VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Baigiamasis magistro darbas

Atsitiktinių signalų identifikavimas naudojant dirbtinius neuroninius tinklus

Paruošė: II Magistratūros kurso B srauto studentas
Mantas Puočiauskas

Darbo vadovas:
Šarūnas Raudys

Vilnius
2001

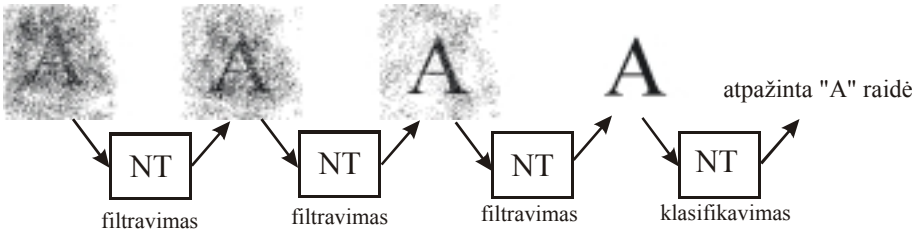
Turinys

Įvadas

Žmonės ir kiti gyvūnai informaciją apdoroja neuroninių tinklų (toliau NT) pagalba. NT yra suformuoti iš milijardų neuronų (nervinių ląstelių), kurie apsikeitinėja elektriniais impulsais, vadinamais aktyvacijos potencialais. Kompiuteriniai algoritmai, kopijuojantys šias biologines struktūras, vadinami dirbtiniais neuroniniais tinklais, tam kad juos atskirti nuo natūralių neuroninių tinklų. Tačiau daugelis mokslininkų ir inžinierių nėra tokie formalūs ir naudoja terminą "neuroninis tinklas" apibūdindami ir biologines, ir nebiologines struktūras.

Neuroninis tinklas – skaičiavimo modelis, kurio pagalba sprendžiami klasifikavimo, prognozavimo, klasterizavimo, duomenų savybių radimo ir kiti uždaviniai.

Pagrindinė problema, apsunkinanti sėkmingą neuroninio tinklo darbą, yra ta, kad pateikiama neuroniniam tinklui apdoroti informacija (duomenys) dažniausiai yra triukšminga (pavyzdžiui, jei turime žmogaus kalbos garso įrašą, tai jame papildomai gali būti aplinkinių žmonių kalba, fone grojanti muzika ir panašiai). Nors teoriškai įmanoma, kad NT tuo pačiu metu ignoruotų triukšmą ir apdorotų informaciją, žymiai praktiškiau būtų suskaidyti uždavinį į du loginius etapus: pirmame etape neuroninis tinklas pašalina kiek įmanoma daugiau triukšmo, sekančiame etape atlieka tai, kam jis yra numatytas.



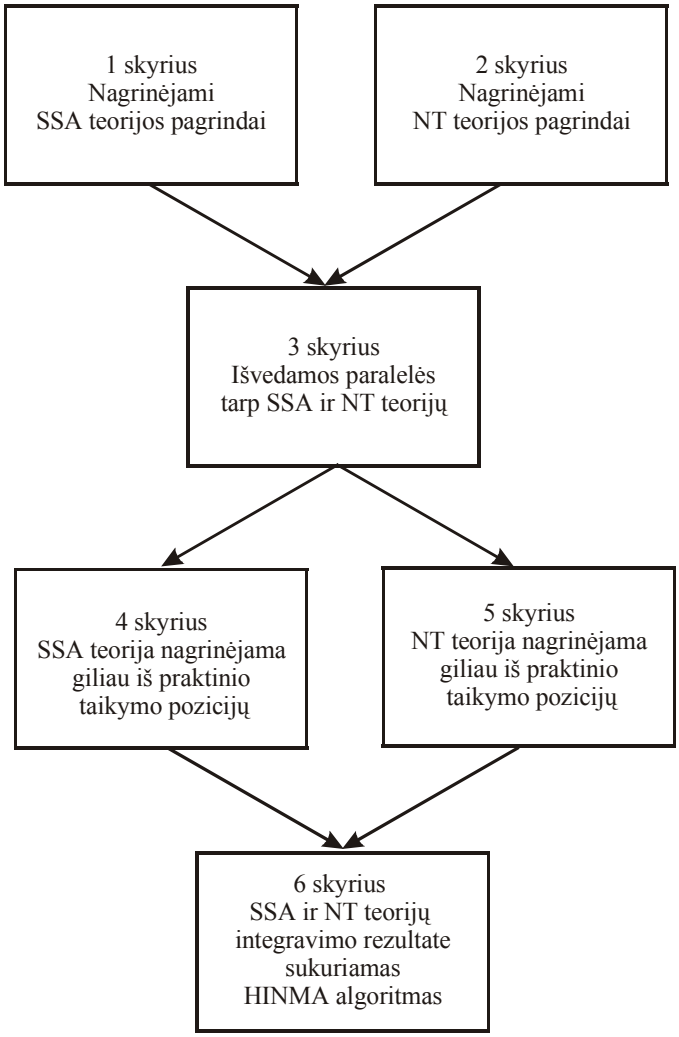
1 pav. Signalo kaskadinio apdirbimo koncepcija. Du loginiai etapai: pirmajame etape šalinamas triukšmas, antrajame etape neuroninis tinklas atlieka tai, kam jis ir numatytas.

Tradiciškai triukšmo šalinimo ir signalo išvalymo problematiką nagrinėja mokslo sritis vadinama skaitmeninių signalų apdorojimu (toliau SSA).

Šio darbo tikslas yra išnagrinėti kaip SSA metodus galima susieti ir pritaikyti neuroniniams tinklams, kurie šalina trukdžius iš triukšmingo informacinio kanalo ir atstato originalų signalą, ir sukurti tam skirtus algoritmus bei parodyti, kad idėjos paimtos iš SSA ir NT gali būti apjungtos ir rezultate duoti efektyvius algoritmus.

Trumpa apžvalga

Šio darbo tikslas yra išnagrinėti kaip SSA teorijos rezultatus būtų galima pritaikyti neuroniniams tinklams, kurie šalina triukšmą iš signalo. Darbe paraleliai nagrinėjamos SSA ir neuroninių tinklų teorijos ir ieškoma sąlyčio taškų, jų abiejų integravimo rezultate sukuriamas naujas algoritmas, įgalinantis pagerinti neuroninio tinklo triukšmo šalinimo iš signalų efektyvumą. Žemiau pavaizduota darbo struktūrinė schema.



2 pav. Darbo struktūrinė schema.

1 skyrius skirtas SSA teorijos pagrindų nagrinėjimui. Nagrinėjama kas tai yra signalai, kaip jie yra keičiami praeidami pro sistemas, įvedamos svarbios signalų nagrinėjimui sąsūkos ir koreliacijos sąvokos.

2 skyriuje pristatomi dirbtiniai neuroniniai tinklai ir jų mokymas, nagrinėjamas dažniausiai praktikoje naudojamas neuroninių tinklų mokymo atbulinio sklidimo algoritmas.

3 skyriuje išvedamos paralelės tarp skaitmeninių signalų apdorojimo metodų ir neuroninių tinklų. Parodoma, kad neuroniniais tinklais galima atlikti skaitmeninių signalų apdorojimo operacijas.

4 skyriuje SSA teorija nagrinėjama giliau iš praktinio taikymo pozicijų. Nagrinėjama kokiais būdais signaluose gali būti koduojama informacija. Pristatoma svarbi signalų teorijoje fazės sąvoka, nagrinėjamas tiesinių filtrų sudarymas, įvedama optimalaus filtravimo sąvoka.

5 skyriuje neuroninių tinklų teorija nagrinėjama giliau iš praktinio taikymo pozicijų – aptariama neuroninių tinklų mokymo problematika ir nagrinėjami mokymo optimizavimo metodai.

6 skyriuje SSA ir neuroninių tinklų teorijų integravimo rezultate, idant pagerinti triukšmo pašalinimo efektyvumą, pasiūlytas hibridinis neuroninio tinklo, šalinančio triukšmą iš signalo, mokymo algoritmas (HINMA), paremtas SSA teorijos panaudojimu dirbtinio neuroninio tinklo mokymui. Pasiūlytas algoritmas palygintas su 12 žinomu, šį uždavinį sprendžiančių, algoritmų.

Rezultatų apibendrinime ir išvadose susumuojami darbo rezultatai.

Darbe kompiuteriniams eksperimentams atlikti naudotas *MATLAB* programinis paketas. *MATLAB* programavimo kalba yra aukšto lygio kalba ir jos konstrukcijos beveik atitinka įprastinę moksle ir inžinerijoje naudojamą matematinę notaciją, todėl *MATLAB* programos yra gana išraiškingos ir lengvai skaitomos. Todėl visų darbe atliktų kompiuterinių eksperimentų *MATLAB* programų tekstai yra pateikti prieduose.

Sutrumpinimai

dB – decibelas, dešimtoji belo dalis. Atitinka dviejų vienavardžių fizikinių dydžių (energijos ar jėgos) santykio 10 ar 20 dešimtainių logaritmų; $1\,dB = 10\log_{10}\frac{P_2}{P_1}$ (jei energija $P_2 = 10P_1$), $1\,dB = 20\log_{10}\frac{F_2}{F_1}$ (jei jėga $F_2 = \sqrt{10}F_1$).
Pavyzdžiui, stiprintuvo įėjimo ir išėjimo galių santykis vadinamas stiprinimo koeficientu ir gali būti išreikštas decibelais sekančiai: $G(dB) = 10\log_{10}\frac{P_{out}}{P_{in}}$. [Lap00]

GFT – greitoji Furjė transformacija
MVK – mažiausias vidutinis kvadratinis (nuokrypis)
NT – neuroninis tinklas
SSA – skaitmeninis signalų apdorojimas

1. Skaitmeninių signalų apdorojimas

Triukšmo šalinimas iš signalų lieka viena iš svarbiausių problemų daugybėje sričių: telekomunikacijose, medicininių duomenų apdorojime, radaruose ir sonaruose, aukštos kokybės muzikos atkūrime ir t.t. Triukšmo šalinimo iš signalų problematiką tradiciškai nagrinėja skaitmeninių signalų apdorojimo (SSA) teorija. Tačiau tai jokiu būdu nereiškia, kad SSA yra vienintelis ir optimalus galimas sprendimas. Viena iš alternatyvų yra neuroninių tinklų panaudojimas. Šio darbo tikslas yra išnagrinėti kaip SSA teorijos rezultatus būtų galima pritaikyti neuroniniams tinklams, kurie šalina triukšmą iš signalo, bet šiam tikslui pasiekti pirmiausia reikia išnagrinėti SSA pagrindines sąvokas ir dėsningumus. Šis skyrius ir nagrinėja SSA teorijos pagrindus.

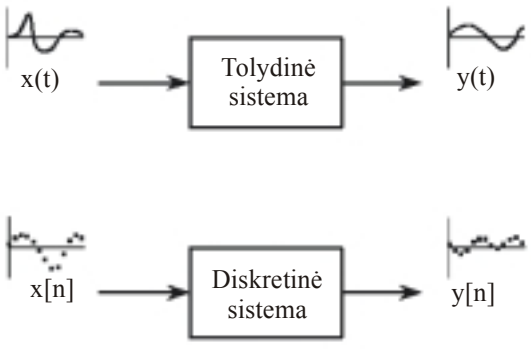
1.1. poskyryje pristatoma fundamentali SSA koncepcija – superpozicija, kuri įgalina sudėtingus signalus nagrinėti kaip daugelio paprastų signalų sumą. SSA teorija nagrinėja kaip signalai yra keičiami tiesinėse sistemose, todėl 1.2. poskyryje pristatoma tiesinės sistemos sąvoka. 1.3. poskyryje nagrinėjama kas tai yra tiesinė sistema ir kokias ji turi savybes. 1.4. poskyryje giliau nagrinėjama superpozicija – sudėtingų signalų skaidymo (dekompozicija) į paprastus būdai ir suskaidytų signalų apjungimas į galutinį signalą (sintezė). 1.5. poskyryje pristatomos pagrindinės SSA operacijos su signalais, kurios savyje realizuoja superpozicijos koncepciją – sąsūka ir koreliacija. 1.6. poskyryje nagrinėjamos autokoreliacijos ir tarpusavio koreliacijos savybės. 1.7. poskyryje pateikiamos išvados.

1.1. Superpozicija – skaitmeninių signalų apdorojimo pagrindas

Skaitmeninių signalų apdorojimo (toliau SSA) metodai yra pagrįsti "skaldyk ir valdyk" strategija, kuri vadinama *superpozicija*. Nagrinėjamas signalas yra suskaidomas į atskirus paprastus komponentus, kiekvienas komponentas individualiai apdorojamas ir gauti rezultatai apjungiami. Šio metodo pranašumas tame, kad sudėtinga problema suskaidoma į keletą paprastų. Superpozicija gali būti taikoma tik *tiesinėms sistemoms* (terminas, nusakantis kad galioja tam tikros matematinės taisyklės). Laimei, daugelis reiškinių, sutinkamų moksle ir inžinerijoje, papuola į šią kategoriją.

Signalas – tai aprašas, kuris nusako kaip vienas parametras priklauso nuo kito parametro. Pavyzdžiui, įtampos kitimas laike elektrinėje grandinėje, arba apšviestumas priklausantis nuo atstumo paveiksle.

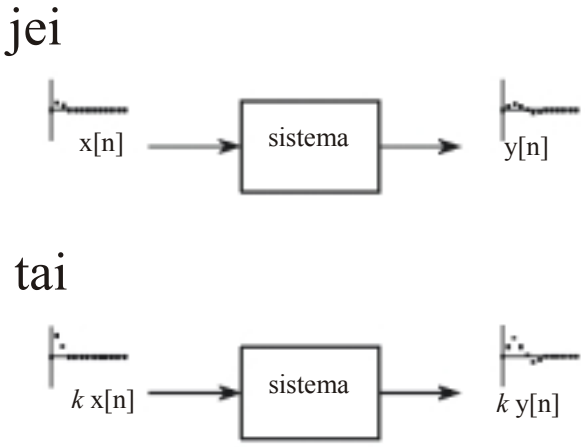
Sistema – tai bet koks procesas, kuris sukuria išėjimo signalą, atreaguodamas į įėjimo signalą. [Smi99]



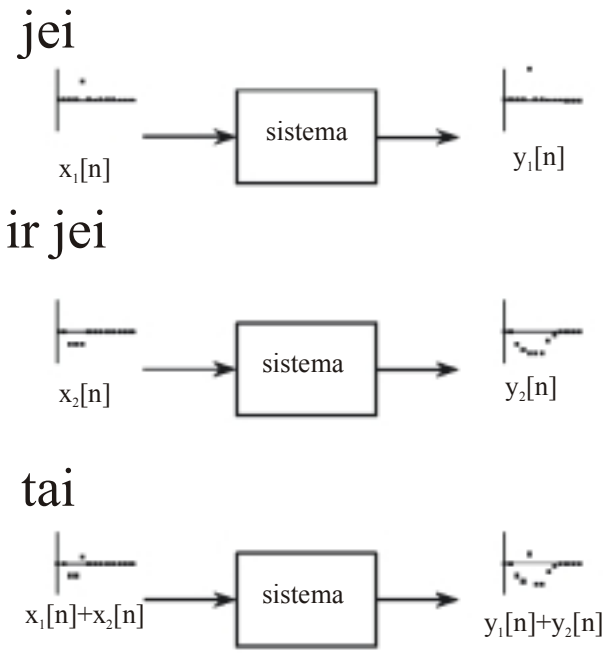
3 pav. Signalų ir sistemų terminologija. Tolydiniai signalai vaizduojami su skliausteliais, o diskretiniai signalai vaizduojami su laužtiniais skliausteliais. Jei nėra geresnio vardo, tai įėjimo signalas vadinamas x(t) arba x[n], o išėjimo - y(t) arba y[n].

1.2. Tiesinė sistema

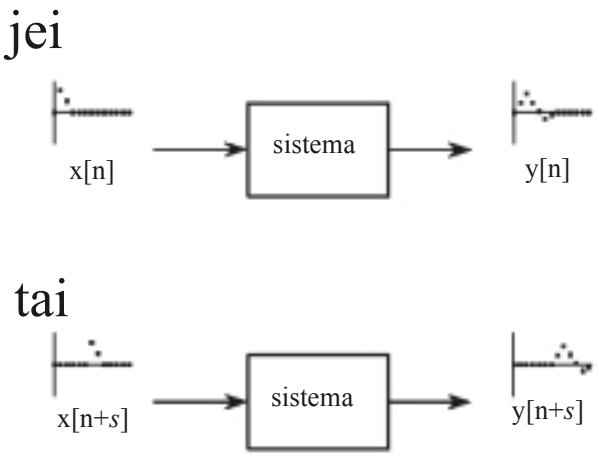
Sistema vadinama *tiesine*, jei ji patenkina dvi matematines savybes: homogeniškumą ir adityvumą. Trečia savybė – poslinkio invariantiškumas nėra griežtas tiesiškumo reikalavimas, tačiau būtina, jei norima pritaikyti daugumą SSA metodų.



4 pav. *Homogeniškumo* apibrėžimas. Sistema vadinama homogeniška, jei amplitudės pasikeitimas įėjime yra identiškas amplitudės pokyčiui išėjime. Tai yra, jei įėjimas x[n] išėjime duoda y[n], tai *k*x[n] duoda *k*y[n], bet kokiam signalui x[n], ir bet kokiai konstantai *k*.



5 pav. *Adityvumo* apibrėžimas. Sistema vadinama adityvia, jei sudedami signalai sklinda sistema nesąveikaudami tarpusavyje. Formaliai, jei $x_1[n]$ išėjime duoda $y_1[n]$, ir jei $x_2[n]$ išėjime duoda $y_2[n]$, tai $x_1[n]+x_2[n]$ duoda $y_2[n]+y_2[n]$.



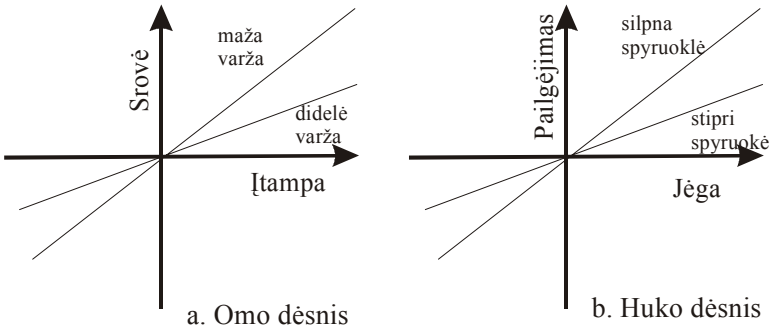
6 pav. *Poslinkio invariantiškumo* apibrėžimas. Sistema vadinama poslinkiui invariantiška, jei poslinkis įėjime sukelia identišką poslinkį išėjime. Tai yra, jei x[n] duoda y[n], tai x[n+*s*] duoda y[n+*s*], bet kokiam signalui x[n], ir bet kokiai konstantai *s*.

Poslinkio invariantiškumas yra svarbus tuo, kad jis reiškia, kad sistemos charakteristikos nesikeičia laike. Pvz., tarkime norime sukurti skaitmeninį filtrą, kuris kompensuotų signalo iškraipymus telefono linijoje. Tarkime pagaminome filtrą, kuris balsą padaro natūralesnį ir aiškesnį. Tačiau keičiantis aplinkos temperatūrai, keičiasi telefono linijos parametrai ir filtras tampa nebesuderintas su telefono linija ir pradeda

blogai veikti. Tokios situacijos reikalauja sudėtingo algoritmo, kuris *adaptuotųsi* prie besikeičiančių sąlygų.

Homogeniškumas, adityvumas ir poslinkio invariantiškumas yra svarbūs, nes suteikia matematinį pagrindą apibrėžiant tiesines sistemas. Tačiau šios savybės daugeliui mokslininkų ir inžinierių nesuteikia intuityvaus suvokimo kas gi yra tiesinė sistema. Todėl naudinga įvesti sąvokas *statinis tiesiškumas*[1] ir *sinusoidinis tikslumas*[2]. Šios sąvokos nėra labai svarbios matematiniu požiūriu, bet padeda žmonėms geriau suvokti tiesines sistemas.

Statinis tiesiškumas apibrėžia, kaip tiesinė sistema reaguoja, kai signalai nesikeičia, t.y. kai jie yra statiniai. Statinis tiesinės sistemos atsakas yra labai paprastas: išėjimas yra įėjimas, padaugintas iš konstantos.



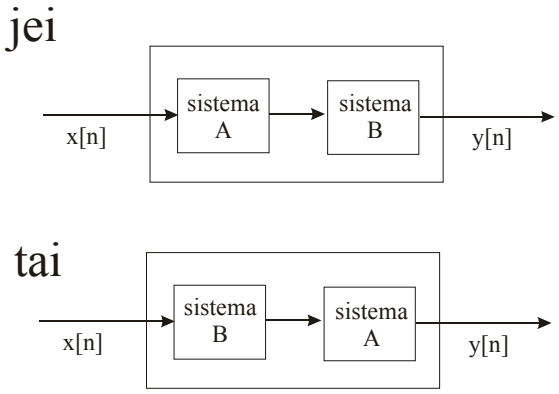
7 pav. Statinio tiesiškumo pavyzdžiai.
a pav. Omo dėsnyje srovė per rezistorių yra lygi įtampai ant rezistoriaus, padalintai iš varžos.
b pav. Huko dėsnyje spyruoklės pailgėjimas yra lygus jėgai padaugintai iš spyruoklės standumo koeficiento.

Visos tiesinės sistemos yra statiškai tiesiškos (tačiau atvirkščias teiginys ne visada teisingas).

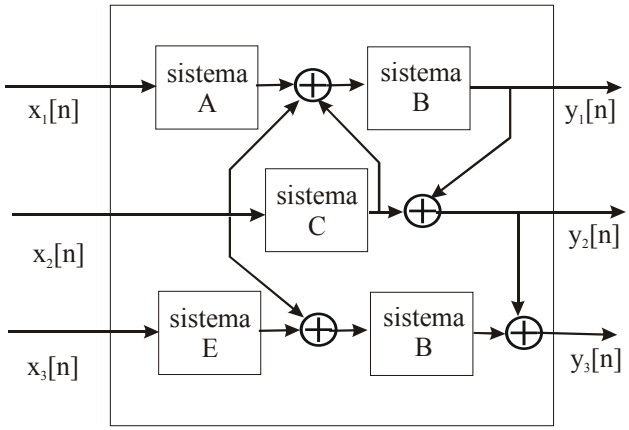
1.3. Tiesinių sistemų savybės

Tiesinių sistemų svarbi charakteristika yra jų reakcija į sinusoidinius signalus. Tai savybė, kuri vadinama *sinusoidiniu tikslumu*: jei tiesinės sistemos įėjime yra sinusoidinė banga, tai išėjime taip pat bus sinusoidinė banga lygiai to paties dažnio kaip ir įėjimo banga. Sinusoidės yra vienintelės bangos, turinčios tokią savybę. Pvz. nėra jokio pagrindo manyti, kad stačiakampė banga tiesinės sistemos įėjime duos stačiakampę bangą išėjime. Nors sinusoidė įėjime garantuoja sinusoidę išėjime, abi jos gali turėti skirtingą amplitudę ir fazę.

[1] iš angl. static linearity
[2] iš angl. sinusoidal fidelity

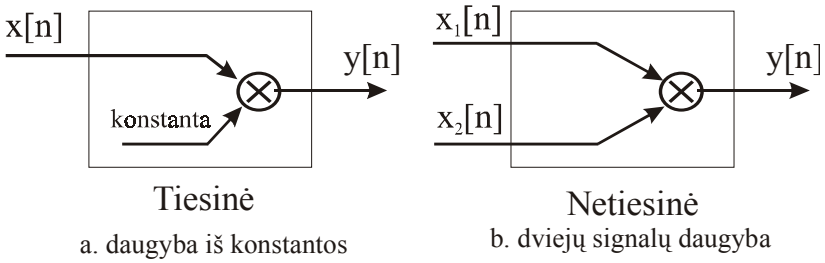


8 pav. Tiesinių sistemų *komutatyvumo* savybė. Kai dvi ar daugiau tiesinių sistemų jungiamos į kaskadą, sistemų išdėstymo tvarka neturi įtakos galutiniam rezultatui.



9 pav. Bet kokia sistema su daugeliu įėjimų ir/arba daugeliu išėjimų bus tiesiška, jei bus sudaryta iš tiesinių sistemų ir jų sumų.

Daugyba tiesinėse sistemose gali būti tiesinė arba netiesinė. Sistema, kuri daugina įėjimo signalą iš konstantos yra tiesinė. Tokia sistema yra stiprintuvas arba slopintuvas, priklausomai nuo to, ar konstanta didesnė ar mažesnė už vienetą. Jei sistema sudaugina signalą iš kito signalo, tai daugyba yra netiesinė.



a. daugyba iš konstantos b. dviejų signalų daugyba

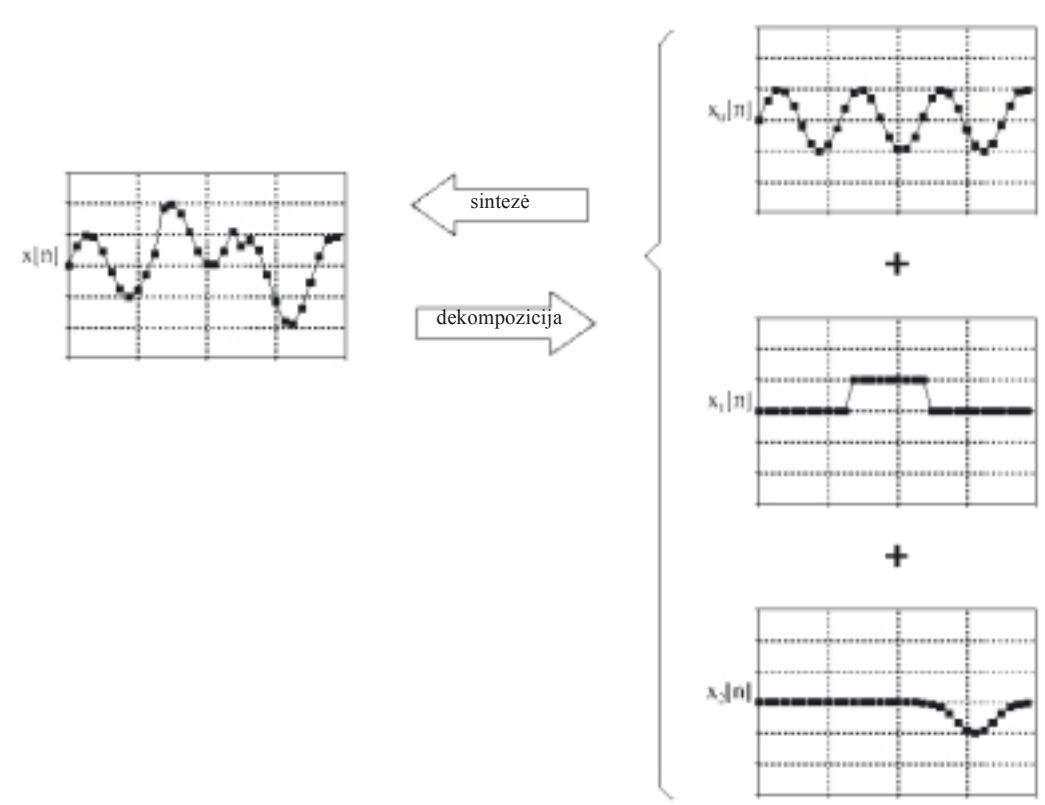
10 pav. Daugyba tiesinėse sistemose

1.4. Sintezė ir dekompozicija

Kai nagrinėjame tiesines sistemas, vienintelis būdas kaip gali būti gaunami signalai yra signalų daugyba ir sudėtis. Šis signalų gavimo būdas vadinamas signalų *sinteze*.

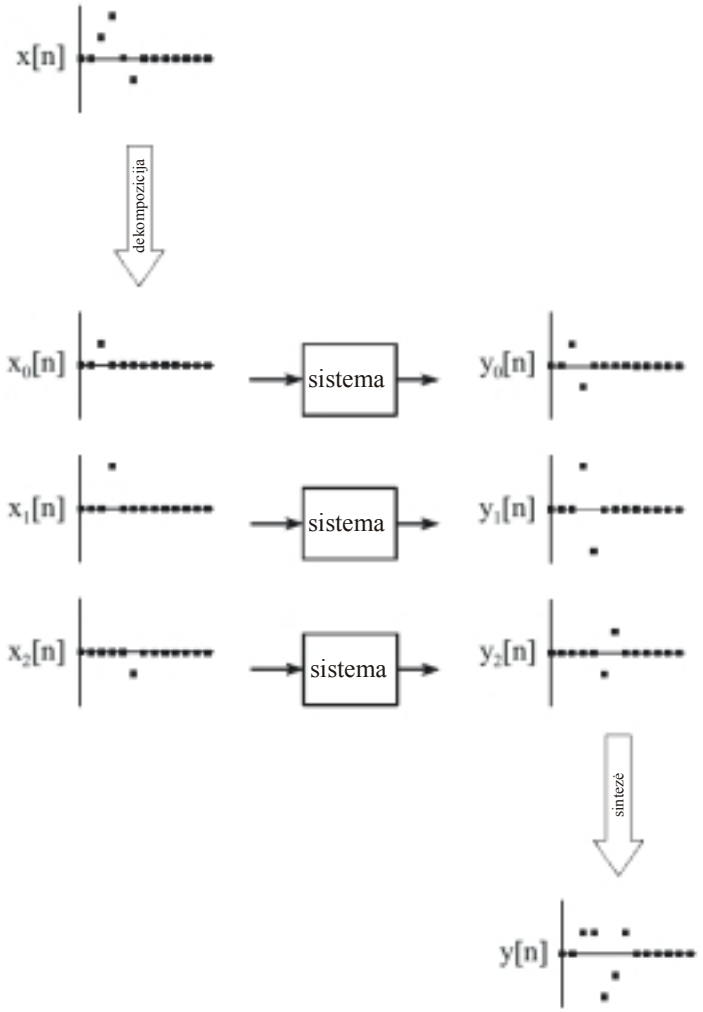
Dekompozicija yra atvirkštinė sintezei operacija, kai signalas suskaidomas į du ar daugiau komponentų.

Pavyzdžiui, skaičiai 15 ir 25 gali būti susintezuoti (sudėti) į skaičių 40. Palyginimui, skaičius 40 gali būti suskaidytas: 1+ 39, 2 + 38 ar 30.5 + 10.5 ir t.t.



11 pav. Signalų sintezė ir dekompozicija. Sintezėje dviejų ar daugiau signalų suma suformuoja signalą. Dekompozicija yra atvirkščias procesas.

Signalų dekompozicija ir sintezė yra SSA pagrindas.

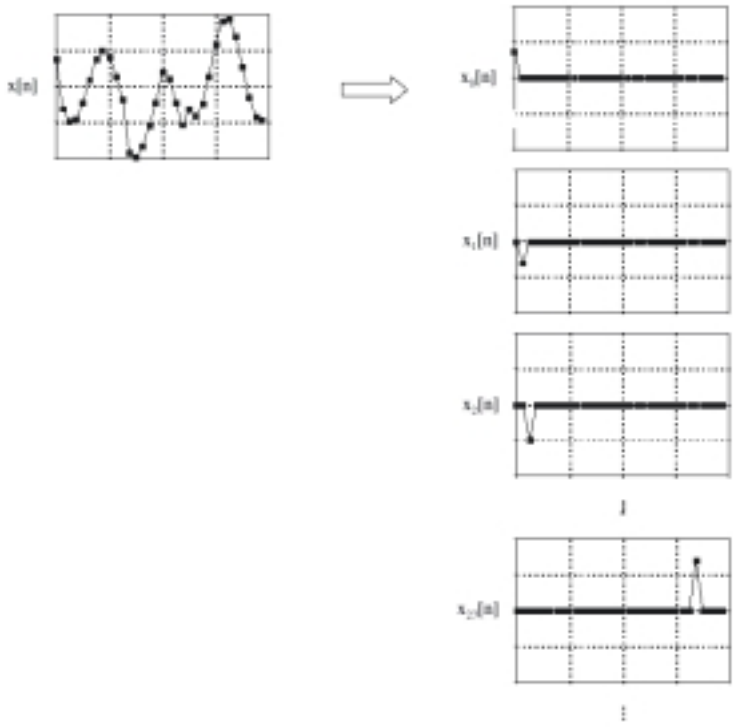


12 pav. SSA pagrindinė koncepcija. Kiekvienas signalas, kaip kad pvz. x[n], gali būti suskaidytas į adityvių komponentų grupę, pavaizduotų paveiksle kaip signalai $x_1[n]$, $x_2[n]$, x3[n]. Šie komponentai praėję tiesinę sistemą duoda rezultate signalus $y_1[n]$, $y_2[n]$, $y_3[n]$. Šių išėjimo signalų *sintezė* (sudėtis) yra lygi signalui, kuri duotų sistema kai įėjime yra signalas x[n]. Tai labai galinga idėja. Vietoj to, kad bandyti suprasti kaip sistemoje yra keičiami *sudėtingi* signalai, viskas ką mums tereikia žinoti tėra kaip sistemoje yra modifikuojami *paprasti* signalai. Kitaip sakant, įėjimo ir išėjimo signalai yra nagrinėjami kaip paprastų bangų superpozicija (suma). Toks požiūris yra pagrindas beveik visoms signalų apdorojimo metodikoms.

