VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
MATEMATINĖS STATISTIKOS KATEDRA

Vilius Kucinas (parašas)

BIOJUTIKLIŲ ATSAKO KREIVIŲ IR MEDŽIAGŲ KONCENTRACIJŲ REGRESINĖ ANALIZĖ
REGRESSION ANALYSIS OF BIOSENSOR RESPONSE CURVES AND LIQUER THICKNESS

Magistrinis baigiamasis darbas

Vilnius, 2009

Darbo vadovas:

Matematikos mokslų daktaras, docentas Pranas Vaitkus (parašas)

Recenzentas:

(parašas)

Registracijos Nr.: _____

Darbo gynimo data: _____

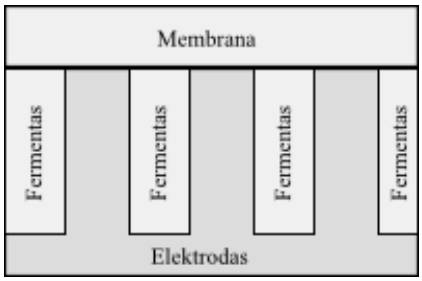
T U R I N Y S

I. ĮVADAS

I.1. Biojutiklis

Biojutiklis yra prietaisas, skirtas medžiagų koncentracijoms matuoti (Wollenberger ir kt., 1997). Jo veikimas pagrįstas fermentine reakcija ir medžiagų difuzija (Scheller, Schubert 1992; Turner ir kt., 1987). Biojutiklio pagrindą sudaro elektrodas ir jo paviršiuje imobilizuotas fermentas. Biojutiklį panardinus į tirpalą, tiriamoji medžiaga – substratas difunduoja į fermentą, fermentas savo ruožtu katalizuoja reakciją, kurioje substratas virsta produktu. Susidaręs produktas elektrodo paviršiuje dalyvauja elektrocheminėje reakcijoje, kurioje išlaisvinami elektronai. Taip sužadinama elektros srovė, ji vėliau yra stiprinama ir vaizduojama biojutiklio naudotojui. Amperometriniuose biojutikliuose vykstantys biocheminiai procesai yra verčiami elektros srove, kurios stiprumas yra tiesiog proporcingas reakcijos produkto koncentracijai (Chaubey, Malhotra, 2002).

Sudėtingesnių analitinių sistemų biojutiklių sandara yra sudėtingesnė (Laurinavičius ir kt., 2004). Jie konstruojami naudojant sudėtingos geometrijos elektrodus, selektyvias ir perforuotas membranas. Vieno tokio biojutiklio profilis pavaizduotas 1 pav. (Baronas ir kt., 2006).



I.1 pav. Biojutiklio, sudaryto iš elektrodo, kurio įpjovos užpildytos fermentu, ir selektyvios membranos, struktūra

Procesas, vykstantis fermente, aprašomas reakcijos ir difuzijos lygtimis su netiesiniu Michaelio–Menteno reiškiniu, aprašančiu fermentinę reakciją

$$\frac{\partial S}{\partial t} = D_S \Delta S - \frac{V_{\max} S}{K_M + S}, \quad \frac{\partial P}{\partial t} = D_P \Delta P + \frac{V_{\max} S}{K_M + S}; \tag{I.1}$$

čia S ir P yra fermentinės reakcijos substrato ir produkto koncentracijos, DS ir DP – substrato ir produkto difuzijos koeficientai fermente, Vmax – maksimalus reakcijos greitis, KM – Michaelio konstanta (Schulmeister, 1990). Tokio pavidalo lygčių analiziniai sprendiniai yra žinomi tik ribiniais atvejais, kai substrato koncentracija tiriamajame tirpale yra labai maža arba labai didelė. Tuomet lygtyje galima panaikinti netiesiškumą.

Bendriausiu atveju biojutiklius modeliuojančias lygtis tenka spręsti skaitiniais metodais (Britz, 2005). Šiame darbe (1) pavidalo lygtys sprendžiamos baigtinių skirtumų metodu (Samarskii, 2001).
Norint modeliuoti biojutiklio veikimą kompiuteriu, visų pirma reikia sudaryti konkretaus biojutiklio matematinį modelį. Tada matematinis modelis keičiamas skirtuminiu ir kuriama programinė įranga, įgyvendinanti skirtuminį biojutiklio modelį (Baronas ir kt., 2003; Bieniasz, Britz, 2004). Šiame straipsnyje pateikiama sudėtinės geometrijos biojutiklių modelių ir juos sprendžiančios programinės įrangos automatizuoto sudarymo metodika.

I.2. Tyrimo tikslas ir uždaviai

Šio darbo tikslas – sukonstruoti netiesinės regresijos modelį su optimaliu koeficientų skaičiumi bei optimaliomis tų koeficientų reikšmėmis, kurio pagalba būtų galima nustatyti tirpalo koncentracijas pagal amperometrinio biojutiklio atsako kreives.

Darbui buvo keliami šie uždaviniai:

- Modelio kūrimui, panaudoti nepilnai apibūdintų stebėjimų analizės metodiką (angl. Semi-supervised learning). **(Kai Yu, Volker Tresp [4])**
- Pritaikant **P.L.Bartlet [1]** darbo idėją, jog netiesinės regresijos modelyje svorių dydis yra svarbiau nei narių kiekis, rasti netiesinės regresijos optimalius svorių dydžius.
- Sukonstruoti matematinį modelį, kuris stochastinės paieškos būdu suskaičiuotų optimalius netiesinės regresijos svorius, bei nustatytų tirpalų koncentraciją pagal biojutiklio atsako signalus.

I.3. Darbo aktualumas ir jo reikšmė

Konstruojant modelį, susiduriama su optimalios architektūros parinkimo problema. Optimalus modelis reiškia optimalų netiesinės regresijos komponenčių skaičių ir svorių reikšmes su kuriomis modelis „geriausiai“ prognozuoja tirpalo koncentracijas. Optimaliam komponenčių skaičiui nustatyti pasitelkiame pažingsninės regresijos metodą. Skaičiavimams buvo pasirinkta netiesinės regresijos sigmoidinė aktyvacijos funkcija, šis pasirinkimas buvo pagrįstas P.L. Bartlet [1] darbo rezultatais.

Modelyje taip pat taikomas nepilnai apibūdintų duomenų analizės metodas, pagrįstas **Volker Tresp [3]** darbo rezultatais. Šis metodas yra plačiai taikomas šiuolaikinėje informacinėse technologijose bei matematinių modelių konstravime, kai turima pilna informacija apie modelių įėjimus tačiau išėjimo duomenys yra žinomi ne visais atvejais. Mūsų atveju, yra žinomi tikslūs amperometro parodymai – srovės stipris tam tikru laiko momentu (realybėje šiuos duomenis nesunku pamatuoti), tačiau duomenys apie tikslias tirpalų koncentracijas yra žinomi ne visi.

Panaudojome pagrindinių komponenčių analizės metodą (PCA) ir sumažinome turimų duomenų rinkinius, taip išsaugodami didžiąją dalį informacijos apie tuos duomenis.

Tam tikri modelio parametrai : λ , σ buvo parenkami bandymų keliu, stebint modelio vidurkių kvadratų sumą(*MSE*).

Gauti rezultatai rodo, kad sukurtas regresijos modelis pakankamai gerai klasifikuoja duomenis, tačiau tiksliems koncentracijų prognozavimams reikalingi papildomi tyrimai.

Viena iš problemų su kuriomis susiduriama, nagrinėjant realius duomenis, yra atsako kreivių matavimo dydžiai, kurių pokyčiai yra labai dideli (pvz. tirpalo minimalus srovės stipris 1.21* 10^{-19} ir maksimalus 7.6* 10^{-7}) bei didelis matavimų kiekis ženkliai padidina skaičiavimo laiko sąnaudas.

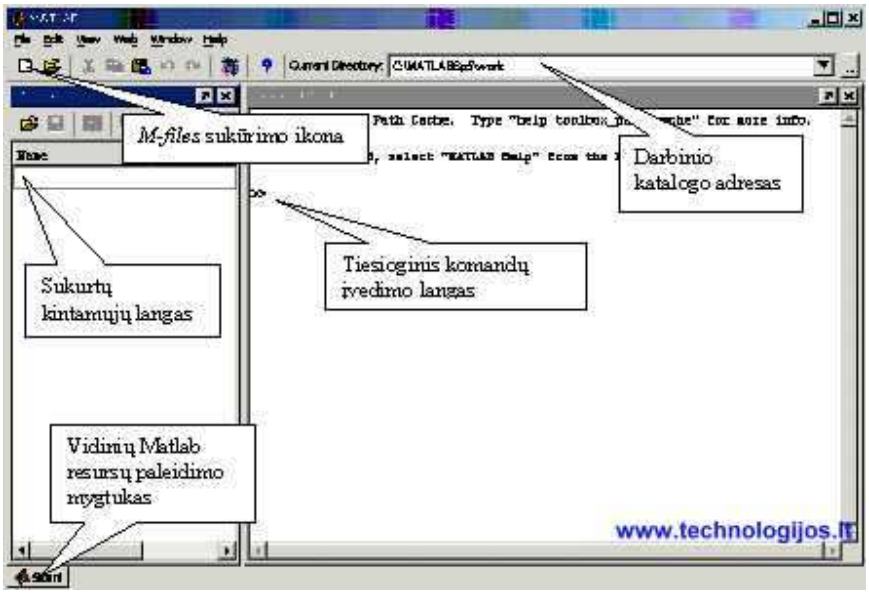
II. PAŽINTIS SU MATLAB SIMULIACINE PROGRAMA

Matlab - tai viena seniausių ir galingiausių specialios paskirties kompiuterinių programų, skirtų automatizuoti mokslinius ir inžinierinius skaičiavimus. Darbas su šia galinga programa nėra sudėtingas, ypač turintiems bent minimalius programavimo įgūdžius. Matlab yra programavimo kalba, naudojama algoritmams koduoti. Operatorių seką galima surašyti į failą ir vykdyti kaip vieną komandą. Programa saugoma tekstiniame faile su prievardžiu „.m“, kuris sukuriamas „File“ meniu komanda: New→M-file. M-failas gali būti tiek scenarijus (script), kuriame saugoma Matlab operatorių seka, tiek funkcija (function), priimanti argumentų reikšmes ir grąžinanti rezultatus.

Matlab terpėje galima dirbti dviem rėžimais:

- Tiesiogiai darbiniame lange arba dar vadinamame kalkuliatoriuje. Komandos įvedamos atskirai. Padarius komandų įvedimo klaidą viską reikia kartoti nuo pradžių. Komandos vykdomos iš karto, o galutinis skaičiavimo rezultatas nepateikiamas iki tol, kol neaptinkama komanda be kabliataškio.
- M-files kūrimo režimu. M-files saugomos vartotojo sukurtos ir vidinės Matlab funkcijos. Skirtingai nuo darbinio lango M-files turi galimybę programą redaguoti, komentuoti, taisyti klaidas, daryti pakeitimus ir išsaugoti kietame diske. M-files vykdomi tik jį sukompiliavus arba padarius tieisioginį kreipimąsį iš darbinio lango arba kitos programos. Kadangi M-files nevykdoma iš karto, klaidos juose aptinkamos tik po kompiliavimo.

Įjungus Matlab programą, ekrane parodomas pagrindinis jos langas (1 pav.).



II.1 pav. Matlab programos pagrindinis langas

Kadangi dirbant tiesioginiam komandų įvedimo režimu nėra galimybės programos teksto išsaugojimui ir redagavimui, dažniausiai naudojamas *M-files* redagavimo langas. Tiesioginis komandų įvedimo langas panaudojamas gauti informacijai apie funkcijas arba tiesioginiam ir greitam funkcijos išbandymui.

Pavyzdžiui norint gauti informaciją apie žinomą funkciją, rašoma komanda help funkcijos_pavadinimas:

```
>> help abs
ABS Absolute value.
ABS(X) is the absolute value of the elements of X. When
X is complex, ABS(X) is the complex modulus (magnitude) of
the elements of X.
See also SIGN, ANGLE, UNWRAP.
Overloaded methods
help iddata/abs.m
help sym/abs.m
>>
```

Kai iš help failo nelabai aišku funkcijos pritaikymas komandinėje eilutėje patogu pasitikrinti funkcijos veikimą:

```
>> abs(-5)
ans =
5
>> abs(4+i*3)
ans =
5
>>
```

Jei nežinomas konkretus funkcijos pavadinimas, pasinaudoti tiesiogiai komanda help funkcijos_pavadinimas nepavyks. Tokiu atveju reikia pasileisti Matlab programos Help langą ir pagal raktinius žodžius atlikti paiešką.

Programavimas Matlab terpėje atliekamas M-files redagavimo lange, kuris paleidžiamas paspaudus ikoną New M-files (1 pav.) arba pasirinkus meniu punktą File–>New–>M-file. Atsidaro naujas redagavimo langas (2 pav.)

II.2 pav. M-file redagavimo langas

Parašius programą ją sukompiliuoti ir įvykdyti galima keliais būdais:

- paspaudus kompiliavimo ikoną (2 pav.);
- pasirinkus komandą Debug–>Run;
- paspaudus klavišą F5;
- komandinėje eilutėje surinkus M-files pavadinimą.

