave v postupe však môžeme rozšíriť pomerne striktné podmienky, ak namiesto inverznej matice použijeme ľavú inverznú maticu .

Z definície vyplýva, že pre maticu existuje také, že iba ak . Jednoduchým odvodením z dostaneme vzťah .

Podmienka v prípade dizajnovej matice znamená, že musí platiť . S týmto na zreteli môžeme odvodiť:

V poslednom kroku odvodenia bol použitý vzťah . Dostali sme teda vzťah analogický ku tomu od prof. Premúdrelého, avšak s voľnejšou podmienkou .

Správnosť tohto vzťahu som overil aj implementačne (zdrojové kódy sú súčasťou prílohy). Princíp programu spočíval v porovnávaní výsledných vektorov získaných klasickým výpočtom a našim vzťahom. Pre demonštráciu aká veľká odchýlka vznikne iba samotným zaokrúhľovaním desatinných čísel počas výpočtu som doplnil aj tretí výpočet, matematicky ekvivalentný s klasickým postupom. Trénovanie teda prebiehalo podľa troch vzorcov:

Počet atribútov sa vyberal náhodne z intervalu a veľkosť trénovacej množiny z intervalu , čím sa zaručila splnenosť podmienky . Trénovacie dáta boli vyberané náhodne, rovnako bol z intervalu vybrané aj hodnoty vektora koeficientov lineárnej kombinácie atribútov, ktorý určovali „nameraný“ výsledok . Celý postup sa následne opakoval tisíc krát, a vypočítala sa priemerná odchýlka vektora resp. voči referenčnému vektoru . Odchýlka bola počítaná podľa noriem a .

Výsledky:

|  |  |  |
| --- | --- | --- |
|  | norma | norma |
| odchýlka | 2.5226e-12 | 1.6823e-24 |
| demonštračná odchýlka | 6.7175e-13 | 1.1019e-25 |

Podľa tabuľky vidíme, že odchýlky od vektora sú skutočne minimálne. Navyše výsledky podľa vzťahu 2 majú takmer rovnakú odchýlku ako podľa vzťahu 3. (ktorý je ekvivalentný s referenčným vzťahom 1), môžeme teda konštatovať že všetky odchýlky vznikli iba zaokrúhľovaním počas výpočtu. Preto môžeme náš vzťah 2. považovať za korektný.

**2.) Pevnosť betónu v tlaku**

*Pozn.: zdrojové kódy k programu nájdete v prílohe na konci dokumentu.*

**e)** Vysvetlite priebehy a rozdiely grafov z častí c) a d)

|  |  |
| --- | --- |
| Graf - priemerná trénovacia chyba pri max. d=3 | Graf - priemerná testovacia chyba pri max. d=3 |

Na grafoch je možné prakticky vidieť niekoľko aspektov kriviek strojového učenia. Ukazujú ako sa správajú priemerné trénovacie a testovacie chyby pri zväčšujúcej sa trénovacej množine, či pri zložitejšej triede hypotéz. Môžeme si všimnúť niekoľko korelácií:

1. Čím viac príkladov, tým väčšia trénovacia chyba. Tento trend si môžeme všimnúť na reze grafu 1, napr. rovinou . Je to spôsobené tým, že hľadaná krivka musí aproximovať viac bodov. Je teda komplikovanejšie takúto krivku nájsť, a vo výsledku má aj väčšiu priemernú odchýlku.
2. Čím viac príkladov, tým menšia testovacia chyba. Tento trend si môžeme všimnúť na reze grafu 2, napr. rovinou . Tu je dôvod prostý – vďaka viac trénovacím príkladom náš regresný algoritmus lepšie odhadol hľadanú krivku.
3. Čím zložitejšia množina hypotéz, tým menšia trénovacia chyba. Tento trend si môžeme všimnúť na reze grafu 1, napr. rovinou . Dôvodom je, že polynómy vyšších rádov vedia lepšie aproximovať aj zložitejšie rozmiestnené dáta. Keby sme vzali stále komplikovanejšie množiny hypotéz, napr. polynómy vyšších stupňov, vedeli by sme stlačiť priemernú trénovaciu chybu pod ľubovoľne malé .
4. Čím zložitejšia množina hypotéz, tým menšia testovacia chyba (do istého bodu). Tento trend si môžeme všimnúť na reze grafu 2, napr. rovinou . Dôvodom je podučenie nášho regresného algoritmu pri príliš jednoduchých funkciách v množine hypotéz. Pri hodnotách nanajvýš 3 vidíme, že testovacia chyba sa s rastúcim neustále zmenšuje. Ľahko však zistíme, že práve 3 je optimálna hodnota pre , a že pri ďalšom zväčšovaní začne testovacia chyba už narastať – nastane preučenie. To si môžeme všimnúť na grafe 4 (dole), rezanom rovinou . Ten ukazuje výsledky získané pri maximálnej hodnote .

|  |  |
| --- | --- |
| Graf 3 - priemerná trénovacia chyba pri max. d=5 | Graf 4 - priemerná testovacia chyba pri max. d=5 |

Príloha: zdrojové kódy k úlohám

*Všetky zdrojové súbory sú dostupné aj online:* http://davinci.fmph.uniba.sk/~holas3/\_etc/HolasDU1.zip