Dekompozicijos tikslas yra sudėtingą problemą pakeisti keliomis paprastomis. Jei dekompozicija kažkokiu būdu nesupaprastina situacijos, tai nieko nelaimima. Dažniausia naudojamos dekompozicijos yra: *impulsų dekompozicija* ir *Furjė dekompozicija*.

Impulsų dekompozicija suskaido *N* imčių signalą į *N* komponentinių signalų, kurių kiekvienas turi *N* imčių. Kiekvienas komponentinis signalas turi vieną imtį iš originalaus signalo, o kitos imtys lygus 0. Šis nelygi nuliui imtis, apsupta nulinių imčių, vadinama *impulsu*. Impulsų dekompozicija yra svarbi tuo, kad leidžia signalą nagrinėti po vieną

imtį vienu metu. Analogiškai, sistemos yra aprašomos, kaip jos atreaguoja į impulsą. Žinant kaip sistema atreaguoja į impulsą, sistemos išėjimas gali būti apskaičiuotas bet kokiam įėjimui. Šis metodas vadinamas sąsūka ir jį aprašysime vėliau.



13 pav. Impulsų dekompozicijos pavyzdys. *N* imčių signalas suskaidomas į *N* komponentinių signalų, kurių kiekvienas turi vieną nelygią nuliui imtį.

Furjė dekompozicija bet kokį *N* imčių signalą suskaido į komponentinius signalus, iš kurių pusė yra sinusoidinės bangos, kita pusė – kosinusoidinės. Žemiausio dažnio kosinusoidė (pavaizduota paveiksle $x_{C0}[n]$), padaro 0 ciklų per *N* imčių. Sekantys komponentiniai signalai, pavadinti $x_{C1}[n]$, $x_{C2}[n]$, $x_{C3}[n]$, padaro atitinkamai 1, 2 ir 3 ciklus, ir taip su visais likusiais (iki *N/2*). Kadangi kiekvienos komponentės dažnis yra fiksuotas, vienintelis dalykas, kuris keičiasi, yra sinusoidžių ir kosinusoidžių *amplitudė*.

Diskreti Furjė transformacija (dekompozicija) atliekama sekančiai:

$$\mathrm{Re}\,X[k] = \frac{2}{N}\sum_{n=0}^{N-1} x[n]\cdot\cos(2\pi kn/N)$$

$$\mathrm{Im}\,X[k] = -\frac{2}{N}\sum_{n=0}^{N-1} x[n]\cdot\sin(2\pi kn/N),$$

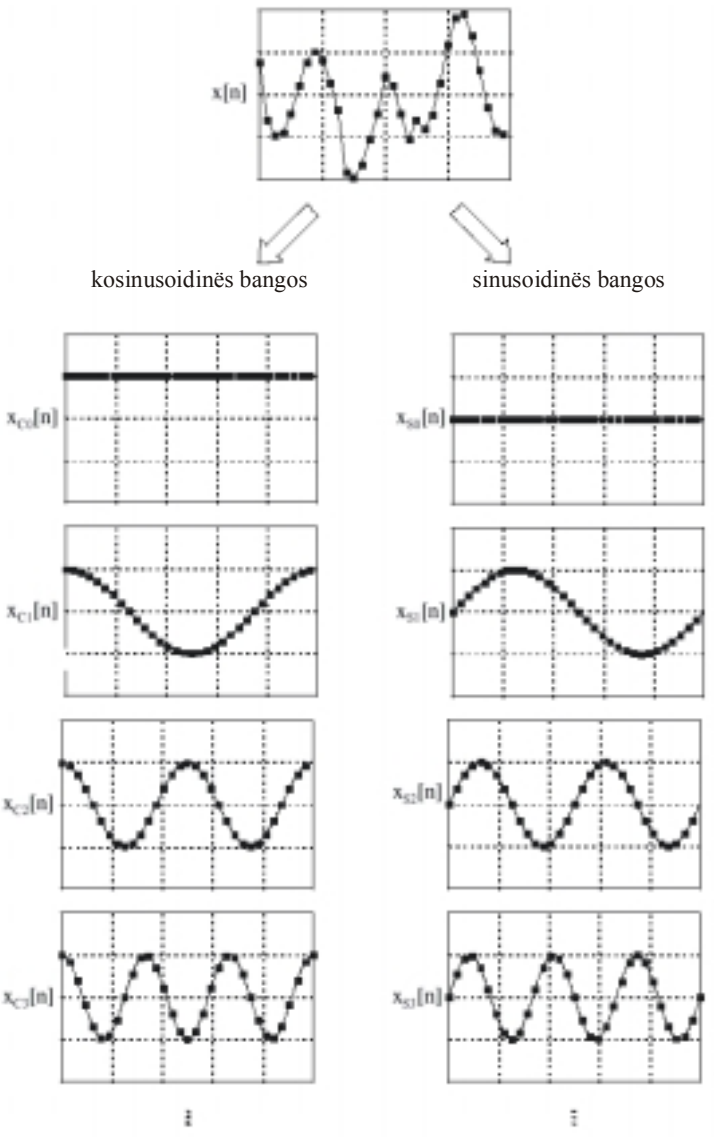
kur k – dažninis indeksas ir kinta nuo 0 iki *N/2*.

Atvirkštinė diskreti Furjė transformacija (sintezė) atliekama sekančiai:

$$x[i] = \sum_{k=0}^{N/2}\mathrm{Re}\,\overline{X}[k]\cos(2\pi ki/N) + \sum_{k=0}^{N/2}\mathrm{Im}\,\overline{X}[k]\sin(2\pi ki/N),$$

kur *x[i]* – sintezuojamas signalas, *i* – indeksas, ir kinta nuo 0 iki *N-1*. $\mathrm{Re}\,\overline{X}$ ir $\mathrm{Im}\,\overline{X}$ yra atitinkamai kosinusoidžių ir sinusoidžių amplitudės, ir k kinta nuo 0 iki *N/2*.

Furjė dekompozicija yra svarbi dėl trijų priežasčių. Pirma, didžioji dauguma signalų (kaip kad pvz. audio signalai) yra sudaryti iš harmoninių signalų sumos. Furjė dekompozicija leidžia tiesiogiai analizuoti informaciją, esančią tokiuose signaluose. Antra, tiesinės sistemos unikaliai reaguoja į sinusoides: sinusoidė įėjime duoda sinusoidę išėjime. Šiuo požiūriu sistemos charakterizuojamos kaip jos keičia sinusoidžių, praeinančių sistemą, amplitudę ir fazę. Kadangi signalas gali būti suskaidytas į sinusoides, tai jei yra žinoma kaip sistema reaguoja į sinusoides, galime rasti sistemos atsaką į įėjimo signalą. Trečia, Furjė dekompozicija yra pagrindas labai plačios ir galingos matematikos srities, vadinamos Furjė analize, bei dar galingesnių Laplaso ir z-transformacijų. Daugelis pažangiausių SSA algoritmų yra pagrįsti šių metodų tam tikrais aspektais.



14 pav. Furjė dekompozicijos pavyzdys. *N* imčių signalas yra suskaidomas į *N+2* komponentinių signalų, kiekvienas kurių turi *N* imčių. Pusė šių signalų yra kosinusoidės, kita pusė – sinusoidės. Sinusoidžių dažniai yra fiksuoti, keistis gali tik jų amplitudės.

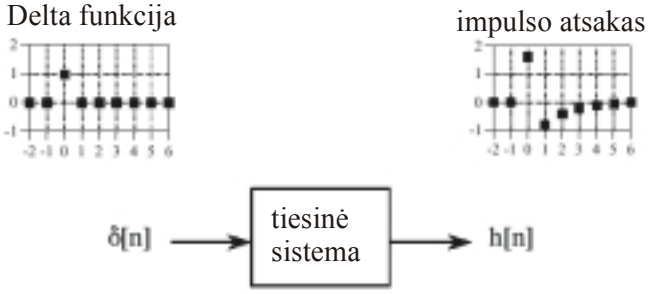
Be impulsų bei Furjė dekompozicijų dar yra naudojamos: šuoliukų dekompozicija[3], lyginė/nelyginė dekompozicija[4], interleisinė dekompozicija[5] bei kitos.

1.5. Sąsūka ir koreliacija

Dabar trumpai išnagrinėsim du pagrindinius SSA metodus: sąsūką ir koreliaciją.

Abu jie pagrįsti ankščiau minėta strategija: (1) išskaidyti signalus į paprastus sudedamuosius komponentus, (2) apdoroti kiekvieną komponentą atskirai, (3) susintezuoti komponentus į galutinį rezultatą.

Sąsūka – tai matematinis metodas iš dviejų signalų suformuojantis trečią. Tai pats svarbiausias SSA metodas. Jame naudojama impulsų dekompozicija. Pagrindiniai terminai, naudojami sąsūkoje, yra delta funkcija ir impulso atsakas.



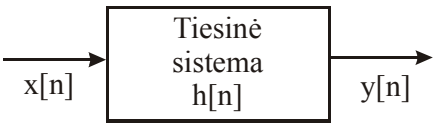
15 pav. *Delta funkcijos* ir *impulso atsako* apibrėžimas. Delta funkcija yra normalizuotas impulsas. Visos vertės lygios nuliui, išskyrus imtį su koordinate 0, kuris turi vertę lygią vienetui. Delta funkcijai pažymėti yra naudojama graikiška raidė delta $\delta[n]$. Tiesinės sistemos *impulso atsakas*, paprastai žymimas *h[n],* yra sistemos atsakas į delta impulsą įėjime.

Sąsūkos metodo požiūriu sistema transformuoja įėjimo signalą į išėjimo signalą sekančiai. Pirmiausia, įėjimo signalas yra nagrinėjamas kaip impulsų seka, kur kiekvienas impulsas yra pakeisto mastelio ir paslinkta delta funkcija. Antra, išėjimas nuo kiekvieno impulso yra pakeisto mastelio ir paslinktas impulso atsakas. Trečia, pilnas sistemos atsakas gali būti rastas sudedant šiuos pakeisto mastelio ir paslinktus impulsų atsakus.

Jei sistema laikoma filtru, tai impulso atsakas vadinamas filtro branduoliu[6], sąsūkos branduoliu[7], ar paprasčiausiai, branduoliu.

---

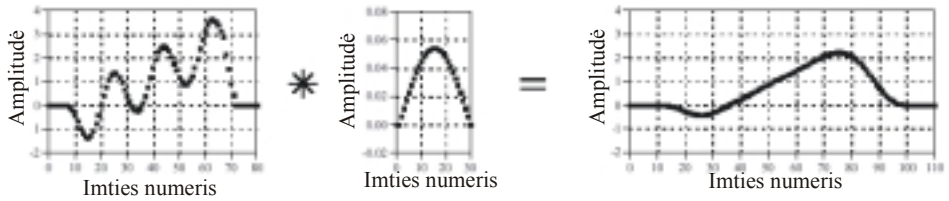
[3] iš angl. step decomposition
[4] iš angl. even/odd decomposition
[5] iš angl. interlaced decomposition
[6] iš angl. filter kernel
[7] iš angl. convolution kernel



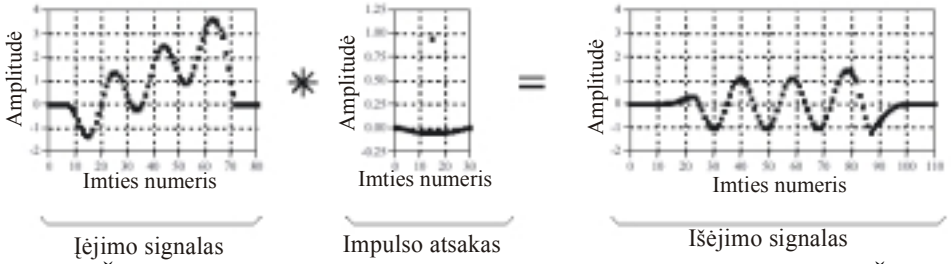
x[n]*h[n]=y[n]

16 pav. Sąsūkos panaudojimas SSA. Tiesinės sistemos išėjimas yra lygus įėjimo signalui susuktam su sistemos impulso atsaku. Sąsūkos operacija yra žymima žvaigždute.

a. Žemadažnis filtras



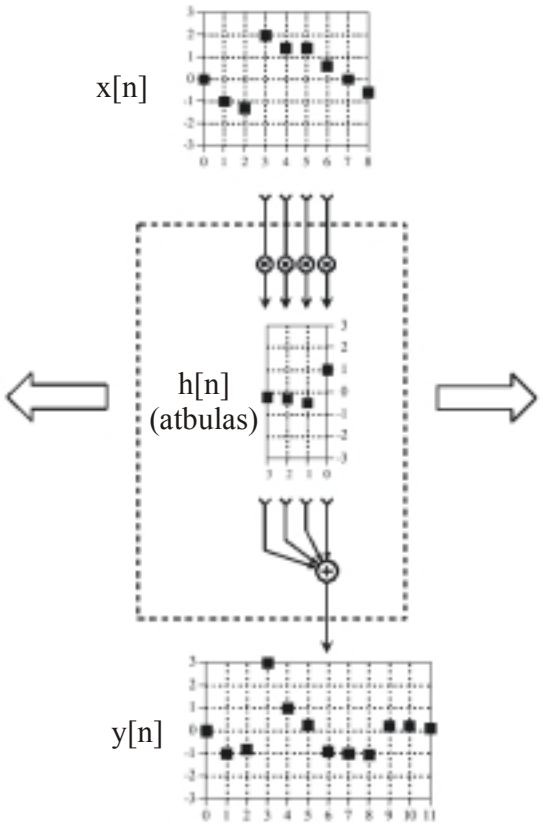
b. Aukštadažnis filtras



17 pav. Žemadažnis ir aukštadažnis filtravimas panaudojant sąsūką. Šiame pavyzdyje įėjimo signalas susideda iš sinusoidės keleto ciklų (aukštas dažnis) plius lėtai kylančio nuolydžio (žemas dažnis). Šie du komponentai atskiriami tinkamai parinkus impulso atsaką.

Sąsūkos algoritmas gali būti pavaizduotas kaip *sąsūkos mašina*. Įėjimo ir išėjimo signalai vaizduojami kaip baigtinio ilgio juostos fiksuotos erdvėj. Sąsūkos mašina gali laisvai judėti į kairę arba dešinę. Pradžioje sąsūkos mašina pastatoma taip, kad rodytų į išėjimo juostos pirmą imtį. Sąsūkos mašina surenka imtis iš įėjimo juostos. Jų surenka tiek, kiek yra impulso atsake imčių. Jei sąsūkos mašinos įėjimai išeina už įėjimo juostos ribų, tai laikoma kad į juos paduodami nuliai [8]. Po to kiekvieną imtį sudaugina iš atitinkamų impulso atsako verčių, gautus rezultatus susumuoja ir įrašo į išėjimą. Po to sąsūkos mašina pastumiama per vieną imtį į dešinę ir procesas kartojamas.

[8] Tai vadinama signalo padėklavimu (iš angl. padding) nuliais. Vietoj to, kad paduoti į įėjimą neegzistuojančias vertes, sąsūkos mašina gauna į įėjimą nulį. Kadangi nuliai yra eliminuojami daugybos metu, tai rezultatas matematiškai yra tas pats, kas neegzistuojančių įėjimų *ignoravimas*.



18 pav. Sąsūkos mašina. Ši diagrama vaizduoja kaip kiekviena išėjimo signalo imtis priklauso nuo įėjimo signalo bei impulso atsako.

Impulso atsako išdėstymas sąsūkos mašinos viduje yra labai svarbus. Impulso atsakas yra apsuktas atbulai iš kairės į dešinę (veidrodinis atspindys). Impulso atsakas aprašo kaip kiekviena išėjimo imtis yra paveikiama įėjimo signalo imčių, *pasvertų* iš apsukto impulso atsako (impulso atsako imtys faktiškai yra *svorio koeficientai,* iš kurių dauginamos įėjimo imtys, o rezultatai susumuojami).

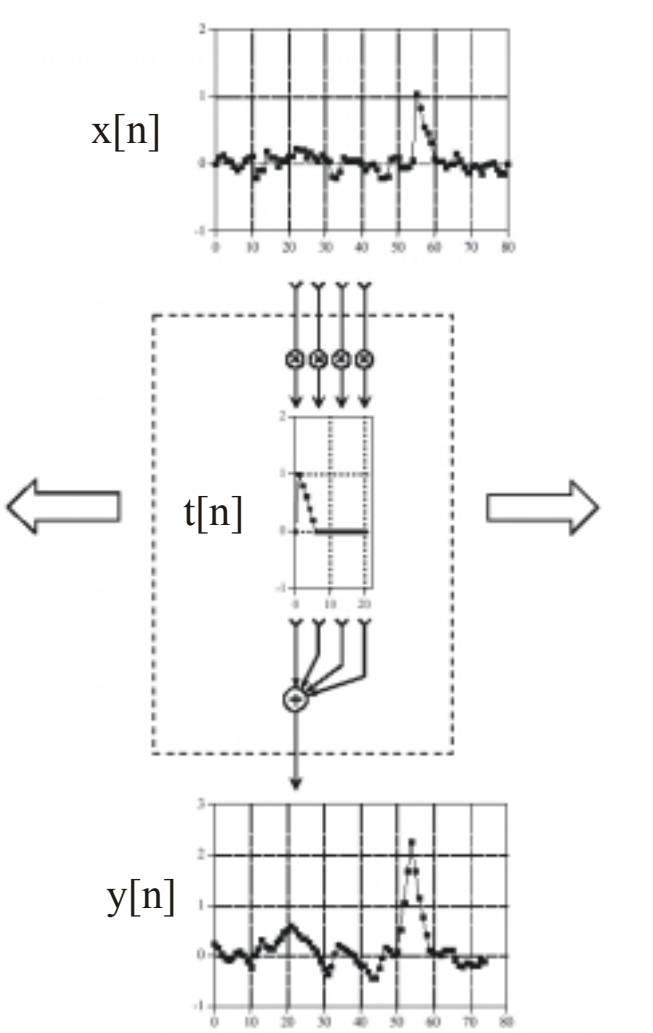
Pasinaudodami sąsūkos mašina užrašysime sąsūkos standartinę lygtį. Jei x[n] yra *N* imčių signalas nuo 0 iki *N*, ir h[n] yra M imčių signalas nuo 0 iki *M-1*, tai jų sąsūka $y[n] = x[n] * h[n]$ bus *N+M-1* imčių signalas nuo 0 iki *N+M-2*, kur kiekviena imtis apskaičiuojama sekančiai:

$$y[n] = \sum_{j=0}^{M-1} h[j]x[i-j].$$

Ši lygtis vadinama sąsūkos suma.

Be sąsūkos mašinos dar yra naudojama koreliacinė mašina. Tarkim turim kažkokį signalą ir norima nustatyti ar šis signalas yra *kitame* signale. Tokio tipo uždaviniams spręsti naudojama koreliacija. Koreliacija yra matematinė operacija, kuri labai panaši į sąsūką. Kaip ir sąsūkoje, iš dviejų signalų suformuojamas trečias. Šis trečiasis signalas

vadinamas dviejų įėjimo signalų tarpusavio koreliacija[9]. Kai signalas koreliuojamas pats su savim, tai gautas išėjime signalas vadinamas *autokoreliacija*.



19 pav. Koreliacinė mašina. Ši diagrama vaizduoja kaip apskaičiuojama dviejų signalų tarpusavio koreliacija. Koreliacinė mašina yra identiška sąsūkos mašinai, tiktais impulso atsakas nėra atbulai apsuktas.

Tarpusavio koreliacijos signalo amplitudė kiekvienoje imtyje rodo panašumą į ieškomą signalą. Tai reiškia, kad maksimali amplitudė bus ten, kur ieškomas signalas yra nagrinėjamame signale.

Tarpusavio koreliacija detektuoja tiktais signalo buvimo faktą, bet neatstato originalaus signalo. Koreliacija yra *optimalus* metodas, kai norima aptikti ieškomą signalą atsitiktiniame triukšme. Išėjimo signalo vertė gaunama panaudojant koreliaciją

[9] iš angl. cross-correlation

yra didesnė, nei kad panaudojant bet kokią kitą tiesinę sistemą (patikslinant, šis metodas optimalus tiktais baltajam triukšmui). Koreliacijos panaudojimas žinomo signalo aptikimui vadinamas atitikimo filtravimu[10].

Tarpusavio *x[n]* ir *y[n]* sekų koreliacija yra seka $r_{xy}[l]$, kuri apibrėžiama kaip:

$$r_{xy}[l] = \sum_{n=i}^{N-|k|-1} x[n]y[n-l] \text{ , } l = 0, \pm1, \pm2, ...$$

arba ekvivalenčiai kaip

$$r_{xy}[l] = \sum_{n=i}^{N-|k|-1} x[n-l]y[n], \; l = 0, \pm1, \pm2, ...,$$

kur $i = l$, $k = 0$, kai $l \geq 0$, ir $i = 0$, $k = l$ kai $l < 0$ [PM96].

Indeksas *l* yra laiko poslinkio (uždelsimo) parametras ir *xy* indeksai tarpusavio koreliacijos sekoje $r_{xy}(l)$ rodo koreliuojamas sekas. Indeksų tvarka, kai *x* yra pirmiau už *y*, rodo kryptį, į kurią viena seka stumiama kitos atžvilgiu. Jei žymima *xy*, tai reiškia, kad *x[n]* seka yra paliekama vietoje, *y[n]* yra pastumiama per *l* imčių į dešinę, jei *l* teigiamas ir *l* imčių į kairę, jei *l* neigiamas. Bet postūmis *x[n]* į kairę yra ekvivalentus *y[n]* postūmiui į dešinę per *l* imčių atžvilgiu *x[n]*. Todėl r_{xy} ir r_{yx} rezultate yra ta pati seka.

Apkeitus *x[n]* ir *y[n]* vietomis, tarpusavio koreliacija užrašoma kaip:

$$r_{yx}[l] = \sum_{n=i}^{N-|k|-1} y[n]x[n-l] \text{ , } l = 0, \pm1, \pm2, ...$$

arba ekvivalenčiai kaip

$$r_{yx}[l] = \sum_{n=i}^{N-|k|-1} y[n+l]x[n], \; l = 0, \pm1, \pm2, ...$$

Sulyginę matome, kad

$$r_{xy}[l] = r_{yx}(-l).$$

Sąsūka ir tarpusavio koreliacija yra susijusios sekančiai:

$$r_{xy}[l] = x[l] * y[-l].$$

Atskiras atvejis yra, kai *y[n]=x[n]*. Tai vadinama *x[n]* autokoreliacija ir apibrėžiama kaip:

$$r_{xx}[l] = \sum_{n=i}^{N-|k|-1} x[n]x[n-l]$$

arba

$$r_{xx}[l] = \sum_{n=i}^{N-|k|-1} x[n+l]x[n].$$

1.6. Autokoreliacijos ir tarpusavio koreliacijos savybės

[10] iš angl. matched filtering

Autokoreliacijos ir tarpusavio koreliacijos sekos turi keletą svarbių savybių. Tarkime turime dvi sekas *x[n]* ir *y[n]* su baigtine energija, iš kurių suformuojame tiesinę kombinaciją:

$$ax[n]+by[n-l]$$

kur *a* ir *b* yra konstantos ir *l* – poslinkis laike. Šio signalo energija yra:

$$\sum_{n=-\infty}^{\infty}[ax[n]+by[n-l]]^2 = a^2\sum_{n=-\infty}^{\infty}x^2[n]+b^2\sum_{n=-\infty}^{\infty}y^2[n-l]+2ab\sum_{n=-\infty}^{\infty}x[n]y[n-l]=$$

$$= a^2 r_{xx}[0]+b^2 r_{yy}[0]+2abr_{xy}[l]$$

Pirma, pastebime, kad $r_{xx}[0]=E_x$ ir $r_{yy}[0]=E_y$, kur *E* yra atitinkamos *x[n]* ir *y[n]* energijos.

Akivaizdu, kad

$$a^2 r_{xx}[0]+b^2 r_{yy}[0]+2abr_{xy}[l]\geq 0\,.$$

Tarkime, kad $b\neq 0$ ir padalinkime iš b^2:

$$r_{xx}[0]\left(\frac{a}{b}\right)^2+2r_{xy}[l]\left(\frac{a}{b}\right)+r_{yy}[0]\geq 0\,.$$

Nagrinėsime šią nelygybę kaip kvadratinę su koeficientais $r_{xx}[0]$, $2r_{xy}[1]$ir $r_{yy}[0]$. Kadangi kvadratinė lygtis yra neneigiama, vadinasi diskriminantas:

$$4\left[r_{xy}[l]^2-r_{xx}[0]r_{yy}[0]\right]\leq 0\,.$$

Todėl, tarpusavio koreliacija tenkina sąlyga:

$$\left|r_{xy}[l]\right|\leq\sqrt{r_{xx}[0]r_{yy}[0]}=\sqrt{E_xE_y}$$

Atskiru atveju kai *y[n] =x[n]* :

$$\left|r_{xx}[l]\right|\leq r_{xx}[0]=E_x\,.$$

Tai reiškia kad autokoreliacija turi maksimalią vertę ties nuliniu uždelsimu. Tai yra pasekmė to, kad signalas atitinka pats save su nuliniu poslinkiu.

Pastebėsime, kad jei vieno ar abiejų signalų, dalyvaujančių tarpusavio koreliacijoje, keičiamas mastelis, autokoreliacijos sekos forma nesikeičia, tik atitinkamai keičiasi tarpusavio koreliacijos sekos mastelis. Kadangi mastelio keitimas nėra svarbus, tai praktikoje dažnai pageidautina sunormalizuoti autokoreliacines ir tarpusavio koreliacijos sekas intervale nuo –1 iki 1. Autokoreliacijos atveju, paprasčiausiai padaliname iš $r_{xx}[0]$. Tokiu būdu normalizuota autokoreliacinė seka yra apibrėžiama kaip:

$$\rho_{xx}[l]=\frac{r_{xx}[l]}{r_{xx}[0]}\,.$$

Atitinkamai, normalizuota tarpusavio koreliacija:

$$\rho_{xy}[l]=\frac{r_{xy}[l]}{\sqrt{r_{xx}[0]r_{yy}[0]}}\,.$$

Tokiu būdu $\left|\rho_{xx}[l]\right|\leq 1$ ir $\left|\rho_{xy}[l]\right|\leq 1$, t.y. šios sekos yra nepriklausomos nuo signalo mastelio keitimo.

Kaip jau buvo parodyta anksčiau, tarpusavio koreliacija tenkina savybę:

$$r_{xy}[l]=r_{yx}[-l]\,.$$

Kai $y[n] = x[n]$, tai gaunama labai svarbi autokoreliacinės sekos savybė:
$r_{xx}[l] = r_{xx}[-l]$,
t.y. autokoreliacija yra lyginė funkcija. Todėl užtenka apskaičiuoti $r_{xx}[l]$ kai $l > 0$ [PM96].

1.7. Išvados

SSA metodų esmė yra "skaldyk ir valdyk" strategija, kuri vadinama *superpozicija* – nagrinėjamas signalas yra suskaidomas (dekompozicija) į atskirus paprastus komponentus, kiekvienas komponentas individualiai apdorojamas ir gauti rezultatai apjungiami (sintezė). Šio metodo pranašumas tame, kad sudėtinga problema suskaidoma į keletą paprastų. Superpozicija gali būti taikoma tik *tiesinėms sistemoms*. Yra daug dekompozicijos būdų, tačiau dažniausiai naudojamos yra: impulsų dekompozicija ir Furjė dekompozicija. Sąsūka ir koreliacija – tai patys svarbiausi SSA metodai. Jie iš dviejų signalų suformuoja trečią ir savyje realizuoja superpozicijos koncepciją. Kai signalas koreliuojamas pats su savim, tai gautas išėjime signalas vadinamas autokoreliacija. Autokoreliacija yra lyginė funkcija, t.y autokoreliacinė seka yra simetriška.

2. Dirbtiniai neuroniniai tinklai ir jų mokymas

1 skyriuje nagrinėjome SSA teoriją, kuri aprašo tiesines sistemas. Tačiau daugelis gamtoje sutinkamų reiškinių yra netiesiniai. Netiesinėms sistemoms aprašyti tinkamas yra neuroninių tinklų modelis. Šiame skyriuje nagrinėjami neuroninių tinklų teorijos pagrindai.

2.1. poskyryje pristatomas neuroninio tinklo apibrėžimas. 2.2. poskyryje nagrinėjamas neuroninio tinklo elementaraus informacijos apdorojimo vieneto – neurono, matematinis modelis. Neuroniniame tinkle neuronai gali būti išdėstyti vienu ar daugeliu sluoksniais. Vienasluoksniai tinklai aprašomi 2.3. poskyryje, o daugiasluoksniai tinklai 2.4. poskyryje. 2.5. poskyryje pristatoma neuroninio tinklo forma – perceptronas. 2.6. poskyryje nagrinėjamas neuroninių tinklų mokymas ir iškylančios problemos, nagrinėjami dažniausiai praktikoje naudojami neuroninių tinklų mokymo atbulo sklidimo bei MVK (mažiausio vidutinio kvadratinio (nuokrypio)) algoritmai. 2.7. poskyryje aptariami neuroninio tinklo architektūros organizavimo klausimai. 2.8. poskyryje pateikiamos išvados.

2.1. Kas tai yra dirbtinis neuroninis tinklas

Dirbtinis neuroninis tinklas – masiškai paralelinis paskirstytas procesorius, sudarytas iš paprastų informacijos apdorojimo vienetų – neuronų. Dirbtinis neuroninis tinklas geba kaupti empirines žinias ir jomis naudotis. Jis panašus į smegenis dviem aspektais:

1. Tinklas gauna žinias iš aplinkos apmokinimo proceso metu.
2. Informacija yra saugoma tarpneuroninių jungčių stiprumų (sinapsinių svorių) pavidalu .

Dirbtinio neuroninio tinklo mokymo procedūra vadinama *mokymo algoritmu*. Tai funkcija, kuri modifikuoja tinklo sinapsinius svorius taip, kad tinklas galėtų pasiekti norimą tikslą.

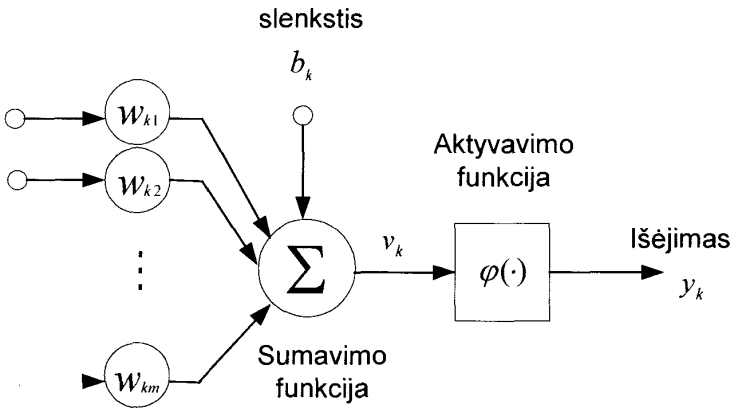
2.2. Neurono modelis

Neuronas yra informacijos apdorojimo vienetas, kuris yra neuroninio tinklo veikimo pagrindas. 20 pav. vaizduoja neurono modelį, kuris yra naudojamas kuriant dirbtinį neuroninį tinklą. Neurono modelyje galima išskirti tris pagrindinius elementus:

Sinapsių aibė arba *ryšiai* yra apibūdinami ryšių *svoriais* arba *stiprumais.* Sinapsės *j*, sujungtos su neuronu *k*, sinapsinis svoris w_{kj} yra dauginamas iš įėjimo signalo x_j, t.y sinapsinio svorio žymėjimo pirmas indeksas nusako neuroną, o antras įėjimo numerį.

Sumatorius sumuoja įėjimo signalus, padaugintus iš atitinkamų sinapsinių svorių;

Aktyvacijos funkcija apriboja neurono išėjimo amplitudę. Dažniausiai neurono išėjimo sritis yra vienetinio ilgio intervalas [0, 1] arba [-1, 1].



20 pav. Neurono modelis

Neurono modelis taip pat turi slenkstį, kuris pažymėtas b_k. Slenkstis b_k priklausomai nuo ženklo didina arba mažina įėjimą aktyvacijos funkcijai.

Neurono modelį nusako ši lygybių pora :

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

ir

$$y_k = \varphi(u_k + b_k),$$

kur $x_1, x_2, ..., x_m$ – įėjimo signalai; $w_{k1}, w_{k2}, ..., w_{km}$ – sinapsiniai neurono k svoriai; u_k – įėjimo signalų tiesinė kombinacija; b_k yra slenkstis; $\varphi(\cdot)$ – aktyvacijos funkcija, o y_k – neurono išėjimo signalas. Slenkstis b_k atlieka tiesinio sumatoriaus u_k afininę transformaciją.

$$v_k = u_k + b_k$$

kur v_k yra k neurono *aktyvacijos potencialas* arba neurono *indukuotas lokalus laukas.* Aktyvacijos funkcija, žymima simboliu $\varphi(v)$, apibrėžia neurono išėjimo priklausomybę nuo v reikšmės.