Matlab programa gali dirbti tik su tomis programomis, kurios patalpintos į jo darbinį katalogą, kurio adresas matomas pagrindiniame programos lange (1 pav.). Jei bandome kreiptis į M-file, kuris patalpintas ne darbiniame kataloge, Matlabas sugeneruos klaidos pranešimą:

```
>> testine
??? Undefined function or variable 'testine'.
>>
```

Todėl M-file reikia saugoti arba darbiniame kataloge (pagal nutylėjimą katalogas work), arba bet kokiame kitame kataloge, tik tuomet reikia Matlab terpėje nurodyti naują darbinį katalogą. Tai atliekama dviem būdais:

- Automatinis darbinio katalogo nustatymas. Šiuo atveju M-file išsaugomas bet kokiame kataloge ir vykdomas failo kompiliavimas (F5 klavišas, kompiliavimo įkona arba meniu punktas). Matlab programa išmeta lentele, kurioje prašoma pasirinkti darbinį katalogą (3 pav.). Atsiradusiame lange nustatymai paliekami pagal nutylėjimą (Change MATLAB current directory) ir spaudžiamas mygtukas OK. Darbinė direktorija pakeičiama į tą, kurioje išsaugotas kompiliuojamas M-file.

3 pav. Darbinio katalogo pasirinkimas M-file kompiliavimo metu

- Rankinis darbinio katalogo nustatymas. Tai galima atlikti pagrindiniame Matlab programos lange skiltyje Current Directory (1 pav.). Šioje skiltyje reikia nurodyti tikslų adresą iki naujo darbinio katalogo.

II.1 Programavimas Matlab aplinkoje

Programos rašomos M-file redagavimo lange. Patartina programos pradžioje parašyti trumpą komentarą apie ką yra ta programa. Matlab aplinkos sintakse labai panaši į C kalbos sintaksę, su specifinėm Matlab galimybėm arba ribojimais. Dažniausiai naudojami programavimo elementai:

- Komentarai rašomi po ženklo % :

```
%testine programa demonstruoti darbui su Matlab aplinka
```

- Reikšmės priskyrimas kintamajam su lygybės ženklu = :

```
k=5
```

Aritmetinės operacijos: +, -, *, / ;

Palyginimo operacijos <, >, <=, >=,~= (nelygu) ,== ;

Trigonometrinės funkcijos sin(α), cos(α), tan(α):

```
>> k=sin(30*pi/180)
k =
0.5000
>>
```

Sisteminiai kintamieji:

i, j - kvadratinė šaknis iš –1 (menamas 1);

pi – skaičius 3.1415926…;

realmin – mažiausias skaičius (2-1022);

realmax – didžiausias skaičius (21023);

inf – mašininė begalybė;

ans – kintamasis, kuriame saugoma paskutinė apskaičiuota reiškinio reikšmė.

išvedamų rezultatų formatą galima parinkti komanda format. Žemiau pateikta lentelė iliustruoja kai kuriuos komandos format naudojimo pavyzdžius, išvedant išraiškos 100/3 rezultatą:

FORMAT SHORT **33.3333**
FORMAT SHORT E **3.3333E+001**
FORMAT LONG **33.33333333333334**
FORMAT LONG E **3.333333333333334E+001**
FORMAT SHORT G **33.333**
FORMAT LONG G **33.3333333333333**
FORMAT HEX **4040AAAAAAAAAAB**

```
>> format short e
>> 100/3
ans =
3.3333e+001
>>
```

sekų formavimas: pradinė_reikšmė:žingsnis:galutinė_reikšmė:

```
>> k=1:2:10
k =
1 3 5 7 9
>>
```

jei žingsnis nenurodomas, Matlab traktuoja jį kaip 1:

```
>> k=1:10
k =
1 2 3 4 5 6 7 8 9 10
>>
```

grafinis duomenų atvaizdavimas:

plot - sukuriamas vektoriaus ar matricos stulpelių grafikas;

fplot - sukuriamas grafikas, kai yra nurodyta funkcijos analitinė išraiška;

stem - sukuriamas vektoriaus ar matricos stulpelių taškinis grafikas;

title - įterpiamas grafiko pavadinimas;

xlabel - įterpiamas x ašies pavadinimas;

ylabel - įterpiamas y ašies pavadinimas;

text - nurodytas tekstas įterpiamas į grafiką;

gtext - nurodytas tekstas įterpiamas į grafiką naudojant pelę;

grid - įjungiamas arba išjungiamas koordinačių tinklelis;

figure - sukuriamas arba suaktyvinamas grafinis langas;

II.2. Ciklinės ir sąlygos struktūros Matlab terpėje

Ciklinę struktūrą Matlab terpėje galima suformuoti dviem būdais – panaudojant ciklus for ir while.

for ciklo struktūra:

```
for i=pradine_reiksme:galine_reiksme
operacija 1;
operacija 2;
....
operacija n;
end
```

```
for i=1:10 %ciklo formavimas
   x(i,1)=5*i; %dvimacio masyvo formavimas
end %ciklo pabaiga
```

while ciklo struktūra:

```
while i<maksimali_reiksme
x(i,1)=5*i;
i=i+1;
end
```

Naudojant while ciklą negalima pamiršti ciklo skaitliuko didinimo operacijos, kitaip gausis amžinas ciklas!

```
while i<10 %ciklo formavimas
   x(i,1)=5*i; %dvimacio masyvo formavimas
   i=i+1; %ciklo skaitliuko didinimas
end %ciklo pabaiga
```

Sąlygos struktūros formavimui dažniausiai naudojama komanda **if**. Jos bendra struktūra:

```
if lyginamas_dydis < ribine_reiksme
    operacija 1;
  operacija 2;
  ..
  operacija n;
else
  operacija 1
  operacija 2
  ..
```

```
  operacija n;
end;
```

Galima naudoti dalinę if komandą:

```
if lyginamas_dydis < ribine_reiksme
  operacija 1;
  operacija 2;
  ..
  operacija n;
end;
```

```
if i<12 %salyga
  x(i,1)=0; %operacija, jei salyga tenkinama
else
  x(i,1)=10; %operacija, jei s1lyga netenkinama
end; %salygos pabaiga
```

III. TIESINĖ PAŽINGSNINĖ REGRESIJA

Dažnai, taikydami regresinę analizę, eksperimentatorius neturi pakankamai informacijos apie tai, kurie iš kintamųjų X_1 ……,X_m labiau tinka a. d. Y prognozuoti. Be to, galutinai parinktas modelis yra tuo vertingesnis, kuo mažesniu skaičiumi kovariančių jis nusakomas. Suprantama, stengiamasi, kad kintamųjų X_1,…, X_m skaičiaus sumažnimas iš esmės nepablogintų prognozės tikslumo. Šiame modelyje pritaikėme vieną iš paprastesnių ir palyginti dažnai taikomų metodų. Sakykime, vektorius Y, X_1,…,Xm pasiskirstęs pagal (m + 1)-matį normalųjį skirstinį. Stebint šį a. v. gauta turio n imtis $(Y_i, X_{1i},\ldots, X_{mi})^T$; i = 1,…, n. Kadangi tiesinio prognozavimo tikslumo matas yra suvestinio koreliacijos koeficiento kvadratas, tai būtų natūralu parinkti tokį kovariančių rinkinį, kad suvestinio koreliacijos koeficiento įvertinys būtų kuo didesnis. Tačiau tokio metodo negalima realizuoti praktiškai, jeigu kovariančių skaičius gana didelis. Pavyzdžiui, jeigu m = 10 , tai galima sudaryti 2^{10}-1 = 1023 skirtingų tiesinės regresijos lygčių (neatsižvelgiant į galimas kovariančių interakcijas). Vienas iš paprasčiausių ir dažniausiai praktiškai taikomų metodų kovariančių rinkiniui parinkti yra pažingsninė regresija.

N u l i n i s ž i n g s n i s. Parenkame du reikšmingumo lygmenis P ir P’; randame koreliacijos koeficientų tarp a. d. Y ir X_i įverčius r(Y; X_i) = r_{YX_i}=r_{0i} ; i = 1,…,m.

P i r m a s i s ž i n g s n i s. Išrenkame maksimalų įvertį

$$\max_{1\le i\le m}|r_{0i}| = |r_{0i_1}|$$

Kintamasis X_{i1} įtraukiamas į kintamųjų Y prognozuoti sąrašą, jeigu teisinga nelygybė

$$F_{0i1} = r^2_{0i1}(n-2)/(1-r^2_{0i1}) > F_p(1,n-2)$$

$F_p(1,n-2)$ yra Fišerio skirstinys su 1 ir *n*- 2 laisvės laipsniais ir kritine reikšme P.

Priešingu atveju daroma išvada, kad Y prognozavimas pagal X_1,.., X_m negalimas.

A n t r a s i s ž i n g s n i s. Randame dalinių koreliacijos koeficientų įverčius

$$rY\ X_i\ (X_{i1}) = r_{0i}\ (i1)$$

ir iš jų išrenkame maksimalų:

$$\max_{i\ne i_1}|r_{0i(i_1)}| = |r_{0i_2(i_1)}|$$

Kintamasis X_{i2} įtraukiamas į sąrašą, jeigu teisinga nelygybė

$$r^2{}_{0i_2(i_1)}(n-3)/(1-r^2{}_{0i_2(i_1)}) > F_p(1,n-3) \quad (2.1)$$

Priešingu atveju sąraše paliekamas tik kintamasis X_{i1} .

Jeigu kintamasis X_{i2} į sąrašą įtrauktas, tai tikrinama, ar nereikia išbraukti iš sąrašo X_{i1} , kai į jį įrašytas X_{i2} . Kintamasis X_{i1} išbraukiamas, jeigu teisinga nelygybė

$$r^2{}_{0i_2(i_2)}(n-3)/(1-r^2{}_{0i_2(i_2)}) < F_{p'}(1,n-3) \quad (2.2)$$

Taigi po antrojo žingsnio a. d. Y gali likti tik kintamasis X_{i1} (nelygybė (2.1.) neteisinga); lieka tik kintamasis X_{i2} (nelygybė (2.1) ir (2.2) teisingos); vektorius (X_{i1} ; X_{i2})T (nelygybė (2.1) teisinga, o nelygybė (2.2)

neteisinga). Pirmuoju atveju procesas užbaigiamas; antruoju ir trečiuoju atvejais pereinama prie trečiojo žingsnio.

T r e č i a s i s, k e t v i r t a s i s, ... , ž i n g s n i a i. Rekurentiškai kartojamas antrasis žingsnis. Tiksliau, remiantis dalinių koreliacijos koeficientų įverčiais, pagal taisykles, analogiškas (2.1), daroma išvada, ar reikia papildyti kintamųjų sąrašą dar vienu kintamuoju. Įrašius naują kintamąjį, tikrinama, ar nereikia kurio nors kintamojo iš sąrašo išbraukti. Procesas užbaigiamas, kai nelygybės, analogiškos (2.1), (2.2), netenkinamos.

Šio darbo modelyje pritaikytos pažingsninės regresijos dalinės koreliacijos koeficientų įverčių bendra išraiška atrodo taip:

$$L=\sum_i\left(\varepsilon_i-w_k x_{(k)}^{(i)}\right)^2=\sum_i\varepsilon_i{}^2-2w_k\sum_i\varepsilon_i x_{(k)}^{(i)}+w_k{}^2\sum_i x^{2(i)}_{(k)}=\left|w_k=\frac{\sum_i\varepsilon_i x_{(k)}^{(i)}}{\sum_i x^{2(i)}_{(k)}}\right|=$$

$$\sum_i\varepsilon_i{}^2-2\sum_i\varepsilon_i x_{(k)}^{(i)}\frac{\sum_i\varepsilon_i x_{(k)}^{(i)}}{\sum_i x^{2(i)}_{(k)}}+\sum_i x^{2(i)}_{(k)}\frac{\left(\sum_i\varepsilon_i x_{(k)}^{(i)}\right)^2}{\left(\sum_i x^{2(i)}_{(k)}\right)^2}=\sum_i\varepsilon_i{}^2-\frac{\left(\sum_i\varepsilon_i x_{(k)}^{(i)}\right)^2}{\sum_i x^{2(i)}_{(k)}}=$$

$$=\sum_i\varepsilon_i{}^2\left(1-\left(\frac{\sum_i\varepsilon_i x_{(k)}^{(i)}}{\sqrt{\sum_i\varepsilon_i{}^2}\sqrt{\sum_i x^{2(i)}_{(k)}}}\right)^2\right)$$

Bendru atveju, gavome dalinių koreliacijos koeficientų išraišką, kuri bus panaudota konstriuojant mūsų modelį.

