# Neuronnät som olinjär modellstruktur för dynamiska system

Jonas Sjöberg
Chalmers

December 4, 1996

## 1 Inledning

Under den senaste tioårsperioden har artificella neuronnät, eller bara neuronnät, blivit mycket populära modeller och de har använts inom de mest skilda områden. Oftast framställs de som konstgjorda hjärnor som "lär sig själva" av sina "egna misstag" och även om den ursprungliga idén var att härma informationsbehandlingen i hjärnan, så är kopplingarna till biologiska neuronnät i de flesta fall obefintlig. Istället beskrivs och analyseras modellerna i regel bättre med de metoder som beskrivs i kursboken, [4].

Vi kommer att beskriva neuronnäten som en speciell typ av modellstruktur men mycket har publicerats inom detta område av forskare som inte ser på dem på det sättet. Inom neuronnätsbranchen hittar man dessutom folk med bakgrund från helt skilda forskningsområden. Detta har medfört att många gamla resultat "upptäckts" på nytt och att många begrepp har fått nya namn. Lyckas man översätta termerna till det språk man är van att höra så blir det inte så mycket nytt att lära sig. Framförallt blir det mycket lättare att förstå vad som egentligen menas.

Begreppsförvirringen kan ibland bli total och med termen "neuronnät" kan refereras till nästan vad som helst. Till exempel finns det de som på nytt undersöker de linjära konfektionsmodellerna som presenteras i kursboken och ser dem som neuronnät. Det gäller alltså att kritiskt granska "nya" idéer och försöka förstå dem med hjälp av begrepp och terminologi från statistik och modellbygge.

Vi ska här på ett par sidor ge en kort förklaring av vad neuronnät är för något och beskriva hur de kan användas för att modellera dynamiska system. Där ej annat anges är hänvisningarna till kursboken.

# 2 Problembeskrivning

Antag att vi har data

$$Z^N = \{[y(t), \varphi(t)];\ t = 1, \ldots N\} \tag{1}$$

genererade av en okänd funktion $f(\cdot)$

$$y(t) = f(\varphi(t)) + e(t) \tag{2}$$

där $e(t)$ är vitt brus. Vi ska snart gå in på vad $\varphi(t)$ är för vektor, nu räcker det dock att konstatera att den är känd vid tiden $t$. Målet blir nu att utifrån data finna en modell som approximerar $f(\cdot)$ så bra som möjligt.

Inom denna allmänna problembeskrivning ryms många problem från flera olika områden. Det kan t.ex. handla om klassificering, modellering av dynamiska system eller detektering. Här intresserar vi oss speciellt för dynamiska system men det kan vara värt att veta att den underliggande problembeskrivningen är den samma för alla problem där en funktion ska anpassas till data.

Som ett första steg i modelleringen ska en modellstruktur väljas ut. De linjära konfektionsmodellerna i kursboken är exempel på olika modellstrukturer och vi ska snart presentera mer allmänna, olinjära, modellstrukturer baserade på bland annat neuronnät. Med valet av modellstruktur bestämmer vi oss inom vilken klass av funktioner som vi ska söka efter en approximation till $f(\cdot)$. Vår modell kan nu uttryckas

$$\hat{y}(t|\theta) = g(\theta, \varphi(t)) \tag{3}$$

där $\theta$ är modellens parametrar och de ska nu skattas så att $g(\hat{\theta}, \cdot)$ "liknar" $f(\cdot)$ så bra som möjligt. Märk att vi inte har sagt någonting om utseendet på $g(\cdot, \cdot)$. Beskrivningen passar in på alla tänkbara modeller, detta inkluderar inte bara de linjära modellerna i kursboken utan även neuronnät som vi strax ska se. Vill vi ha en adaptiv modell så får vi låta $\theta$ bero på tiden.

Att hitta det $\hat{\theta}$ som ger den bästa approximationen är ett statistiskt skattningsproblem, snarare än ett inlärningsproblem som det ofta kallas i neuronnätslitteraturen. Som mått på approximationens godhet väljs oftast kvadratfelet vilket beskrivs i avsnitt 9.3.

$$\begin{aligned} V_N(\theta) &= \frac{1}{N}\sum_{t=1}^{N}\varepsilon^2(t,\theta) \\ \varepsilon(t,\theta) &= y(t) - \hat{y}(t|\theta) \end{aligned} \tag{4}$$

Att minimera $V_N(\theta)$, givet data $Z^N$, är ett minimeringsproblem, även om det i neuronnätskretsar oftast kallas för ett inlärningsproblem. Den vanligaste minimeringsalgoritmen inom neuronnätsområdet går under namnet *backpropagation-error algorithm* och överensstämmer med brantastelutnings-, eller som det också kallas,

gradientmetoden. Det är ett vanligt missförstånd att backpropagation-error algoritmen skulle vara något speciellt för neuronnät. Vad man gör är i princip att beräkna derivatan av $V_N$ med avseende på $\theta$ och sedan ta ett litet steg i dess negativa riktning vilket man kan göra med vilken deriverbar modell som helst. Oftast får man dock en betydligt effektivare minimering om man använder en andra ordningens metod då även andraderivatan används, se avsnitt 9.3 i kursboken.