Pagrindiniai aktyvacijos funkcijų tipai :

1. Binarinė unipoliarinė funkcija (slenksčio funkcija)

$$\varphi(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases}$$

Pritaikius šią funkciją neuronui k gausime išraišką:

$$y_k = \begin{cases} 1, & v_k \geq 0 \\ 0, & v_k < 0 \end{cases},$$

kur v_k yra k neurono *aktyvacijos potencialas* arba neurono *indukuotas lokalus laukas.*

2. Rampos funkcija
Dalinai tiesinę funkciją apibrėžia formulė :

$$\varphi(v)=\begin{cases}1, & v\geq+\frac{1}{2}\\ v, & +\frac{1}{2}>v>-\frac{1}{2}\\ 0, & v\leq-\frac{1}{2}\end{cases}$$

3. Sigmoidinė unipoliarinė funkcija
Sigmoidinę funkciją apibrėžia formulė:

$$\varphi(v)=\frac{1}{1+\exp(-av)},$$

kur *a* yra sigmoidinės unipoliarinės funkcijos *šlaito parametras.* Riboje didinant parametrą *a*, funkcija tampa paprasčiausia slenksčio funkcija.

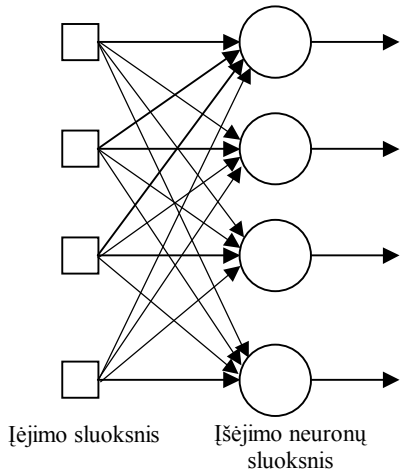
Slenksčio funkcijos reikšmių sritis yra dvi reikšmes 0 arba 1, o sigmoidinės unipoliarinės funkcijos reikšmių sritis yra intervalas nuo 0 iki 1. Be to sigmodinė unipoliarinė funkcija yra diferencijuojama, o binarinė unipoliarinė – nediferencijuojama.

4. Hiperbolinio tangento (sigmoidinė bipoliarinė) funkcija
Ji apibrėžiama:

$$\varphi(v)=\tanh(v)$$

2.3. Vienasluoksniai tinklai

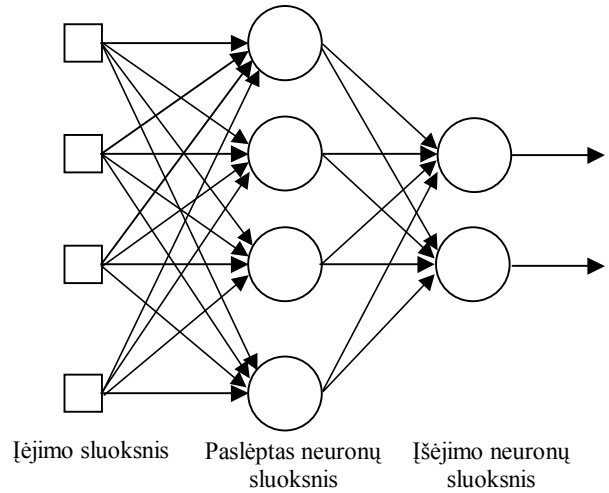
Sluoksniuotame neuroniniame tinkle neuronai yra išdėstyti sluoksniais. Paprasčiausiu atveju įėjimo sluoksnis projektuojamas į išėjimo neuronų sluoksnį (skaičiuojantys mazgai), bet ne atvirkščiai. Kitaip sakant, tinklas yra *acikliškas*. 4 paveiksle pavaizduotas tinklas turi po keturis mazgus įėjimo sluoksnyje ir išėjimo neuronų sluoksnyje. Toks tinklas vadinamas *vieno sluoksnio tinklu*, turint omeny vieną skaičiuojančių mazgų (neuronų) sluoksnį. Įėjimo sluoksnis neskaičiuojamas, kadangi jame neatliekami skaičiavimai.



21 pav. Vieno sluoksnio tinklas

2.4. Daugiasluoksniai tinklai

Kita neuroninių tinklų klasė skiriasi tuo, kad turi vieną ar daugiau *paslėptų sluoksnių*, kuriuose skaičiuojantys mazgai yra vadinami *paslėptais neuronais*. Paslėpti neuronai yra tarpininkai tarp įėjimo mazgų ir išėjimo neuronų sluoksnio.



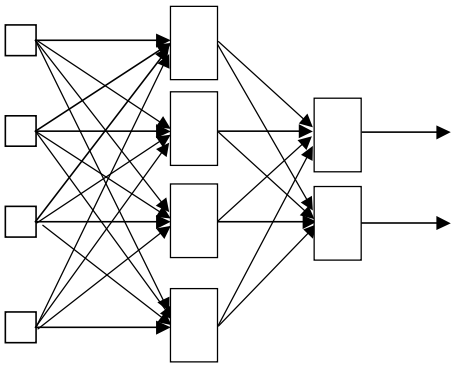
22 pav. Daugiasluoksnis tinklas turintis vieną paslėptą ir vieną išėjimo sluoksnį

2.5. Perceptronas

Perceptronas yra neuroninio tinklo forma. Paprasčiausiu atveju perceptronas yra vienas neuronas (vienasluoksnis tinklas su vienu išėjimu), tačiau jis gali būti sudarytas iš *n* neuronų (vienasluoksnis tinklas su *n* išėjimų).

Toks perceptronas vadinamas vienasluoksniu perceptronu .

Daugiasluoksnis perceptronas susideda iš kelių vienasluoksnių perceptronų sluoksnių.



23 pav. Daugiasluoksnis perceptronas

2.6. Neuroninių tinklų mokymas

Svorius, reikalingus, kad NT galėtų atlikti konkrečią užduotį, suranda *mokymo algoritmas*, kartu su *pavyzdžiais* kaip sistema *turėtų* veikti. Pavyzdžiui, sonaro uždavinio pavyzdžiais būtų duomenų bazė iš kelių šimtų (ar daugiau) 1000 imčių ilgio pavyzdžių. Kai kurie pavyzdžiai atitiktų povandeninius laivus, kiti banginius, treti atsitiktinį triukšmą ir t.t. Mokymo algoritmas naudoja šiuos pavyzdžius svorių rinkinio konkrečiai užduočiai suradimui. Terminas *mokymas* yra labai plačiai paplitęs NT literatūroje aprašant šį procesą; tačiau tikslesnis apibūdinimas galėtų būti: optimizuoto svorių rinkinio nustatymas pasiremiant pavyzdžių statistika.

Yra sukurta daug mokymo algoritmų, tačiau populiariausias praktikoje naudojamas algoritmas yra atbulo sklidimo algoritmas. Kad geriau suprasti kas yra mokymas, išnagrinėkime smulkiau atbulo sklidimo algoritmą.

Atbulo sklidimo algoritmas yra gradientinis optimizavimo metodas, pritaikytas minimizuoti nuostolių funkciją, kuri tiesiogiai susijusi su perceptrono daromų klaidų skaičiumi. Klaidos signalas išėjimo neurono *j* iteracijoje *n* (pateikiant *n*-tąjį mokymo pavyzdį) yra apibrėžiamas taip:

$$e_j = d_j(n) - y_j(n),$$

kur $d_j(n)$ yra trokštamas rezultatas, o $y_j(n)$ – neurono išėjimas.

Klaidai įvertinti imamas klaidos signalo kvadratas. Kodėl imami nuokrypių kvadratai, o ne paprasti nuokrypiai? Tai išlaukia iš signalų fizikinių dėsnių. Pavyzdžiui, kai elektrinėje grandinėje atsitiktinio triukšmo signalai susisumuoja, galutinis triukšmas yra lygus individualių signalų *energijų* sumai, o ne *amplitudžių* sumai, o energija proporcinga signalo amplitudės kvadratui:

$E = \frac{1}{2}kA^2$ (energijos priklausomybė nuo nuokrypio amplitudės, kur E – energija, k – proporcingumo koeficientas, A – nuokrypio amplitudė).

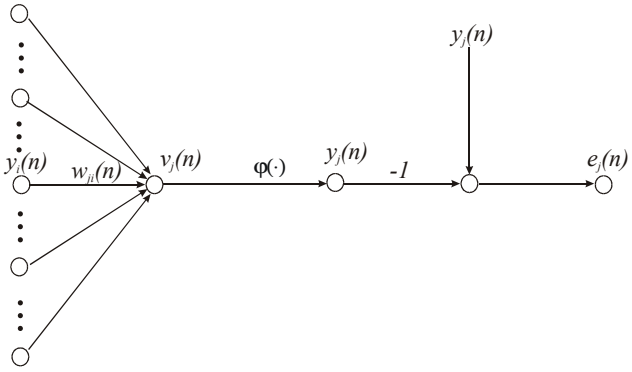
Neurono *j* klaidos energijos reikšmė apibrėžiama kaip $\frac{1}{2}e_j^2(n)$. Atitinkama momentali klaidos energija *Er(n)* apibrėžiama, kaip suma $\frac{1}{2}e_j^2(n)$ visiems išėjimo sluoksnio neuronams:

$$Er(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n)$$

kur *C* yra aibė sudaryta iš viso tinklo išėjimo neuronų numerių. Tarkime, kad *N* yra mokymo pavyzdžių (mokymo duomenų aibėje) skaičius. Vidutinę klaidos kvadrato energiją apibrėžiame taip *Er(n)* sumą visų *n* ir normalizuotą iš aibės dydžio *N*:

$$Er_{av}(n) = \frac{1}{N}\sum_{n=1}^{N} e_j^2(n).$$

Tiek momentali klaidos energija, tiek vidutinė klaidos kvadrato energija yra funkcija nuo laisvų tinklo parametrų (tinklo neuronų sinapsiniai svoriai ir slenksčiai). Jei yra pateikta mokymo aibė, tai Er_{av} nusako nuostolių funkciją, kuri nusako mokymosi efektyvumą. Mokymosi proceso tikslas yra surasti laisvų tinklo parametrų reikšmes, kurios minimizuotų Er_{av}. Nagrinėsime paprastą mokymo metodą, kuris keičia tinklo svorius pateikiant vis naują vienos epochos mokymo pavyzdį. Svorių pakeitimai yra atliekami priklausomai nuo klaidų apskaičiuotų kiekvienam tinklui pateiktam mokymo pavyzdžiui. Aritmetinis vidurkis šių individualių svorių pokyčių yra lygus įverčiui tikro pokyčio, kuris būtų gautas keičiant svorius minimizuojant nuostolių funkciją Er_{av} visai mokymo aibei.



24 pav. *j*-tojo neurono signalų srautų diagrama

Atbulo sklidimo algoritmas keičia sinapsinius svorius w_{ij} pokyčiu Δw_{ij} , kuris yra proporcingas dalinei išvestinei $\frac{\partial Er(n)}{\partial w_{ij}(n)}$. Atlikus skaičiavimus gauname:

$$\frac{\partial Er(n)}{\partial w_{ij}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n),$$

o w_{ij} pokytis išėjimo sluoksnio neuronams pagal *delta* taisyklę yra lygus:

$$\Delta w_{ij}(n) = -\eta\frac{\partial Er(n)}{\partial w_{ij}(n)},$$

kur η yra atbulo sklidimo algoritmo mokymo greičio parametras. Ši taisyklė keičia svorius priešinga gradientui kryptimi (taip yra mažinama nuostolių funkcija *Er*). Analogiškai galima suskaičiuoti pokyčio formules paslėpto sluoksnio neuronams, kurių lokalus gradientas lygus:

$$\partial_j = -\frac{\partial Er(n)}{\partial y_j(n)} \varphi_j'(v_j(n)).$$

Šis algoritmas iliustruoja, kokiais principais yra mokomas daugiasluoksnis perceptronas. Nepaisant visų teigiamų šio algoritmo savybių, jo taikymas yra komplikuotas dėl sekančių neigiamų savybių. Taikant dideles mokymo greičio reikšmes gaunamas nestabilus mokymo procesas, o taikant mažas – gaunamas lėtas mokymo algoritmo konvergavimas. Jei klaidos paviršius turi sričių, kuriose maži svorių pokyčiai iššaukia staigų klaidos padidėjimą, tai yra grėsmė, kad algoritmas neras globalaus minimumo, o baigs darbą radęs tik lokalų minimumą. [Hay98]

Neuroninių tinklų su tiesine aktyvacijos funkcija mokymui naudojamas MVK (mažiausio vidutinio kvadratinio (nuokrypio)) algoritmas.

MVK algoritmas (dar vadinamas Widrow-Hoff mokymo algoritmu) yra pagrįstas nuostolių funkcijos momentinėm vertėm:

$$\varepsilon(w) = \frac{1}{2} e^2(n),$$

kur *e(n)* yra klaidos signalas (nuokrypis) *n* laiko momentu. Diferencijuojant *ε(w)* pagal svorių vektorių *w* duoda:

$$\frac{\partial \varepsilon(w)}{\partial w} = e(n) \frac{\partial e(n)}{\partial w}.$$

Klaidos signalas (nuokrypis) išreiškiamas kaip:

$$e(n) = d(n) - x^T(n) w(n)$$

Tuo būdu:

$$\frac{\partial e(n)}{\partial w(n)} = -x(n)$$

ir

$$\frac{\partial \varepsilon(n)}{\partial w(n)} = -x(n) e(n)$$

Panaudojus paskutinę formulę kaip gradiento vektoriaus įvertį:

$$\hat{g}(n) = -x(n) e(n)$$

Įstatę šią formulę į greičiausio nusileidimo algoritmo formulę $w(n+1) = w(n) - \eta g(n)$, gauname

$$\hat{w}(n+1) = \hat{w}(n) - \eta x(n) e(n),$$

kur η yra teigiama konstanta ir vadinama mokymo žingsniu.

Paskutinėje formulėje naudojome $\hat{w}(n)$ vietoj $w(n)$, norėdami pabrėžti faktą, kad MVK algoritmas duodą *įvertį* svorių vektoriaus, kuris būtų gautas naudojant greičiausio nusileidimo metodą. To pasėkoje MVK algoritmas praranda greičiausio nusileidimo išskirtinę savybę. Greičiausio nusileidimo algoritme, svorių vektorius $w(n)$ juda griežtai apibrėžta trajektorija svorių erdvėje. Kaip kontrastas, MVK algoritme svorių vektorius

$w(n)$ juda atsitiktine trajektorija. Todėl MVK algoritmas kartais vadinamas "stochastinio gradiento algoritmu". Kai MVK algoritme iteracijų skaičius artėja į begalybę, $\hat{w}(n)$ juda atsitiktine trajektorija (Brauno judėjimas) aplink Wiener'io[11] sprendinį w_0. MVK algoritmas modifikuoja svorius taip, kad minimizuotų vidutinį kvadratinį nuokrypį. Vidutinio kvadratinio nuokrypio funkcija yra kvadratinė funkcija. Todėl, ji turi arba vieną globalų minimumą, silpną minimumą arba jokio minimumo, priklausomai nuo mokymo duomenų vektorių.

Tiesinis mažiausių kvadratų filtras asimptotiškai artėja link Wiener'io filtro, kai mokymo duomenų vektorių skaičius auga į begalybę. [Hay98]

Pabaigai keletas pastabų. Priversti NT konverguoti apmokinant gali būti labai sunku. Jei tinklo klaida stabiliai nemažėja, programa turi būti nutraukta, pakeista ar perstartuota. Gali prireikti keleto bandymų sėkmei pasiekti. Norint paveikti konvergavimą gali būti pakoreguoti trys dalykai: (1) iteracijos žingsnio dydis, (2) pradiniai svoriai, (3) paslėptų neuronų skaičius.

Pats kritiškiausias neuroninio tinklo sudarymo momentas yra apmokančiųjų pavyzdžių *pagrįstumas*. Pavyzdžiui, kai kuriami nauji komerciniai produktai, vieninteliai prieinami testavimo duomenys yra iš prototipų, simuliacijų, empirinių spėjimų ir t.t. Jei NT yra apmokinamas remiantis tiktais šia preliminaria informacija, NT gali neveikti galutinėje aplikacijoje.

2.7. Neuroninio tinklo architektūra

Neuroninis tinklas gali būti traktuojamas kaip metodas sužymintis regionus *parametrų erdvėje*[12]. Tarkim turim sonaro sistemos neuroninį tinklą su 1000 įėjimų ir vienu išėjimu. Tinkamai parinkus svorius, išėjimas bus netoli vieneto, jei detektuotas aidas nuo povandeninio laivo ir išėjimas bus netoli nulio, jei įėjime tiktais triukšmas. Neuroninis tinklas yra metodas, priskiriantis vertes hipererdvės taškams. T.y. 1000 įėjimo verčių nusako *vietą* (adresą) hipererdvėje, o neuroninio tinklo išėjimas nusako tos vietos *vertę*. Šią užduotį idealiai galėtų atlikti dešifravimo lentelė[13] (joje saugoma išėjimo vertė bet kokiam įėjimo adresui), tačiau neuroninis tinklas pranašesnis tuo, kad jis apskaičiuoja vertes kiekvienam adresui, vietoj to, kad saugoti kiekvieną vertę. Faktiškai, neuroninio tinklo architektūra dažnai yra vertinama pagal tai, kaip sėkmingai jis atskiria hipererdvę esant tam tikram svorių skaičiui. Šis požiūris taip pat nusako, kiek reikia turėti neuronų paslėptame sluoksnyje. *N* matavimų parametrinei erdvei reikia *2N* skaičių norint nusakyti sritį (t.y. minimali ir maksimali vertė ant kiekvienos ašies, nusakanti hipererdvės kūno ribas). Pavyzdžiui, šie paprasti įvertinimai rodo, kad neuroniniam tinklui su 1000 įėjimų reikia 2000 svorių, kad atskirti vieną hipererdvės sritį nuo kitos. Pilnai sujungtame tinkle tai reikalautų 2 paslėptų neuronų. Sričių skaičius priklauso nuo uždavinio formulavimo, bet kaip taisyklė galima tikėtis, kad jis žymiai mažesnis, nei parametrų erdvės matavimų skaičius. Nors tai tik grubūs įvertinimai, tai paaiškina, kodėl daugelis neuroninių tinklų

[11] Kas tai yra Wiener'io filtras aprašyta 4.7. poskyryje ("Optimalūs filtrai")
[12] iš angl. parameter space
[13] iš angl. look-up table

gali veikti, jei paslėpto sluoksnio dydis lygus maždaug nuo 2% iki 30% įėjimo sluoksnio dydžio.

2.8. Išvados

Dirbtinis neuroninis tinklas yra sudarytas iš elementarių informacijos apdorojimo vienetų – neuronų. Neuronas yra supaprastintas biologinio neurono matematinis modelis. Dirbtinis neuroninis tinklas informaciją saugo tarpneuroninių jungčių stiprumų (sinapsinių svorių) pavidalu. Dirbtinio neuroninio tinklo mokymo procedūra vadinama mokymo algoritmu. Tai funkcija, kuri modifikuoja tinklo sinapsinius svorius taip, kad tinklas galėtų pasiekti norimą tikslą. Dažniausiai naudojamas praktikoje NT mokymo atbulo sklidimo algoritmas yra gradientinis optimizavimo metodas, pritaikytas minimizuoti nuostolių funkciją, kuri tiesiogiai susijusi su perceptrono daromų klaidų skaičiumi. Priversti NT konverguoti apmokinant gali būti labai sunku. Norint paveikti konvergavimą gali būti pakoreguoti trys dalykai: (1) iteracijos žingsnio dydis, (2) pradiniai svoriai, (3) paslėptų neuronų skaičius.

3. Neuroninių tinklų ir skaitmeninio signalų apdorojimo integracija

Pirmuose dviejuose skyriuose išnagrinėti SSA ir neuroninių tinklų teorijų pagrindai. Iš pirmo žvilgsnio gali pasirodyti, kad SSA ir NT yra pakankamai skirtingos mokslo šakos ir jas mažai kas sieja. Tačiau taip nėra, tai labai glaudžiai persipynę dalykai. Šiame skyriuje išvedamos paralelės tarp skaitmeninių signalų apdorojimo metodų ir neuroninių tinklų, nagrinėjama kaip neuroniniais tinklais galima realizuoti pagrindines SSA operacijas.

3.1. poskyryje lyginamos praktinių uždavinių sprendimo, taikant SSA ir NT metodus, ideologijos. Kaip jau minėta anksčiau, SSA teorija aprašo tiktais tiesines sistemas, o NT teorija – netiesines. 3.2. poskyryje analizuojama kada ir kaip netiesinė sistema gali būti aproksimuota į tiesinę. 3.3. poskyryje parodoma, kad neuroniniu tinklu įmanoma realizuoti pagrindines SSA operacijas – koreliaciją, sąsūką ir Furjė analizę. 3.4. poskyryje pateiktas kompiuterinis eksperimentas, kuris parodo, kad NT gali atlikti diskrečią Furjė transformaciją. 3.5. poskyryje nagrinėjama kuo neuroniniai tinklai yra pranašesni už skaitmeninių signalų procesorius. Parodoma, kad NT pagalba galima realizuoti *ultraspartų* skaitmeninių signalų apdorojimą. 3.6. poskyryje nagrinėjama kaip galima būtų apjungti NT ir SSA teorijų geriausias savybes ir jas panaudoti praktinių uždavinių sprendimui. 3.7. poskyryje pateikiamos išvados.

3.1. Algoritmai ir parametrai

Tradiciniai skaitmeniniai signalų apdorojimo metodai yra pagrįsti *algoritmais*, kurie žingsnis po žingsnio transformuoja duomenis iš vienos formos į kitą . Tam kad šie metodai veiktų, daugeliui metodų reikalingi *parametrai*. Pavyzdžiui: skaitmeniniuose filtruose naudojami dažniniai *koeficientai*; klasifikatorių (savybių) radimui gali būti panaudoti *slenksčiai* ir koreliacijos; nuo *šviesumo* ir *kontrasto* verčių priklauso paveikslo atvaizdas ir t.t. Algoritmai aprašo kas turi būti padaryta, tuo tarpu parametrai naudojami kaip pagrindas lyginant gautus išėjime duomenis.

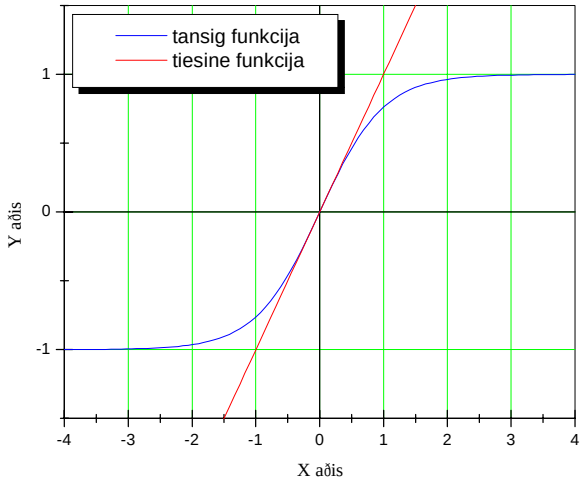
Geras parametrų parinkimas dažnai yra net svarbesnis nei pats algoritmas.

Neuroninių tinklų modelis šią idėją panaudoja maksimaliai – neuroniniai tinklai naudoja santykinai paprastus algoritmus, tačiau su labai daug optimizuotų parametrų. Ir tai yra revoliucinis atotrūkis nuo tradicinių mokslo ir inžinerijos šakų: matematinės logikos ir teorizavimo ir iš jo sekusio eksperimentavimo. Šias tradicines uždavinių sprendimo strategijas neuroniniai tinklai pakeičia "bandymų ir klaidų" pragmatiškais sprendimais ir "šitas veikia geriau nei anas" metodologija.

3.2. Superpozicija ir netiesinės sistemos

Kaip jau minėta anksčiau, SSA pagrindas – superpozicija, o ji gali būti taikoma tik *tiesinėms sistemoms*. Ar tai reiškia, kad neuroniniams tinklams su netiesinėm aktyvacijos funkcijomis negalima taikyti SSA metodų? Galima, tačiau vienintelė strategija nagrinėjant netiesines sistemas yra padaryti jas panašias į tiesines. Yra trys pagrindiniai būdai šiam tikslui pasiekti:

1. Ignoruoti netiesiškumą. Jei netiesiškumas nedidelis, sistema apytiksliai gali būti laikoma tiesine. Klaidos, kylančios dėl šios prielaidos yra laikomos kaip triukšmas ir ignoruojamos;
2. Įėjimo signalai padaromi mažos amplitudės. Daugelis netiesinių sistemų elgiasi kaip tiesinės, jei signalai keičiasi mažose ribose. Pavyzdžiui tranzistoriai yra labai netiesiški savo pilname darbiniame diapazone, tačiau stiprina labai tiesiškai, kai įėjimo signalai keičiami kelių milivoltų ribose;
3. Galima pritaikyti tiesinančią[14] transformaciją. Pavyzdžiui, du signalai sudauginami ir suformuojamas trečias: a[n] = b[n] × c[n] . Signalų išlogaritmavimas pakeičia signalų netiesinę daugybą į tiesinę sudėtį log(a[n])= log(b[n]) + log(c[n]). Šis metodas vadinamas homomorfiniu signalų apdorojimu[15].



25 pav. Tam tikrame reikšmių intervale (maždaug -0.5…0.5) neuroninio tinklo aktyvacijos netiesinė *tansig* funkcija ($y = 2\frac{1}{1-e^{x}} - 1$) gali būti aproksimuota *tiesine* funkcija ($y = x$).

[14] iš angl. linearizing
[15] iš angl. homomorphic signal processing

3.3. Koreliacija, sąsūka ir Furjė analizė neuroniniuose tinkluose

Į neuroninį tinklą galima žiūrėti iš SSA koreliacijos pozicijos. Kaip minėta anksčiau, koreliacija yra optimalus būdas detektuoti ieškomą signalą kitame signale. Kuo didesnis koreliacinės mašinos išėjimo signalas, tuo didesnė tikimybė, kad ieškomas signalas aptiktas. Neuroninis tinklas lygiai taip pat koreliuoja įėjimo duomenis su neuronų svoriais. Jei ieškomas šablonas aptiktas, paduodama į sigmoidę suma bus didelė, kitu atveju – maža.

Tradiciniai SSA algoritmai pagristi dviem metodais: sąsūka ir Furjė analize. Neuroninis tinklas gali atlikti abi šias operacijas, plius daug daugiau.

Tarkim *N* imčių signalas yra filtruojamas ir gaunamas kitas *N* imčių signalas. Iš sąsūkos pozicijos, kiekviena išėjimo signalo imtis yra pasvertų įėjimo signalo imčių suma. Toliau tarkim turim dviejų sluoksnių tinklą su *N* neuronų kiekviename sluoksnyje. Kiekvieno išėjimo sluoksnio neurono išėjimo vertė yra taip pat pavertų įėjimo sluoksnio neuronų suma. Jei kiekvienas išėjimo neuronas turi tokius pat svorius kaip ir kiti išėjimo neuronai, tai neuroninis tinklas darys tiesinę sąsūką.

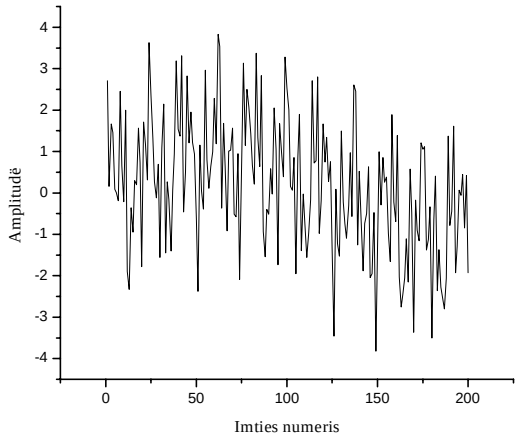
Panašiai, diskretinė Furjė transformacija gali būti apskaičiuota dviejų sluoksnių neuroninio tinklo su *N* neuronų kiekvienam sluoksnyje. Kiekvienas išėjimo sluoksnio neuronas ieškos vieno dažnio komponentės amplitudės. Tai gali būti atlikta, padarant kiekvieno išėjimo neurono svorius lygius sinusoidei, kurios amplitudės yra ieškoma. Rezultate tinklas koreliuos įėjimo signalą su kiekviena bazine sinusoide ir tuo būdu atliks diskrečią Furjė transformaciją.

Tai reiškia, kad neuroniniai tinklai gali atlikti ne tik tiesinį, bet ir netiesinį signalų apdorojimą.

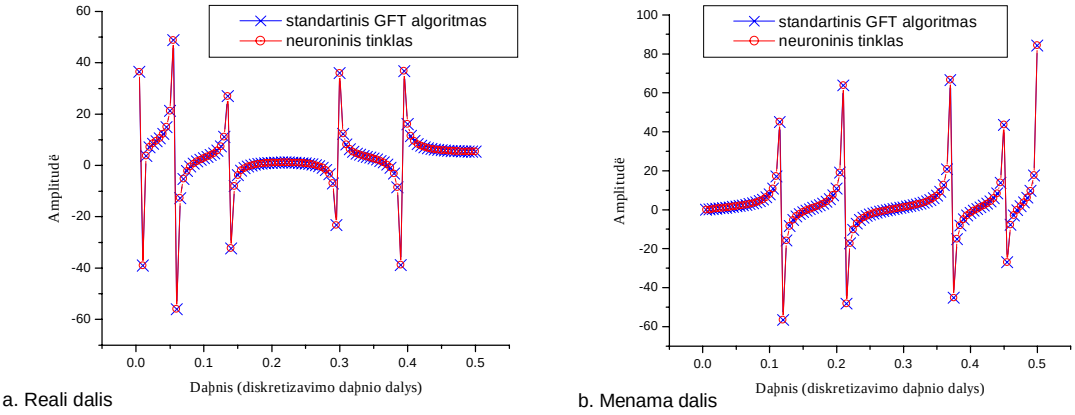
3.4. Kompiuterinis eksperimentas. NT, atliekantis diskrečią Furjė transformaciją

Žemiau pateikti paveikslai vaizduoja neuroninio tinklo atliekamą diskrečią Furjė transformaciją.[16]

[16] šio bei tolimesnių kompiuterinių eksperimentų *MATLAB* programų tekstai pateikti prieduose.



26 pav. Tiriamasis signalas. Sinusoidžių su periodais 1, 13.3, 33.4, 73.6 ir 73.6 suma.



27 pav. Diskretinė Furjė analizė atlikta standartinio GFT algoritmo ir neuroninio tinklo su tiesine aktyvacijos funkcija.

3.5. Neuroninių tinklų perspektyva – ultraspartus skaitmeninių signalų apdorojimas

Skaitmeninių signalų procesorius faktiškai yra *von Neuman'o*[17] mašina, t.y vienu taktu atlieka tiktais vieną operaciją (pvz. sumavimą). Todėl laikas, reikalingas atlikti uždaviniui, yra proporcingas taktų (t.y. operacijų) skaičiui. [Bri00]

Pavyzdžiui, norint atlikti N imčių diskrečią Furjė transformaciją, skaitmeninis signalų procesorius turi atlikti N^2 kompleksinių skaičių daugybos veiksmų. Panaudojus greitąją

[17] von Neumann, John (1903–1957)

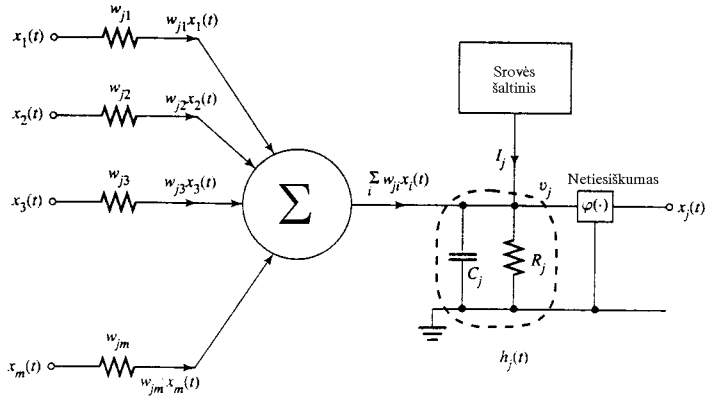
Furjė transformaciją (FFT), reikės $\left(\frac{N}{2}\right)\log_2(N)$ kompleksinių skaičių daugybos veiksmų. [Kes00]

Neuroninis tinklas, skirtingai nuo von Neuman'o mašinos, yra masiškai paralelinė skaičiavimo mašina, jame visi neuronai dirba vienu metu paraleliai, todėl jis *N* imčių diskrečią Furjė transformaciją gali atlikti *vienu taktu*.