IV. NETIESINĖ REGRESIJA

Neretai netiesinis regresijos modelis (dar vadinamas išretintu Gauso regresijos modeliu), atlikus tam tikras transformacijas, yra paverčiamas tiesiniu ir tada skaičiuojama tiesinė regresija.

$$y=f(\vec{x})=\sum_{j=1}^{M}\alpha_j\varphi_j(\vec{x}|\vec{\theta}_j)+\varepsilon,\quad \varepsilon\sim N(0,\sigma^2),\qquad (IV.1)$$

kur $\vec{x}\in\mathbb{R}^1$; $\theta_j\in\mathbb{R}^1, 1\le j\le M$ yra modelio svoriai M – regresorių skaičius, kurį nustatysime algoritmo pagalba; $\varphi_j(\vec{x})$ yra bendru atveju netiesinės regresijos atvaizdis, priklausantis nuo nežinomų parametrų.

Pagrindinė netiesinės regresijos modelio idėja yra ta, kad nuosekliai (pažingsniui) mažinsime modelio paklaidos dispersijos įvertinį, prijungdami naujus narius.

Čia funkcijos gali būti parinktos taip:

a) $\varphi_j(\vec{x}|\vec{\theta}_j) = \exp\left\{-\frac{1}{2\sigma^2}\left\|\vec{x}-\vec{\mu}_j\right\|^2\right\}$, ir $\vec{\theta}_j = (\vec{\mu}_j, \vec{\sigma}_j)$, taip vadinama radialinė bazinė funkcija.

b) $\varphi_j(\vec{x}|\vec{\theta}_j) = \frac{1}{1+\exp(-(\vec{x}^T\vec{\theta}_j))}$, taip vadinama sigmoidinė bazinė (perdavimo) funkcija.

y – tirpalo koncentracija.

$\vec{x}$ – stiprio rodmenų vektorius, kiekvieną stebimo proceso sekundę.

k – pažingsninės netiesinės regresijos žingsnis.

n – stiprio vektorių skaičius.

l – stiprio vektorių skaičius, kai žinomos koncentracijos.

$$y = f^{(k)}(\vec{x}) = \sum_{j=1}^{k} \alpha_j \varphi_j(\vec{x}|\vec{\theta}_j) = \sum_{j=1}^{k-1} \alpha_j \varphi_j(\vec{x}|\vec{\theta}_j) + \alpha_k \varphi_k(\vec{x}|\vec{\theta}_k) = f^{(k-1)}(\vec{x}) + \alpha_k \varphi(\vec{x}|\vec{\theta}_k)$$

Pastaba. Dabar plačiai vystomojoje atraminių vektorių problematikoje perdavimo funkcijos vadinamos branduoliais, o vietoje nežinomų parametrų $\vec{\theta}_j$ yra imami vektoriai $\vec{x}_j$, kurie sudaro turimą imtį. Tada netiesinės regresijos lygtis paprastai užrašoma taip:

$$f(x) = \sum_{k=1}^{n} \alpha_k K(\vec{x}, \vec{x}_k),$$

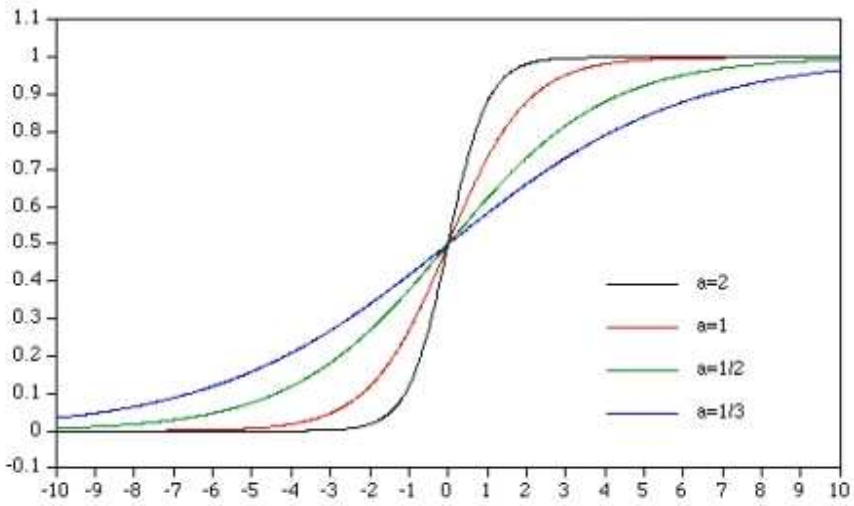
Savo modelyje, transformacijai mes naudosime logistinę sigmoidinę aktyvacijos funkciją :

$$\varphi(\vec{x}|\vec{\theta}_k) = K(\vec{x}, \vec{\theta}_k) = \frac{1}{1 + e^{-<\vec{x}\vec{\theta}_k^T>}},$$

Čia k – pažingsninės regresijos žingsnis. $\vec{\theta}_j$ – vektorius, kurio koordinatės parenkamos stochastinės paieškos būdu iš intervalo $\left[-\frac{1}{2}, \frac{1}{2}\right]$, su kuriuo tikslo funkcija pasiekia lokalų optimumo tašką k-ąjame žingsnyje.

Peter L. Bartlett („ The Sample Complexity of Pattern Classification with Neural Networks: The Sizeof the Weights is More Important than the Size of the Network“) **[1]** įrodė, jog netiesinės regresijos su sigmoidine logistine funkcija rezultatų „gerumas“ priklauso ne nuo narių skaičiaus, o nuo jų dydžio. T.y. , modelio tikslumas labiausiai priklauso nuo to, kiek yra mažos parametrų $\vec{\theta}_j$ koordinatės.

Mūsų turimi duomenys, vektoriaus $\vec{x}$ koordinatės, yra labai maži, todėl aktyvacijos funkcijos reikšmės yra labai arti $\frac{1}{2}$. Parametras $a=<\vec{x}\overline{\theta_k}^T>$ dar vadinamas sigmoidinės funkcijos greičiu arba šlaito parametru **[8]** , kuris aktyvacijos funkcijai suteikia jautrumo. Kai duomenys yra labai maži , nedideli parametro *a* pokyčiai duoda reikšmingus funkcijos reikšmių pokyčius.



IV.2 Pav. Sigmoidinės aktyvacijos funkcijos priklausomybė nuo greičio *a*

Riboje, didinant parametrą *a* ,funkcija tampa paprasčiausia slenksčio funkcija. Slenksčio funkcijos reikšmių sritis yra dvi reikšmės 0 arba 1.

V. NEPILNAI APIBŪDINTI STEBĖJIMAI

(angl. Semi-supervised learning)

Pagrindinė stebimų apmokymo reikšmių problema yra prognozės funkcijos erdvė $\mathcal{H} = \{f: \chi \to \mathbb{R}\}$ apibrėžta taip: $\chi \subseteq \mathbb{R}^d$. Norint apmokyti funkciją $f \subset \mathcal{H}$ pagal turimus duomeis $\{x_i, y_i\}_{i=1}^{l}$, kur x_i yra įėjimai ir y_i yra išėjimų reikšmės $f(x_i)$, reikia rasti minimumą:

$$\hat{f} = \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{l} [y_i - f(x_i)]^2 + \lambda\Omega(f) \qquad \text{(V.1)}$$

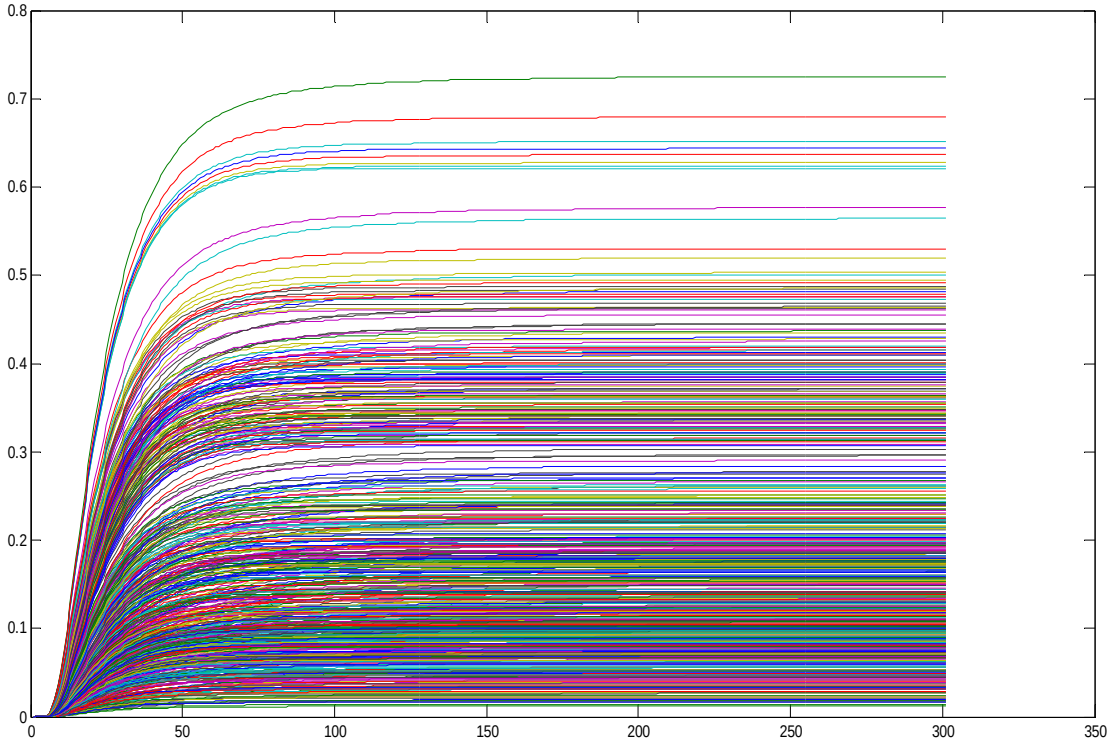
Nesunkiai galime pastebėti, jog pirmasis dėmuo yra ne kas kita, kaip mažiausių kvadratų metodas, t.y. sumuojamos klaidos, kurias vėliau minimizuosime.

Čia, $\Omega: f \to \mathbb{R}^+ \cup 0$ matuoja funkcijų sudėtingumą srityje $\mathcal{H}$. Pirma užduotis paklaidos finkcijai (V.1) , "priversti" funkciją f gerai atspindėti turimus stebėjimus. Antra – užtikrina, jog $f(x)$ yra pakankamai glotni (reguliarizuota), t.y. mažiau sudėtinga.Viena iš reguliarizavimo prielaidų – tam tikri įėjimai, turi tam tikrus išėjimus.

VI. MODELIO FORMALIZAVIMAS

VI.1. Netiesinės regresijos modelio tikslo funkcijos išvedimas

Nagrinėjame situaciją, kai imtis susideda iš dviejų dalių. Pirmoje dalyje žinome šių vektorių kovariantes $\vec{x}_k$ ir atsakus y_k , t.y. $(y_k, \overrightarrow{x_k})$, k=1,2, …,l.



VI.1pav. Vektorių $\vec{x}_k$ atsako kreivės tam tikru laiko momentu

Antroje dalyje žinome tik kovariantes, t.y. $(\overrightarrow{x_k},\ k = l+1, l+2, \dots, n)\ X_k \in \Re$, todėl tikslo funkciją ir sudarysime iš dviejų dėmenų:

$$L = \sum_{k=1}^{l} \left(y_k - f(\overrightarrow{x_k})\right)^2 + \lambda\Omega(\vec{f})$$

Čia

$$\Omega = \Omega_g(\vec{f}) = \frac{1}{2}\sum_{i,j=1}^{n} W_{ij}\left(\frac{f(\overrightarrow{x_i})}{\sqrt{D_i}} - \frac{f(\overrightarrow{x_j})}{\sqrt{D_j}}\right)^2 \qquad \textbf{(VI.1)}$$

arba

$$\Omega = \Omega_c(\vec{f}) = \frac{1}{2}\sum_{i,j=1}^{n} W_{ij}\left(f(\overrightarrow{x_i}) - f(\overrightarrow{x_j})\right)^2 \qquad \textbf{(VI.2)}$$

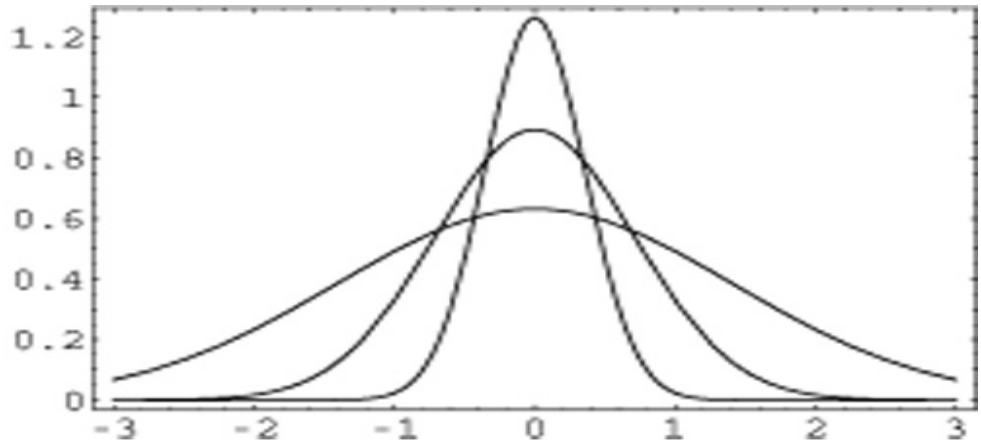
Ω – reguliarizacijos dėmuo (Kai Yu ir Volker Tresp „Semi – supervised learning“) **[4].** (Kiekvienam turimam $x \in \Re$ modelis generuoja netiesinį atvaizdį $y \in \Re$. $D_i = \sum_{j=1}^{n} w_{ij}$; $D_j = \sum_{i=1}^{n} w_{ij}$;

Svorių matrica apskaičiuojama remiantis Gauso funkcijos išraiška :

$$W_{ij} = \exp\left\{-\frac{1}{2\sigma^2}\|\vec{x_i} - \vec{x_j}\|^2\right\}; \qquad \textbf{(VI.3)}$$

Matricos **W** elementai yra ne neigiami ir simetriški, t.y. $W_{ij} = W_{ji}$ ir pagrindinė įstrižainė $W_{ii} = 1$. Svorio funkcija, kiekvienam stebėjimo vektoriui x_s , kur s= $1, 2, \dots, l$ suteikia svorius, atsižvelgiant į atstumus nuo šių vektorių iki kitų stebėjimų vektorių. σ – hiperparametras, kuris nusako Gauso funkcijos statumą. T.y. jei σ reikšmė yra maža, modelis kreips didesnį dėmesį į artimiausius kaimynus ir Gauso funkcijos kreivė bus statesnė (pav.VI. 2), jei σ yra didelis, kreivė bus labiau prigludusi ir modelis didelį dėmesį kreips į visas funkcijas.