# 3 Linjära modeller och linjär regression

I kursboken presenteras flera linjära konfektionsmodeller (avsnitt 9.2) och tack vare linjäriteten så följer många trevliga egenskaper. Av dessa modeller är det bara *ARX* som dessutom är linjär i parametrarna vilket betyder att den kan uttryckas som linjär regression

$$\hat{y}(t|\theta) = g(\theta, \varphi(t)) = \theta^T \varphi(t) \tag{5}$$

där

$$\theta = [a_1 \; a_2 \ldots a_{n_a} \;\; b_1 \ldots b_{n_b}]$$
$$\varphi(t) = [-y(t-1) \ldots -y(t-n_a) \;\; u(t-n_k) \ldots u(t-n_k-n_b+1)]^T \tag{6}$$

$\theta$ är parametervektorn och $\varphi(t)$ är regressionsvektorn. Elementen i $\varphi(t)$ kallas för regressorer.

Om en modell kan skrivas som linjär regression så kan parameterskattningen som minimerar $V_N$ beräknas analytiskt vilket är en stor fördel. I annat fall måste iterativa metoder utnyttjas vilket blir betydligt mer beräkningskrävande och dessutom kan man få problem med lokala minima. Se avsnitt 9.3.

Det är alltså bra om modellen är en linjär regression, vilket inte inskränker den till att vara en linjär modell. Vi kan mycket väl ha olinjära regressorer, som i exemplet i avsnitt 10.6.

Enligt Weierstrass kan varje kontinuerlig funktion approximeras godtyckligt bra med ett polynom av tillräckligt högt gradtal och detta kan vi nu uttnyttja i flera dimensioner. Vi illustrerar detta med ett exempel.

**Exempel 3.1** Polynomutveckling
Antag att det underliggande systemet kan uttryckas på följande sätt

$$y(t) = f(y(t-1), u(t-1)) + e(t) \tag{7}$$

där $f$ är en okänd olinjär funktion i två variabler. En modellstruktur får vi genom att ansätta ett polynom i $y(t-1)$ och $u(t-1)$[1].

[1]Polynomapproximationen till ett dynamiskt system kallas för *Volterraserien* för systemet och är den begreppsmässigt lättaste metoden att generalisera teorin från linjära till olinjära system. Se t.ex. [1].

$$g(\theta,[y\ u]) = \theta_1 y + \theta_2 u + \theta_3 y^2 + \theta_4 u^2 + \theta_5 yu + \theta_6 y^3 + \theta_7 u^3 + \theta_8 y^2 u + \theta_9 y u^2 + \ldots \quad (8)$$

där vi låtit bli att skriva ut tidsindex för att få en kompaktare notation. När modellen uppnått tillräckligt bra approximationsförmåga avbryts polynomutvecklingen så att polynomet får ändligt gradtal. Denna modell kan nu skrivas som linjär regression med regressorn

$$\varphi(t) = [y\ u\ y^2\ u^2\ yu\ y^3\ u^3\ y^2u\ yu^2 \ldots]^T \quad (9)$$

Regressorn är lätt att beräkna från givna data och eftersom vi har linjär regression så beräknas parameterskattningen på samma sätt som för $ARX$ modellen. □

Från exemplet är det uppenbart att olinjära konfektionsmodeller inte är något nytt och polynomutveckling är bara en av många olika möjligheter att representera en allmän funktion, i kurserna i transformteori och funktionalanalys presenteras ytterligare några exempel.

I nästa avsnitt ska vi diskutera de svårigheter som tillkommer då man går över från linjära- till olinjära konfektionsmodeller. Därefter presenterar vi en neuronnätsmodell och ser hur den kan användas för att modellera ett dynamiskt system.

# 4 Svårigheter med olinjära modeller

Vi ska här inte utreda alla de svårigheter som tillkommer för olinjära konfektionsmodeller, utan bara ge en ytlig bild av problemen.

## 4.1 Frekvenstolkning

Den för linjära modeller så fruktsamma frekvenstolkningen försvinner. För ett linjärt system så kan varje frekvens behandlas för sig (det är precis det som visas i ett Bodediagram). Detta gäller inte längre för ett olinjärt system utan frekvenserna blandas. Om insignalen till ett olinjärt system är en ren sinussignal så kommer utsignalen att i allmänhet bli en blandning av flera frekvenser.