Būtent todėl žmogaus smegenys sugeba realiam laike atpažinti vaizdus bei spręsti kitus didelių skaičiavimų reikalaujančius uždavinius, nors smegenų neuronų taktinis dažnis yra lygus tiktais maždaug apie 10 Hz.

Naujos kartos mikroschemos, dirbančios kaip dirbtinis neuroninis tinklas, dėl paralelizmo galėtų apdoroti signalus mažiausiai keliom eilėm sparčiau nei įprastinės *von Neuman'o* mašinos pagrindu veikiančios mikroschemos.

Žemiau pavaizduotas naujos kartos mikroschemos struktūrinis elementas.



28 pav. Principinė elektrinio neurono modelio schema.

Šiuolaikinė technologija įgalina pagaminti mikroschemas, kurios yra neuroninių tinklų elektroninis ekvivalentas, kuriose dirbtiniai neuronai dirba paraleliai, ir tokios eksperimentinės mikroschemos jau gaminamos, kaip kad pavyzdžiui: firmos "Intel" modelis 80170NW (Electrically trainable Analog Neural Network), firmos "Micro Devices" modelis MD1220 (Neural Bit Slice) ir t.t.

3.6. NT ir SSA kaip vienas kitą papildantys metodai

Tarkime vienas iš SSA metodų panaudojamas neuroninio tinklo svoriams rasti. Ar galima tvirtinti, kad neuroninis tinklas yra optimalus? Tradiciniai SSA metodai paprastai yra pagrįsti įėjimo signalo charakteristikomis. Pavyzdžiui Wiener'io filtravimas yra optimalus maksimizuojant signalo ir triukšmo energijų santykį, kai signalo ir triukšmo spektrai yra žinomi, koreliacija yra optimali norint aptikti ieškomą signalą, kai triukšmas yra baltas ir t.t. Problema yra tame, kad dažnai nėra pilnų žinių apie įėjimo signalą. Nors SSA matematika ir yra elegantiška, sistemos darbo našumas kritiškai priklauso nuo to, kaip yra interpretuojami duomenys.

Tarkim, tradicinis SSA algoritmas testuojamas su tam tikrais įėjimo duomenimis. Po to algoritmas šiek tiek pakeičiamas (pvz. pakeičiant parametrų vertes vienu procentu). Jei

antrasis testas duoda geresnius rezultatus, nei pirmasis, reiškia originalus algoritmas nebuvo optimizuotas duotai užduočiai. Dauguma tradicinių SSA algoritmų gali būti pagerinti bandymų ir klaidų metodu koreguojant mažais dydžiais algoritmo parametrus. O būtent tokia ir yra neuroninio tinklo strategija.

Iš kitos pusės, NT mokymo klasikiniai algoritmai yra skirti universaliam mokymui ir neatsižvelgia į konkrečių taikymų ypatumus. Tačiau, jei NT yra skirtas šalinti triukšmui, tai jam galioja panašūs dėsningumai kaip ir skaitmeniniam filtrui, o nei vienas klasikinis NT mokymo algoritmas į juos neatsižvelgia.

Toliau darbe (6 skyriuje) parodysim, kad pasinaudojant SSA teorijos dėsningumais įmanoma pagreitinti NT mokymą ir pagerinti galutinį NT darbą.

3.7. Išvados

SSA ir NT teorijos kaip mokslo sritys užgimė nepriklausomai viena nuo kitos, ir turi skirtingą ideologiją. Abu – ir SSA, ir NT metodai turi savų privalumų ir savų minusų. Šių abiejų ideologijų integracija, išnaudojant kiekvienos privalumus, padeda išspręsti praktinius uždavinius, kurių negalima išspręsti taikant vien SSA ar NT metodus. Šiame skyriuje parodyta, kad neuroniniais tinklais galima realizuoti pagrindines SSA operacijas, o aparatūrinė neuroninių tinklų realizacija įgalintų atlikti *ultraspartų* skaitmeninių signalų apdorojimą.

4. Tiesinis signalų apdorojimas

1 skyriuje nagrinėjome SSA teorijos pagrindines sąvokas. Šiame skyriuje SSA teorija nagrinėjama giliau iš praktinio taikymo pozicijų.

Norint sėkmingai filtruoti naudingą informaciją iš triukšmingo signalo, pirmiausia reikia išsiaiškinti kokiu būdu informacija gali būti koduojama signaluose. Galimi informacijos kodavimo būdai aptariami 4.1. poskyryje. Iš SSA pozicijų žmogaus kalba yra sudėtingas signalas ir 4.2. poskyryje aprašomas žmogaus kalbos modelis. 4.3. poskyryje nagrinėjama kaip žmogus suvokia garsinę informaciją, kokie garsinio signalo parametrai yra svarbūs, o kurie – ne. 4.4. poskyryje nagrinėjama polinė notacija, kuri geriau padeda suvokti signalų charakteristikas. 4.5. poskyryje nagrinėjama kokią įtaką signalams turi fazės charakteristikos. Tada pereinama prie SSA teorijos praktinio taikymo realių uždavinių sprendimui – 4.6. poskyryje nagrinėjami tiesinių skaitmeninių filtrų sudarymo klausimai. Kokį filtrą reiktų kurti konkrečiam praktiniam uždaviniui, priklauso nuo siekiamų tikslų – optimalaus filtravimo problematika nagrinėjama 4.7. poskyryje. Tai, kad signalų prigimties supratimas įgalina sėkmingai juos apdoroti, parodoma 4.8. poskyryje pateiktame kompiuteriniame eksperimente, kur pristatomas balto triukšmo šalinimo iš žmogaus kalbos gretimų spektrinių segmentų sumavimo algoritmas. 4.9. poskyryje pateikiamos išvados.

4.1. Informacijos kodavimo signaluose būdai

Norint sėkmingai apdoroti signalus, svarbiausias dalykas yra suprasti kokiu būdu informacija yra užkoduota signaluose su kuriais yra dirbama. Yra daug informacijos kodavimo būdų. Tai ypač pasakytina apie žmogaus sukurtus dirbtinius signalus, kaip kad pavyzdžiui amplitudinė moduliacija (AM), dažninė moduliacija (FM), impulsinė moduliacija, impulso pločio moduliacija ir t.t. Šį sąrašą galima tęsti ir tęsti. Tačiau, laimei, yra tiktais du *esminiai* būdai kaip informacija gali būti koduojama. Informacija gali būti koduota dažnio srityje arba laiko srityje.

Informacija koduota laiko srityje parodo įvykio laiką ir jo amplitudę. Pavyzdžiui, matuojamas saulės šviesos skaistis. Kas fiksuotą laiko momentą atliekamas matavimas. Kiekviena matavimo imtis neša informaciją, kuri gali būti interpretuota nepriklausomai nuo kitų imčių. Net jei turima tik viena imtis iš signalo, vis vien galima žinoti, kas buvo išmatuota. Tai yra pats paprasčiausias informacijos kodavimo būdas.

Kaip kontrastas, informacija užkoduota dažnio srityje yra ne tokia tiesioginė. Tarkime, turime skambančios stygos įrašą. Pagrindinis dažnis ir harmonikos tiesiogiai susijusios su stygos mase ir medžiagos elastingumu. Viena tokio įrašo imtis neneša jokios informacijos apie stygą. Informacija yra koduota įrašo imčių *tarpusavio santykyje*.

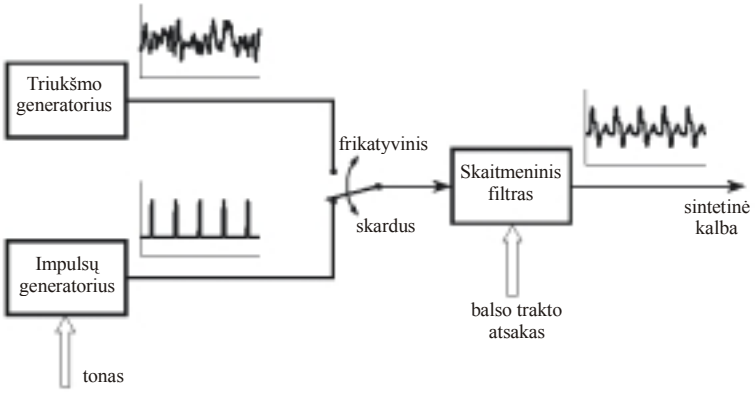
Būtent todėl tokie svarbūs yra impulso ir dažninis atsakai. Impulso atsakas nusako kaip informacija, koduota laiko srityje, yra sistemos modifikuojama. Dažninis atsakas nusako kaip informacija yra keičiama dažninio srityje. Šis skirtumas yra labai kritiškas filtrų kūrime, nes yra neįmanoma optimizuoti filtro abiem taikymams. Geras laiko srities

filtras duoda blogus rezultatus dažnio srityje ir atvirkščiai. Jei kuriamas filtras skirtas pašalinti trukdžius iš kardiogramos (informacija koduota laiko srityje), tai impulso atsakas yra svarbiausias parametras, dažnio atsakas yra nesvarbus. Jei kuriamas filtras skirtas pagerinti klausai (informacija koduota dažnio srityje), tai svarbus yra dažnio atsakas, o impulso atsakas nesvarbus.

4.2. Žmogaus kalbos modelis

Didžioji žmogaus kalbos garsų dalis gali būti suklasifikuota kaip *skardūs* arba *frikatyviniai(pūčiamieji)* garsai. Skardūs garsai suformuojami, kai oras išeina iš plaučių, keliauja per balso stygas, ir išeina per burną ir/arba nosį. Balso stygos yra dvi plonytės audinio plėvelės, esančios oro srauto kelyje, prie pat Adomo obuolio. Priklausomai nuo raumenų įtempimo, balso stygos gali vibruoti nuo 50 iki 1000 Hz, suformuodamos periodinius oro gūsius, kurie patenka į burną. Frikatyviniai garsai kyla kaip atsitiktinis triukšmas, o ne nuo balso stygų virpėjimo, kai oro srautas atsimuša į liežuvį, lūpas, ir/arba dantis – priešais kliūtį susidaro turbulenciniai oro sūkuriai .

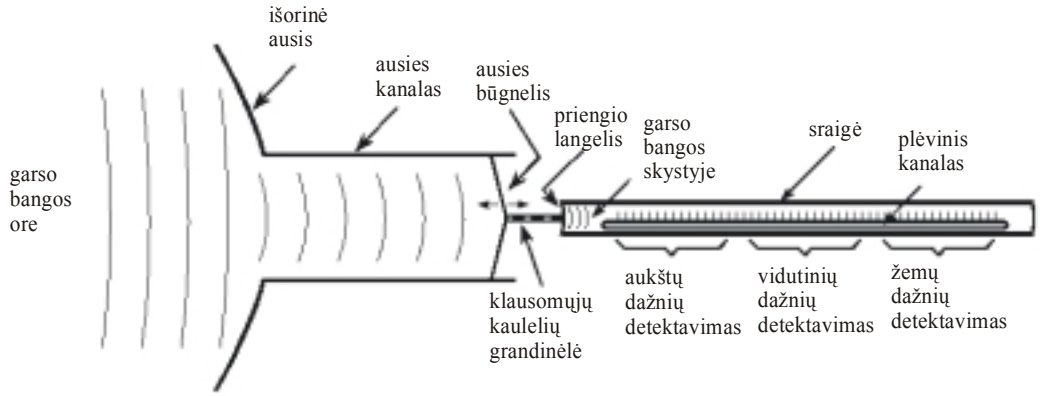
Abu šie garso šaltiniai yra toliau modifikuojami akustinėse ertmėse, kaip kad pvz. liežuvis, lūpos, burna, nosiaryklė. Kadangi garso sklidimas šiose struktūrose yra tiesinis procesas, tai jis gali būti atvaizduotas kaip tiesinis filtras su atitinkamai parinktu impulso atsaku. Kadangi akustinių ertmių dydžiai yra kelių centimetrų eilės, tai jų dažninis rezonansas yra kelių kilohercų eilės.



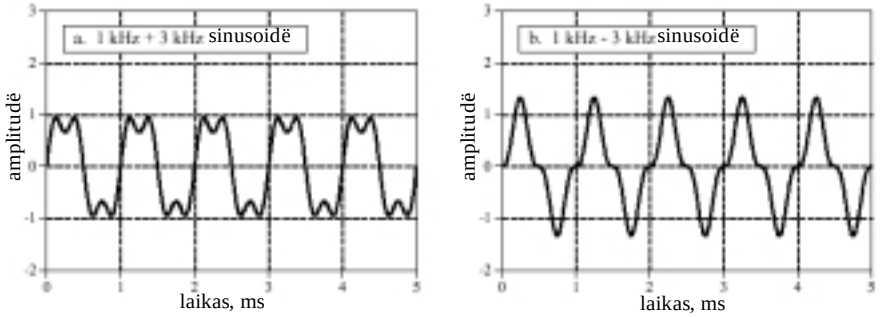
29 pav. Trumpame laiko intervale (maždaug nuo 2 iki 40 milisekundžių), kalba gali būti modeliuojama trimis parametrais: (1) parenkant triukšmingą arba harmoninį sužadinimą, (2) harmoninio sužadinimo (tono) dažniu, (3) skaitmeninio filtro, modeliuojančio garso trakto atsaką, koeficientais.

4.3. Žmogaus ausis – dažninio spektro analizatorius

Išnagrinėsime kaip žmogus suvokia garsinę informaciją.



30 pav. Žmogaus ausies funkcinė diagrama. Garso bangos, pasiekusios ausį, suvirpina būgnelį. Jo virpesiai per klausomųjų kaulelių grandinėlę ir prieangio langelį perduodami sraigės perilimfai ir endolimfai (vandeningi skysčiai). Šių skysčių bangavimas transformuojamas spiraliniame organe į nervinį impulsą, kuris neuronų grandine nueina į galvos smegenų žievę ir ten sukelia garso pojūtį. Sraigės plėvinio kanalo vijų ertmėje yra klausos receptorius – spiralinis organas, kuris susideda iš maždaug 12,000 nervinių ląstelių. Kadangi plėvinis kanalas yra kintamo stangrumo, tai kiekviena nervinė ląstelė reaguoja tik į siaurą garso dažnių juostą, ir tokiu būdu ausis atlieka dažnio spektro analizatoriaus vaidmenį. [Bro00]



31 pav. Žmogaus ausis ir fazės detektavimas. Žmogaus ausis yra labai nejautri santykinei komponentinių sinusoidžių fazei. Pavyzdžiui, abu šie signalai žmogaus ausiai skamba vienodai kadangi komponenčių amplitudės yra vienodos, nepaisant to, kad santykinės fazės skiriasi.

Žmogaus ausies nejautrumas fazei gali būti paaiškintas nagrinėjant kaip garsas sklinda per aplinką. Tarkim jūs klausotės žmogaus kalbančio kambary. Didelė garso dalis pasiekia jūsų ausį atsispindėjusi nuo sienų, lubų ir grindų. Kadangi garso sklidimas priklauso nuo dažnio (kaip kad: slopinimas, atspindys ir rezonansas), skirtingi dažniai pasiekia jūsų ausį nusklidę skirtingus kelius. Tai reiškia, kad santykinė kiekvieno dažnio fazė keisis, kai jūs judėsite kambaryje. Kadangi ausis ignoruoja šiuos fazinius pokyčius, žmogus suvokia garsą kaip *nesikeičiantį,* kai jis juda kambaryje. Iš fizikos požiūrio, garso

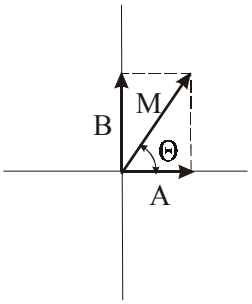
bangų fazės tampa atsitiktinai išblaškytos, kai garsas sklinda sudėtinga aplinka. Kitaip sakant, žmogaus ausis nejautri fazei, kadangi ji neša mažai naudingos informacijos.

4.4. Polinė notacija

Kaip jau rašyta anksčiau, dažnio sritis yra grupė kosinusinių ir sinusinių bangų amplitudžių. Toks informacijos pateikimas vadinamas *stačiakampe notacija*[18]. Alternatyviai dažnio sritis gali būti išreikšta poline forma. Šioje notacijoje reali dalis *ReX[]* ir menama dalis *ImX[]* yra pakeisti *X[] Amplitude*, žymima lygtyse kaip *MagX[]*, ir *X[] Faze*, žymima lygtyse kaip *PhaseX[]*.

Sudėjus kosinusinę ir sinusinę bangas (to paties dažnio), rezultatas yra banga su kita amplitude ir fazės postūmiu:

$$A \cdot \cos(x) + B \cdot \sin(x) = M \cdot \cos(x + \Theta).$$



32 pav. Konvertavimas iš stačiakampės į polinę notaciją. Sinusinės ir kosinusinės bangos susideda kaip paprasti vektoriai.

Sekančios formulės nusako stačiakampės notacijos vertimą į polinę.

$$MagX[k] = \sqrt{\mathrm{Re}\,X[k]^2 + \mathrm{Im}\,X[k]^2},$$

$$PhaseX[k] = \arctan\left(\frac{\mathrm{Im}\,X[k]}{\mathrm{Re}\,X[k]}\right).$$

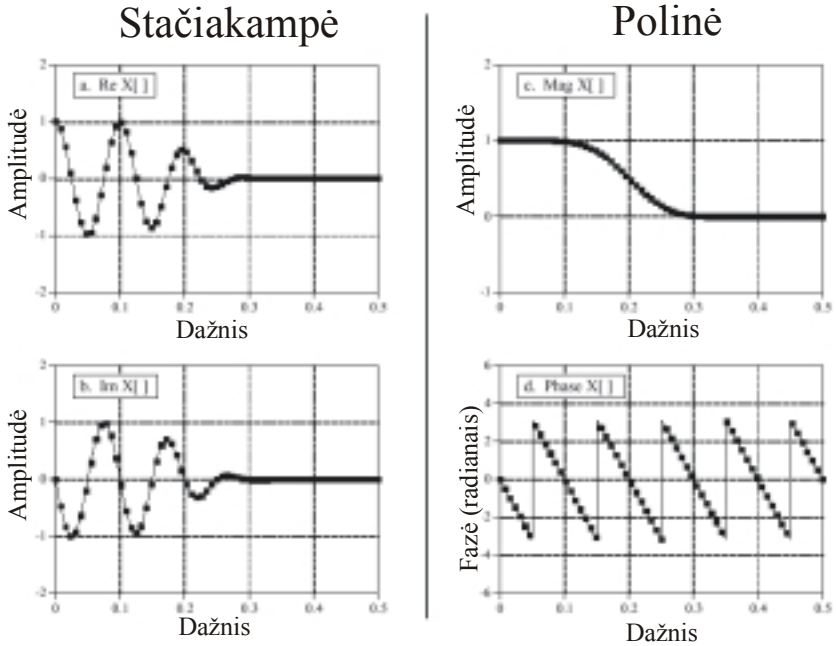
Bei polinės notacijos vertimą į stačiakampę:

$$\mathrm{Re}\,X[k] = MagX[k] \cdot \cos(PhaseX[k]),$$

$$\mathrm{Im}\,X[k] = MagX[k] \cdot \sin(PhaseX[k])$$

Kodėl reikalinga polinė notacija? Prisiminkime *sinusoidinio tikslumo* savybę: jei sinusoidė patenka į tiesinę sistemą, tai išėjime bus taip pat sinusoidė, lygiai to paties dažnio, tik su pakeista amplitude ir faze. Polinė notacija tai tiesiogiai atvaizduoja.

[18] iš angl. rectangular notation



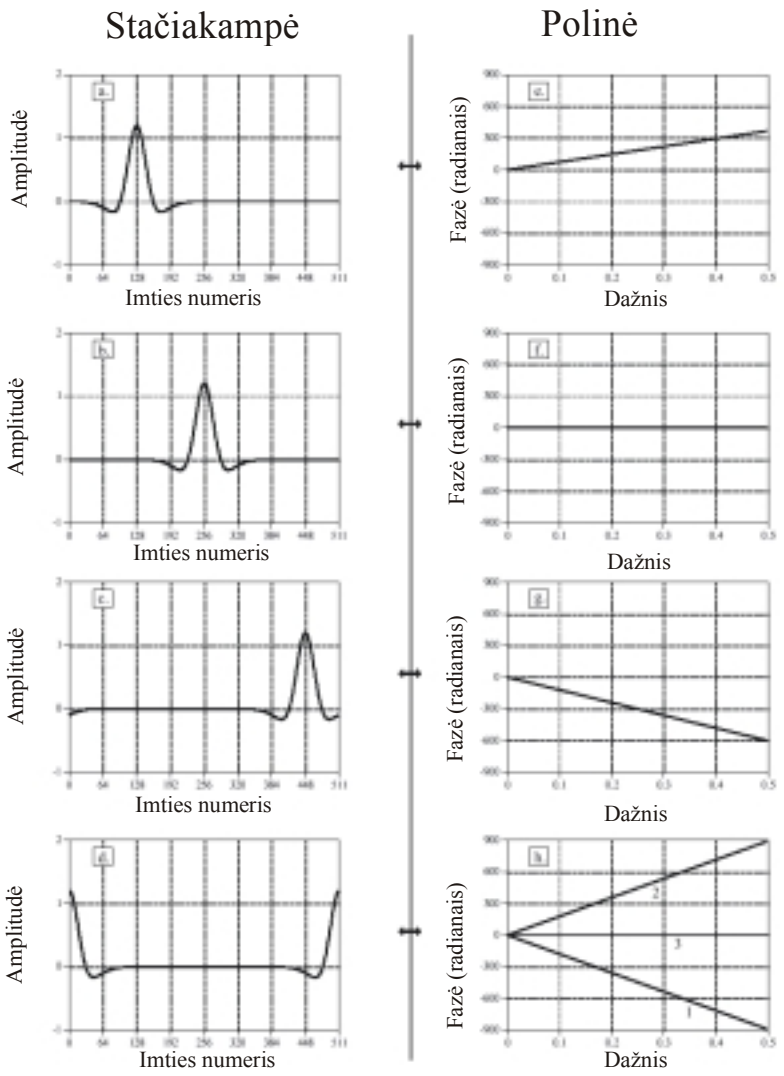
33 pav. Šis pavyzdys rodo dažnio sritį, išreikštą per stačiakampę ir polinę notacijas. Polinė notacija suteikia geresnį supratimą apie signalo charakteristikas. Stačiakampė notacija naudojama, kai reikalinga atlikti matematinius skaičiavimus.

4.5. Fazės charakteristikos

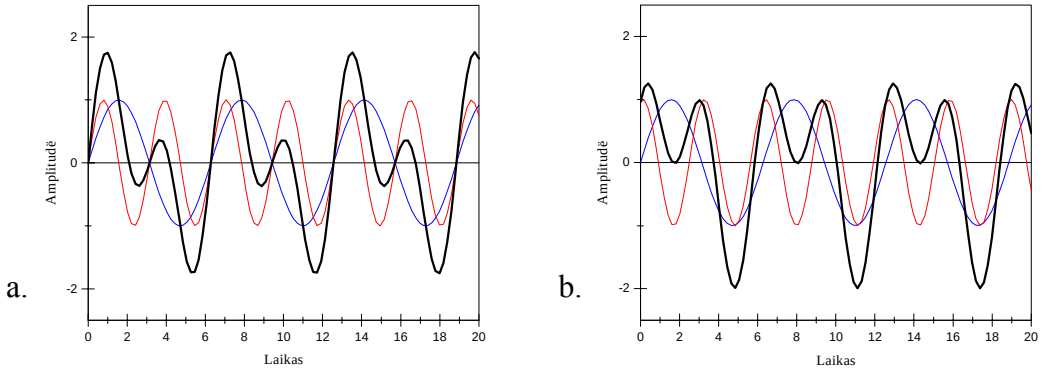
Jei $x[n] \leftrightarrow MagX[f]\ \&\ Phase[f]$, tai poslinkis laiko srityje duos $x[n+s] \leftrightarrow MagX[f]\ \&\ Phase[f] + 2 \cdot \pi \cdot s \cdot f$, kur f yra diskretizavimo dažnio dalis ir kinta nuo 0 iki 0.5. Kitaip sakant poslinkis per s laiko srityje paliks amplitudę nepakeistą, bet pridės prie fazės $2 \cdot \pi \cdot s \cdot f$.

Jei laiko srityje signalas yra simetriškas vertikalios ašies atžvilgiu, tai dažnio srityje fazė bus tiesi linija. Tokia simetrija vadinama *faziškai tiesine*[19]. Signalai, kurie neturi tokios simetrijos vadinami *faziškai netiesiniais*[20]. Kai laiko srityje signalas pastumiamas į dešinę, tai fazė lieka tiese, tačiau jos nuolydis sumažėja, o kai pastumiamas į kairę – padidėja. Jei laiko srityje signalas yra pilnai simetriškas vertikalios ašies atžvilgiu, tai dažnio spektre fazė bus visur lygi 0.

[19] iš angl. linear phase
[20] iš angl. nonlinear phase



34 pav. Fazės pasikeitimas dėl impulso atsako poslinkio laiko srityje.



35 pav. Skirtingo fazės postūmio skirtingiems dažniams įtaka galutiniam signalui.

a pav. Parodyta sinusoidžių *sin(x)* ir *sin(2x)* suma,
b pav. Parodyta sinusoidžių *sin(x)* ir *sin(2x+1.3)* suma, t.y. antros sinusoidės fazė pastumta.
Rezultate išėjimo suminio signalo forma tampa skirtinga.

4.6. Tiesiniai skaitmeniniai filtrai

Skaitmeninio filtro branduolio koeficientai randami sekančiai:

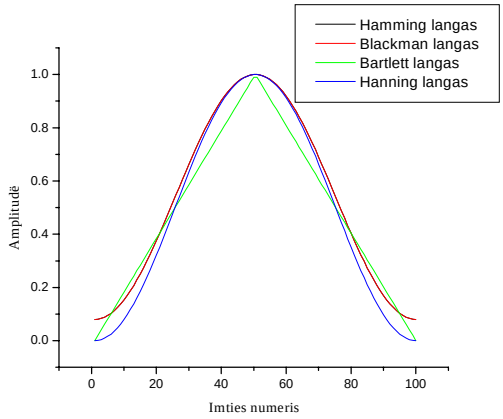
- Sudaromas filtro norimas dažninis atsakas;
- Suskaičiuojama jo atvirkštinė Furjė transformacija;
- Gautas rezultatas ir yra filtro koeficientai.

Tačiau tokiu būdu gautas filtro atsakas dar turi būti *paslinktas*, *apkarpytas* ir *padaugintas iš lango funkcijos*. Tai reikalinga dėl to, kad kuriant filtrą, norimas dažnio atsakas yra užduodamas masyvo elementais. Dabar pasvarstykime kaip dažnio atsakas elgiasi *tarp* šių užduotų masyvo elementų. Paprastumo dėlei, galima įsivaizduoti du atvejus: "gerą" ir "blogą". "Geru" atveju, dažnio atsakas bus glotni kreivė tarp dviejų užduotų masyvo elementų. "Blogu" atveju – bus didelės fliuktuacijos. Norint išgauti gerą dažninį atsaką, masyvo elementų skaičius turi būti begalinis (kad priartėti prie tolydinės kreivės). Po atvirkštinės Furjė transformacijos, impulso atsako ilgis irgi bus *begalinis*. Kitaip sakant, "geras" dažninis atsakas yra tai, ko praktikoje negalima realizuoti (begalinio dydžio masyvo). Kai *N/2+1* imčių ilgio dažnio atsaką norima atvaizduoti į *N* imčių ilgio impulso atsaką, rezultatas yra tas, kad begalinio ilgio atsakas *susisuka*[21] i *N* imčių. Kai tai įvyksta, dažninis atsakas iš "gero" tampa "blogu". Tačiau, padauginimas iš lango funkcijos stipriai sumažina ši susisukimą, tuo būdu kreivė tarp dažnio srities imčių tampa glotni.

Norint išgauti aštrų dažninį atsaką, reikia kad dažniniame spektre būtų kuo daugiau imčių, todėl atvirkštinė Furjė transformacija turės daug filtro branduolio koeficientų. Filtro koeficientų skaičių galima sumažinti atsižvelgiant į sekančias savybes: 1) didėjant imties numeriui, filtro koeficientai artėja į nulį; 2) daug koeficientų turi mažas vertes, todėl juos galime atmesti kaip nereikšmingus.

Tačiau filtro branduolio apkarpymas reiškia, kad apkarpomas signalas, o tai savo ruoštu reiškia signalo dažninio spektro išplatėjimą. Susiaurinti dažninį spektrą galima padauginant jį iš *lango funkcijos*.

[21] iš angl. aliased



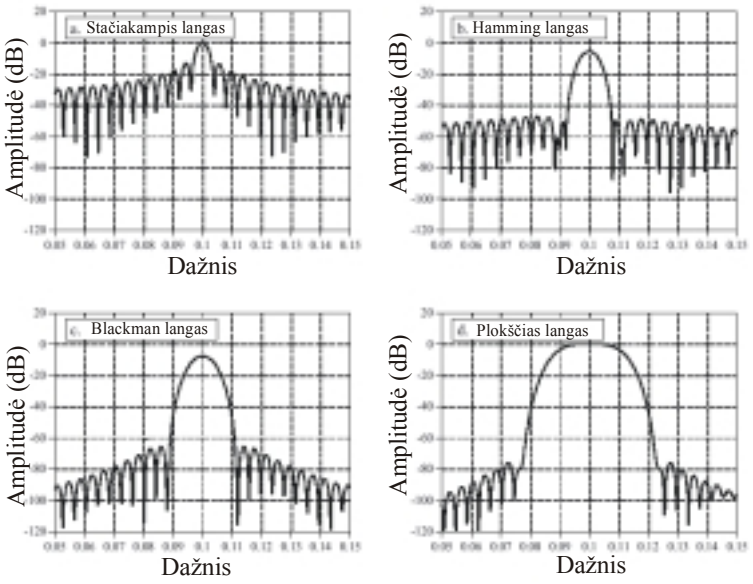
36 pav. Lango funkcija. Jų yra daug, bet dažniausiai vartojamos yra Hamming, Blackman, Bartlett, Hanning.

Hamming langas: $w[i] = 0.54 - 0.46\cos(2\pi i / M)$, kur i kinta nuo 0 iki M.

Blackman langas: $w[i] = 0.42 - 0.5\cos(2\pi i / M) + 0.08\cos(4\pi i / M)$

Hanning langas: $w[i] = 0.5 - 0.5\cos(2\pi i / M)$

Bertlett langas yra paprasčiausias trikampis.



37 pav. Lango funkcijų dažninis spektras. Stačiakampis langas (a) turi siauriausią pagrindinį lapelį, tačiau didžiausią šalutinių lapelių amplitudę. Hamming (b), Blackman (c) turi mažesnius šalutinius lapelius, tačiau pagrindinio lapelio išplatėjimo sąskaita. Plokščias langas (d) yra naudojamas, kai centrinio lapelio amplitudė turi būti tiksliai išmatuota. Kurią lango funkciją pasirinkti, priklauso nuo siekiamo tikslo. [Smi99]

4.7. Optimalūs filtrai

Įprastinė filtravimo užduotis yra atskirti signalą (pavyzdžiui eksponentinį impulsą) paslėptą atsitiktiniame triukšme.

Apžvelgsime tris filtrus, kurių kiekvienas yra "geriausias" (optimalus) iš skirtingų pozicijų.

Judančio vidurkio filtro[22] lygtis yra:

$$y[i] = \frac{1}{M} \sum_{j=0}^{M-1} x[i+j],$$

kur *x[]* – įėjimo signalas, *y[]* – išėjimo signalas, *M* – filtro branduolio imčių skaičius. Pvz.: 5 imčių judančio vidurkio filtre, 80-ta signalo imtis apskaičiuojama:

$$y[80] = \frac{x[80] + x[81] + x[82] + x[83] + x[84]}{5}$$

Šis filtras optimalus tuo požiūriu, kad labiausiai sumažina triukšmą išlaikydamas aštriausią šuoliuko atsaką.

Atitikimo filtro[23] koeficientai yra ieškomo signalo formos veidrodinis atspindys (iš kairės į dešinę). Šio filtro idėja yra koreliacija, ir veidrodinis atspindys reikalingas norint išgauti koreliaciją, naudojant sąsūką. Kiekvienoje išėjimo signalo imtyje, amplitudė yra nusakoma kiek gerai filtro branduolys atitinka signalo segmentą. Šio filtro išėjimo signalas neatitinka pradinio signalo formos, tačiau tai nesvarbu. Jei naudojamas atitikimo filtras, tai ieškomo signalo forma jau yra žinoma. Šis filtras optimalus tuo atžvilgiu, kad išėjimo signalo pikas yra pakilęs aukščiau virš triukšmo, nei bet kurio kito tiesinio filtro.

Wiener'io filtras atskiria signalus pasiremiant jų dažnio spektrais. Jis yra optimalus tuo atžvilgiu, kad maksimizuoja signalo galios santykį su triukšmo galia (viso signalo ilgyje, ne atskiruose taškuose).

Wiener'io filtro dažnio atsakas *H[f]* yra nusakomas signalo *S[f]* ir triukšmo *N[f]* dažnio spektrais.

$$H[f] = \frac{S[f]^2}{S[f]^2 + N[f]^2}$$

Svarbios tiktais amplitudės, vizos fazės lygios nuliui.