Taip priklasifikuojamos atsako funkcijos $f(\vec{x_m}), kur\ m = l+1, l+2, \dots, n$ (kurių reikšmės $f(x_m)$ nėra žinomos) su $f(\vec{x_s}),\ kur\ s = 1, 2, \dots, l$, (reikšmės mums žinomos). Bendrai galime pasakyti, kuo mokymo aibės vektoriai yra arčiau vienas kito, tuo didesnis svoris jam suteikiamas ir kuo mokymo aibės vektoriai toliau vienas kito, tuo mažesnį svorį jis gauna. Pabrėžiame, jog $w_{ij} \in (0,1)$.



VI.2. Pav. Gauso kreivių priklausomybė nuo parametro σ.

Pertvarkome reguliarizacijos narį **(VI.2)**:

$$\frac{1}{2}\sum_{i,j}^{n} w_{ij}\,(f_i - f_j)^2 = \frac{1}{2}\sum_{i,j}^{n} w_{ij}\,({f_i}^2 - 2f_i f_j + {f_j}^2) \quad = \frac{1}{2}\sum_{i,j}^{n} w_{ij}{f_i}^2 - \sum_{i,j}^{n} w_{ij} f_i f_j + \frac{1}{2}\sum_{i,j}^{n} w_{ij}{f_j}^2$$

$$= \frac{1}{2}\sum_{i=1}^{n} {f_i}^2 \sum_{j=1}^{n} w_{ij} - \sum_{i,j}^{n} w_{ij} f_i f_j + \frac{1}{2}\sum_{j=1}^{n} {f_j}^2 \sum_{i=1}^{n} w_{ij} = \sum_{i=1}^{n} {f_i}^2 D_i - \sum_{i,j}^{n} w_{ij} f_i f_j$$

Žymime $W = (w_{ij})_{i,j=\overline{1,n}}$, $\vec{f} = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}$, tada $\vec{f}^T = (f_1 \quad \dots \quad f_n)$.

Tada

$$\vec{f}^T W \vec{f} = (f_1 \quad \dots \quad f_n)\begin{pmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nn} \end{pmatrix}\begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} = (\textstyle\sum_{i=1}^{n} f_i w_{i1} \quad \sum_{i=1}^{n} f_i w_{2i} \quad \dots \quad \sum_{i=1}^{n} f_i w_{ni})\begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} =$$

$$\textstyle\sum_{j=1}^{n}(\sum_{i=1}^{n} f_i w_{ji}) f_j = \sum_{j=1}^{n}\sum_{i=1}^{n} w_{ji} f_i f_j \quad .$$

Taigi

$$\sum_{i,j=1}^{n} w_{ij} f_i f_j = \vec{f}^T W \vec{f}$$

Tarkime

$$\sum_{i=1}^{n} {f_i}^2 = (f_1 \quad \dots \quad f_n)\begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} = \vec{f}^T \vec{f}$$

Diagonalinė matrica:

$$D = \operatorname{diag}(D_i) = \begin{pmatrix} D_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & D_n \end{pmatrix}$$

$$D\vec{f} = \begin{pmatrix} D_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & D_n \end{pmatrix}\begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} D_1 f_1 \\ \vdots \\ D_n f_n \end{pmatrix}$$

Skaliarinė sandauga:

$$\vec{f}^T D \vec{f} = (f_1 \quad \dots \quad f_n)\begin{pmatrix} D_1 f_1 \\ \vdots \\ D_n f_n \end{pmatrix} = \sum_{i=1}^{n} {f_i}^2 D_i$$

Todėl $\frac{1}{2}\sum_{i,j}^{n} w_{ij}\,(f_i - f_j)^2 = \vec{f}^T D \vec{f} - \vec{f}^T W \vec{f} = \vec{f}^T (D - W)\vec{f}$. Matrica $D - W$ vadinama Laplasianu.

Matricos $F_n \in \Re^{m\,x\,n}$ elementai yra $\varphi_j(\vec{x}|\vec{\theta}_j), j = \overline{1,m}$, o $i = \overline{1,n}$. Tada vektorių $\vec{f} = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} f(\vec{x_1}) \\ \vdots \\ f(\vec{x_n}) \end{pmatrix}$

užrašome taip : $\vec{f} = \begin{pmatrix} \sum_{j=1}^{m} w_j \varphi_j(\vec{x}_1|\vec{\theta}_j) \\ \vdots \\ \sum_{j=1}^{m} w_j \varphi_j(\vec{x}_n|\vec{\theta}_j) \end{pmatrix} = F_n{}^T\vec{w} = \begin{pmatrix} \varphi_1(\vec{x}_1|\vec{\theta}_1) & \cdots & \varphi_m(\vec{x}_1|\vec{\theta}_m) \\ \vdots & \ddots & \vdots \\ \varphi_1(\vec{x}_n|\vec{\theta}_1) & \cdots & \varphi_m(\vec{x}_n|\vec{\theta}_m) \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix}$.

Vektorių $\vec{f}^T = (f(\vec{x_1}) \quad \dots \quad f(\vec{x_n})) = \left(\sum_{j=1}^{m} w_j \varphi_j(\vec{x}_1|\vec{\theta}_j) \quad \dots \quad \sum_{j=1}^{m} w_j \varphi_j(\vec{x}_n|\vec{\theta}_j)\right) = \vec{w_j}^T F_n =$

$= (w_1 \quad \dots \quad w_m) \begin{pmatrix} \varphi_1(\vec{x}_1|\vec{\theta}_1) & \cdots & \varphi_m(\vec{x}_1|\vec{\theta}_m) \\ \vdots & \ddots & \vdots \\ \varphi_1(\vec{x}_n|\vec{\theta}_1) & \cdots & \varphi_m(\vec{x}_n|\vec{\theta}_m) \end{pmatrix}$.

Taigi,

$\pi_c(\vec{f}) = \frac{1}{2}\sum_{i,j=1}^{n} W_{ij}\left(f(\vec{x_i}) - f(\vec{x_j})\right)^2 = \vec{f}^T(D - W)\vec{f} = \vec{w}^T F_n \Delta F_n{}^T \vec{w}$, kur $\Delta = D - W$.

Žymėkome $F_l \in \Re^{m\,x\,l}$ matrica, kurios elementai yra $\varphi_j(\vec{x}_i|\vec{\theta}_j)$, $j = \overline{1,m}$, o $i = \overline{1,l}$; o

$\vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_l \end{pmatrix}$, o $\vec{f_l} = \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_l) \end{pmatrix} = F_l\vec{w}$, tada pirmoji tikslo funkcijos (2) dalis atrodys taip:

$$\sum_{k=1}^{l}\left(y_k - f(\vec{x_k})\right)^2 = \|\vec{y} - \vec{f_l}\|^2 = \left(\vec{y} - \vec{f_l}\right)^T\left(\vec{y} - \vec{f_l}\right) = \vec{y}^T\vec{y} - 2\vec{f_l}^T\vec{y} + \vec{f_l}^T\vec{f_l} = = \|\vec{y}\|^2 - 2\vec{w}^T F_l\vec{y} + \vec{w}^T F_l F_l{}^T\vec{w}.$$

Taigi, galutinė tikslo funkcijos (VI.2) išraiška yra :

$$L = \|\vec{y}\|^2 - 2\vec{w}^T F_l\vec{y} + \vec{w}^T F_l F_l{}^T\vec{w} + \lambda\vec{w}^T F_n \Delta F_n{}^T\vec{w},$$

Ji įgyja minimumą, kai $\vec{w} = \hat{w}$, kur $\hat{w} = (F_l F_l{}^T + \lambda F_n \Delta F_n{}^T)^{-1} F_l\vec{y}$.

VI.2. MODELIO ALGORITMAS

k-tajame žingsnyje turime netiesinės regresijos lygtį:

$$y = f^k(\vec{x}) = f^{(k-1)}(\vec{x}) + \alpha_k\varphi(\vec{x}|\vec{\theta}_k) \quad k - \text{tajame žingsnyje}$$

Tikslo funkcija:

$$I(\alpha_k, \vec{\theta}_k) = \sum_{i=1}^{l} \left(y_i - f^k(\vec{x}_i)\right)^2 + \lambda\Omega(\vec{f})$$

Kur $\Omega(\vec{f}) = \sum_{i,j=1}^{n} W_{ij}\left(\frac{f^k(\vec{x}_i)}{\sqrt{D_i}} - \frac{f^k(\vec{x}_j)}{\sqrt{D_j}}\right)^2$;

1. Skleidžiame tikslo f-jos pirmąjį narį:

$$\sum_{i=1}^{l}\left(y_i - f^k(\vec{x}_i)\right)^2 = \sum_{i=1}^{l}\left(y_i - f^{k-1}(\vec{x}_i) - \alpha_k\varphi(\vec{x}_i|\vec{\theta}_k)\right)^2$$
$$= \sum_{i=1}^{l}\left(y_i - f^{k-1}(\vec{x}_i)\right)^2 + 2\alpha_k\sum_{i=1}^{l}\varphi(\vec{x}_i|\vec{\theta}_k)\left(y_i - f^{k-1}(\vec{x}_i)\right) + {\alpha_k}^2\sum_{i=1}^{l}\varphi^2(\vec{x}_i|\vec{\theta}_k)$$
$$= \sum_{i=1}^{l}\left(\varepsilon^{k-1}(\vec{x}_i) - \alpha_k\varphi(\vec{x}_i|\vec{\theta}_k)\right)^2$$

Kur $\varepsilon^{k-1}(\vec{x}_i) = y_i - f^{k-1}(\vec{x}_i)$.

2. Skleidžiame antrąjį narį:

$$\lambda\Omega(\vec{f}) = \sum_{i,j=1}^{n} W_{ij}\left(\frac{f^k(\vec{x}_i)}{\sqrt{D_i}} - \frac{f^k(\vec{x}_j)}{\sqrt{D_j}}\right)^2 =$$
$$= \lambda\sum_{i,j=1}^{n} W_{ij}\left(\frac{f^{k-1}(\vec{x}_i)}{\sqrt{D_i}} - \frac{f^{k-1}(\vec{x}_j)}{\sqrt{D_j}} + \frac{\alpha_k\varphi(\vec{x}_i|\vec{\theta}_k)}{\sqrt{D_i}} - \frac{\alpha_k\varphi(\vec{x}_j|\vec{\theta}_k)}{\sqrt{D_j}}\right)^2 = \lambda\sum_{i,j=1}^{n} W_{ij}\left(H_{ij} + \alpha_k L_{ij}(\vec{\theta}_k)\right)^2$$

Kur $H_{ij} = \left(\frac{f^{k-1}(\vec{x}_i)}{\sqrt{D_i}} - \frac{f^{k-1}(\vec{x}_j)}{\sqrt{D_j}}\right)$, o $L_{ij}(\vec{\theta}_2) = \left(\frac{\varphi(\vec{x}_i|\vec{\theta}_k)}{\sqrt{D_i}} - \frac{\varphi(\vec{x}_j|\vec{\theta}_k)}{\sqrt{D_j}}\right)$

Taigi, ieškome tikslo funkcijos išvestinę pagal α_k ir prilyginę ją 0 išsireiškiame α_k:

$$\frac{\partial}{\partial\alpha_k} I(\alpha_k, \vec{\theta}_k) = \left(\sum_{i=1}^{l}\left(\varepsilon^{(k-1)} - \alpha_k\varphi(\vec{x}_i|\vec{\theta}_k)\right)^2 + \lambda\sum_{i,j=1}^{n} w_{ij}\left(H_{ij} + \alpha_k L_{ij}(\vec{\theta}_k)\right)^2\right)'_{\alpha_k} = 0$$

$$\alpha_k = \frac{\sum_{i=1}^{l}\varepsilon^{(k-1)}\varphi(\vec{x}_i|\vec{\theta}_k) - \lambda\sum_{i,j=1}^{n} W_{ij}H_{ij}L_{ij}(\vec{\theta}_k)}{\sum_{i=1}^{l}\varphi^2(\vec{x}_i|\vec{\theta}_k) + \lambda\sum_{i,j=1}^{n} W_{ij}{L_{ij}}^2(\vec{\theta}_k)}$$