**1.b) Bonus**

errL1prem = 0;

errL2prem = 0;

errL1demo = 0;

errL2demo = 0;

maxIter = 1000; % 1000 interacii

for i = 1:maxIter

% priprava trenovania

n = floor(1 + rand(1)\*19); % pocet atributov 1..20

t = floor(20 + rand(1)\*980); % pocet prikladov 21..1000 (vzdy teda plati t >= n)

X = rand(t, n); % dizajnova matica

coefs = -100 + rand(n, 1)\*200; % koeficienty -100.0..+100.0

y = X \* coefs; % vektor vysledkov

% trenovanie klasickym sposobom

tetha = (X' \* X) \ (X' \* y);

% trenovanie premudrelym sposobom

Xl = inverse(X' \* X) \* X'; % lava inverzna matica k X

tethaPrem = Xl \* y;

% trenovanie klasickym sposobom pre demonstraciu numerickych nepresnosti

Xtrans = ((X'\*5)\*0.2); % realne plati X' = ((X'\*5)\*0.2), kvoli presnosti nie

tethaDemo = (Xtrans \* X) \ (X' \* y);

% porovnanie odchylky

for j = 1:n

errL1prem += abs(tetha(j) - tethaPrem(j));

errL1demo += abs(tetha(j) - tethaDemo(j));

endfor

errL2prem += (tetha - tethaPrem)'\*(tetha - tethaPrem);

errL2demo += (tetha - tethaDemo)'\*(tetha - tethaDemo);

endfor

errL1prem /= maxIter;

errL2prem /= maxIter;

errL1demo /= maxIter;

errL2demo /= maxIter;

disp('Priemerna odchylka postupu prof. Premudreleho podla normy L1:');

disp(errL1prem);

disp('Priemerna demonstracna odchylka podla normy L1:');

disp(errL1demo);

disp('Priemerna odchylka postupu prof. Premudreleho podla normy L2:');

disp(errL2prem);

disp('Priemerna demonstracna odchylka podla normy L2:');

disp(errL2demo);

**2) Pevnosť betónu v tlaku**

Kvôli prehľadnosti kódu bol program rozdelený do dvoch súborov. Hlavný program:

source('functions.m');

% nastavenie parametrov

d = 3;

data = dlmread('Concrete\_Data.csv', ';');

% vybratie trenovacej a testovacej mnoziny (poduloha a)

[rows, cols] = size(data);

randOrder = randperm(rows);

Ttrain = data(randOrder(1:800), :);

Ttest = data(randOrder(801:end), :);

% spocitanie trenovacej a testovacej chyby

[trainErrs, testErrs] = computeErrors(Ttrain, Ttest, d, 100:100:800);

% vykreslenie grafov (poduloha c, d)

surf(trainErrs);

view(130, 30);

xlabel('\*100 = |T|'); ylabel('d'); zlabel('chyba (lin. mierka)');

print -dpng trainErrs.png;

surf(testErrs);

set(gca, 'zscale', 'log');

view(130, 30);

xlabel('\*100 = |T|'); ylabel('d'); zlabel('chyba (log. mierka)');

print -dpng testErrs.png;

% vypisanie (poduloha c, d)

format bank;

trainErrs = [(0:d)', [100:100:800; trainErrs]]

testErrs = [(0:d)', [100:100:800; testErrs]]

Súbor functions.m:

1; % aby to Octave nebral ako function file

function [trainErrs, testErrs] = computeErrors(Ttrain, Ttest, dMax, tOptions)

trainErrs = [];

testErrs = [];

for d = 1:dMax

tNum = 1;

for t = tOptions

% vyber podmnoziny trenovacich prikldov, separacia atributov

X = Ttrain(1:t, 1:8);

y = Ttrain(1:t, 9);

% trenovanie

tetha = linregd(X, y, d);

% spocitanie trenovacej chyby na jeden priklad (poduloha c)

yPredicted = h(X, d, tetha);

err = ((yPredicted - y)' \* (yPredicted - y)) / t;

trainErrs(d, tNum) = err;

% spocitanie testovacej chyby na jeden priklad (poduloha d)

X = Ttest(:, 1:8);

y = Ttest(:, 9);

yPredicted = h(X, d, tetha);

err = ((yPredicted - y)' \* (yPredicted - y)) / t;

testErrs(d, tNum) = err;

tNum++;

endfor

endfor

endfunction

% poduloha b

function tetha = linregd(X, y, d)

Phi = generatePhi(X, d);

tetha = (Phi' \* Phi) \ (Phi' \* y);

endfunction

function result = h(X, d, tetha)

Phi = generatePhi(X, d);

result = Phi \* tetha;

endfunction

% ostatne

function Phi = generatePhi(X, d)

[t, n] = size(X);

exponents = exponents(d, n);

Phi = [];

for i = 1:t

Phi = [Phi; prodPowerRow(X(i,:), exponents)];

endfor

endfunction

function result = prodPowerRow(x, exponents)

x = [1, x];

[rows, cols] = size(exponents);

result = [];

for i = 1:rows

result = [result, prod(x .^ exponents(i,:))];

endfor

endfunction

function result = exponents(d, n)

if n == 0

result = [d];

return;

endif

result = [];

for i = 0:d

nextExps = exponents(d-i, n-1);

[rows, cols] = size(nextExps);

result = vertcat(horzcat(ones(rows, 1) \* i, nextExps), result);

endfor

endfunction