Nors optimalių filtrų matematinės idėjos yra elegantiškos, tačiau jos dažnai yra nepraktiškos. Kiekvienas šių filtrų yra optimalus *tam tikru* aspektu. Ir negalima teigti, kad kažkuris filtras yra pilnai optimalus. Pvz. biomedicinos inžinierius galėtų naudoti Wiener'io filtrą, norėdamas maksimizuoti signalo ir triukšmo energijų santykį. Tačiau nėra akivaizdu, kad šis filtras optimizuotų daktaro galimybes aptikti nereguliarų širdies darbą žiūrint į signalą.

Šiame darbe optimumo kriterijumi pasirinktas signalo ir triukšmo energijų santykis, kadangi jis turi gilią fizikinę prasmę ir todėl labiausiai tinka dažniausiai pasitaikančių praktinių uždavinių sprendimui. Pvz., jei į radijo siųstuvo stiprintuvą patenka signalas ir triukšmas, tai siųstuvo spinduliavimui sunaudojama galia yra proporcinga signalo ir triukšmo galių (galia yra tiesiškai proporcinga energijai) sumai, ir norint minimizuoti siųstuvo sunaudojamą galią, reikia maksimizuoti signalo ir triukšmo galių santykį. Todėl šiame darbe optimaliu filtru laikomas Wiener'io filtras.

---

[22] iš angl. moving average filter

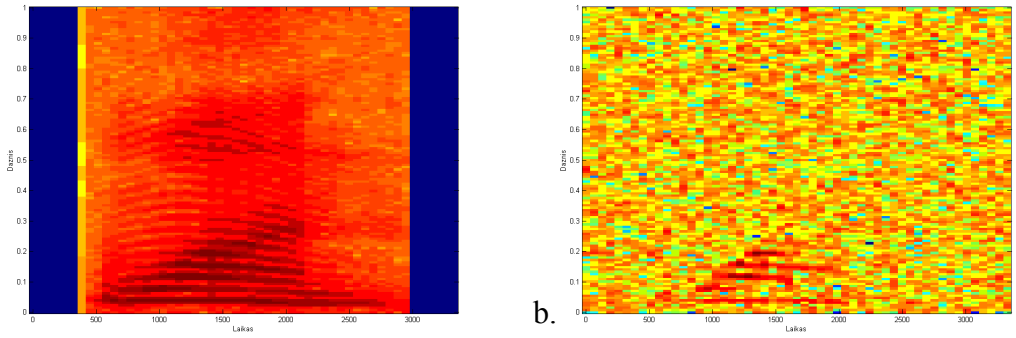
[23] iš angl. matched filter

4.8. Kompiuterinis eksperimentas. Balto triukšmo šalinimas iš žmogaus kalbos gretimų spektrinių segmentų sumavimo metodu

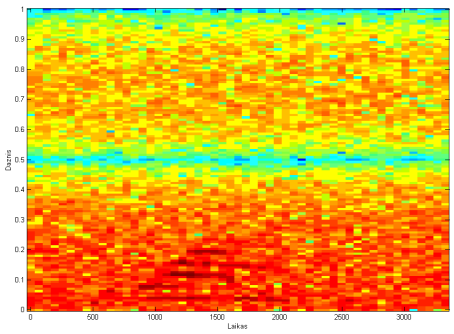
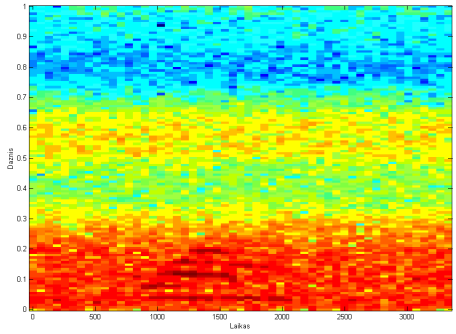
Pasinaudojant tuo, kad trumpame laiko intervale (maždaug nuo 2 iki 40 milisekundžių), žmogaus kalbos signalas yra daugmaž pastovus (spektro atžvilgiu), baltą triukšmą iš žmogaus kalbos įrašo galima šalinti sekančiu nauju metodu:

1. Imama kalbos signalo *N* pirmų imčių;
2. Atliekama šios signalo segmento Furjė transformacija (gaunamas dažnio spektras);
3. Paslenkama viena imtimi į šoną ir procesas kartojamas iki signalo pabaigos (gaunamos persidengiančių segmentų Furjė transformacijos);
4. Tada imama *k* pirmų gretimų Furjė transformacijų segmentų ir jie sudedami kaip kompleksinių skaičių masyvai. To pasėkoje kalbos signalo, esančio gretimuose segmentuose spektras susisumuoja ir išryškėja, o baltas triukšmas gretimuose spektro segmentuose yra nekoreliuotas ir todėl *k* koeficientui artėjant į begalybę balto triukšmo spektro suma artėtų į nulį;
5. Paslenkama vienu segmentu į šoną ir procesas kartojamas iki segmentų pabaigos;
6. Šiam susumuotų Furjė transformacijų segmentų masyvui atliekama atvirkštinė Furjė transformacija;
7. Gauti laiko srities persidengiantys segmentai turi problemą, kad jų kraštai nėra suderinti su kaimyniniais segmentais. Tai galima išspręsti padauginant segmentus iš lango funkcijos (pvz. Hamming). Šiuos, padaugintus iš lango funkcijos, persidengiančius segmentus susumuojame į galutinį išfiltruotą signalą.

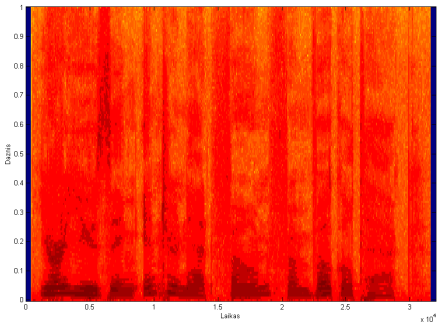
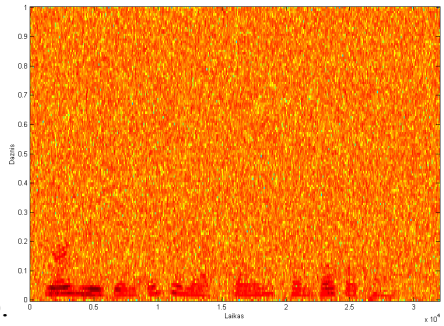
a. b.



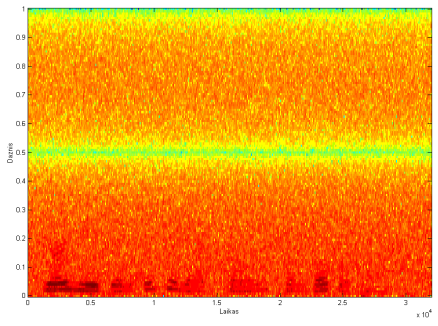
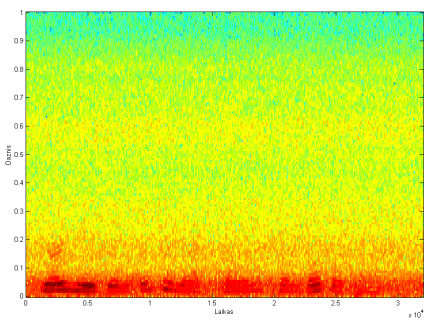
38 pav. Balso spektrograma.
a paveiksle parodytas originalus signalas – angliškas žodis “one” (moteriškas balsas).
b paveiksle prie originalaus signalo (angliškas žodis “one”) pridėtas baltas triukšmas, kurio amplitudė lygi 0.5 originalaus signalo amplitudės.

a.  b. 

39 pav. Balso spektrograma. Balto triukšmo šalinimas iš žmogaus kalbos.
a paveiksle – gretimų spektrinių segmentų sumavimo metodu. Segmento ilgis N=128, sumavimo koeficientas k=4. Triukšmas sumažintas 2.0578 dB
b paveiksle – naudojant Wiener'io filtrą. Triukšmas sumažintas 3.3092 dB

a.  b. 

40 pav. Balso spektrograma.
a paveiksle parodytas originalus signalas – angliškas sakinys "You're using Internet Phone from Vocaltec" (moteriškas balsas).
b paveiksle prie originalaus signalo pridėtas baltas triukšmas, kurio amplitudė lygi 0.5 originalaus signalo amplitudės.

a.  b. 

41 pav. Balso spektrograma. Balto triukšmo šalinimas iš žmogaus kalbos.
a paveiksle – gretimų spektrinių segmentų sumavimo metodu. Segmento ilgis N=128, sumavimo koeficientas k=4. Triukšmas sumažintas 4.1290 dB

b paveiksle – naudojant Wiener'io filtrą. Triukšmas sumažintas 2.4598 dB.

Žmogaus kalbos signalas yra labai sudėtingas ir jo charakteristikos stipriai keičiasi laikui bėgant. Todėl jei naudojame Wiener'io filtrą (kurio dažninis atsakas apskaičiuotas pasinaudojant visu kalbos įrašu), tai jei įrašas yra trumpas (kaip šiame eksperimente – vienas žodis), tai Wiener'io filtras duoda geresnį rezultatą, nei kad gretimų spektrinių segmentų sumavimo metodas. Tačiau ilgėjant įrašo trukmei Wiener'io filtro triukšmo šalinimo efektyvumas krenta. Tuo tarpu gretimų spektrinių segmentų sumavimo metodo efektyvumas nepriklauso nuo įrašo ilgio, nes jis "dirba" su trumpais įrašo segmentais.

Todėl gretimų spektrinių segmentų sumavimo metodas turi pranašumą prieš Wiener'io filtrą, jei filtruojamas žmogaus kalbos signalas yra ilgos trukmės (t.y. ilgesnis nei vienas ar keli žodžiai). Taip pat jo pranašumas yra tame, kad jis gali būti panaudotas filtruoti baltą triukšmą iš signalo realaus laiko sistemose, nes jam nereikalinga *a priori* pilna informacija apie signalo charakteristikas.

4.9. Išvados

Sėkmingai dirbti su signalais įmanoma tiktais suvokiant kokiu būdu yra koduojama juose informacija. Yra tiktais du esminiai informacijos kodavimo būdai: dažnio srityje arba laiko srityje. Optimalus filtravimas tegali būti "optimalus" kažkieno atžvilgiu, neįmanoma, kad skaitmeninis filtras būtų optimalus visais aspektais. Dažniausiai pasitaikančių praktinių uždavinių sprendimui geriausiai tinka Wiener'io filtras, kadangi jis turi gilią fizikinę prasmę – maksimizuoja signalo ir triukšmo energijų santykį, todėl šiame darbe optimaliu filtru laikomas Wiener'io filtras.

Žmogaus kalba yra sudėtingas signalas (jame informacija koduota dažninėje srityje), tačiau jam galioja tam tikros taisyklės. Šiame skyriuje pristatytas balto triukšmo šalinimo iš žmogaus kalbos gretimų spektrinių segmentų sumavimo algoritmas parodo, kad signalų prigimties supratimas ir šių žinių panaudojimas įgalina sėkmingai filtruoti naudingą informaciją iš triukšmingo signalo.

5. NT darbo efektyvumo optimizavimo metodai

2 skyriuje nagrinėjome neuroninių tinklų teorijos pagrindus. Šiame skyriuje neuroninių tinklų teorija nagrinėjama giliau iš praktinio taikymo pozicijų – kokiais būdais galima pagerinti neuroninio tinklo mokymą ir galutinį darbą.

NT mokymą galima vaizduoti kaip vienos dimensijos kelią parametrų (svorių) erdvėje. 5.1. poskyryje aptariama kokie veiksniai įtakoja judėjimą šiuo keliu vykstant mokymui. Neuroninio tinklo mokymas priklauso nuo daugybės faktorių, kuriems optimalių sprendinių nėra žinoma, todėl NT mokymo rezultatai yra įtakoti kūrėjo personalinės empirinės patirties. Tačiau tam tikri dėsningumai vis dėlto yra pastebėti ir 5.2. poskyryje nagrinėjamos euristikos, kurios pagerina mokymo proceso darbą:

1. Svorių keitimas nuosekliu vietoj paketinio režimu;
2. Mokymo duomenų vektorių informacijos turinio maksimizavimas;
3. Neurono aktyvacijos funkcijos parinkimas;
4. Tinkamas trokštamų išėjimų verčių parinkimas;
5. Įėjimo duomenų normalizavimas;
6. Neuroninio tinklo parametrų pradinio inicijavimo įtaka prieš mokant;
7. Neuronų mokymo greičio parinkimas;
8. Mokymas "užuominomis" – *a priori* informacijos išnaudojimas NT mokymo procese.

Nepaisant to, kad yra prikurta daug euristinių metodų, dažnai būna, kad jų taikymas praktinių uždavinių sprendimui neduoda trokštamo rezultato, o tai reiškia, kad reikalingi papildomi euristiniai metodai. 5.3. poskyryje pristatomas dar vienas euristinis metodas, pagrįstas evoliuciniu NT mokymu, kai kiekvienoje pakopoje keičiami mokymo parametrai ir prie mokymo duomenų pridedamas vis didesnis triukšmas. Kompiuteriniu eksperimentu pademonstruojamas šio metodo efektyvumas. 5.4. poskyryje pateikiamos išvados.

5.1. NT mokymas – kelias parametrinėje erdvėje

Kai neuroninis tinklas yra apmokinamas su apmokymo pavyzdžiais, jis turi mokėti dirbti su tokio pat tipo duomenimis, tik kurių jis niekada anksčiau nėra matęs. Neuroninio tinklo savybė teisingai dirbti su duomenimis, kurių jis nėra matęs vadinama neuroninio tinklo *generalizavimo kokybe*[24].

Modeliuojant tinklus, tinklo generalizavimo kokybei yra kritiškai svarbu, kad apmokymo duomenų vektorių skaičius būtų didesnis nei tinklo parametrų skaičius (tinklo svorių bei slenksčių skaičius). Tinkamas duomenų vektorių skaičius yra maždaug dešimt kart didesnis nei tinklo laisvų parametrų skaičius, o apatinė riba – maždaug trys kartai.

[24] iš angl. generalization performance

[Hay98]. Tačiau esant tam tikroms situacijoms (atitinkamai paruošiant mokymo duomenų vektorius), mokymo duomenų vektorių skaičius gali būti *netgi mažesnis* nei tinklo laisvų parametrų skaičius. Raudys [Rau01] parodė, kad pvz. neuroniniam tinklui, kurio laisvų parametrų skaičius lygus 100, užtenka tiktais *penkių* mokymo vektorių kiekvienai klasei (klasių skaičius lygus 10).

Sekanti interpretacija yra naudinga norint geriau suprasti atbulo sklidimo algoritmo savybes. NT mokymo netiesinė optimizavimo procedūra nusako vienos dimensijos kelią parametrų (svorių) erdvėje. Atbulo sklidimo algoritme judėjimas šiuo keliu (gradiento kryptimi) užtikrina klaidos mažėjimą. Taigi galimi tinklo sprendimai atitinka šio kelio taškus. Akivaizdu, kad kelias priklauso nuo sekančių faktorių:

a) apmokymo duomenų aibė;
b) netiesinių funkcijų aibė, t.y. NT architektūra;
c) pradinis kelio taškas, t.y. pradinės parametrų vertės (inicijuojantys svoriai) ;
d) algoritmo sustojimo taisyklė, nuo kurios priklauso galutinis kelio taškas.

Norint išanalizuoti optimizavimo algoritmo efektus, laikysime kad (a) ir (b) faktoriai yra fiksuoti. Kadangi NT klaidos paviršius turi daug lokalių minimumų, tai optimizacijos metodo surastas sprendinys (lokalus minimumas) priklausys nuo (c) ir (d) parinkimo.

Paprastai rekomenduojama inicijuoti tinklo svorius *mažomis* atsitiktinėmis vertėmis. Tačiau nusakyti "geras" mažas pradines vertes yra pakankamai komplikuota, kadangi tai turi neišvengiamą efektą galutiniam rezultatui (t.y. tinklo apmokymui).

Jei pradiniai inicijuojantys svoriai yra maži, tai atbulo sklidimo algoritmas linkęs konverguoti į lokalų minimumą su mažais svoriais.

Jei kaip sustojimo taisyklė naudojamas maksimalus gradientinio nusileidimo žingsnių skaičius, tai tada efektyviai atkertami taškai kelyje (parametrų erdvėje) esantys tolyn nuo pradinio taško (t.y. inicijuojančių parametrų verčių).

Kadangi abu faktoriai (inicijuojantys parametrai ir sustojimo taisyklė) užduoda apribojimus parametrų erdvei, tai jie apjungtai įtakoja ir galutinį sprendimą. [CS98]

Pradinių svorių parinkimas yra tiesiogiai susijęs su lokalių minimumų problema. Sėkmingas tinklo inicijavimas padeda konverguoti į geresnį lokalų minimumą ir tuo pačiu sumažina apmokymo ir testavimo klaidų kiekį. Inicijuojančio svorių vektoriaus tikslus parinkimas yra labai svarbus, nes inicijuojantis svorio vektorius neša informaciją ir ši informacija gali būti panaudota. [RA98]

5.2. Euristikos, pagerinančios atbulo sklidimo algoritmo darbą

Dažnai sakoma, kad NT kūrimas panaudojant atbulo sklidimo algoritmą yra daugiau menas nei mokslas, turint omeny, kad daugybė faktorių yra įtakoti kūrėjo personalinės empirinės patirties. Tačiau, nepaisant to, sekantys metodai gali labai pagerinti atbulo sklidimo algoritmo efektyvumą:

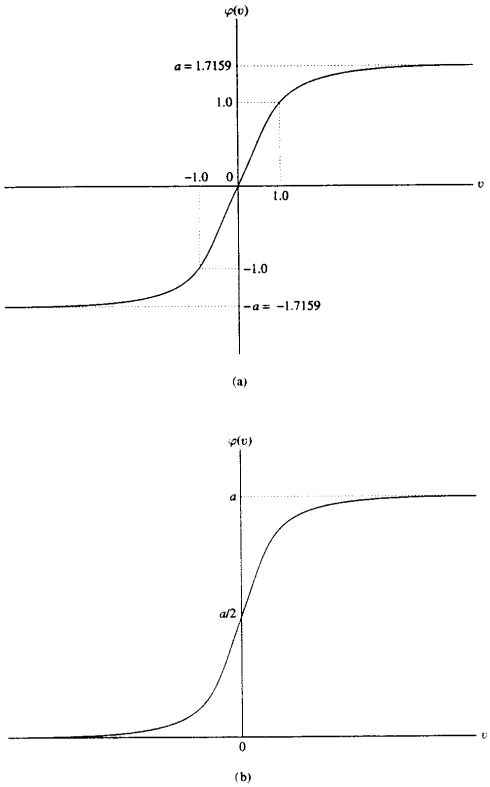
1. *Nuoseklus*[25] *vietoj paketinio*[26] *svorių keitimas*. Atbulo sklidimo algoritme nuoseklus svorių keitimo režimas yra skaičiavimo atžvilgiu greitesnis nei

[25] iš angl. sequential
[26] iš angl. batch

paketinis režimas. Tai ypač pasakytina, kai mokymo duomenų aibė yra didelė ir perteklinė. [Hay98]

2. *Informacijos turinio maksimizavimas*. Kaip taisyklė, kiekvienas mokymo pavyzdys turėtų būti parinktas taip, kad neštų kuo daugiau informacijos duotai užduočiai [LeC93]. Yra du būdai tam pasiekti:
   - Naudoti pavyzdį, kuris duodą didžiausią klaidą išėjime;
   - Naudoti pavyzdį, kuris kardinaliai skiriasi nuo visų ankščiau buvusių.

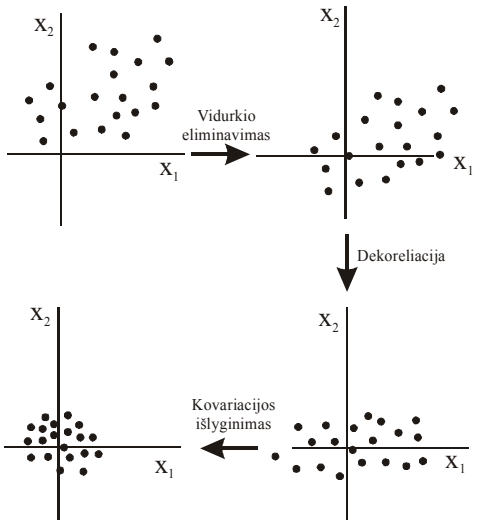
3. *Aktyvacijos funkcija*. NT, mokomas atbulinio sklidimo algoritmu, kaip taisyklė mokosi greičiau (mokymo iteracijų skaičiaus prasme), jei neuronų sigmoidinė funkcija yra antisimetrinė, nei kad kai ji būna nesimetrinė. Sakoma, kad aktyvacijos funkcija $\varphi(v)$ yra antisimetrinė, jei $\varphi(-v) = -\varphi(v)$.
   Ši sąlyga neišlaikoma, jei naudojama standartinė *logsig* funkcija (pavaizduota 42 *b* paveiksle).
   Antisimetrinės aktyvacijos funkcijos pavyzdys būtų hiperbolinio tangento funkcija $\varphi(v) = a\tanh(bv)$, kur *a* ir *b* yra konstantos. Pvz. tinkamos konstantų vertės būtų *a*=1.7159 ir *b*=2/3. [LeC93]
   Tokia hiperbolinio tangento funkcija turi naudingas savybes:
   - $\varphi(1) = 1$ ir $\varphi(-1) = -1$;
   - Antroji $\varphi(v)$ išvestinė turi maksimalią vertę, kai $v = 1$.



42 pav. (a) Antisimetrinė aktyvacijos funkcija. (b) Nesimetrinė aktyvacijos funkcija.

4. *Trokštami išėjimai*. Svarbu, kad trokštami išėjimai būtų pasirinkti sigmoidinės aktyvacijos funkcijos rėžiuose. Tiksliau sakant, d_j trokštamas išėjimas *j* neuronui išėjimo sluoksnyje turi būti pastumtas per dydį ε nuo sigmoidės maksimalios reikšmės (neigiamos arba teigiamos). Kitu atveju atbulinio sklidimo algoritmas linkęs NT laisvus parametrus nustumti į begalybę ir tuo būdu sulėtinti mokymo procesą. Kad būtų aiškiau, panagrinėsime antisimetrinę aktyvacijos funkciją pavaizduotą 42 *a* paveiksle. Ribojančiai $+a$ vertei parenkame $d_j = a - \varepsilon$ ir ribojančiai $-a$ vertei parenkame $d_j = -a + \varepsilon$, kur ε yra atitinkama teigiama konstanta. Pvz. anksčiau paminėtai vertei a=1.7159, galime parinkti ε=0.7159 ir tuo būdu tikslo vertė (siekiamas tikslas) d_j gali būti patogiai pasirinkta kaip ±1, kaip parodyta 42 *a* paveiksle.

5. *Įėjimo normalizavimas*. Kiekvienas įėjimo duomenų vektorius gali būti apdorotas taip, kad vidurkis (visos apmokymo duomenų aibės) būtų arti nulio arba kad jis būtų mažas palyginus su standartiniu nuokrypiu. Norint pagreitinti mokymo procesą, įėjimo normalizavime dar reikia imtis sekančių dviejų priemonių [LeC93]:
    - Įėjimo duomenų vertės turi būti dekoreliuotos;
    - Dekoreliuotų įėjimo verčių mastelis turi būti pakeistas[27] taip, kad *kovariacijos būtų apytiksliai lygios*, tuo būdu užtikrinant, kad NT skirtingi sinapsiniai svoriai mokytųsi maždaug tuo pačiu greičiu.



43 pav. Trys duomenų normalizacijos žingsniai: vidurkio eliminavimas, dekoreliacija ir kovariacijos išlyginimas dviejų matavimų įėjimo erdvei.

[27] iš angl. scaled

6. *Inicijavimas*. Geras inicijuojančių parametrų parinkimas labai įtakoja sėkmingą NT mokymą. Klausimas: kas yra geras parinkimas?
Kai sinapsiniams svoriams yra priskiriamos didelės vertės, tai yra labai tikėtina, kad NT neuronai įsisotins. Kai taip įvyksta, tai atbulo sklidimo algoritmo lokalus gradientas tampa mažas, o tai savo ruoštu stabdo mokymo procesą. Tačiau, jei sinapsiniams svoriams priskirti mažas inicijuojančias vertes, tai atbulo sklidimo algoritmas dirbs ant labai plokščio paviršiaus, tai ypač pasakytina apie antisimetrines aktyvacijos funkcijas, kaip kad hiperbolinio tangento funkcija. Kad būtų aiškiau, išnagrinėkime MLP perceptroną su hiperbolinio tangento aktyvacijos funkcijomis. Tarkime visų neuronų slenksčiai yra 0. Tada *j* neurono indukuotas laukas

$v_j = \sum_{i=1}^{m} w_{ji} y_i$.

Tarkime įėjimai į kiekvieną neuroną turi nulinį vidurkį ir vienetinę dispersiją:
$\mu_y = E[y_i] = 0$ visiems *i,*
ir $\sigma^2 = E[(y_i - \mu_i)^2] = E[y_i^2] = 1$ visiems *i*.
Tarkime kad įėjimai nekoreliuoti:

$$E[y_i y_k] = \begin{cases} 1, & \text{jei } k = i \\ 0, & \text{jei } k \neq i \end{cases}$$

ir kad sinapsiniai svoriai yra tolygiai pasiskirsčiusių skaičių aibė su nuliniu vidurkiu
$\mu_w = E[w_{ji}] = 0$ visoms (*i,j*) poroms,
ir dispersija:
$\sigma_w^2 = E[(w_{ji} - \mu_w)^2] = E[w_{ji}^2]$ visoms (*i,j*) poroms.
Atitinkamai galima išreikšti indukuoto lauko v_j vidurkį ir dispersiją:

$$\mu_v = E[v_j] = E\left[\sum_{i=1}^{m} w_{ji} y_i\right] = \sum_{i=1}^{m} E[w_{ji}][y_i] = 0$$

ir

$$\sigma_v^2 = E[(v_j - \mu_v)^2] = E[v_j^2] = E\left[\sum_{i=1}^{m}\sum_{i=1}^{m} w_{ji} w_{jk} y_i y_k\right] =$$

$$= \sum_{i=1}^{m}\sum_{i=1}^{m} E[w_{ji} w_{jk}] E[y_i y_k] = \sum_{i=1}^{m} E[w_{ji}^2] = m\sigma_w^2,$$

kur *m* yra neurono sinapsinių ryšių skaičius.

Gera sinapsinių svorių inicijavimo strategija yra inicijuoti sinapsinius svorius taip, kad indukuotas lokalus laukas butų tarp aktyvacijos sigmoidės funkcijos tiesinės ir įsotintos dalių. Pvz. anksčiau nusakytoms *a* ir *b* vertėms šis tikslas pasiekiamas, parenkant $\sigma_v = 1$ (paskutinėje lygtyje) ir tada gauname $\sigma_w = m^{-\frac{1}{2}}$. Tai yra palanku tolydiniam pasiskirstymui, iš kurio parenkami sinapsiniai svoriai, kurie turi nulinį vidurkį ir dispersiją, lygią atvirkštiniam dydžiui nuo neurono sinapsinių

ryšių skaičiaus. [LeC93]

7. *Mokymo greičiai*. Idealiai visi MLP neuronai turėtų mokytis vienodu greičiu. Paskutiniai sluoksniai paprastai turi didesnius lokalius gradientus negu pirmieji sluoksniai. Tuo būdu, paskutinių sluoksnių mokymosi greičio parametras η turėtų būti mažesnis nei pirmesnių sluoksnių. Neuronai su daug įėjimų turėtų turėt mažesnę mokymosi greičio vertę, nei kad neuronai su mažai įėjimų, tam kad palaikyti daugmaž vienodą visų neuronų mokymosi greitį. [Hay98]

8. *Mokymas "užuominomis"*. Mokymas pavyzdžiais faktiškai reiškia nežinomos įėjimo-išėjimo projekcijos funkcijos *f(·)* ieškojimą. Faktiškai, mokymo procesas pasinaudoja informacija mokymo pavyzdžiuose apie *f(·)* funkciją, kad dedukuoti funkcijos apytikslę aproksimaciją. Pagerinti rezultatus galima mokymo procese pasinaudojant *a priori* turima informacija apie *f(·)* funkciją. [Mos95]
   Tokia informacija gali būti invariantinės savybės, simetrijos (*a priori* žinomų simetrijų panaudojimas vadinamas *svorių dalinimusi*[28]) arba bet kokios kitos žinios apie *f(·)* funkciją, kurios gali būti panaudotos apytikslės aproksimacijos funkcijos suradimo *pagreitinimui*, ir visų svarbiausia, kad tai *pagerina* galutinį tinklo darbą.

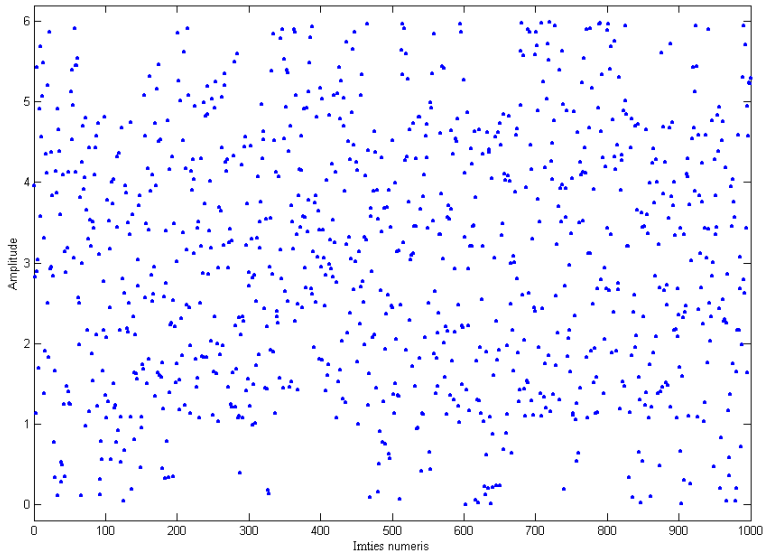
5.3. Kompiuterinis eksperimentas. Evoliucinis NT mokymas kaip dar vienas euristinis metodas.

Generuojamas stačiakampių impulsų signalas, kuris savybės sekančios:

1. Impulso amplitudė gali įgauti reikšmę 0 arba 1;
2. Impulso ilgis yra tarp 20 ir 24, jei signalas lygus 0; impulso ilgis yra tarp 20 ir 26, jei signalo amplitudė lygi 1.

Signalas užteršiamas baltu triukšmu, kurio amplitudė yra 5 kart didesnė už originalų signalą.

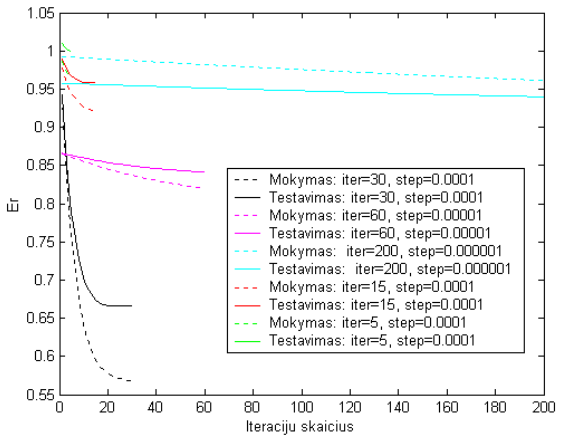
[28] iš angl. weight sharing



44 pav. Signalas, užterštas baltu triukšmu, kurio amplitudė yra 5 kart didesnė už originalų signalą.

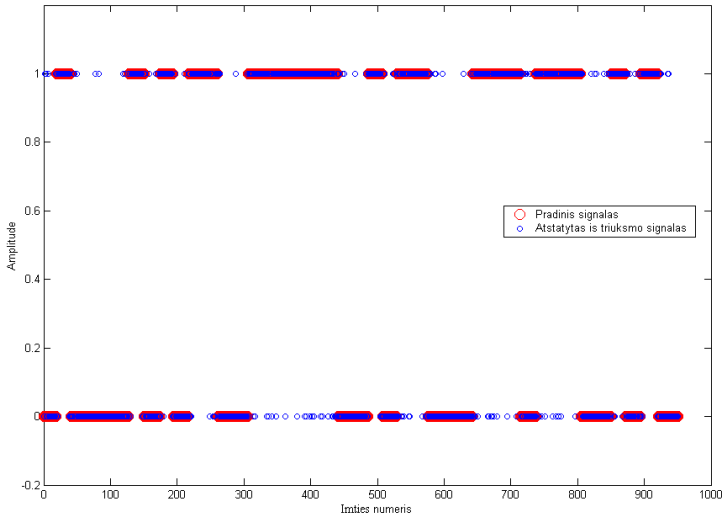
Jei tinklą bandoma mokyti su šiais duomenimis atstatyti originalų signalą, rezultatai būna nesėkmingi, nepriklausomai nuo mokymo parametrų (mokymo žingsnis ir iteracijų skaičius).