Gautą α_k išraišką statome į tikslo funkciją, tada nežinomi parametrai $\vec{\theta}_k$ turi minimizuoti pertvarkytą tikslo funkciją:

$$\sum_{i=1}^{l}\left(\varepsilon^{(k-1)}+\lambda\sum_{i,j=1}^{n}W_{ij}H_{ij}{}^{2}\right)\ -\frac{\left(\sum_{i=1}^{l}\varepsilon^{(k-1)}\varphi(\vec{x}_i|\vec{\theta}_k)-\lambda\sum_{i,j=1}^{n}W_{ij}H_{ij}L_{ij}(\vec{\theta}_k)\right)^{2}}{\sum_{i=1}^{l}\varphi^{2}(\vec{x}_i|\vec{\theta}_k)+\lambda\sum_{i,j=1}^{n}W_{ij}L_{ij}{}^{2}(\vec{\theta}_k)}[2-\lambda]\rightarrow\min_{\vec{\theta}_k} \qquad \text{(VI.2.3)}$$

Kadangi k-ąjame žingsnyje jau žinome pirmojo dėmens reikšmę, tad norint minimizuoti (3) išraišką, galime maksimizuoti antrąjį išraiškos dėmenį pagal nežinomus $\vec{\theta}_k$:

$$Y(\vec{\theta}_k)=\frac{\left(\sum_{i=1}^{l}\varepsilon^{(k-1)}\varphi(\vec{x}_i|\vec{\theta}_k)-\lambda\sum_{i,j=1}^{n}W_{ij}H_{ij}L_{ij}(\vec{\theta}_k)\right)^{2}}{\sum_{i=1}^{l}\varphi^{2}(\vec{x}_i|\vec{\theta}_k)+\lambda\sum_{i,j=1}^{n}W_{ij}L_{ij}{}^{2}(\vec{\theta}_k)}[2-\lambda]\rightarrow\max_{\vec{\theta}_k}$$

Norėdami maksimizuoti išraišką $Y(\vec{\theta}_k)$ pasinaudosime stochastine paieška.

Siekdami sumažinti turimų duomenų rinkinius ir išsaugoti didžiąją dalį informacijos apie tuos duomenis atlikome pagrindinių komponenčių analizę ir transformavome juos į mažesnės dimensijos erdvę, kuri susideda iš nekoreliuotų kintamųjų – pagrindinių komponenčių, kurios yra išdėstomos svarbos mažėjimo tvarka.

VI.3. PAGRINDINIŲ KOMPONENČIŲ ANALIZĖ (PCA)

Pagrindinių komponenčių analizė

Pagrindinių komponenčių analizė (PCA) (angl. Principal Component Analysis) dėl savo paprastumo ir yra dažniausia praktikoje naudojamas dimensijos mažinimo būdas.

Apibrėžimas ir pagrindinių komponenčių radimas

Turime vektorių $\mathbf{X}\epsilon\mathbb{R}^{n}$. Paėmus visus turimus n duomenis, turėsime duomenų rinkinį, kuris teiks 100 % informacijos, tačiau matematinių modelių konstravimui ne visada yra tikslinga ir patogu imti visas n komponentes. PCA metodo tikslas yra rasti keletą kintamųjų, atspindinčių daugiausia informacijos, išreikštos žinoma dispersija, koreliacija ir kovariacija.

Nagrinėkime atsitiktinį vektorių

$$\mathbf{X}:\mathbf{X}=(\mathbf{X}_1,\mathbf{X}_2,\dots\mathbf{X}_n)^{\mathrm{T}} \qquad \text{(VI.3.1)}$$

Ir tiesines kombinacijas:

$$Y_1 = a_1^T X = a_{11}X_1 + \cdots + a_{1n}X_n = \sum_{i=1}^{n} a_{1i}X_i$$

$$\vdots \qquad \text{(VI.3.2)}$$

$$Y_n = a_n^T X = a_{n1}X_1 + \cdots + a_{nn}X_n = \sum_{i=1}^{n} a_{ni}X_i$$

Kur i-oji tiesinė kombinacija yra vadinama i-taja pagrindine komponente. Tegu $\text{cov}(X,X) = \Sigma = [\sigma_{ij}]_{n\times n}$ yra a.v. (VI.3.1) kovariacijų matrica. Tada atsitiktinio dydžio $Y_i = a_i^T X$ dispersija yra $\mathbb{D}Y_i = \mathbb{D}(a_n^T X) = \text{cov}(a_n^T X, a_n^T X) = a_n^T \text{cov}(X,X) a_i = a_n^T \Sigma a_i$, kurią galima padaryti kiek norint didele didinant vektoriaus a_i ilgį. Todėl natūralu nagrinėti tik tokias transformacijas, kai vektorius a_i yra normuotas, t.y. tenkina sąlygą:

$$a_i^T a_i = 1 \qquad \text{(VI.3.3)}$$

Teorema 1.1.

Pagrindinės komponentės $Y_1 = a_1^T Y_2 = a_2^T X, \ldots Y_n = a_n^T X$ gaunamos tada, kai $a_1, \ldots, a_n$ yra tikriniai vektoriai

$$\sum a_i = \lambda_i a_i, i = 1, \ldots, n \qquad \text{(VI.3.4)}$$

Atitinkantys charakteringosios lygties

$$|\Sigma - \lambda I| = 0 \qquad \text{(VI.3.5)}$$

Kur $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_i$ .Pagrindinės komponentės yra ne koreliuotos ir išreikštos taip, kad jų dispersijos:

$$\mathbb{D}Y_i Y_i = a_i^T \sum a_i = \lambda_i, \ i=1,\ldots,n$$

Tenkina sąlygą:

$$\mathbb{D}Y_1 \geq \mathbb{D}Y_2 \geq \cdots \geq \mathbb{D}Y_n$$

Perėjimas nuo a.v. $X^T = (X_1, X_2, \ldots, X_n)$ prie a.v. $Y^T = (Y_1, Y_2, \ldots, Y_n)$ nepakeičia apibendrintosios dispersijos ir koordinačių dispersijų sumos:

$$|\mathbb{D}Y| = |\mathbb{D}X| = |\Sigma|, \ \sum_{i=1}^{n} \mathbb{D}X_i = \sum_{i=1}^{n} \mathbb{D}Y_i \qquad \text{(VI.3.6)}$$

Įrodymas .

Norint rasti pirmąją pagrindinę komponentę, vektorių a_1 reikia pasirinkti taip, kad $\mathbb{D}Y_1$ būtų maksimali, su sąlyga (5.3). Tam patogu būtų pasinaudoti Lagranžo daugiklių metodu.

Lagranžo funkcija pirmajai mažiausiai dispersijai rasti:

$$L(a_1) = a_1^T \Sigma\, a_1 - \lambda(a_1^T a_1 - 1) \qquad \text{(VI.3.7)}$$

Randame jos išvestinę ir prilyginame 0:

$$\frac{\delta L(a_1)}{\delta a_1} = 2\Sigma\, a_1 - 2\lambda_1\, a_1 = 0 \qquad \text{(VI.3.8)}$$

Gauname, kad :

$$\Sigma\, a_1 - \lambda_1\, a_1 = 0$$

Arba

$$(\Sigma - \lambda I)\, a_1 = 0 \qquad \text{(VI.3.9)}$$

Kur I yra $n \times n$ matavimo vienetinė matrica. Lygybės (5.9) abi puses dauginant iš a_1^T, gauname

$$a_1^T \Sigma\, a_1 - a_1^T \lambda_1\, a_1 = 0 \qquad \text{(VI.3.10)}$$

Iš išplaukia, kad $\lambda_1 = a_1^T \Sigma\, a_1 = \mathbb{D}Y_1$. Taigi galima sakyti, kad λ_1 yra didžiausia tikrinė kovariacinės matricos Σ reikšmė ir yra lygi didžiausiai dispersijai $\mathbb{D}Y_1$, o a_1 yra atitinkamas tikrinis vektorius. Analogiškai galima sudaryti Lagranžo funkciją dispersijai $\mathbb{D}Y_2 \le \mathbb{D}Y_1$ rasti, turint omenyje, kad pirma ir antra pagrindinės komponentės nekoreliuoja, t.y., kad $\mathrm{cov}(a_1^T X, a_2^T X) = 0$ ir $a_2^T a_2 = 1$. Jau buvo minėta, kad $\mathrm{cov}(a_i^T X, a_j^T X) = a_i^T \Sigma\, a_j$ todėl :

$$\mathrm{cov}(a_1^T X, a_2^T X) = a_1^T \Sigma\, a_2 = a_2^T \Sigma\, a_1 = a_2^T a_1^T\, a_1 \Sigma\, a_1 = a_2^T\, a_1 a_1^T \lambda_1\, a_1 = \lambda_1 a_2^T\, a_1 = 0$$

t.y. $a_2^T\, a_1 = 0$. Sudarome Lagranžo funkciją:

$$L(a_1) = a_2^T \Sigma\, a_2 - \lambda_1 a_2^T\, a_1 - \lambda_2 (a_2^T\, a_2 - 1) \qquad \text{(VI.3.11)}$$

Randame jos išvestinę ir prilyginame 0 :

$$\frac{\delta L(a_2)}{\delta a_2} = a\Sigma\, a_2 - \lambda_1\, a_1 - 2\lambda_2\, a_2 = 0 \qquad \text{(VI.3.12)}$$

Abi lygybės (VI.3.12) puses padauginame iš a_2^T irgauname :

$$a_2^T \Sigma\, a_2 - a_2^T \lambda_2\, a_2 - a_2^T \frac{\lambda_2}{2}\, a_1 = 0$$

Žinodami, jog $a_2^T\, a_1 = 0$, seka

$$a_2^T\, \Sigma\, a_2 - a_2^T \lambda_2\, a_2 = 0 \qquad \text{(VI.3.12)}$$

Ir

$$(\Sigma - \lambda_2 \mathbf{I})\, a_2 = 0$$

Kur I vėl yra $\mathbf{n} \times \mathbf{n}$ matavimo vienetinė matrica. Iš (5.13) gauname, kad $\lambda_2 = a_2^T\, \Sigma\, a_2 = \mathbb{D}Y_2$. Taigi, λ_2 yra antra pagal didumą tikrinė kovariacinės matricos Σ reikšmė ir lygi dispersijai $\mathbb{D}Y_2$, o a_2 yra atitinkamas tikrinis vektorius. Analogiškai randama ir $\lambda_3, \lambda_4, \ldots, \lambda_n$. Imkime k-ąją pagrindinę komponentę $Y_k = a_k^T X$ ir sudarome Lagranžo funkciją dispersijai $\mathbb{D}Y_k \leq \mathbb{D}Y_{k-1} \leq \cdots \leq \mathbb{D}Y_1$ rasti, žinant, kad $\mathrm{cov}(a_k^T X, a_l^T X) = 0$, kai $k \neq l$ ir $a_k^T\, a_k = 1$. Užrašome Lagranžo funkciją:

$$L(a_k) = a_k^T \Sigma\, a_k - \lambda_k \left(a_k^T\, a_k - 1\right) - \sum_{i=1}^{k-1} \lambda_i a_k^T\, a_i \qquad \text{(VI.3.14)}$$

Randame išvestinę ir prilyginame 0,

$$\frac{\delta L(a_k)}{\delta a_k} = 2\Sigma\, a_k - 2\lambda_k\, a_k - \sum_{i=1}^{k-1} \lambda_i a_i = 0 \qquad \text{(VI.3.15)}$$

Abi lygybės (5.15) puses dauginant iš a_k^T gauname

$$a_k^T \Sigma\, a_k = a_k^T \lambda_k\, a_k - a_k^T \sum_{i=1}^{k-1} \frac{\lambda_i}{2}\, a_i = 0 \qquad \text{(VI.3.16)}$$

Atsižvelgiant į tai, kad $a_k^T\, a_l = 0$, kai $k \neq l$, gauname

$$\Sigma\, a_k - \lambda_k\, a_k = 0 \qquad \text{(VI.3.17)}$$

O tai yra ekvivalentu

$$(\Sigma - \lambda_k \mathbf{I})\, a_k = 0$$

Iš (VI.3.17) gauname, kad $\lambda_k = a_k^T \Sigma a_k = \mathbb{D}Y_k$. Taigi, $\mathbb{D}(a_k^T X) = \lambda_k$, k=1,…,n kur a_k yra k-tasis tikrinis kovariacinės matricos Σ vektorius, o λ_k- jį atitinkanti tikrinė reikšmė, lygi k-tosios komponentės dispersijai $\mathbb{D}Y_k$.

Pagrindinių komponenčių vektorius $Y = (Y_1, Y_2, \ldots, Y_n)$ yra gaunamas atliekant transformaciją $Y = A^T X$, kur A-matrica, kurios stulpeliai yra tikriniai vektoriai $a_1, a_2, \ldots, a_n$. Remiantis tuo, kad $\lambda_k = a_k^T \Sigma a_k == \mathbb{D}Y_k$ ir $\mathrm{cov}(Y_k, Y_1) = a_k^T \Sigma a_1 = 0$, kai $k \neq 1$ vektoriaus Y kovariacinė matrica yra diagonali su diagonaliaisiais elementais $\lambda_1, \lambda_2, \ldots, \lambda_n$ yra kovariacijų matricos Σ tikrinės reikšmės su atitinkamais tikriniais vektoriais a_1, a_2,…. a_n, iš to seka, kad $\Sigma = A\Lambda A^T$, kur Λ yra tikrinių reikšmių diagonalinė matrica, o A=(a_1, a_2,…., a_n) ir $AA^T = I$, tada

$$\sum_{i=1}^{n} DX_i = \sum_{i=1}^{n} \sigma_{ii} = tr(A\Lambda A^T) = tr(\Lambda AA^T) = tr(\Lambda) \sum_{i=1}^{n} \lambda_i = \sum_{i=1}^{n} \mathbb{D}Y_i$$

VII. PROGRAMA

VII.1. MatLab kodas

Modelis buvo programuojamas su MATLAB 7.4.0 matematinio modeliavimo programa. Žemiau pateikiame viso modelio atskirus „m“ modulius. Programos versiją rasite kompaktinėje plokštelėje (priedas prie darbo).

VI.1. Pagalbine funkcija :

Funkcija, skaičiuoja modelio Euklidinius skirtumus, kurie bus panaudoti matricos koeficienų W_{ij} radimui:

```
function B = EuklSk2(A)
tmp = size(A);
vektoriu = tmp(1);
B = zeros(vektoriu,vektoriu);
for i = 1:vektoriu
for j = i:vektoriu
tmp1 = (A(i,:) - A(j, :)).^2;
tmp = sum(tmp1');
B(i,j)=tmp(1,1);
B(j,i)=tmp(1,1);
end
end
end
```

VI.2. Pagalbine funkcija:

Skaičiuojami atstumai tarp vektorių X koordinačių, kurie vėliau bus panaudoti matricos W_{ij} radimui:

```
function ats = funkcijaL(Phi, D)
   tmp = Phi'./sqrt(D);
   tmp2 = size(tmp);
   tmp2 = tmp2(2);
   ats = zeros(tmp2,tmp2);
   for i = 1:tmp2;
      for j = i:tmp2;
         ats(i,j)=tmp(i)-tmp(j);
         ats(j,i)=tmp(j)-tmp(i);
      end;
   end;
end
```

VI.3. Pagalbine funkcija :

Skaičiuojamos modelio išraiška $\mathbf{K}(\vec{x},\vec{\theta}_k) = \frac{1}{1+e^{<x\vec{\theta}_k^T>}}$:

```
function ats = funkcijaPhi(x, teta)
   ats = (-1.0)*(x * teta');
   ilgis = size(ats);
   ilgis = ilgis(1);
   ats = (ones(ilgis,1)+exp(ats)).^(-1);
end
```

VI.4. Pagalbine funkcija :

Skaičiuojama modelio išraiška $\mathbf{H}_{ij} = \left(\frac{f^{k-1}(\vec{x}_i)}{\sqrt{D_i}} - \frac{f^{k-1}(\vec{x}_j)}{\sqrt{D_j}}\right)$:

```
function H = gaunamH(f,D)
   tmp = f./sqrt(D);
   tmp2 = size(tmp);
   tmp2 = tmp2(2);
   H = zeros(tmp2,tmp2);
   for i = 1:tmp2;
      for j = i:tmp2;
         H(i,j)=tmp(i)-tmp(j);
         H(j,i)=tmp(j)-tmp(i);
      end;
   end;
end
```

VI.5. Pagalbine funkcija :

Vektoriaus $\vec{\theta}_k$ atsitiktinių koordinačių radimas:

```
function ats = generuojamTeta(vilgis)
   ats = rand(1,vilgis)-((1/2)*ones(1,vilgis));
end
```

VI.6. Pažingsninė regresija :

„Lambda „ yra modelio parametras λ, kuris parenkamas bandymų keliu.

„sigma“ yra modelio parametras σ, kuris parenkamas bandymų keliu.

„k“ – pažingsninės regresijos žingsnių skaičius (sveikas, laisvai pasirenkamas skaičius).

„kiek“ – skaičius atsitiktinai generuojamų vektorių θ , vienam pažingsninės regresijos žingsniui, iš kurių parenkamas optimalus θ vektorius.

```
function [alpha, epsilon, teta] = pazingsnine(y, x, lambda, sigma, k, kiek)
   n = size(x);
   vilgis=n(2);
   n = n(1);
   l = size(y);
   l = l(1);
   alpha= zeros(1,1);
   teta = zeros(1,vilgis);
   epsilon = zeros(1,1);
   W=EuklSk2(x);
   A=exp(1).^((-1/(2*(sigma^2)))*W);
   D=sum(A);
   f = zeros(1,n);
   for i = 1:k
      kteta=generuojamTeta(vilgis);
      tmpteta=kteta;
      %eps = y - f(1:l);
      eps = y - f(1:l)';
      %epsilon=[epsilon;sum(sum(eps.^2))];
      epsilon=[epsilon;sum(eps.^2)];
      Phi = funkcijaPhi(x, tmpteta);
      H = gaunamH(f,D);
      L = funkcijaL(Phi,D);
      %n1=sum(sum(eps.*Phi(1:l,1)'))-lambda*sum(sum((W.*H).*L));
      n1=sum(sum(eps.*Phi(1:l,1)))-lambda*sum(sum((A.*H).*L));
      %n2=sum(sum(Phi(1:l,1)'.^2))+lambda*sum(sum((L.^2).*W));
      n2=sum(sum(Phi(1:l,1).^2))+lambda*sum(sum((L.^2).*A));
      RO=((n1^2)/n2)*(2-lambda);
      for j= 2:kiek
         tmpteta=generuojamTeta(vilgis);
         Phi = funkcijaPhi(x, tmpteta);
         L = funkcijaL(Phi,D);
         %tmpn1=sum(sum(eps.*Phi(1:l,1)'))-lambda*sum(sum((W.*H).*L));
         tmpn1=sum(sum(eps.*Phi(1:l,1)))-lambda*sum(sum((A.*H).*L));
         %tmpn2=sum(sum(Phi(1:l,1)'.^2))+lambda*sum(sum((L.^2).*W));
         tmpn2=sum(sum(Phi(1:l,1).^2))+lambda*sum(sum((L.^2).*A));
         tmpRO=((tmpn1^2)/tmpn2)*(2-lambda);
         if (tmpRO>RO)
            kteta=tmpteta;
            RO=tmpRO;
            n1=tmpn1;
            n2=tmpn2;
         end;
      end;
      kalpha = n1/n2;
      Phi = funkcijaPhi(x,kteta);
      f = f+kalpha*(Phi');
      alpha=[alpha;kalpha];
      teta=[teta;kteta];
   end;
   epsilon = epsilon(2:k+1,:);
   alpha = alpha(2:k+1,:);
```

```
teta = teta(2:k+1,:);
end
```

VI.7. Pagalbine funkcija :

Atliekame pagrindinių komponenčių analizę

```
function f =doPCA(X,d)
[rows,cols] = size(X);
 % standartizuojam
f.mean = mean(X);
f.std = std(X);
X = X - repmat(mean(X),rows,1);
X = X ./ repmat(std(X),rows,1);
% randam kovariaciju matrica
C = cov(X);
% randam jos tikrinius vektorius
[P D] = eig(C);
% juos surusiuojam
[P,D] = sortem(P,D);
D = D(1:d,1:d);
f.P = P(:,1:d);
% randam naujas kovariantes
f.Z = X*f.P;
```

VI.8. Pagalbine funkcija :

Pagalbinė funkcija PCA metodui atlikti

```
function [P2,D2]=sortem(P,D)
D2=diag(sort(diag(D),'descend')); % make diagonal matrix out of sorted diagonal values of input D
[c, ind]=sort(diag(D),'descend'); % store the indices of which columns the sorted eigenvalues come from
P2=P(:,ind);
```

VI.10. Pagrindinė funkcija :

Funkcija apjungianti pažingsninę bei prognozuojamų Y-kų (su testavimo duomenimis) funkcijas.

„l“ – yra modelio parametras λ, kuris parenkamas bandymų keliu.

„s“ yra modelio parametras σ, kuris parenkamas bandymų keliu.

„laip“ – skaičius Y reikšmių, kurias žinome.

```
function [ats1,ats2] = vykdyk(y,x,l,s,laip)
sigma = s;
lambda = l;
 [eil stulp] = size(y);
eil = laip;
pY = y(1:eil,:);
[a,e,t]=pazingsnine(pY,x,lambda,sigma,20,1000);
prog = prognoze(x,a,t);
ats1 = [y,prog];
ats2 = [a,e,t];
end
```

VI.11. Prognozės funkcija :

Funkcija skaičiuojanti prognozuojamas Y koncentracijų reikšmes pagal X atsako kreivių duomenis, su pažingsninės regresijos metu apskaičiuotais koeficientais α , θ, λ .

```
function prognozuojamos = prognoze(x,alpha,teta)
[eil stulp] =size(x);
f = zeros(eil,1);
[eil1 stulp] = size(alpha);
prognozuojamos = zeros(eil,1);
for j = 1:eil1
   Phi = funkcijaPhi(x,teta(j,:));
   f = f+(alpha(j,1)*(Phi'))';
   prognozuojamos = [prognozuojamos,f];
end;
prognozuojamos = prognozuojamos(:,2:eil1+1);
end
```

VI.2. Programos paleidimas

Norint paleisti modelio veikimą Matlab programos terpėje, reikia turėti visas bylas. Jų sąrašas:

- ☐ EuklSk2.m
- ☐ funkcijaL.m
- ☐ funkcijaPhi.m
- ☐ gaunamH.m
- ☐ generuojamTeta.m
- ☐ pazingsnine.m
- ☐ x7.mat (duomenys)
- ☐ y7.mat (duomenys)
- ☐ doPCA.m
- ☐ sortem.m
- ☐ vykdyk1.m
- ☐ prognoze.m

Taigi, pasileidę programą Matlab, visų pirma įsikeliame duomenis komandomis:

>>load x7.mat

Taip sukeliame X matricą.

>>load y7.mat

Sukeliame Y vektorių (vieno stulpelio matricą).

>> X=1.0e+5*X(:,2:301);

Pasirenkame naudoti visus stulpelius išskyrus 1-ąjį, nes pirmojo stulpelio duomenys yra visi 0, patogumo dėlei visą matricą padauginame iš 10^5, nes duomenys X yra labai maži;

```
>> Y1=Y(3997:4096,:);
>> X1=X(3997:4096,:);
```

Iš turimų duomenų bazių, sudarome testines duomenų bazes Y1 ir X1, kurios bus naudojamos modelio testavimui (šiuo atveju atsitiktinai atrinkome 100 koncentracijų);

```
>> Y=Y(1:3996,:);
>>X=X(1:3996,:);
```

Iš likusių duomenų sudarome matricas, kurios bus naudojamos modelio apmokymui.

```
>> csvwrite('Apmokymui.csv',[Y,X]);
>> csvwrite('Prognozavimui.csv',[Y1,X1]);
```

Atskiriame apmokymo bei testavimo duomenų bazes;

```
>> [ats1,ats2] = vykdyk(Y,X,1,0.5,500);
```

```
>>csvwrite('ivertiniai1.csv',ats2);
```

Išvedame sugeneruotas reikšmes, atitinkamai pirmas stulpelis – α (stulpelis), antras – ε (stulpelis), visi likę stulpeliai – pažingsninės regresijos metu atrinktos optimalios θ reikšmės;

```
>>csvwrite('prognoze2.csv',prog2);
```

Išvedame Y1 testavimo reikšmes, bei kiekviename *k* pažingsninės regresijos žingsnyje gautas prognozuojamas Y reikšmes

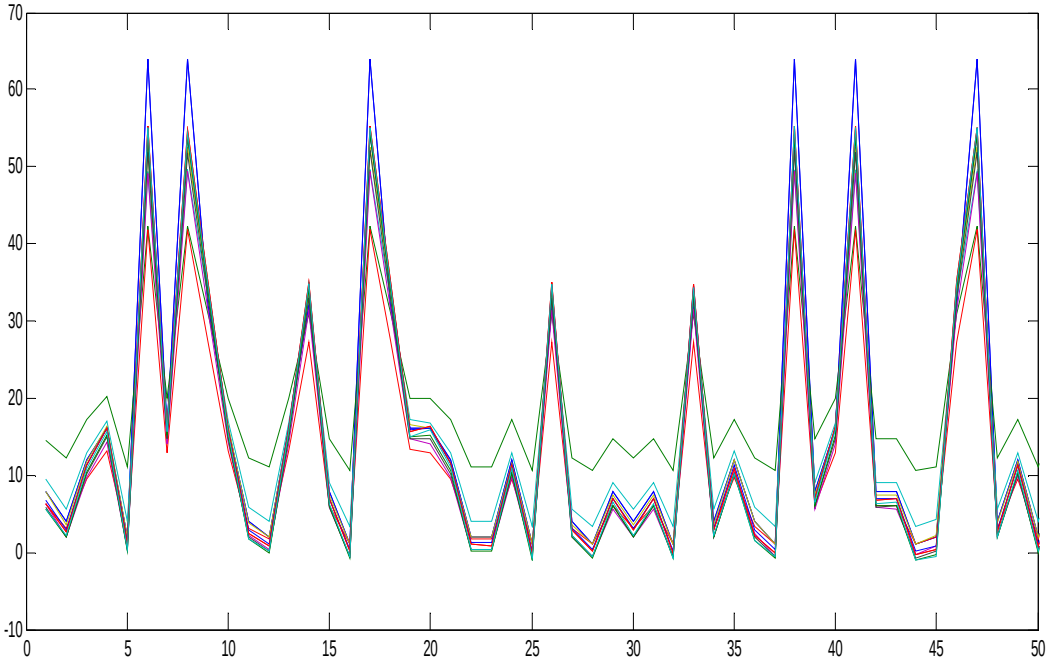
VI.3. Gauti rezultatai ir išvados

Paeiliui buvo atliekami prognozavimai keičiant parametrus λ, σ, atsitiktinių vektorių θ generavimų skaičių, bei pažingsninės regresijos žingsnių skaičių ir stebimos MSE reikšmės. Žemiau pateikti geriausi darbo rezultatai:

VI.3.1.Nustatome pasirenkamuosius parametrus “>> [ats1,ats2] = vykdyk(Y,X,1,0.5,500);

λ=1; σ=0.5; *l*=500;

Žemiau pateikiami grafikai kiekviename *k* žingsnyje.