Tačiau tinklas gali būti sėkmingai išmokytas, jei jis bus mokomas evoliucinėmis pakopomis, pamažu didinant triukšmą su kiekviena pakopa[29].

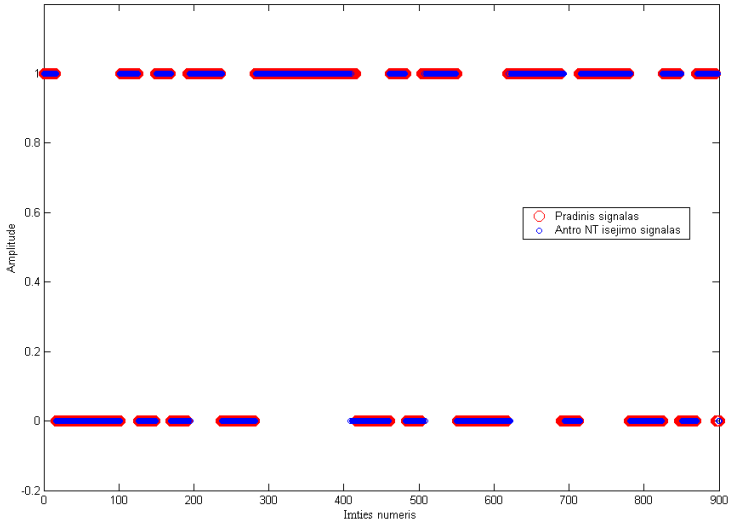


45 pav. Evoliucinis NT mokymas. Kiekvienoje evoliucinėje pakopoje prie pradinio signalo pridedamas triukšmas, kurio amplitudė su kiekviena pakopa vis didesnė. Kiekvienoje pakopoje individualiai parenkami mokymo parametrai. Mokymas kiekvienoje pakopoje stabdomas, kai tik testavimo duomenų klaida pradeda didėti. Pačioje pradžioje NT turi nulinį svorių vektorių. Sekančiose evoliucinėse pakopose NT pradinis svorių vektorius yra lygus praeitos pakopos svorių vektoriui.

[29] prieš tinklo mokymą ir testavimą duomenys buvo specialu būdu apdoroti – normalizuoti.



46 pav. Evoliucinių pakopų metodu išmokyto NT darbo rezultatai. Pavienės imtys yra neteisingai atstatytos, todėl signalo užglotninimui panaudojamas antras NT.



47 pav. Antrojo (užglotninančio) NT darbo rezultatai.

Matome, kad MLP, organizuotas kaip šių dviejų NT junginys ir išmokytas evoliucinėm pakopom gali sėkmingai atstatyti stačiakampių impulsų signalą iš triukšmo, tuo tarpu mokant viena pakopa, tinklą išmokyti nepavyksta.

5.4. Išvados

NT mokymas yra daugiau menas, nei mokslas ir mokymo rezultatai yra labai įtakoti kūrėjo personalinės empirinės patirties. Tačiau tam tikri dėsningumai vis dėlto yra pastebėti ir yra prikurta keletas euristikų, kurios pagerina mokymo proceso darbą. Tačiau

dažnai šių euristikų taikymas praktinių uždavinių sprendimui neduoda trokštamo rezultato, o tai reiškia, kad reikalingi papildomi euristiniai metodai. Tokio praktinio uždavinio pavyzdys yra stačiakampių impulsų signalo atstatymas iš triukšmo. Šiame skyriuje pristatytas dar vienas euristinis metodas, pagrįstas evoliuciniu NT mokymu. Kompiuterinis eksperimentas parodė, kad jei NT bandoma mokyti viena pakopa tai, rezultatai būna nesėkmingi, nepriklausomai nuo mokymo parametrų (mokymo žingsnis ir iteracijų skaičius). Tačiau tinklas gali būti sėkmingai išmokytas, jei bus mokomas evoliucinėmis pakopomis, pamažu didinant triukšmą su kiekviena pakopa ir kiekvienoje pakopoje individualiai parenkant mokymo parametrus. Mokymas kiekvienoje pakopoje stabdomas, kai tik testavimo duomenų klaida pradeda didėti. Pačioje pradžioje NT turi nulinį svorių vektorių. Sekančiose evoliucinėse pakopose NT pradinis svorių vektorius yra lygus praeitos pakopos galutiniam svorių vektoriui.

6. Hibridinis neuroninio tinklo, šalinančio triukšmą iš signalo, mokymo algoritmas (HINMA)

Ankstesniuose skyriuose buvo išnagrinėtos SSA ir neuroninių tinklų teorijos. Jų abiejų integravimo rezultate, šiame skyriuje, idant pagerinti triukšmo pašalinimo efektyvumą, pasiūlytas hibridinis neuroninio tinklo, šalinančio triukšmą iš signalo, mokymo algoritmas, paremtas SSA teorijos panaudojimu dirbtinio neuroninio tinklo mokymui.

Algoritmas pavadintas **HINMA** – **Hi**bridinis **N**euroninio Tinklo, Šalinančio Triukšmą iš Signalo, **M**okymo **A**lgoritmas.

Pasiūlytas algoritmas palygintas su 12 žinomu, šį uždavinį sprendžiančių, algoritmų.

NT su tiesine aktyvacijos funkcija buvo mokytas MVK algoritmu, kuris aprašytas 2.6. poskyryje ("Neuroninių tinklų mokymas"), o NT su netiesine aktyvacijos funkcija buvo mokytas 11 skirtingų standartinių klasikinių algoritmų, kurie svorių modifikavimui naudoja nuostolių funkcijos gradientą. Žemiau pateiktas šių 11 algoritmų trumpas aprašymas:

1) *Gradientinis nusileidimas*.
   Yra daug atbulo sklidimo algoritmo variacijų. Paprasčiausiame variante mokymas keičia svorius ta kryptimi, kuria nuostolių funkcija greičiausiai mažėja. Viena šio algoritmo iteracija užrašoma kaip:
   $x_{k+1} = x_k - \alpha_k g_k$,
   kur x_k yra dabartinis svorių vektorius, g_k – dabartinis gradiento vektorius ir α_k – mokymo greitis.

2) *Gradientinis nusileidimas su inercija*.
   Šis algoritmas konverguoja greičiau, nei paprastas gradientinis nusileidimas, nes inercija leidžia NT reaguoti ne tik į lokalius minimumus, bet ir paskutines tendencijas klaidos paviršiuje. Inercija veikia kaip žemo dažnio filtras ir leidžia NT ignoruoti smulkias klaidos paviršiaus savybes. NT be inercijos gali užstrigti negiliame lokaliame minimume, o NT su inercija jį prašoka.
   Inercija atbulo sklidimo algoritme gali būti pridėta padarant svorių pokyčius lygius daliai praeito svorio pokyčio plius naujas svorių pokytis, kurį siūlo atbulo sklidimo taisyklė.

3) *Adaptyvus mokymo žingsnis*.
   Paprastame gradientiniame nusileidime mokymo greitis, vykstant mokymui, yra konstanta. Algoritmo veikimas labai priklauso nuo šios konstantos parinkimo. Jei mokymo greitis per didelis, algoritmas gali pradėti osciliuoti ir tapti nestabilus. Jei

mokymo greitis per mažas, tai algoritmas labai lėtai konverguos. Praktiškai neįmanoma parinkti optimalų mokymo greitį prieš mokymą, ir faktiškai optimalus mokymo greitis keičiasi mokymo metu, kai algoritmas juda nuostolių funkcijos paviršiumi.
Gradientinio nusileidimo algoritmas gali būti pagerintas leidžiant mokymo greičiui keistis mokymo metu. Adaptyvaus mokymo žingsnio algoritmas stengiasi palaikyti mokymo žingsnį kiek įmanoma didesnį ir tuo pat metu išlaikyti stabilų mokymą. Mokymo žingsnis padaromas priklausomu nuo nuostolių funkcijos paviršiaus sudėtingumo.

4) *Elastingas atbulo sklidimo*.
Daugiasluoksniai tinklai paslėptuose sluoksniuose paprastai naudoja sigmoidinę aktyvacijos funkciją. Šios funkcijos dažnai vadinamos "spaudžiančiomis" funkcijomis, kadangi jos suspaudžia begalinę įėjimo sritį į baigtinę išėjimo sritį. Sigmoidinės funkcijos nuolydis artėja į nulį, kai įėjimas yra didelis. Tai sukelia problemų, kai daugiasluoksniui tinklui mokyti naudojamas gradientinis nusileidimas, kadangi gradientas tampa mažas ir todėl svoriai koreguojami mažais dydžiais, net jei jie yra toli nuo optimalių verčių. Elastingas atbulo sklidimo algoritmas eliminuoja dalinių išvestinių amplitudžių efektą. Svorių koregavimui yra naudojamas tiktais išvestinės ženklas, išvestinės dydis neturi jokios įtakos. Pokyčio dydis yra apsprendžiamas koks buvo išvestinės ženklas prieš tai – jei ženklas buvo toks pats, pokytis didinamas dydžiu Δinc, ir mažinamas dydžiu Δdec, jei ženklas pasikeitė. Jei išvestinė lygi nuliui, pokyčio dydis paliekamas tas pats. Kai svoriai pradeda osciliuoti, pokyčio dydis mažinamas. Jei svoris keičiamas ta pačia kryptimi kelis kartus, tai pokyčio dydis padidinamas.

5) *Fletcher-Reeves jungtinio gradiento algoritmas.*
Visi jungtinio gradiento algoritmai pradeda paiešką pirmoje iteracijoje greičiausio nusileidimo kryptimi (neigiamas gradientas)
$p_0 = -g_0$
Tada ieškoma linija, kuri nusako optimalų judesio atstumą dabartine paieškos kryptimi:
$x_{k+1} = x_k + \alpha_k p_k$
Tada sekanti paieškos kryptis yra padaroma tokia, kad būtų jungtinė ankstesnėms paieškos kryptims. Naujos paieškos krypties bendra procedūra yra apjungti naują greičiausio nusileidimo kryptį su praeita paieškos kryptimi:
$p_k = -g_k + \beta_k p_{k-1}$
Įvairios jungtinio gradiento algoritmo versijos skiriasi tuo, kaip skaičiuojama konstanta β_k . Fletcher-Reeves jungtinio gradiento algoritme ji skaičiuojama sekančiai:
$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}$.

6) *Polak-Ribiére jungtinio gradiento algoritmas*.
Kita jungtinio gradiento algoritmo versija. Kaip ir Fletcher-Reeves jungtinio gradiento algoritme, paieškos kryptis kiekvienoje iteracijoje yra nusakoma:
$p_k = -g_k + \beta_k p_{k-1}$.
Polak-Ribiére algoritme, konstanta β_k yra skaičiuojama sekančiai:
$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{g_{k-1}^T g_{k-1}}.$$

7) *Powell-Beale jungtinio gradiento algoritmas*.
Visuose jungtinio gradiento algoritmuose paieškos kryptis periodiškai nukreipiama neigiamo gradiento kryptimi. Standartinis nukreipimo taškas yra kai iteracijų skaičius tampa lygus tinklo parametrų skaičiui, bet yra kiti nukreipimo metodai, kurie gali padidinti mokymo efektyvumą. Vienas iš tokių yra Powell-Beale metodas. Šiame algoritme, mes perstartuosim jei liko labai mažai ortogonalumo tarp dabartinio gradiento ir praeito gradiento. Testavimas atliekamas su sekančia nelygybe:
$\left| g_{k-1}^T g_k \right| \geq 0.2 \left\| g_k \right\|^2$.
Jei ši sąlyga patenkinama, tai paieškos kryptis yra nukreipiama neigiamo gradiento kryptimi.

8) *Keičiamo mastelio jungtinio gradiento algoritmas*.
Kiekviename jungtinio gradiento algoritme, kuriuos aptarėme, buvo reikalinga linijos paieška kiekvienoje iteracijoje. Ši linijos paieška skaičiavimo atžvilgiu yra brangi, kadangi reikia, kad tinklo atsakas į visus mokymo duomenis būtų suskaičiuotas kelis kartus kiekvienai paieškai. Keičiamo mastelio jungtinio gradiento algoritmas, buvo sukurtas tam, kad išvengti laiką užimančios linijos paieškos. Šis algoritmas per daug sudėtingas, kad aprašyti keliomis eilutėmis, bet pagrindinė idėja yra Levenberg-Marquardt algoritmo irjungtinio gradiento algoritmo ideologijų apjungimas.

9) *BFGS kvazi-Newton metodas*.
Newton metodas yra alternatyva jungtinio gradiento metodams greitai optimizacijai. Pagrindinis Newton metodo žingsnis yra:
$x_{k+1} = x_k - A_k^{-1} g_k$.
kur A_k yra esamų svorių ir slenksčių darbo sėkmingumo įverčio Hessian matrica (antros išvestinės). Newton metodas dažnai konverguoja greičiau nei jungtinio gradiento metodai. Tačiau jis yra sudėtingas ir yra brangu suskaičiuoti neuroninio tinklo Hessian matricą. Yra klasė algoritmų, kurie yra pagrįsti Newton metodu, tačiau nereikalauja antros išvestinės skaičiavimų. Jie vadinami kvazi-Newton (arba kirstinės) metodais. Jie koreguoja apytikslę Hessian matricą kiekvienoje algoritmo iteracijoje. Korekcija yra skaičiuojama kaip gradiento funkcija.

10) *Vieno žingsnio kirstinės metodas*.
Kadangi BFGS algoritmas reikalauja daugiau atminties ir skaičiavimų kiekvienoje iteracijoje, nei jungtinio gradiento algoritmai, tai yra poreikis kirstinės aproksimacijai su mažesniais atminties ir skaičiavimų reikalavimais. Vieno žingsnio kirstinės metodas yra bandymas užpildyti tarpą tarp jungtinio gradiento algoritmų ir kvazi-Newton algoritmų. Šis algoritmas nesaugo pilnos Hessian matricos, jis laiko, kad kiekvienoje iteracijoje, praeita Hessian matrica buvo vienetinė matrica. Šio metodo papildomas privalumas, kad nauja paieškos kryptis gali būti suskaičiuota neskaičiuojant atvirkštinės matricos.

11) *Levenberg-Marquardt algoritmas*.
Kaip ir kvazi-Newton metodas, Levenberg-Marquardt algoritmas buvo sukurtas norint pagreitinti mokymą, neskaičiuojant Hessian matricos.
Kai nuostolių funkcija yra kvadratų suma (kaip kad tipiškai būna NT mokyme), tai Hessian matrica gali būti aproksimuota kaip:
$H = J^T J$
ir gradientas gali būti suskaičiuotas kaip:
$g = J^T e$,
kur J yra Jacobian matrica, kurioje saugomos tinklo klaidų pirmos išvestines svorių ir slenksčių atžvilgiu ir e yra tinklo klaidų vektorius. Jacobian matrica gali būti suskaičiuota standartiniu atbulo sklidimo metodu, kuris yra paprastesnis, nei Hessian matricos skaičiavimas.
Levenberg-Marquardt algoritmas naudoja šią Hessian matricos aproksimaciją į Newton panašioje korekcijoje:
$x_{k+1} = x_k - \left[J^T J + \mu I\right]^{-1} J^T e$.
Kai skaliaras μ lygus nuliui, tai yra tik Newton metodas, naudojantis Hessian matricos aproksimaciją. Kai μ yra didelis, šis metodas tampa gradientiniu nusileidimu su mažu žingsniu. Newton metodas yra greitesnis ir tikslesnis šalia klaidos minimumo, todėl tikslinga pasistumti link Newton'o metodo taip greitai, kaip įmanoma. Tuo būdu μ yra mažinamas po kiekvieno sėkmingo žingsnio (nuostolių funkcijos mažinimas) ir yra didinamas, kai bandomasis žingsnis padidintų nuostolių funkciją. Tuo būdu, nuostolių funkcija visada bus mažinama kiekvienoje algoritmo iteracijoje.

NT mokymui yra sukurta keliolika efektyvių algoritmų, kurie mokymo metu automatiškai (be žmogaus įsikišimo) keičia mokymo greičio (atbulo sklidimo algoritmas) ir kitus parametrus [DB00], mokymui naudoja ne tik klaidos gradientą, bet ir antros eilės nuostolių funkcijos išvestines (kvazi-Newton [DB00]), naudoja efektyviai suskaičiuojamą atvirkštinės antros eilės išvestinių matricos įvertį (Lavenberg-Marquardt), atsižvelgia į poslinkio ir dispersijos problemą. Nors antros eilės nuostolių funkcijos išvestinės panaudojimas labai paspartina tinklo mokymą, bet visų gradientinio pobūdžio algoritmų veikimas labai priklauso nuo pradinių sąlygų ir turi tendenciją patekti į lokalų minimumą. [Hay98]

Tačiau jei neuroninio tinklo paskirtis yra šalinti triukšmą iš signalo, tai kaip buvo parodyta 3.3. poskyryje ("Koreliacija, sąsūka ir Furjė analizė neuroniniuose tinkluose"), neuroninis tinklas dirba panašiai kaip skaitmeninis filtras, ir todėl neuroniniam tinklui galioja panašūs dėsningumai kaip ir skaitmeniniam filtrui (kuriuos aprašo SSA teorija).

O nei vienas iš klasikinių (gradientinio pobūdžio) NT mokymo algoritmų nesinaudoja SSA teorijos dėsningumais.

Šio darbo metu sukurtas HINMA algoritmas, kuris naudoja SSA teorijos rezultatus NT mokymo proceso pagreitinimui ir galutiniam NT darbo pagerinimui.

Eksperimentiškai buvo parodyta, kad HINMA algoritmas padeda išvengti dalies lokalių minimumų ir paspartina NT mokymo procesą, ko naudojant vien tik klasikinius mokymo algoritmus nepavyksta gauti.

Toliau aptarsime HINMA algoritmo veikimo pricipus ir jo pagalba gaunamus rezultatus.

6.1. HINMA algoritmo veikimo prielaidos

HINMA algoritmo veikimo principas pagrįstas šiomis prielaidomis:

1. Kaip buvo parodyta 3.3. poskyryje ("Koreliacija, sąsūka ir Furjė analizė neuroniniuose tinkluose"), jei kiekvienas vienasluoksnio perceptrono išėjimo neuronas turi tokius pat svorius kaip ir kiti išėjimo neuronai, tai vienasluoksnis perceptronas atlieka sąsūkos operaciją, t.y. tą patį, ką ir skaitmeninio filtro branduolys.

2. Kaip buvo parodyta 4.1. poskyryje ("Fazės charakteristikos"), jei laiko srityje impulso atsakas yra pilnai simetriškas vertikalios ašies atžvilgiu, tai dažnio spektre fazė bus visur lygi 0, t.y. skirtingo dažnio sinusoidės praeidamos filtro branduolį, nebus pastumiamos viena kitos atžvilgiu. Todėl, norint filtravimo metu neiškraipyti signalo formos, būtina sąlyga yra filtro branduolio simetriškumas.

3. Kaip buvo parodyta 5.1. poskyryje ("Euristikos, pagerinančios atbulo sklidimo algoritmo darbą", 8 punktas), pagerinti rezultatus neuroninio tinklo mokymo procese galima pasinaudojant *a priori* turima informacija apie $f(\cdot)$ funkciją. Tokia informacija gali būti invariantinės savybės, simetrijos arba bet kokios kitos žinios apie $f(\cdot)$ funkciją, kurios gali būti panaudotos apytikslės aproksimacijos funkcijos suradimo *pagreitinimui*, ir visų svarbiausia, kad tai *pagerina* galutinį tinklo darbą.

4. Kaip buvo parodyta 1.1. poskyryje ("Autokoreliacijos ir tarpusavio koreliacijos savybės"), sekos sąsūka su savo veidrodiniu atspindžiu yra autokoreliacinė seka, kuri turi savybę
   $r_{xx}[l] = r_{xx}[-l]$ (lyginė funkcija).
   T.y. sekos sąsūka su savo veidrodiniu atspindžiu yra simetriška.

5. Kaip buvo parodyta 4.6. poskyryje (“Tiesiniai skaitmeniniai filtrai”), kadangi filtro branduolys yra baigtinio ilgio (*N* imčių), tai padarius dažninio atsako atvirkštinę Furjė transformaciją begalinio ilgio atsakas susisuka i *N* imčių atsaką. Tačiau, padauginimas iš lango funkcijos stipriai sumažina ši susisukimą. Taip pat padauginimas iš lango funkcijos susiaurina filtro branduolio dažninį spektrą, kuris išplatėja dėl to, kad filtro branduolys yra baigtinio ilgio.

6.2. HINMA algoritmo aprašymas

Žemiau pateiktas algoritmo aprašymas, kuris iliustruoja esminius algoritmo veikimo principus.

1. Vienasluoksnis perceptronas mokomas įprastiniais standartiniais metodais (pvz. NT su tiesine aktyvacijos funkcija mokomas MVK algoritmu, NT su netiesine aktyvacijos funkcija – bet kuriuo gradientinio pobūdžio algoritmu);

2. Padaroma gautų vienasluoksnio perceptrono svorių sąsūka su savo veidrodiniu atspindžiu (jei vienasluoksnis perceptronas turėjo *N* įėjimų, tai po sąsūkos su savo veidrodiniu atspindžiu gaunama simetriška *2·N-1* imčių ilgio sąsūkos seka);

3. Iš gautos sąsūkos sekos paimamos imtys nuo $\frac{N}{4}$ iki $\frac{3}{4}N$, o likusios atmetamos;

4. Likusi *N* imčių ilgio seka padauginama iš lango funkcijos (kuo vėliau pritaikomas HINMA algoritmas, tuo labiau lango funkcija turi tolti nuo Bertlett lango[30] link stačiakampio lango[31]);

5. Gautoji *N* imčių ilgio seka ir yra naujieji vienasluoksnio perceptrono svoriai.

6.3. Kompiuterinis eksperimentas. HINMA algoritmo efektyvumo demonstracija

Sugeneruojami 600 imčių ilgio mokymo ir testavimo signalai – prie ieškomo signalo (dviejų sinusoidžių suma) pridėdamas baltas triukšmas, kurio amplitudė viršija signalo amplitudę 5 kartus.

Nagrinėjamas ir lyginamas filtravimas dviem metodais:

1. Vienasluoksniu perceptronu su tiesine aktyvacijos funkcija (kuris mokomas MVK algoritmu);

[30] Bertlett langas yra paprasčiausias trikampis

[31] stačiakampio lango funkcija faktiškai yra tas pats, kas netaikymas jokios lango funkcijos

2. Vienasluoksniu perceptronu su netiesine aktyvacijos funkcija (kuris mokomas 11 skirtingų standartinių klasikinių algoritmų, kurie svorių modifikavimui naudoja nuostolių funkcijos gradientą).

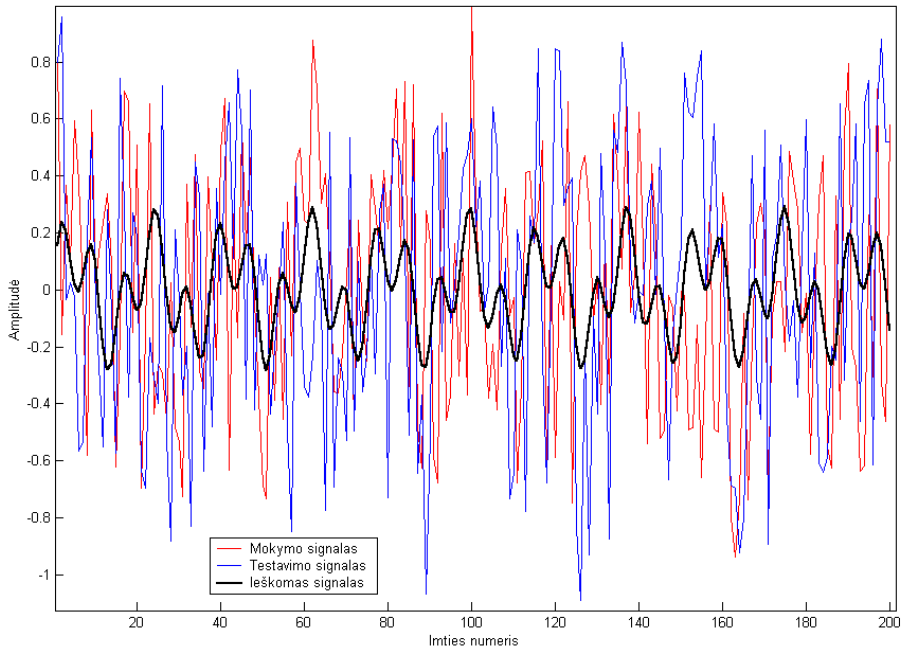
Bei paraleliai lyginama su optimaliu tiesiniu filtru.

Eksperimente kaip lango funkcija naudotas Hamming'o langas.

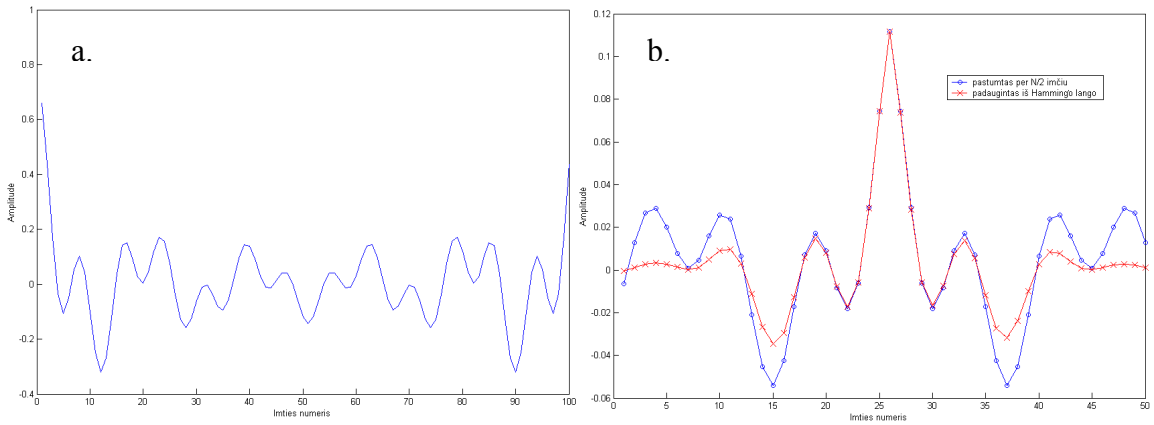
Kad geriau įvertinti HINMA algoritmo efektyvumą, eksperimentas atliekamas dviems etapais:

1. imama dalis sugeneruoto signalo (1/3 signalo, t.y. 200 pirmų imčių) ir atliekamas mokymas ir testavimas su šiais duomenim;
2. imamas visas sugeneruotas signalas (600 imčių) ir atliekamas mokymas ir testavimas su šiais duomenim.

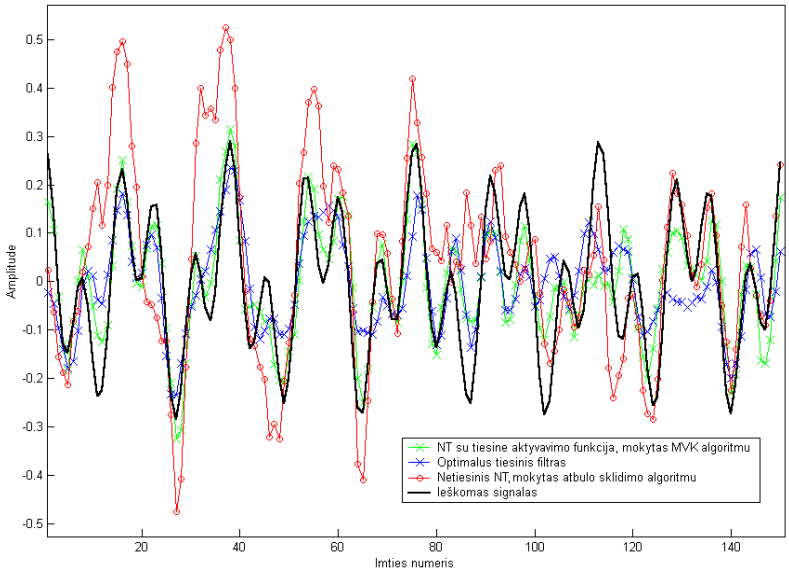
1 etapas. Eksperimentas su 1/3 signalo, t.y. 200 pirmų imčių.



48 pav. Ieškomas signalas yra dviejų sinusoidžių su periodais 13.3 ir 33.4 suma. Prie signalo pridėtas baltas triukšmas, kurio amplitudė viršija signalo amplitudę 5 kartus. Mokymo ir testavimo signaluose prie originalaus signalo pridėtas skirtingas baltas triukšmas.

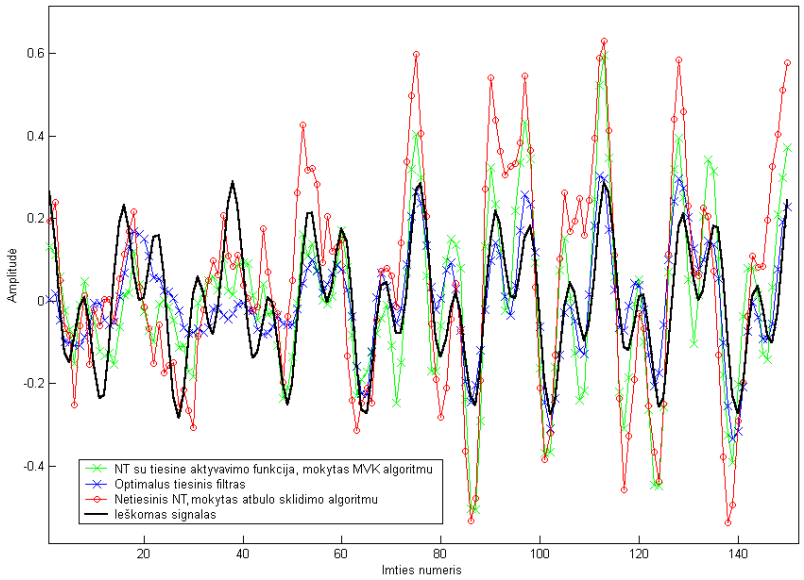


49 pav. Optimalaus tiesinio filtro sudarymas. Imama ieškomo signalo dažninės srities atvirkštinė Furjė transformacija (a pav.), po to gautas impulso atsakas pastumiamas per *N/2* imčių, apkarpomas, ir padauginamas iš Hamming'o lango (b pav.).

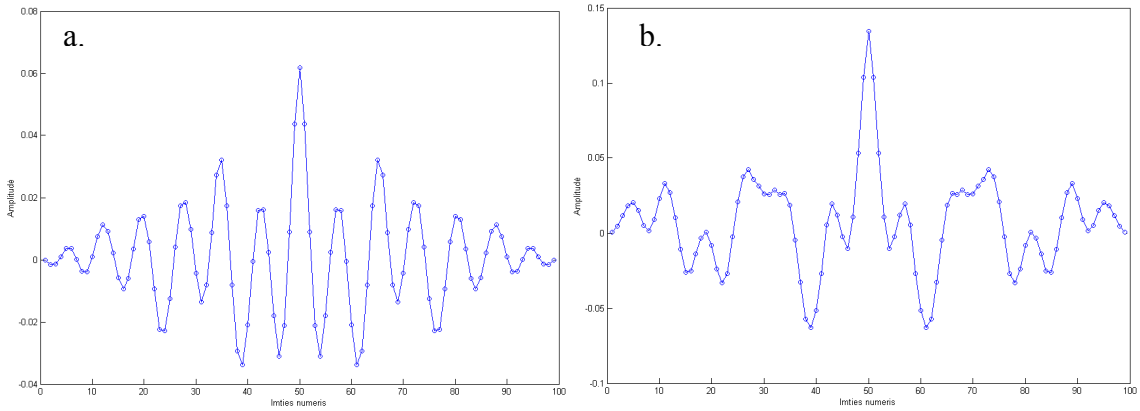


50 pav. Originalaus signalo atstatymas iš balto triukšmo (*mokymo* signalas) prieš taikant HINMA algoritmą. NT su tiesine aktyvacijos funkcija, NT su netiesine[32] aktyvacijos funkcija ir optimalaus tiesinio filtro darbo rezultatai.

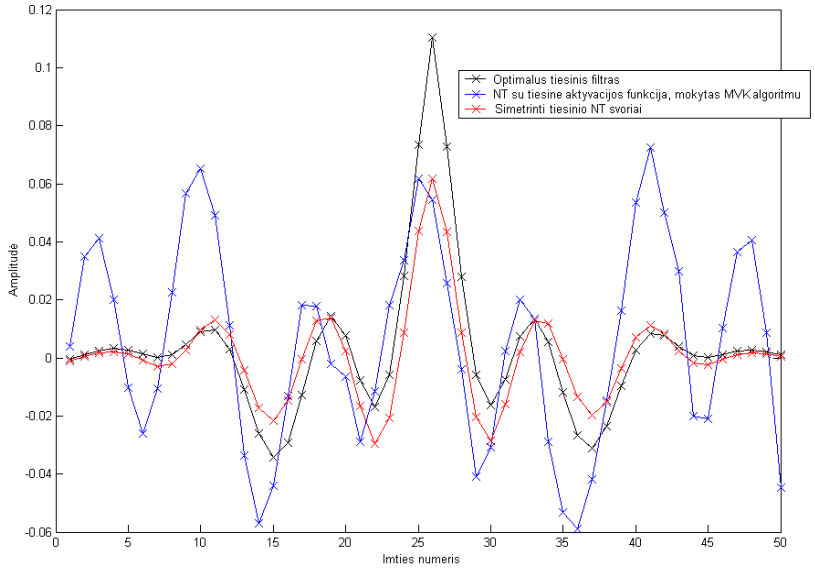
[32] Pateiktuose paveiksluose vaizduojamas NT su netiesine aktyvacijos funkcija, kuris mokytas atbulo sklidimo algoritmu. NT su netiesine aktyvacijos funkcija, mokintų likusiais 10 klasikinių algoritmų, rezultatai pateikti lentelėje ir grafiškai nepavaizduoti, kadangi jie labai panašūs į atbulo sklidimo algoritmu mokinto NT rezultatus.



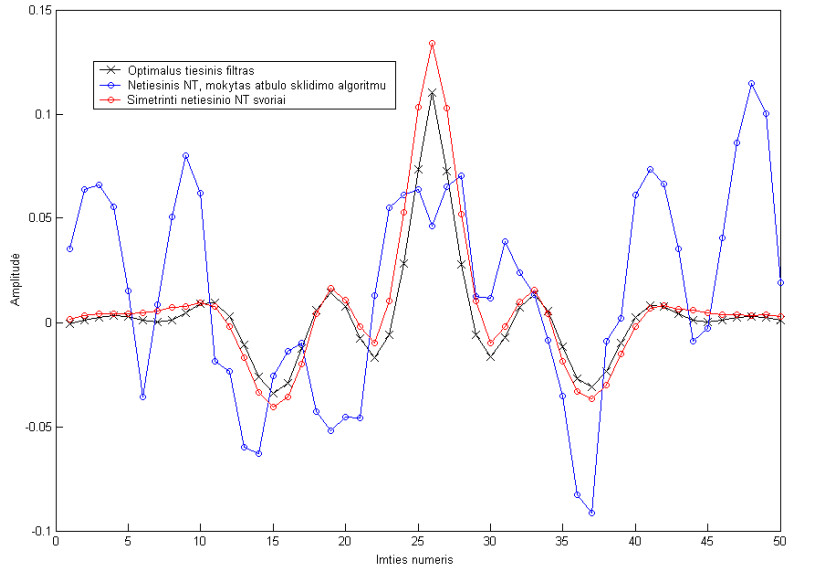
51 pav. Originalaus signalo atstatymas iš balto triukšmo (*testavimo* signalas) prieš taikant HINMA algoritmą. NT su tiesine aktyvacijos funkcija, NT su netiesine aktyvacijos funkcija ir optimalaus tiesinio filtro darbo rezultatai.



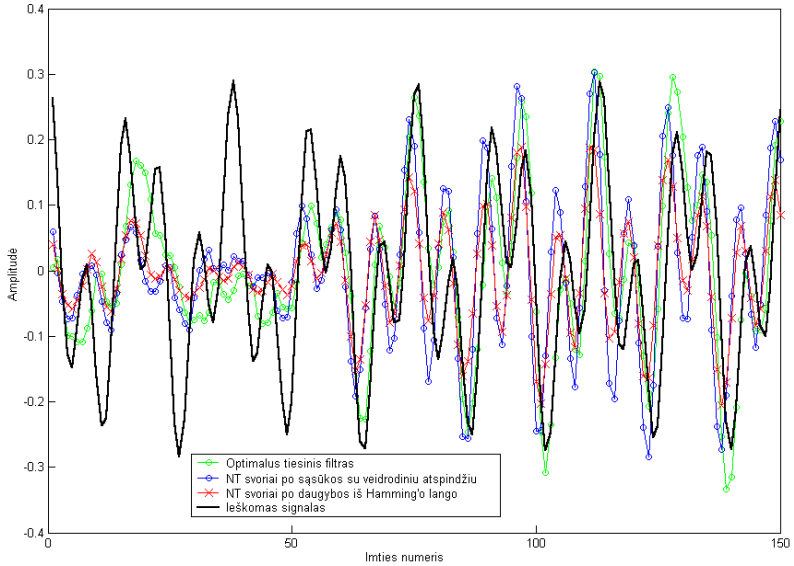
52 pav. Vienasluoksnio perceptrono svorių sąsūka su savo veidrodiniu atspindžiu – pirmasis HINMA algoritmo etapas.
a pav. parodyta vienasluoksnio perceptrono su *tiesine* aktyvacijos funkcija svorių sąsūkos su savo veidrodiniu atspindžiu seka.
b pav. parodyta vienasluoksnio perceptrono su *netiesine* aktyvacijos funkcija svorių sąsūkos su savo veidrodiniu atspindžiu seka.



53 pav. Vienasluoksnio perceptrono su *tiesine* aktyvacijos funkcija svoriai prieš HINMA algoritmo taikymą (po mokymo MVK algoritmu) ir po HINMA algoritmo taikymo (simetrinti svoriai). Taip pat parodyti optimalaus tiesinio (Wiener'io) filtro branduolio koeficientai. Matyti, kad HINMA algoritmas svorius padaro panašesnius į optimalaus tiesinio filtro koeficientus.
Po HINMA algoritmo taikymo NT šalina triukšmą 3.45 dB efektyviau nei kad vien po mokymo MVK algoritmu.



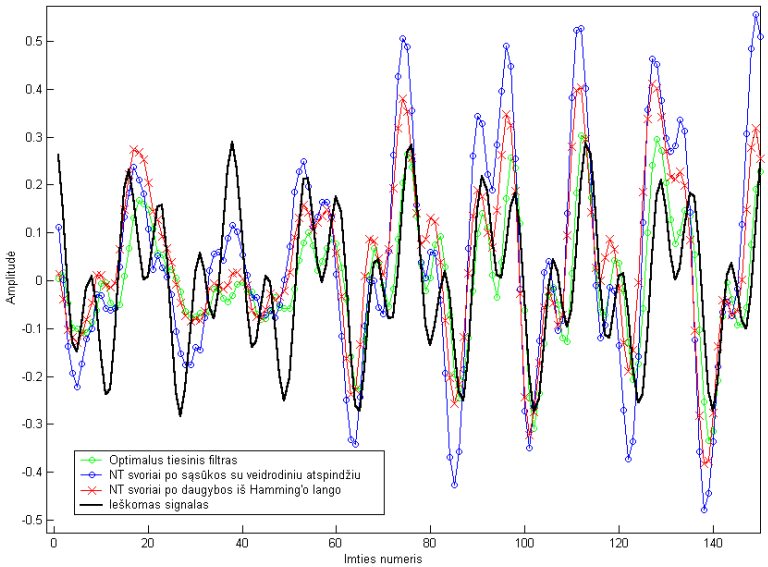
54 pav. Vienasluoksnio perceptrono su *netiesine* aktyvacijos funkcija svoriai prieš HINMA algoritmo taikymą (po mokymo atbulo sklidimo algoritmu) ir po HINMA algoritmo taikymo (simetrinti svoriai). Taip pat parodyti optimalaus tiesinio (Wiener'io) filtro branduolio koeficientai. Matyti, kad HINMA algoritmas svorius padaro panašesnius į optimalaus tiesinio filtro koeficientus.
Po HINMA algoritmo taikymo NT šalina triukšmą 1.99 dB efektyviau nei kad vien po mokymo atbulo sklidimo algoritmu.



55 pav. HINMA algoritmo etapų eksperimentinė analizė. NT su *tiesine* aktyvacijos funkcija. Signalo filtravimas iš testinio signalo.

Padarius NT svorių sąsūka su savo veidrodiniu atspindžiu, NT pašalino triukšmą 2.55 dB efektyviau nei kad NT prieš sąsūką (lyginant su NT vien tik mokytu MVK algoritmu). Tačiau jei naujieji svoriai dar padauginami iš Hamming'o lango, tai efektyvumas išauga iki 3.45 dB (lyginant su NT vien tik mokytu MVK algoritmu).

T.y. svorių padauginimas iš Hamming'o lango duodą žymų teigiamą efektą.

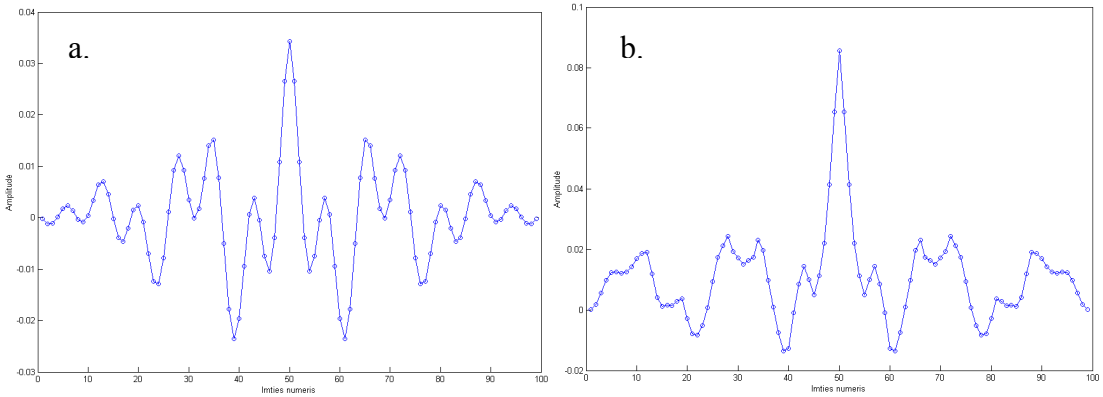


56 pav. HINMA algoritmo etapų eksperimentinė analizė. NT su *netiesine* aktyvacijos funkcija. Signalo filtravimas iš testinio signalo.

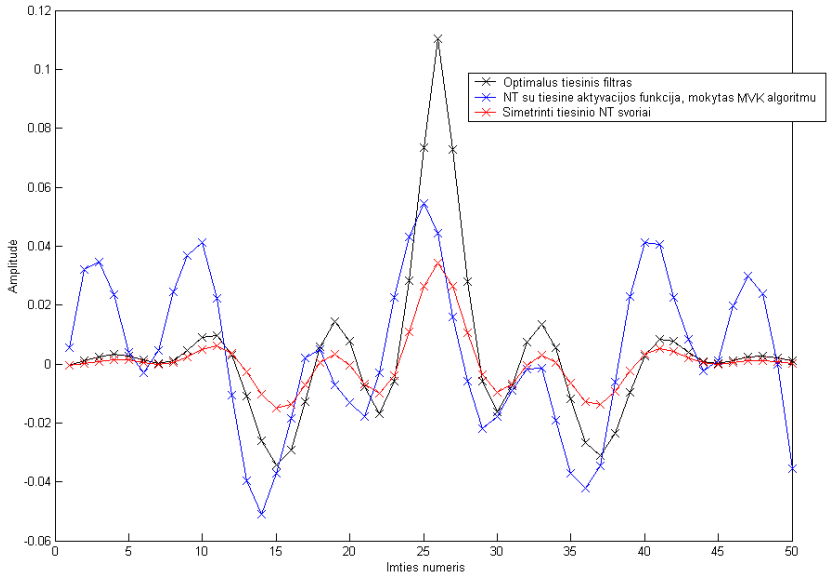
Padarius NT svorių sąsūka su savo veidrodiniu atspindžiu, NT pašalino triukšmą 0.22 dB efektyviau nei kad NT prieš sąsūką (lyginant su NT vien tik mokytu atbulo sklidimo algoritmu).

Tačiau jei naujieji svoriai dar padauginami iš Hamming'o lango, tai efektyvumas išauga iki 1.99 dB (lyginant su NT vien tik mokytu atbulo sklidimo algoritmu).
T.y. svorių padauginimas iš Hamming'o lango duodą žymų teigiamą efektą.

2 etapas. Eksperimentas su visu signalu, t.y. 600 imčių.

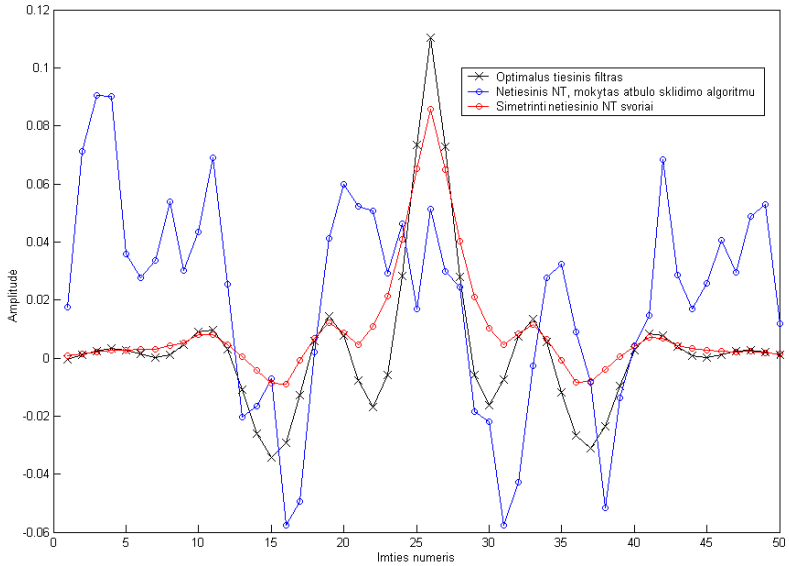


57 pav. Vienasluoksnio perceptrono svorių sąsūka su savo veidrodiniu atspindžiu – pirmasis HINMA algoritmo etapas.
a pav. parodyta vienasluoksnio perceptrono su *tiesine* aktyvacijos funkcija svorių sąsūkos su savo veidrodiniu atspindžiu seka.
b pav. parodyta vienasluoksnio perceptrono su *netiesine* aktyvacijos funkcija svorių sąsūkos su savo veidrodiniu atspindžiu seka.



58 pav. Vienasluoksnio perceptrono su *tiesine* aktyvacijos funkcija svoriai prieš HINMA algoritmo taikymą (po mokymo MVK algoritmu) ir po HINMA algoritmo taikymo (simetrinti svoriai). Taip pat parodyti optimalaus tiesinio (Wiener'io) filtro branduolio koeficientai. Matyti, kad HINMA algoritmas svorius padaro panašesnius į optimalaus tiesinio filtro koeficientus.

Po HINMA algoritmo taikymo NT šalina triukšmą 2.71 dB efektyviau nei kad vien po mokymo MVK algoritmu.



59 pav. Vienasluoksnio perceptrono su *netiesine* aktyvacijos funkcija svoriai prieš HINMA algoritmo taikymą (po mokymo atbulo sklidimo algoritmu) ir po HINMA algoritmo taikymo (simetrinti svoriai). Taip pat parodyti optimalaus tiesinio (Wiener'io) filtro branduolio koeficientai. Matyti, kad HINMA algoritmas svorius padaro panašesnius į optimalaus tiesinio filtro koeficientus.

Po HINMA algoritmo taikymo NT šalina triukšmą 2.37 dB efektyviau nei kad vien po mokymo atbulo sklidimo algoritmu.

Žemiau pateikti keleto eksperimentų su NT su tiesine aktyvacijos funkcija rezultatai.

1 lentelė. HINMA algoritmo efektyvumo eksperimentinis tyrimas. NT su *tiesine* aktyvacijos funkcija mokytas MVK algoritmu. Lentelėje parodyta, kiek efektyviau NT šalino triukšmą po HINMA algoritmo taikymo, nei kad netaikius HINMA algoritmo.

Bandymo Nr.	Po sąsūkos su savo veidrodiniu atspindžiu (200 imčių signalas), dB	Po daugybos iš Hamming'o lango (200 imčių signalas), dB	Po sąsūkos su savo veidrodiniu atspindžiu (600 imčių signalas), dB	Po daugybos iš Hamming'o lango (600 imčių signalas), dB	Triukšmo šalinimo efektyvumas pritaikius HINMA algoritmą po mokymo su 200 imčių signalu ir netaikius HINMA algoritmo, bet mokant su 600 imčių signalu. Santykis kartais
1	5.0705	6.3187	5.9676	6.0573	0.5736
2	2.7838	2.8412	3.6809	2.5799	0.6340
3	3.0568	3.3871	3.9539	3.1257	0.4962
4	2.9811	3.6230	3.8782	3.3616	0.6194
5	2.9811	3.6230	3.8782	3.3616	0.6032
6	2.5486	2.9701	3.4457	2.7088	0.5620
7	5.3179	7.1919	6.6624	6.4334	0.7010
8	3.5760	3.3661	4.3354	3.3005	0.6783
9	3.3498	3.6028	4.7171	3.5322	0.6311
10	3.6057	4.4302	4.5616	3.7441	0.4784
11	3.2607	3.6924	4.2258	3.4078	0.5124
12	3.3494	3.9571	3.6453	2.7993	0.6312
Vidurkis	**3.490117**	**4.083633**	**4.412675**	**3.701017**	**0.5934**

2 lentelė. HINMA algoritmo efektyvumo eksperimentinis tyrimas. NT su *netiesine* aktyvacijos funkcija mokytas 11 skirtingų standartinių klasikinių algoritmų, kurie svorių modifikavimui naudoja nuostolių funkcijos gradientą. Lentelėje parodyta, kiek efektyviau NT šalino triukšmą po HINMA algoritmo taikymo, nei kad netaikius HINMA algoritmo.

	Po sąsūkos su savo veidrodiniu atspindžiu (200 imčių signalas), dB	Po daugybos iš Hamming'o lango (200 imčių signalas), dB	Po sąsūkos su savo veidrodiniu atspindžiu (600 imčių signalas), dB	Po daugybos iš Hamming'o lango (600 imčių signalas), dB	Triukšmo šalinimo efektyvumas pritaikius HINMA algoritmą po mokymo su 200 imčių signalu ir netaikius HINMA algoritmo, bet mokant su 600 imčių signalu. Santykis kartais
Gradientinis nusileidimas[33]	0.8633	1.5852	-0.4178	0.8292	1.1357
Gradientinis nusileidimas su inercija[34]	0.8590	1.8329	-2.27E-04	0.4217	1.2158
Adaptyvus mokymo žingsnis[35]	0.9178	1.8859	0.1237	0.2991	1.2383
Elastingas atbulo sklidimo[36]	-1.7565	1.0831	2.6165	4.0447	1.5557
Fletcher-Reeves jungtinio gradiento algoritmas[37]	-0.2066	1.8339	2.6112	2.2607	1.4058
Polak-Ribiére jungtinio gradiento algoritmas[38]	-0.2066	1.8339	2.6112	2.2607	1.4058
Powell-Beale jungtinio gradiento	-0.2066	1.8339	2.6112	2.2607	1.4058

[33] iš angl. Basic gradient descent
[34] iš angl. Gradient descent with momentum
[35] iš angl. Adaptive learning rate
[36] iš angl. Resilient backpropagation
[37] iš angl. Fletcher-Reeves conjugate gradient algorithm
[38] iš angl. Polak-Ribiére conjugate gradient algorithm

algoritmas [39]					
Keičiamo mastelio jungtinio gradiento algoritmas [40]	-0.0447	1.9629	3.0298	2.7833	1.5421
BFGS kvazi-Newton metodas[41]	-0.1929	1.8175	3.0426	2.8266	1.6023
Vieno žingsnio kirstinės metodas[42]	-0.1929	1.8175	3.0426	2.8266	1.6023
Levenberg-Marquardt algoritmas[43]	0.2199	1.9856	2.6826	2.3737	1.5831

[39] iš angl. Powell-Beale conjugate gradient algorithm
[40] iš angl. Scaled conjugate gradient algorithm
[41] iš angl. BFGS quasi-Newton method
[42] iš angl. One step secant method
[43] iš angl. Levenberg-Marquardt algorithm

Taip pat buvo palygintas NT triukšmo sumažinimo efektyvumas mokant NT su 1/3 duomenų ir pritaikius HINMA algoritmą, bei mokant su visais duomenim, bet netaikant HINMA algoritmo.

Eksperimento rezultatai rodo, kad NT su *tiesine* aktyvacijos funkcija, po mokymo su 3 kart didesniu mokymo vektorių skaičiumi, šalino triukšmą efektyviau, nei kad pritaikius HINMA algoritmą po 1/3 mokymo vektorių aibės.

Taip yra todėl, kad kaip parodyta 2.6. poskyryje ("Neuroninių tinklų mokymas"), NT su tiesine aktyvacijos funkcija, turi nuostolių funkcijos vieną globalų minimumą, silpną minimumą arba jokio minimumo, priklausomai nuo mokymo duomenų vektorių. NT su tiesine aktyvacijos funkcija neturi lokalių nuostolių funkcijos minimumų ir tinklo svoriai asimptotiškai artėja link Wiener'io filtro, kai mokymo duomenų vektorių skaičius auga į begalybę. [Hay98] T.y. NT pats pakankamai sėkmingai artėja link globalaus minimumo.

Tačiau nesvarbu, ankščiau ar vėliau pritaikomas HINMA algoritmas, NT triukšmo šalinimo efektyvumas vis vien *padidinamas*.

Taip pat rezultatai rodo, kad jei HINMA algoritmas pritaikomas vėliau, tai padauginimas iš Hamming'o lango nebeduoda teigiamo efekto, o netgi šiek tiek sumažina efektyvumą. Faktiškai tai reiškia, kad kuo vėliau pritaikomas algoritmas, tuo labiau lango funkcija HINMA algoritme turi tolti nuo Bertlett lango link stačiakampio lango.

Eksperimento rezultatai rodo, kad HINMA algoritmas nors duoda teigiamą efektą NT su tiesine aktyvacijos funkcija, tačiau jis yra efektyvesnis neuroniniams tinklams su *netiesine* aktyvacijos funkcija. NT su *netiesine* aktyvacijos funkcija buvo mokytas 11 skirtingų standartinių klasikinių algoritmų, kurie svorių modifikavimui naudoja nuostolių funkcijos gradientą, ir *visais atvejais* HINMA algoritmas pagerino NT triukšmo šalinimo efektyvumą, bei HINMA algoritmo pritaikymas po 1/3 mokymo duomenų davė geresnius rezultatus, nei mokymas klasikiniais algoritmais, tačiau su visais duomenim. Taip yra todėl, kad NT su netiesine aktyvacijos funkcija turi lokalių nuostolių funkcijos minimumų, į kuriuos papuola klasikiniai mokymo algoritmai (gradientinių algoritmų taikymas labai komplikuotas dėl lokalių minimumų problemos, bet jie yra efektyvūs lokalioje aplinkoje).

Skirtingai nuo tiesinių skaitmeninių filtrų, netiesiniams filtrams nėra žinomų optimalių sprendinių, todėl griežtai kalbant tiesiogiai lyginti netiesinį NT su Wiener'io filtru negalima, tačiau galima empiriškai teigti, kad netiesinis optimalus filtras yra panašus į Wiener'io filtrą. HINMA algoritmas netiesinio NT svorius padaro panašius į Wiener'io filtro branduolio koeficientus ir eksperimentiniai duomenys rodo, kad HINMA algoritmas "permeta" NT jei į ne globalų minimumą, tai bent jau į "geresnį" lokalų minimumą.

6.4. Išvados

Šio darbo metu buvo sukurtas naujas algoritmas HINMA, su šiuo algoritmu atlikti eksperimentai parodė, kad jei neuroninio tinklo paskirtis yra šalinti triukšmą iš signalo, tai:

1. HINMA algoritmas pagerina neuroninių tinklų tiek su tiesine, tiek ir netiesine aktyvacijos funkcija, darbą, nepriklausomai kokiais klasikiniais gradientiniais

algoritmais jie buvo mokyti.
NT su tiesine aktyvacijos funkcija buvo mokytas MVK algoritmu, o NT su netiesine aktyvacijos funkcija buvo mokytas 11 skirtingų standartinių klasikinių algoritmų, kurie svorių modifikavimui naudoja nuostolių funkcijos gradientą, ir *visiems 12-kai algoritmų* HINMA algoritmas pagerino NT triukšmo šalinimo efektyvumą.

2. HINMA algoritmas yra efektyvesnis neuroniniams tinklams su netiesine aktyvacijos funkcija, nei kad neuroniniams tinklams su tiesine aktyvacijos funkcija, kadangi NT su tiesine aktyvacijos funkcija, turi nuostolių funkcijos vieną globalų minimumą, silpną minimumą arba jokio minimumo ir todėl NT pats sėkmingai prie jo artėja, tuo tarpu, kai NT su netiesine aktyvacijos funkcija, mokymo metu gali užstrigti lokaliam minimume. HINMA algoritmas "permeta" netiesinį NT jei į ne globalų minimumą, tai bent jau į "geresnį" lokalų minimumą.
3. HINMA algoritmas *visiems 12-kai algoritmų* duoda teigiamą efektą nepriklausomai nuo to, kada (ankščiau ar vėliau) pritaikomas.
   Kuo vėliau pritaikomas HINMA algoritmas, tuo labiau lango funkcija HINMA algoritme turi tolti nuo Bertlett lango link stačiakampio lango.
4. HINMA algoritmo efektyvumą galima paaiškinti tuo, kad jis išnaudoja skaitmeninių signalų apdorojimo teorijos dėsningumus, o klasikiniai gradientiniai algoritmai į juos neatsižvelgia.

Rezultatų apibendrinimas ir išvados

1. Šio darbo metu buvo sukurtas naujas hibridinis neuroninio tinklo, šalinančio triukšmą iš signalo, mokymo algoritmas (HINMA), kuris padeda panaudoti papildomą *a priori* informaciją (simetriją), gaunamą iš SSA teorijos, neuroninio tinklo mokymo efektyvumui pagerinti.
   Kompiuteriniu eksperimentu parodyta, kad HINMA algoritmas gali pagerinti 12-os skirtingų standartinių klasikinių mokymo algoritmų darbą.
2. HINMA algoritmas yra efektyvesnis neuroniniams tinklams su netiesine aktyvacijos funkcija, nei kad neuroniniams tinklams su tiesine aktyvacijos funkcija, kadangi NT su tiesine aktyvacijos funkcija, turi nuostolių funkcijos vieną globalų minimumą, silpną minimumą arba jokio minimumo ir todėl NT pats sėkmingai prie jo artėja, tuo tarpu, kai NT su netiesine aktyvacijos funkcija, mokymo metu gali užstrigti lokaliam minimume. HINMA algoritmas “permeta” netiesinį NT jei į ne globalų minimumą, tai bent jau į “geresnį” lokalų minimumą.
3. Aparatūrinė neuroninių tinklų realizacija įgalintų atlikti *ultraspartų* skaitmeninių signalų apdorojimą.

Summary

Artificial neural networks are used for interpretation, prediction, diagnosis, planning, monitoring, debugging, repair, instruction, control, categorization and pattern recognition. However, the performance of neural network is reduced due the noise that is usually present in data.

Reducing noise in signals remains one of the biggest challenges in broad range of fields: communications, medical imaging, radar & sonar, high fidelity music reproduction, and oil processing, to name just a few.

This work investigates how to apply artificial neural networks for noise reduction in the signals via integrating digital signal processing and neural networks approaches.

The new algorithm to speed up the learning process of neural network which filter out noise from the signal has been proposed based on the ideas taken from digital signal processing theory.

Literatūra

[Bri00] Encyclopedia Britannica, Inc. , CD-ROM , 2000.

[Bro00] Joseph D. Bronzino. *The Biomedical Engineering Handbook, Second Edition*. CRC Press LLC, 2000.

[CS98] Vladimir Cherkassky, Robert Shepherd. Regularization Effect of Weight Initialization in Back Propagation Networks. *1998 IEEE World Congress on Computational Intelligence. Proceedings of IJCNN '98 FUZZ-IEEE '98 ICEC '98.*

[DBU00] *DSP Blockset User's Guide Version 4.0*. The MathWorks, Inc, 2000.

[DB00] Howard Demuth, Mark Beale. *Neural Network Toolbox User's Guide Version 4.0*. The MathWorks, Inc, 2000.

[DHS01] Richard O. Duda, Peter E. Hart, David G. Stork. *Pattern Classification*. John Wiley & Sons , Inc. 2001.

[Dor00] Richard C. Dorf, *The Electrical Engineering Handbook*. CRC Press LLC, 2000.

[FDT00] *Filter Design Toolbox User's Guide Version 3.0*. The MathWorks, Inc, 2000.

[Hay98] Simon Haykin. *Neural Networks: a comprehensive foundation*. Prentice Hall, 1998.

[HS96] Hann T.H. & Steurer E. Much ado about nothing? Exchange rate forecasting: Neural Networks vs. linear models using monthly and weekly data. *Neurocomputing*, 10, pp. 323-339, 1996.

[NH98] Akiko Nakashima, Akira Hirabayashi and Hidemitsu Ogawa. Noise Suppression in Training Data for Improving Generalization. *1998 IEEE World Congress on Computational Intelligence. Proceedings of IJCNN '98 FUZZ-IEEE '98 ICEC '98.*

[Kem96] Povilas Kemėšis. *Kalbos signalo algoritmai*. Technologija, Kaunas, 1996.

[Kes00] Walt Kester. *Mixed-signal and DSP Design Techniques*. Analog Devices Inc, 2000.

[Lap00] Ed. Phillip A. Laplante. *Electrical Engineering Dictionary*. CRC Press LLC, 2000.

[LeC93] LeCun, Y. *Efficient Learning and Second-order Methods. A Tutorial at NIPS 93*. Denver, 1993.

[Mos95] Abu Mosfata, Y. S. Hints. *Neural computation*, vol. 7, pp. 639-671, 1995.

[PM96] John G. Proakis, Dimitris G. Manolakis. *Digital Signal Processing, Principles, Algorithms, and Applications*. Prentice Hall, 1996.

[RA98] Šarunas Raudys, Shun-ichi Amari. Effect of Initial Values in Simple Perception. *1998 IEEE World Congress on Computational Intelligence. Proceedings of IJCNN '98 FUZZ-IEEE '98 ICEC '98.*

[Rau01] Šarūnas Raudys. *Statistical and Neural Classification Algorithms*. Springer, London, 2001.

[Rud98] Vytautas Evaldas Rudžionis. *Kalbos atpažinimas fonetinių vienetų pagrindu*. Kaunas, 1998.

[RWR97] Fred Rieke, David Warland, Rob de Ruyter van Steveninck, Willliam bialek. *Spikes. Exploring the neural code*. The MIT press, Cambridge, Massachusetts, 1997.

[Smi99] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, San Diego, California, 1999.

[SPT00] *Signal Processing Toolbox User's Guide Version 5.0*. The MathWorks, Inc, 2000.