Gauti įverčiai bei MSE:

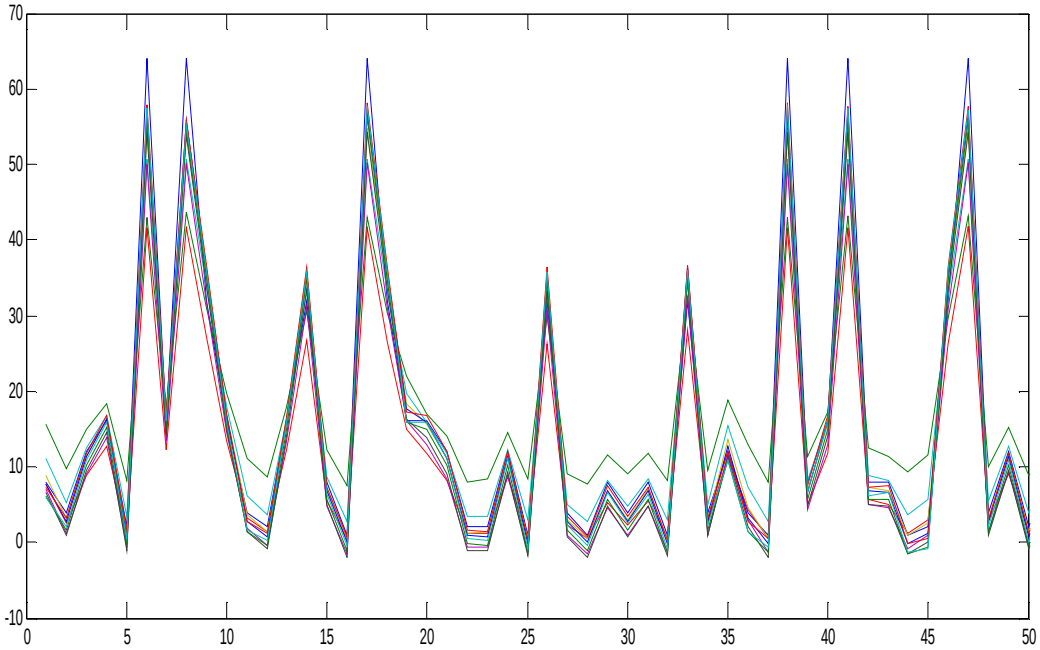
k	α	MSE
1	44.82	3.40E+05
2	-11.506	56602
3	8.1997	32265
4	-4.8156	20439
5	3.696	16057
6	-3.0539	12905
7	2.4795	11423
8	-1.8795	9739.1
9	1.5642	8987.6
10	-1.2742	8091.4

θ vektoriai su kuriais buvo apskaičiuotos prognozes, pateikti elektroniniame priede.

VI.3.2. >> [ats1,ats2] = vykdyk(Y,X,0.5,0.1,500);

λ=0.5; σ=1; l=500;

Žemiau pateikiami grafikai kiekviename k žingsnyje.

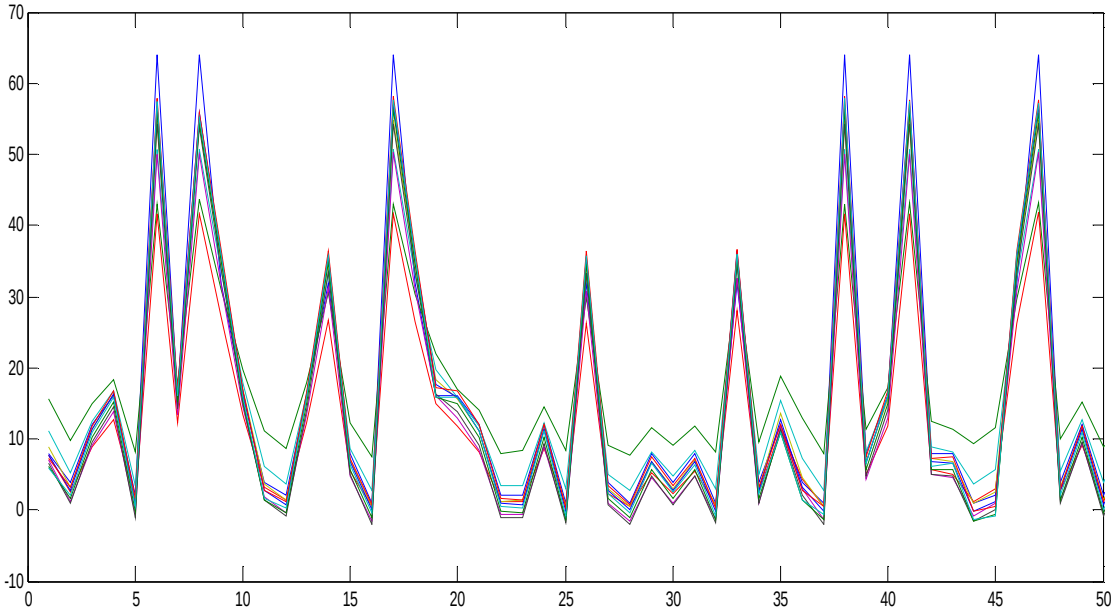


Gauti MSE:

k	□	MSE
1	45.062	3.40E+05
2	-12.467	57485
3	9.4473	34840
4	-6.3088	22647
5	5.1337	16543
6	-3.6719	12992
7	3.2507	11031
8	-3.343	9577.9
9	2.463	8057.8
10	-1.9555	7151.4

θ vektoriai su kuriais buvo apskaičiuotos prognozes, pateikti elektroniniame priede.

VI.3.3. >> [ats1,ats2] = vykdyk(Y,X,0.5,0.1,500);

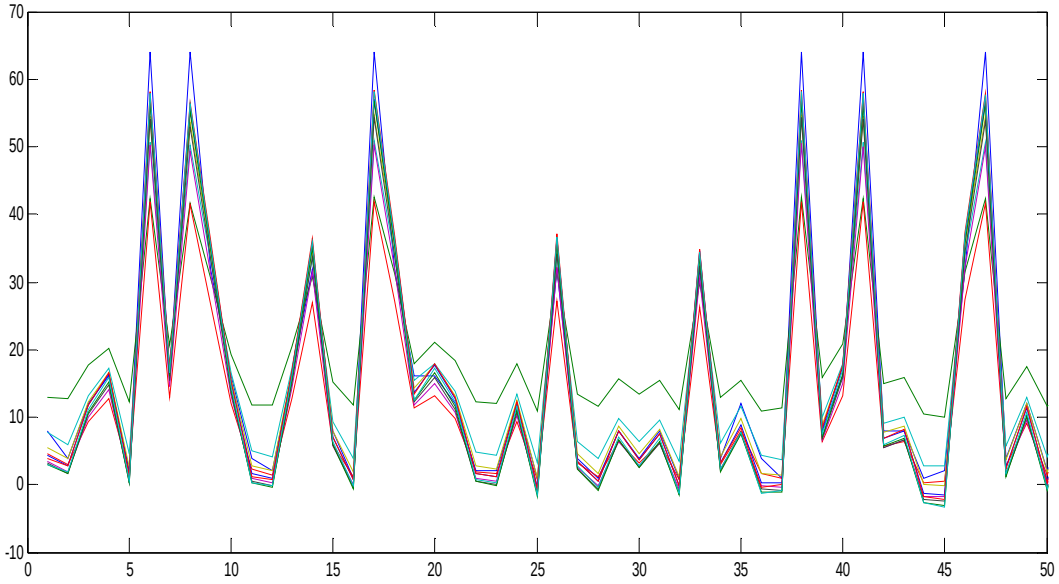


Gauti MSE:

k	□	MSE
1	44.681	3.40E+05
2	-12.346	61781
3	8.7474	33515
4	-5.6955	21394
5	4.2452	15568
6	-3.0075	12380
7	2.7065	10585
8	-1.8158	8911.5
9	1.348	8297.6
10	-1.3084	7764.7

□ vektoriai su kuriais buvo apskaičiuotos prognozes, pateikti elektroniniame priede.

VI.3.4.>> [ats1,ats2] = vykdyk(Y,X,0.5,0.01,500);



Gauti MSE:

k	□	MSE
1	45.155	3.40E+05
2	-12.762	61846
3	9.53	33242
4	-6.2643	20910
5	4.9272	14993
6	-3.8327	11724
7	3.0722	9635.8
8	-2.5231	8358.3
9	2.0538	7469.6
10	-1.4741	6992.9

θ vektoriai su kuriais buvo apskaičiuotos prognozes, pateikti elektroniniame priede.

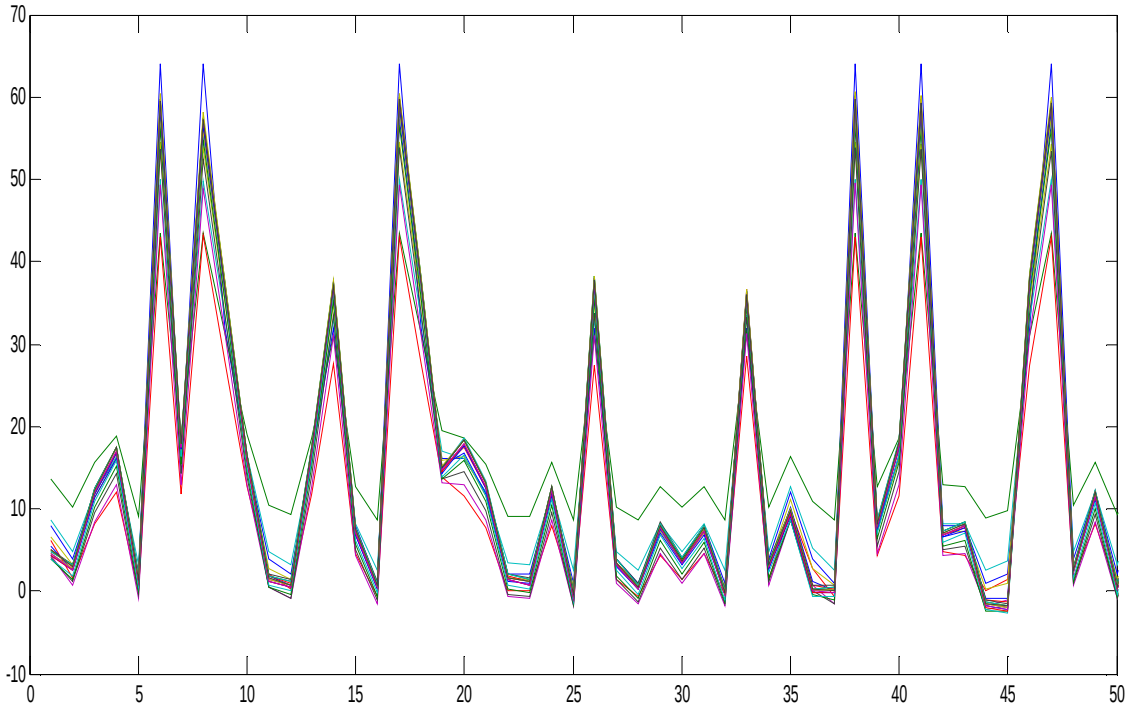
Toliau, fiksavome σ su kuriuo modelis pasiekia lokalu minimumą (gerai prognozuoja koncentracijas) ir

stabėdami kaip kinta MSE keitėme σ reikšmes. Suradę geriausią variantą, padidinome pažingsninės regresijos žingsniu skaičių bei atsitiktinių vektorių σ generavimo skaičių.