Priedai

1. Neuroninis tinklas, atliekantis diskrečią Furjė transformaciją

Bylos “NNfurje.m” tekstas

```
cd c:\Diplominis\MatLab\NNFurje\;
clear all;
close all;


time = (0.025:0.025:5)';
T1 = sin(time);
T2 = sin(time*13.3);
T3 = sin(time*33.4);
T4 = sin(time*73.6);
T5 = sin(time*97.3);

NM = length(time);
in=T1+T2+T3+T4+T5;


figure;
Furje = fft(in);

RealFurje = real(Furje) ;
ImFurje = imag(Furje) ;

for i=1:NM;   %Daznis (fraction of sampling rate)
x_asis(i)=i/NM;
end;



%%%%%%%%%%%%%%%%%%%%%%
N=NM; % tinklo iejimu skaicius

for i=1:N
   input_range(i,1)=-5 ; input_range(i,2)=5;
end;

netFurje = newlin(input_range,(N+2));


for i=1:N  % i row
       sin_masyvas(1, i)=1; % RE X    kosinusoide su 0 ciklu. Visur
verte lygi 1
end;
```

```
for j=2:(N/2+1) % j column    % j = tinklo isejimu skaicius
        for i=1:N  % i row
           sin_masyvas(j, i)=cos((i-1)*(pi*(j-1)/(N/2))); % RE X
        end;
end;


for i=1:N  % i row
       sin_masyvas((N/2+1), i)=0; % IM X    sinusoide su 0 ciklu. Visur
verte lygi 0
end;


for j=(N/2+3):(N+2) % j column    % j = tinklo isejimu skaicius
        for i=1:N  % i row
           sin_masyvas(j, i)=-sin((i-1)*(pi*(j-1)/(N/2))); % IM X
        end;
end;


netFurje.IW{1,1} = sin_masyvas;
outNNFurje = sim(netFurje, in);
%%%%%%%%%%%%%%%%%%



%plot (abs(Furje(1:NM/2)));
plot (RealFurje(1:NM/2), '-xb', 'MarkerSize', 10);
title('Reali dalis');
hold on;
plot(outNNFurje(1:NM/2), '-or', 'MarkerSize', 5);
legend('Standartinis FFT','Neuroninis tinklas', 0);
%plot (Furje((NM/2+1):NM));
%plot(x_asis(1:NM/2), Furje((NM/2+1):NM)  );
%xlabel('Daznis (fraction of sampling rate)');
%ylabel('Amplitude');



%plot(x_asis(1:NM/2), Furje((NM/2+1):NM)  );


figure;

plot (ImFurje((NM/2+1):NM), '-xb', 'MarkerSize', 10);
hold on;
title('Menama dalis');
plot(outNNFurje((N/2+1):(N)), '-or', 'MarkerSize', 5);
legend('Standartinis FFT','Neuroninis tinklas', 0);

hold off;
```

2. Balto triukšmo šalinimas iš žmogaus kalbos gretimų spektrinių segmentų sumavimo metodu

Bylos “runSummingFilter.m” tekstas

```
cd c:\Diplominis\MatLab\SummingFiltering\;
clear all;
close all;


soundGenerator;

%if sampling rate = 12 kHz, then
% 1 sec <-> 12,000 samples
% (2:40) ms <-> x samples
% x= 12E3 * (2:40)E-3 = 12 * (2:40) = 24:480 samples


N=128;

NUZ = length(uzterstas);
atstatytas=zeros(NUZ, 1);

SN=4; %sumos gylis
gabaliukas=zeros(N, SN);
H=hamming(N);
H=H';

for i=1:(NUZ-N)


    for j=1:(SN-1)
      gabaliukas(:, j)=gabaliukas(:, j+1);
    end;

gabaliukas(1:N, SN)=uzterstas(i:(N+i-1));
tempfft = repmat(complex(0) ,[N, 1]);

    for j=1:SN
      tempfft=fft(gabaliukas(:, j))+tempfft;
    end;

tempatstatytas(1:N)=real(ifft(tempfft));


tempatstatytas=tempatstatytas.*H;


tempA=atstatytas(i:(N+i-1));
atstatytas(i:(N+i-1))=tempA+tempatstatytas';
%atstatytas(i:(N+i-1))=atstatytas(i:(N+i-1))+tempatstatytas(1:N);
```

```
end;


maks=max(atstatytas);
%maks2=abs(min(atstatytas));
%maks=max(maks, maks2);


atstatytas=atstatytas/maks;
wavwrite(atstatytas, Fs, 'outFiltered.wav');



ArtificialImpulseResponse;
atstatytas2=conv(uzterstas, ImResponse);
atstatytas2=atstatytas2/maks;

NUZ2 = length(atstatytas2);
atstatytas2=atstatytas2(N/2:(NUZ2-N/2));
wavwrite(atstatytas2, Fs, 'outFiltered2.wav');


figure;
plot(uzterstas, 'r');
hold on ;
plot(atstatytas, 'g');
plot(ieskom, 'b');


PlotSpecgram;
efficiency;
```

Bylos “ArtificialImpulseResponse.m” tekstas

```
FrResponseLong=abs(real(fft(ieskom)));

compr=round(NM/N/2);

for i=1:(2*N)
FrResponse(i)=max(FrResponseLong((compr*(i-1)+1):(compr*i)));
end;



figure;
plot(FrResponse);
hold on;
ImResponse= (real(ifft(FrResponse)));
plot(real(fft(ImResponse)), 'r');
legend('Dazninis atsakas','Dazninis atsakas nuo REAL dalies ImResponse',
0);
xlabel('Imties numeris'); ylabel('Amplitude');
```

```
figure;
plot(ImResponse);
%title('Impulso atsakas');
xlabel('Imties numeris'); ylabel('Amplitude');


%%%%% SHIFTING--ROTATING ARRAY BEGIN
M=N/2;  % per kiek poziciju stumiam
TempArray=ImResponse;
ImResponse(1:M)=TempArray((2*N-M+1):2*N);
ImResponse((M+1):2*N)=TempArray(1:(2*N-M));
%%%%% SHIFTING--ROTATING ARRAY END


ImResponse=ImResponse(1:N);

ImResponse=ImResponse*10;

figure;
plot(ImResponse, '-ob', 'MarkerSize', 5);
hold on;


fliplr(ImResponse);
H=hamming(N);
H=H';
ImResponse=ImResponse.*H;

plot(ImResponse, '-or', 'MarkerSize', 5);

%%%% Atvirkstine funkcija tansig  log=ln ; log = Natural logarithm
for i=1:N
ImResponseTanSig(i)=(-1/2)*log(-(ImResponse(i)-1)/(ImResponse(i)+1));
end;

%plot(ImResponseTanSig, '-xg', 'MarkerSize', 10);


%title('Impulso atsakas');
legend('pastumtas per N/2 imciu','padaugintas is Hammingo langu', 0);
xlabel('Imties numeris'); ylabel('Amplitude');
```

Bylos “soundGenerator.m” tekstas

```
[ieskom,Fs,bits] = wavread('one.wav');
%[ieskom,Fs,bits] = wavread('about.wav');


ieskom=cat (1, zeros(1000, 1), ieskom);
ieskom=cat (1, ieskom, zeros(1000, 1));

NM = length(ieskom);
```

```
triuksmas=(rand(NM, 1)-.5)*2*.5;

%ieskom=ieskom/2;
uzterstas=ieskom+triuksmas;
maks=max(uzterstas);
%maks2=abs(min(uzterstas));
%maks=max(maks, maks2);

uzterstas=uzterstas/maks;
wavwrite(uzterstas, Fs, 'OutWithNoise.wav');
```

Bylos “PlotSpecgram.m” tekstas

```
figure;
specgram(ieskom);
xlabel('Laikas'); ylabel('Daznis');
figure;
specgram(uzterstas);
xlabel('Laikas'); ylabel('Daznis');
figure;
specgram(atstatytas);
xlabel('Laikas'); ylabel('Daznis');
figure;
specgram(atstatytas2);
xlabel('Laikas'); ylabel('Daznis');
```

Bylos “efficiency.m” tekstas

```
uzTriuksmas=uzterstas-ieskom;
atsTriuksmas=atstatytas-ieskom;
atsTriuksmas2=atstatytas2-ieskom;

uzNoisePower=0;
atsNoisePower=0;
atsNoisePower2=0;

for i=1:NM
uzNoisePower=uzNoisePower+uzTriuksmas(i)^2;
atsNoisePower=atsNoisePower+atsTriuksmas(i)^2;
atsNoisePower2=atsNoisePower2+atsTriuksmas2(i)^2;
end;


Efektyvumas=10*log10(uzNoisePower/atsNoisePower)
Efektyvumas2=10*log10(uzNoisePower/atsNoisePower2)
```

3. Evoliucinis NT mokymas kaip dar vienas euristinis metodas

Bylos “runStaciakampiai.m” tekstas

```
cd c:\Diplominis\MatLab\StaciakampiaiImpulsai\;
clear all;
close all;
staciakampiai;
uzlyginti;
```

Bylos “staciakampiai.m” tekstas

```
%pradiniu duomenu generavimas
NM=1000;
koordinate=0;


while koordinate<NM
   signalas=round(rand); % generuoti ar signalas yra 0 ar 1
   if signalas==0 ilgis=round(20 + rand*4); end;% generuoti atsitiktini
signalo ilgi (t) [20..24]
   if signalas==1 ilgis=round(20 + rand*6); end;% generuoti atsitiktini
signalo ilgi (t) [20..26]

   for i2=1:ilgis
      koordinate=koordinate+1;
      if koordinate<=NM prduom(1,koordinate)=signalas; end;
   end;
end;


pr=1; pa=0;

while pa<NM
   pa=pr+50;
   train(pr, [1:51])=prduom(:,[pr:pa]);
   pr=pr+1;
end;



%%%%%%%% ciklas
for evol=1:5
%for evol=1:20


switch evol
case 1, iter=30; step=0.0001; pradinis=zeros(1,51); %'y-'
case 2, iter=60; step=0.00001; pradinis=W; %'m-'
```

```
case 3, iter=200; step=0.000001; pradinis=W; %'c-'
case 4, iter=15; step=0.0001; pradinis=W; %'r-'
case 5, iter=5; step=0.0001; pradinis=W;   %'g-'
otherwise iter=25; step=0.0001; pradinis=W; %'b-'
end;



triuksmas=evol*1;

for i=1:950
train2(i, [1:51])= train(i, [1:51])+rand(1,51)*triuksmas;
end;


for i=1:950
test(i, [1:51])= train(i, [1:51])+rand(1,51)*triuksmas;
end;



a=(train2-
ones(size(train2,1),1)*mean(train2))./(ones(size(train2,1),1)*std(train2
));
a=a(:,[[1:24],[26:51],25]);
C=cov(a(:,1:50));[u,d,v]=svd(C);
T=u*inv(sqrtm(d));
b=a(:,1:50)*T; %T-transformavimo matrica
figure(2);
plot(b(:,1), b(:,2),'k.')



a2=(test-
ones(size(test,1),1)*mean(test))./(ones(size(test,1),1)*std(test));
a2=a2(:,[[1:24],[26:51],25]);
C2=cov(a2(:,1:50));[u2,d2,v2]=svd(C2);
b2=a2(:,1:50)*u2*inv(sqrtm(d2));
figure(3);plot(b2(:,1),b2(:,2),'k.')


ad=a(:,51);
bd=b(:,1:50);
at=a2(:,51);
bt=b2(:,1:50);

[W,er,et]=percstud(bd,ad,bt,at,iter,step,pradinis,0.01);
figure(4); hold on;


switch evol
case 1, clf; plot([1:iter],er,'k:',[1:iter],et,'k-');
case 2, plot([1:iter],er,'m:',[1:iter],et,'m-');
case 3, plot([1:iter],er,'c:',[1:iter],et,'c-');
case 4, plot([1:iter],er,'r:',[1:iter],et,'r-');
case 5, plot([1:iter],er,'g:',[1:iter],et,'g-');
```

```
otherwise plot([1:iter],er,'b:',[1:iter],et,'b-');
end;

legend('Mokymas: iter=30, step=0.0001', 'Testavimas: iter=30,
step=0.0001', 'Mokymas: iter=60, step=0.00001', 'Testavimas: iter=60,
step=0.00001', 'Mokymas:  iter=200, step=0.000001', 'Testavimas:
iter=200, step=0.000001', 'Mokymas: iter=15, step=0.0001', 'Testavimas:
iter=15, step=0.0001', 'Mokymas: iter=5, step=0.0001', 'Testavimas:
iter=5, step=0.0001', 0);
xlabel('Iteraciju skaicius'); ylabel('Er');

figure(5);
plot(W);


end; %%%%%%%%ciklas


% pradiniu duomenu atvaizdavimas
figure(1); % clf; hold on;
pr=1; pa=0;
while pa<NM
   pa=pr+50;
   galduom(:,[pr:pa])=train2(pr, [1:51]);
   pr=pr+1;
end;
plot(galduom, 'b.');    %  - melyni taskai
ylim([-0.2 6.2]);
xlabel('Imties numeris'); ylabel('Amplitude');


%figure(6);
%plot(pradinis); % pradiniai svoriai


eil=size(b,1);
X=[b,ones(eil,1)];


Y=round(logsig(X*W')); %skaiciuoja neurono funkcija
Y=Y';
%figure(8);plot(1:950,prduom(26:975),'r.',1:950,Y+0.03,'b.');

figure(8);plot(1:950,prduom(26:975),'or', 'MarkerSize', 10)
hold on;
plot(1:950,Y,'ob', 'MarkerSize', 5);
legend('Pradinis signalas', 'Atstatytas is triuksmo signalas', 0);
ylim([-0.2 1.2]);
xlabel('Tasko numeris'); ylabel('Amplitude');
```

Bylos “percstud.m” tekstas

```
% Find robust regression by
%                   nonlinear single layer perceptron
% author            Sarunas Raudys <raudys@ktl.mii.lt>
% A input N*p array - training-set
% Y target N*1 array- training-set
% At input Nt*p array - test-set
% Yt target Nt*1 array- test-set
% iter - a number of iterations
% step - learning-step
% Wstart - 1*(p+1) starting weight vector
% alfa - a scaling parameter
% W -  1*(p+1) final weight vector
% et - generalization error history in "iter" training iterations
% prior to training we recommend:
% - to substract from A,Y, and At,Yt the sample means of A, Y,
% - to use Wstart = zeros(1,p+1);

function [W,er,et]=percstud(A,Y,At,Yt,iter,step,Wstart,alfa)
[N , p ] = size(A );
[Nt, pt] = size(At);
W=Wstart;
stepalfa=step/alfa;
AA=[A, ones(N,1)];AAt=[At, ones(Nt,1)];
for i=1:iter
      dist=alfa*(Y- AA * W');
      ind=find(abs(dist)<pi);
      W=W+stepalfa * sin(dist(ind))'*AA(ind',:);
      dt=AAt*W'-Yt;
      et(i)=sqrt( dt'*dt./Nt);
      dr=AA*W'-Y;
      er(i)=sqrt( dr'*dr./N);

end
return
```

Bylos “simSlp.m” tekstas

```
function Y=simSlp(X,W)

eiluciu=size(X,1);
Y=logsig([X,ones(eiluciu,1)]*W');

return
```

Bylos “uzlyginti.m” tekstas

```
%%%%%%%Antras DNT tinklas BEGIN


pr=1; pa=0;
```

```
% pirmo tinklo duomenis verciam matrica
while pa<(NM-50)
   pa=pr+50;
   train3(pr, [1:51])=Y(:,[pr:pa]);
   train3(pr, [25])=train(pr,[50]);   % istatom i 25 lastele originalu
signala
   pr=pr+1;
end;


for i=1:900
test(i, [1:51])= train3(i, [1:51]);
end;


a3=(train3-
ones(size(train3,1),1)*mean(train3))./(ones(size(train3,1),1)*std(train3
));
a3=a3(:,[[1:24],[26:51],25]);
C3=cov(a3(:,1:50));[u3,d3,v3]=svd(C3);
T3=u3*inv(sqrtm(d3));
b3=a3(:,1:50)*T3; %T-transformavimo matrica
figure(2);
plot(b3(:,1), b3(:,2),'k.')


a4=(test-
ones(size(test,1),1)*mean(test))./(ones(size(test,1),1)*std(test));
a4=a4(:,[[1:24],[26:51],25]);
C4=cov(a4(:,1:50));[u4,d4,v4]=svd(C4);
b4=a4(:,1:50)*u4*inv(sqrtm(d4));
figure(3);plot(b4(:,1),b4(:,2),'k.')


ad=a3(:,51);
bd=b3(:,1:50);
at=a4(:,51);
bt=b4(:,1:50);

pradinis=zeros(1,51);
iter=17;

[W,er,et]=percstud(bd,ad,bt,at,iter,step,pradinis,0.01);


figure(9); hold on;
clf; plot([1:iter],er,'g:',[1:iter],et,'g-');



eil3=size(b3,1);
X3=[b3,ones(eil3,1)];


Y3=round(logsig(X3*W')); %skaiciuoja neurono funkcija
Y3=Y3';
```

```
%figure(10);plot(1:900,prduom(51:950),'r.',1:900,Y3+0.03,'b.');



figure(10);plot(1:900,prduom(51:950),'or', 'MarkerSize', 10)
hold on;
plot(1:900,Y3,'ob', 'MarkerSize', 5);
legend('Pradinis signalas', 'Antro NT isejimo signalas', 0);
ylim([-0.2 1.2]);
xlabel('Imties numeris'); ylabel('Amplitude');



%figure(10);
%plot(W);

%%%%%%Antras DNT tinklas END
```

4. HINMA algoritmo efektyvumo demonstracija

Bylos “runEW.m” tekstas

```
cd c:\Diplominis\MatLab\EnhanceWeightsE\
clear all;
close all;


signalasSINsuma;

N=50;

TrainLong=Train;
TestLong=Test;
ieskomLong=ieskom;


Train=Train(1:NM/3);
Test=Test(1:NM/3);
ieskom=ieskom(1:NM/3);
NM=NM/3;


trainTanSigNN;
ArtificialImpulseResponse;
RunNNtansig;
enhanceTansigNN;
EWefficiency;

save 1.mat;


ShortSNratioTrain=SNratioTrain;
ShortSNratioTest=SNratioTest;
ShortEfektyvumasEW=EfektyvumasEW;
ShortEfektyvumasEWH=EfektyvumasEWH;
ShortEfektyvumasEWLin=EfektyvumasEWLin;
ShortEfektyvumasEWHLin=EfektyvumasEWHLin;
ShortEfektyvumasNetiesinioVsDSP=EfektyvumasNetiesinioVsDSP;
ShortEfektyvumasLinVsDSP=EfektyvumasLinVsDSP;




NM=NM*3;
Train=TrainLong;
Test=TestLong;
ieskom=ieskomLong;

netLin.IW{1,1}=zeros(1,50);
```

```
net.IW{1,1}=zeros(1,50);

trainTanSigNN;
ArtificialImpulseResponse;
RunNNtansig;
enhanceTansigNN;
EWefficiency;



ShortSNratioTrain
SNratioTrain
ShortSNratioTest
SNratioTest
ShortEfektyvumasEW
EfektyvumasEW
ShortEfektyvumasEWH
EfektyvumasEWH
ShortEfektyvumasEWLin
EfektyvumasEWLin
ShortEfektyvumasEWHLin
EfektyvumasEWHLin
ShortEfektyvumasNetiesinioVsDSP
EfektyvumasNetiesinioVsDSP
ShortEfektyvumasLinVsDSP
EfektyvumasLinVsDSP

save 2.mat;

compareEWefficiency;
```

Bylos “signalasSINsuma.m” tekstas

```
time = (0.025:0.025:15)';
T1 = sin(time);
T2 = sin(time*13.3);
T3 = sin(time*33.4);
T4 = sin(time*2.3);



NM = length(time);

triuksmas=(rand(NM, 1)-.5)*10;

Train=T2+T3+triuksmas;


triuksmas2=(rand(NM, 1)-.5)*10;
Test=T2+T3+triuksmas2;
maks= max(Test);
```

```
Test=Test/maks;



maks= max(Train);
Train=Train/maks;

ieskom=(T2+T3)/maks;


figure;
plot(Train(1:NM/3), 'r');
hold on;
plot(Test(1:NM/3), 'b');
hold on;
plot(ieskom(1:NM/3), 'k', 'LineWidth', 2);
%title('Signalas');
legend('Mokymo signalas', 'Testavimo signalas', 'Ieskomas signalas', 0);
xlabel('Imties numeris'); ylabel('Amplitude');


figure;
FrRespSpektrasTrain=abs(real(fft(Train)));
plot(FrRespSpektrasTrain(1:NM/2), 'r');
hold on;

FrRespSpektrasTest=abs(real(fft(Test)));
plot(FrRespSpektrasTest(1:NM/2), 'b');
hold on;

FrRespSpektrasIeskom=abs(real(fft(ieskom)));
plot(FrRespSpektrasIeskom(1:NM/2), 'k', 'LineWidth', 2);
%title('Signalo spektras');
legend('Mokymo signalas', 'Testavimo signalas', 'Ieskomas signalas', 0);
xlabel('Imties numeris'); ylabel('Amplitude');
```

Bylos “trainTanSigNN.m” tekstas

```
N=50; % tinklo iejimu skaicius


for i=1:N
   input_range(i,1)=-4 ; input_range(i,2)=4;
end;

net = newff(input_range, [1], {'tansig'});

%net = newff(input_range, [1], {'tansig'},'trainbr');



%traingd
```

```
%Basic gradient descent. Slow response, can be used in incremental mode
training.

%traingdm

%Gradient descent with momentum. Generally faster than traingd. Can be
used in incremental mode training.

%traingdx

%Adaptive learning rate. Faster training than traingd,but can only be
used in batch mode training.

%trainrp

%Resilient backpropagation. Simple batch mode training algorithm with
fast convergence and minimal storage requirements.


%traincgf

%Fletcher-Reeves conjugate gradient algorithm. Has smallest storage
requirements of the conjugate gradient algorithms.

%traincgp

%Polak-Ribiére conjugate gradient algorithm. Slightly larger storage
requirements than traincgf. Faster convergence on some problems.


%traincgb

%Powell-Beale conjugate gradient algorithm. Slightly larger storage
requirements than traincgp. Generally faster convergence.

%trainscg

%Scaled conjugate gradient algorithm. The only conjugate gradient
algorithm that requires no line search. A very good general purpose
training algorithm.

%trainbfg

%BFGS quasi-Newton method. Requires storage of approximate Hessian
matrix and has more computation in each iteration than conjugate
gradient algorithms, but usually converges in fewer iterations.


%trainoss

%One step secant method. Compromise between conjugate gradient methods
and quasi-Newton methods.

%trainlm
```

```
% Levenberg-Marquardt algorithm. Fastest training algorithm for networks
of moderate size. Has memory reduction feature for use when the training
set is large.



ieskom2=ieskom(N/2:(NM-N/2-1));


%%%%%%%%%%%%%%%%%


net.trainParam.epochs = 100;
net.trainParam.goal = 0.01;


figure; hold off;
for i=1:(NM-N)
    hold off;
    plot (net.IW{1,1}, '-ob', 'MarkerSize', 5);
    title('Neuroninio tinklo svoriai');
    net = train(net,Train((i):(i+N-1)),ieskom2(i));
end;

hold on;
%%%%%%%%%%%%%%%%%%%


%%%%%%%Linear network BEGIN
for i=1:(NM-N)
    PP(:, i)=Train((i):(i+N-1));
    TT(1, i)=ieskom2(i);
end;

netLin = newlind(PP,TT);

%%%%%%%Linear network END
```

Bylos “ArtificialImpulseResponse.m” tekstas

```
FrResponse=abs(real(fft(ieskom(1:2*N))));

figure;
plot(FrResponse);
hold on;
ImResponse= (real(ifft(FrResponse)));
plot(real(fft(ImResponse)), 'r');
legend('Dazninis atsakas','Dazninis atsakas nuo REAL dalies ImResponse',
0);
xlabel('Imties numeris'); ylabel('Amplitude');
```

```
figure;
plot(ImResponse);
xlabel('Imties numeris'); ylabel('Amplitude');


%%%%% SHIFTING--ROTATING ARRAY BEGIN
M=N/2;  % per kiek poziciju stumiam
TempArray=ImResponse;
ImResponse(1:M)=TempArray((2*N-M+1):2*N);
ImResponse((M+1):2*N)=TempArray(1:(2*N-M));
%%%%% SHIFTING--ROTATING ARRAY END


ImResponse=ImResponse(1:N);

ImResponse=ImResponse/6;

figure;
plot(ImResponse, '-ob', 'MarkerSize', 5);
hold on;


ImResponse=ImResponse';


fliplr(ImResponse);
H=hamming(N);
H=H';
ImResponse=ImResponse.*H;

plot(ImResponse, '-xr', 'MarkerSize', 10);


legend('pastumtas per N/2 imciu','padaugintas is Hammingo langu', 0);
xlabel('Imties numeris'); ylabel('Amplitude');
```

Bylos “RunNNtansig.m” tekstas

```
netDSP = newff(input_range, [1], {'tansig'});
netDSP.IW{1,1} = ImResponseTanSig;

net.trainParam.epochs = 100;
net.trainParam.goal = 0.01;


for i=1:(NM-N)
   outTrain(i, 1) = sim(net, Train((i):(i+N-1)));
end;

outTrainDSP=conv(ImResponseTanSig, Train);
```

```
NUZ2 = length(outTrainDSP);
outTrainDSP=outTrainDSP(N:(NUZ2-N));


for i=1:(NM-N)
   outTest(i, 1) = sim(net, Test((i):(i+N-1)));
end;



outTestDSP=conv(ImResponseTanSig, Test);
NUZ3 = length(outTestDSP);
outTestDSP=outTestDSP(N:(NUZ3-N));


for i=1:(NM-N)
   outTrainLin(i, 1) = sim(netLin, Train((i):(i+N-1)));
end;


for i=1:(NM-N)
   outTestLin(i, 1) = sim(netLin, Test((i):(i+N-1)));
end;


figure;
plot(outTrainLin, '-xg', 'MarkerSize', 10);
hold on;
plot(outTrainDSP, '-xb', 'MarkerSize', 10);
hold on;
plot(outTrain, '-or', 'MarkerSize', 5);
hold on;
plot(ieskom2, 'k', 'LineWidth', 2);
title('Mokymo signalas');
legend('NT su tiesine aktyvacijos funkcija, mokytas MVK algoritmu',
'Optimalus tiesinis filtras', 'Netiesinis NT, mokytas atbulo sklidimo
algoritmu', 'Ieskomas signalas', 0);
xlabel('Imties numeris'); ylabel('Amplitude');


figure;
plot(outTestLin, '-xg', 'MarkerSize', 10);
hold on;
plot(outTestDSP, '-xb', 'MarkerSize', 10);
hold on;
plot(outTest, '-or', 'MarkerSize', 5);
hold on;
plot(ieskom2, 'k', 'LineWidth', 2);
title('Testavimo signalas');
legend('NT su tiesine aktyvacijos funkcija, mokytas MVK algoritmu',
'Optimalus tiesinis filtras', 'Netiesinis NT, mokytas atbulo sklidimo
algoritmu', 'Ieskomas signalas', 0);
xlabel('Imties numeris'); ylabel('Amplitude');


NLmirror=fliplr(net.IW{1,1});
```

```
NLImResponse = cat(2,net.IW{1,1}, NLmirror);

NLFrRespSpektras=abs(real(fft(NLImResponse)));
```

Bylos “enhanceTansigNN.m” tekstas

```
savesvoriai=net.IW{1,1};
svoriai=savesvoriai;

svoriaiFl=fliplr(svoriai);
enhanced=conv(svoriai, svoriaiFl);

figure;
plot(enhanced, '-ob', 'MarkerSize', 5);
xlabel('Imties numeris'); ylabel('Amplitude');

enhanced=enhanced(N/2:(N+N/2-1));

enhancedH=enhanced.*H;
net.IW{1,1}=enhanced;



%%%%%Linear Begin
savesvoriaiLin=netLin.IW{1,1};
svoriaiLin=savesvoriaiLin;

svoriaiFlLin=fliplr(svoriaiLin);
enhancedLin=conv(svoriaiLin, svoriaiFlLin);

figure;
plot(enhancedLin, '-ob', 'MarkerSize', 5);
%title('Enhancing neuroninio tinklo svorius');
xlabel('Imties numeris'); ylabel('Amplitude');

enhancedLin=enhancedLin(N/2:(N+N/2-1));

enhancedHLin=enhancedLin.*H;
netLin.IW{1,1}=enhancedLin;
%%%%Linear End


figure;
plot(ImResponseTanSig, '-xk', 'MarkerSize', 10);
hold on;
plot(savesvoriaiLin, '-xb', 'MarkerSize', 10);
hold on;
plot(enhancedHLin, '-xr', 'MarkerSize', 10);
%title('Neuronio tinklo svoriai');
legend('Optimalus tiesinis filtras', 'NT su tiesine aktyvavimo funkcija,
mokytas MVK algoritmu', 'Simetrinti tiesinio NT svoriai', 0);
```

```
xlabel('Imties numeris'); ylabel('Amplitude');


figure;
plot(ImResponseTanSig, '-xk', 'MarkerSize', 10);
hold on;
plot(savesvoriai, '-ob', 'MarkerSize', 5);
hold on;
plot(enhancedH, '-or', 'MarkerSize', 5);
hold on;
legend('Optimalus tiesinis filtras', 'Netiesinis NT, mokytas atbulo
sklidimo algoritmu','Simetrinti netiesinio NT svoriai', 0);
xlabel('Imties numeris'); ylabel('Amplitude');


%net.IW{1,1}=fliplr(net.IW{1,1});


NLmirror2=fliplr(net.IW{1,1});
NLImResponse2 = cat(2,net.IW{1,1}, NLmirror2);


figure;
plot(NLFrRespSpektras(1:N), 'b');
hold on;
NLFrRespSpektras2=abs(real(fft(NLImResponse2)));
plot(NLFrRespSpektras2(1:N), 'r');
hold on;
FrResponseResized=FrResponse/5;
plot(FrResponseResized(1:N), 'k', 'LineWidth', 2);
%title('TanSig neuroninio tinklo pseudo Fr response');
legend('NT mokytas atbulo sklidimo algoritmu','Simetrinti svoriai',
'Ieskomas signalas', 0);
hold off;




for i=1:(NM-N)
   outTestEW(i, 1) = sim(net, Test((i):(i+N-1)));
end;

net.IW{1,1}=enhancedH;

for i=1:(NM-N)
   outTestEWH(i, 1) = sim(net, Test((i):(i+N-1)));
end;




%%%Linear BEGIN
for i=1:(NM-N)
   outTestEWLin(i, 1) = sim(netLin, Test((i):(i+N-1)));
end;
```

```
netLin.IW{1,1}=enhancedHLin;

for i=1:(NM-N)
   outTestEWHLin(i, 1) = sim(netLin, Test((i):(i+N-1)));
end;
%%%Linear BEGIN



figure;
plot(outTestDSP, '-og', 'MarkerSize', 5);
hold on;
plot(outTestEWLin, '-ob', 'MarkerSize', 5);
hold on;
plot(outTestEWHLin, '-xr', 'MarkerSize', 10);
hold on;
plot(ieskom2, '-k', 'LineWidth', 2);
%title('Testavimo signalas');
legend('Optimalus tiesinis filtras', 'NT svoriai po su veidrodiniu
atspindziu', 'NT svoriai po is Hammingo lango', 'Ieskomas signalas', 0);



figure;
plot(outTestDSP, '-og', 'MarkerSize', 5);
hold on;
plot(outTestEW, '-ob', 'MarkerSize', 5);
hold on;
plot(outTestEWH, '-xr', 'MarkerSize', 10);
hold on;
plot(ieskom2, '-k', 'LineWidth', 2);
%title('Testavimo signalas');
legend('Optimalus tiesinis filtras', 'NT svoriai po sasukos su
veidrodiniu atspindziu', 'NT svoriai po daugybos is Hammingo lango',
'Ieskomas signalas', 0);


net.IW{1,1}=savesvoriai;
netLin.IW{1,1}=savesvoriaiLin;
```

Bylos “EWefficiency.m” tekstas

```
ilgis=size(outTest, 1);


TrainCut=Train(N/2:(NM-N/2-1));
TestCut=Test(N/2:(NM-N/2-1));

NoiseTrain=(outTrain-ieskom2);
NoiseTest=(outTest-ieskom2);
```

```
PNoiseTrain=0;
PNoiseTest=0;
Pieskom2=0;


for i=1:ilgis
    PNoiseTrain=PNoiseTrain+(NoiseTrain(i))^2;
    PNoiseTest=PNoiseTest+(NoiseTest(i))^2;
    Pieskom2=Pieskom2+(ieskom(i))^2;
end;


SNratioTrain=10*log10(PNoiseTrain/Pieskom2);
SNratioTest=10*log10(PNoiseTest/Pieskom2);


%%%%%%%%%%%%%%%%%%%


eoutTrain=(outTrain-ieskom2);
eoutTrainDSP=(outTrainDSP-ieskom2);
eoutTest=(outTest-ieskom2);
eoutTestDSP=(outTestDSP-ieskom2);
eoutTestLin=(outTestLin-ieskom2);
eoutTestEW=(outTestEW-ieskom2);
eoutTestEWH=(outTestEWH-ieskom2);
eoutTestEWLin=(outTestEWLin-ieskom2);
eoutTestEWHLin=(outTestEWHLin-ieskom2);



PeoutTrain=0;
PeoutTrainDSP=0;
PeoutTest=0;
PeoutTestDSP=0;
PeoutTestLin=0;
PeoutTestEW=0;
PeoutTestEWH=0;
PeoutTestEWLin=0;
PeoutTestEWHLin=0;


for i=1:ilgis
    PeoutTrain=PeoutTrain+(eoutTrain(i))^2;
    PeoutTrainDSP=PeoutTrainDSP+(eoutTrainDSP(i))^2;  %
    PeoutTest=PeoutTest+(eoutTest(i))^2;
    PeoutTestDSP=PeoutTestDSP+(eoutTestDSP(i))^2; %
    PeoutTestLin=PeoutTestLin+(eoutTestLin(i))^2;
    PeoutTestEW=PeoutTestEW+(eoutTestEW(i))^2;    %
    PeoutTestEWH=PeoutTestEWH+(eoutTestEWH(i))^2;  %
    PeoutTestEWLin=PeoutTestEWLin+(eoutTestEWLin(i))^2;
    PeoutTestEWHLin=PeoutTestEWHLin+(eoutTestEWHLin(i))^2;
end;


EfektyvumasEW=10*log10(PeoutTest/PeoutTestEW);
```

```
EfektyvumasEWH=10*log10(PeoutTest/PeoutTestEWH);

EfektyvumasEWLin=10*log10(PeoutTest/PeoutTestEWLin);
EfektyvumasEWHLin=10*log10(PeoutTest/PeoutTestEWHLin);


EfektyvumasNetiesinioVsDSP=10*log10(PeoutTestEWH/PeoutTestDSP);
EfektyvumasLinVsDSP=10*log10(PeoutTestEWHLin/PeoutTestDSP);
```

Bylos “compareEWefficiency.m” tekstas

```
load 1;

a=PeoutTest;
b=PeoutTestEWH;
c=PeoutTestEWHLin;
d=PeoutTestLin;

load 2;

a2=PeoutTest/3;
b2=PeoutTestEWH/3;
c2=PeoutTestEWHLin/3;
d2=PeoutTestLin/3;

EW=a2/b
EWLin=d2/c

a/a2
b/b2
c/c2
```