Gautų prognozuojamų reikšmių grafikai pateikti žemiau:

VI.3.5. >> [ats1,ats2] = vykdyk(Y,X,0.5,0.01,500);

λ=0.5; σ=0.01.

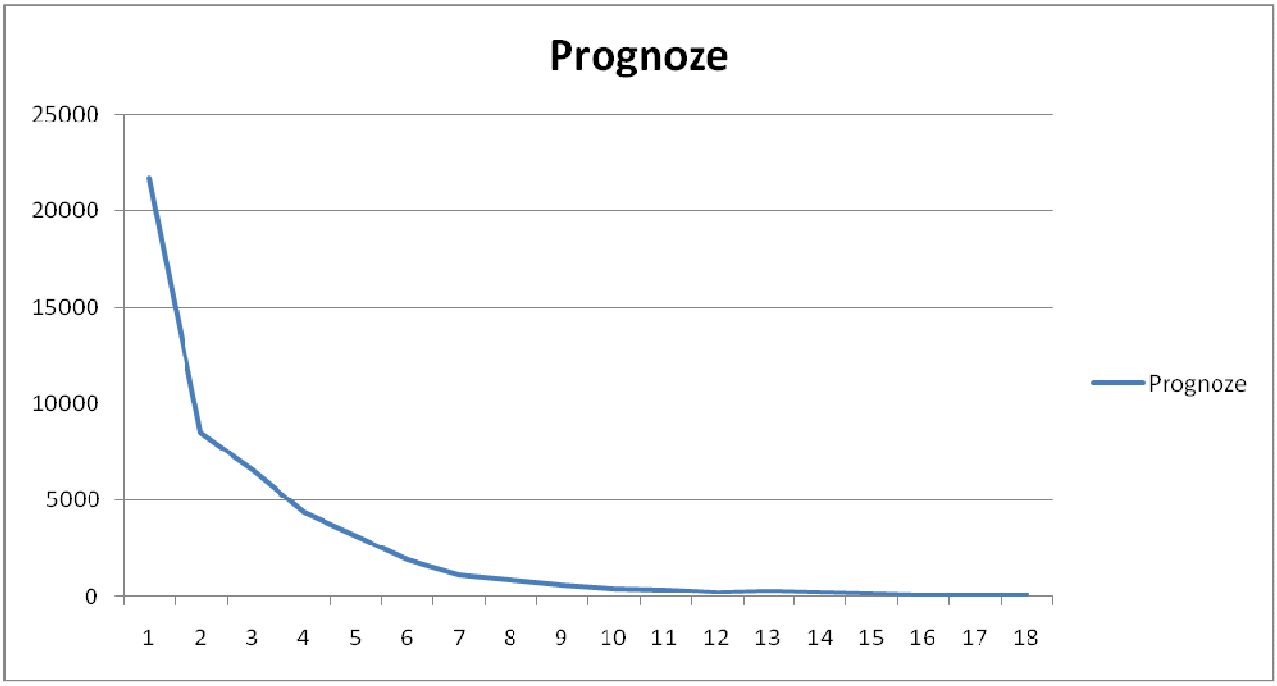


Gauti įverčiai:

k	α	MSE
1	45.369	3.40E+05
2	-10.917	55844
3	7.9354	34172
4	-6.6328	25661
5	5.6657	19114
6	-4.7849	14761
7	3.7521	10625
8	-2.9051	8714
9	2.4779	6567.7
10	-2.0629	5696.8

k	α	MSE
11	1.732	4602.1
12	-1.4497	4093.8
13	1.1848	4228.3
14	-1.4306	4140.8
15	1.1814	4653.8
16	-0.95553	4665.9
17	0.60468	4533.8
18	-0.65307	4437.2
19	0.86212	4351.8
20	-0.84414	4252.6

Modelio paklaida mažėja, didėjant *k* žingsnių skaičiui, tačiau optimalus modelis reiškia ir optimalų regresorių sakičių. Pastebėjome jog MSE pokyčiai mažėja didėjant *k* :



VI.3. Pav. MSE pokyčių grafikas

Iš grafiko matome jog kai *k* =10 MSE pokyčiai yra labai maži, todėl koncentracijoms prognozuoti paėmėme 10 pažingsninės regresijos žingsnių.

Skaitinė išraiška prognozuotų koncentracijų pateikta elektroniniame priede.

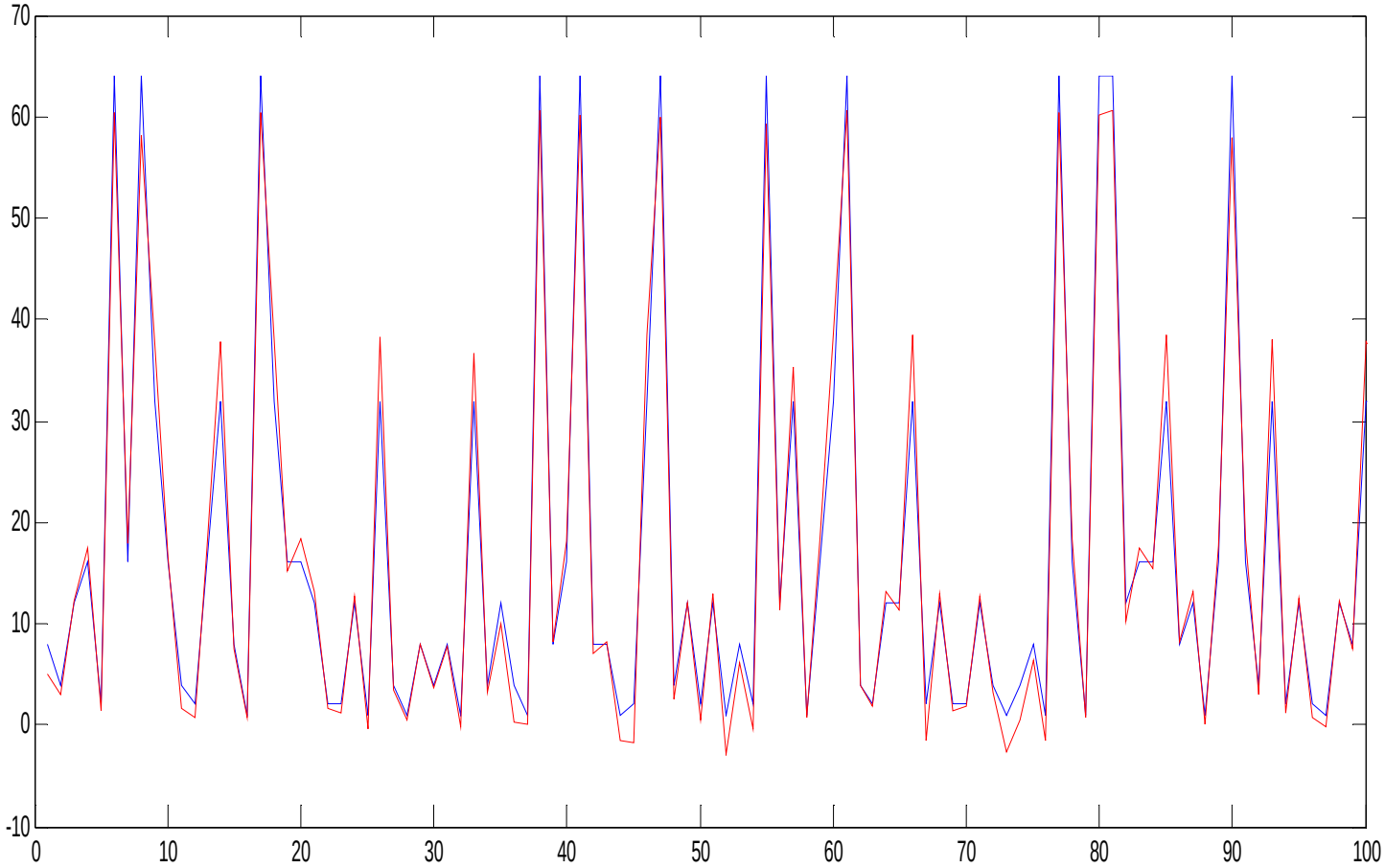
Suskaičiavome sumin koreliacijos koeficientą tarp realiųjų duomenų kreivės bei prognozuojamų:

```
>> corrcoef(ats1(:,1),ats1(:,end))

   1.0000    0.9871
   0.9871    1.0000
```

Aukštas koreliacijos koeficientas rodo, jog kreivės koreliuoja tarpusavyje, o sąryšis yra labai stiprus.

Žemiau pateikiame realių Koncentracijų reikšmes bei suprognuozuotas:



V.I.2 Gautų prognozuotų koncentracijų bei tikrų koncentracijų kreivės

DARBO IŠVADA

Pritaikėme nepilnai apibūdintų stebėjimų analizės metodiką (angl. Semi-supervised learning) **(Kai Yu, Volker Tresp [4])** bei **P.L.Bartlet [1]** darbo idėją, jog netiesinės regresijos modelyje svorių dydis yra svarbiau nei narių kiekis, panaudodami PCA (Pagrindinių komponenčių analize) sumažinome turimų duomenų rinkinius, taip išsaugodami didžiąją dalį informacijos apie tuos duomenis. Sukonstravome pažingsninės regresijos matematinį modelį, kuris gerai prognozuoja tirpalo koncentracijas, radome netiesinės regresijos optimalius svorių dydžius,. Atlikome daug iteracijų su įvairiomis parametrų reikšmėmis ir stebėjome kaip kinta MSE (klaidų kvadratų vidurkį) keičiant minėtus parametrus, išrinkome optimalų regresorių k=12 skaičkų su tam tikrais parametrais (žiūrėti punktą **VI.3.5.).** Vidutinė prognozuojamos koncentracijos paklaida gaunama ~2 mol., tačiau ši paklaida gali būti mažinama atitinkamai didinant generuojamų atsitiktinių vektorių θ skaičių, didinant pažingsninės regresijos skaičių k , bei paieškos bandymų būdu randant optimalius σ ir λ parametrus.

Literatūra

[1] Bartlett, P.L. *„The sample complexity of pattern classification with neuralnetworks: the size of the weights is more important than the size of thenetwork“*(Information Theory, IEEE Transactions on).

[2] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Sch¨olkopf *„Learning with Local and Global Consistency“*, Max Planck Institute for Biological Cybernetics, 72076 Tuebingen, Germany.

[3] John Fox, *Nonlinear Regression and Nonlinear Least Squares* (Appendix to An R and S-PLUS Companion to Applied Regression).

[4] Kai Yu, Volker Tresp *„Semi-supervised Induction with Basis Functions”* (Corporate Technology, Siemens AG Otto-Hahn-Ring 6, 81739, Munich, Germany)

[5] Marius Mocevičius “*SAS/IML programos panaudojimas realizuojant naujus regresijos ir klasterizavimo modelius“* (Vilnius, 2008).

[6] M.Puočiauskas *„Atsitiktinių signalų identifikavimas naudojant dirbtinius neuroninius tinklus“*. VU baigiamasis magistro darbas, 2001.

[7] R. Maslovskis *„Biocheminių procesų klasifikavimas naudojant dirbtinius neuroninius tinklus“*, Vilnius 2007, (Daktaro disertacija, VU).

[8] R.Baronas, J.Christensen, F.Ivanauskas, J.Kulys *„Computer simalation of amperometric biosensor response to mixtures of compounds“***.** Nonliner analysis: modelling and control, 7(2), 2002, pp.3-14

[9] R.Baronas, F.Ivanauskas, R.Maslovskis, P.Vaitkus *„Tirpalų mišinių koncentracijų klasifikavimas“*(Liet.matem.rink.,44,spec. Nr., 2004, 682-686).

SUMMARY

The purpose of this work is to create non liner regression model with optimal number of coefficients and optimal values of these coefficients, and predict liquor concentration having values of amperimetric data.

The main task of the work:

- To use Semi-supervised learning algorithm while creating the model (Kai Yu, Volker Tresp [4])
- To apply P.L. Bartlet [1] idea, that in non-linear regression model the value of the weights is more important than number of these weights.
- To create mathematical model that could calculate optimal non-liner regression' weights using stochastic search, and could determine the concentration of liquor according biosensors response curve

While creating the model, we met model optimization problem. The optimal model means the optimal number of model' weights and the values of weights with whom the model best predicts the concentration of the liquor. We used step wise regression method to determine the optimal number of weights. For the calculations we choose non-liner regressions sigmoid function according P.L. Bartlet [1] work results. We also implemented semi supervised data analysis method that was chosen based on Volker Tresp [3] work results. These methods are widely applied in the mathematical modeling and information technologies. What is more, we also applied partial component analysis and reduced the data sets of response curve, without loosing significant information about these data.

The parameters λ, σ of the model we chosen in experimental way while observing the models MSE.

The results showed that this model predicts the values of the concentration very good with appropriate parameters and appropriate number of regressors.