Linjäriteten innebär i princip att insignalen måste innehålla de frekvenser som vi vill identifiera vid, se ekv (9.48) i kursboken. Vi kan alltså titta på frekvensinnehållet på insignalen för att avgöra om den är bra för identifiering av ett dynamiskt system. För ett olinjärt system räcker det inte med att titta på frekvensinnehållet utan vi måste även titta på $\varphi$ i ekv. (3) och den delmängd $D \subset \mathbb{R}^n$, där $n = \dim\varphi$, som innesluter alla $\varphi(t)$ i identifieringsdata. Eftersom vi inte har någon

information om $f(\cdot)$ utanför $D$ kan vi inte heller lita på modellen där. En förnuftig tumregel är: *Samla in data under omständigheter som liknar de förhållanden som modellen senara ska arbeta under.* I så fall gäller $\varphi \in D$ även under användandet av modellen och det finns en möjlighet att modellen beskriver systemet.

## 4.2 Flexibilitet

Modellfelet kan delas upp i ett varians- och ett biasbidrag, se avsnitt 9.4. Om det existerar ett "sant" parametervärde $\theta_0$,

$$f(\cdot) = g(\theta_0, \cdot) \tag{10}$$

så ligger den okända funktionen i modellklassen. I detta fall så kommer hela det slutliga modellfelet bestå av endast variansfel som beror på brus i mätningarna. I de flesta fall existerar inget $\theta_0$ som uppfyller (10) och ett biasfel tillkommer, vilket beror på modellens oförmåga att approximera den okända funktionen.

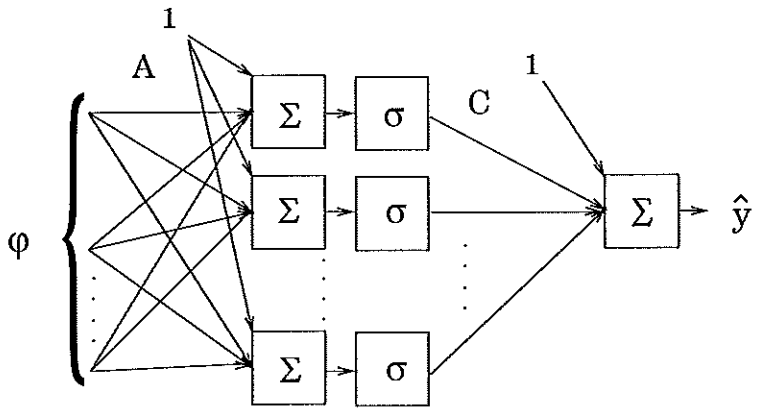
Genom att göra modellen flexiblare, dvs genom att införa fler frihetsgrader (=parametrar) så minskar vi biasfelet. I exempel 3.1 motsvarar det att ta med flera termer i polynomutvecklingen. Samtidigt med flexibiliteten ökar också variansfelet och därför bör modeller vara sparsamma med parametrar. Jämför diskussionen om "Att välja modellstruktur" i avsnitt 10.4. För många parametrar ger överanpassning till data, vilket betyder att man har anpassat $g(\theta, \cdot)$ inte bara till den "nyttiga" informationen utan också till bruset i mätningarna.

Det är en huvudfråga att avgöra lämplig grad av flexibilitet för att få en så bra avvägning mellan bias- och variansfelet som möjligt och det finns många idéer på hur det ska göras. För linjära modeller presenteras i kursboken ett antal kriterier för att göra denna avvägning och de kan till viss utsträckning även användas för olinjära modeller. Man kan också starta med många parametrar och sedan endast skatta de som visar sig vara viktiga. Resterande parametrar som inte skattats kan sedan ses som konstanter och de bidrar inte till variansfelet. Hur detta görs, och speciellt med neuronmodeller, beskrivs i [5].

# 5 Neuronnät som modellstruktur

Det finns många olika typer av neuronnät och vi ska här presentera den populäraste varianten som kallas "feed-forward"-nät. Allmän information om olika typer av neuronnät kan hittas i någon introduktionsbok i området, till exempel [2, 3, 6]. Ingen av dessa skriver dock med det språk vi är vana vid från den här kursen.

I figur (1) visas ett neuralt nätverk bestående av enheter (eller neuroner) indelade i tre lager, ett inlager, ett dolt lager och ett utlager. Nätverket är av "feed-forward"-typ vilket betyder att enheterna i ett tidigare lager påverkar nästa lager, men inte




Figur 1: nätverk av "feed-forward"-typ med ett dolt lager.

tvärt om. Det dolda lagret kan vara godtyckligt brett, dvs antalet dolda enheter är en designvariabel som måste bestämmas av användaren. Man kan även ha flera dolda lager mellan inlagret och utlagret. Här kommer vi endast ha en utsignal, dvs utlagret består av endast en enhet, men en generalisering till flera utsignaler är rättfram.

Vid en given insignal $\varphi$ (=värden på enheterna i inlagret) beräknas utsignalen $\hat{y}$ på följande sätt: I varje enhet i det dolda lagret görs en viktad summation. Därefter passerar denna summa en olinjäritet av mättnadstyp, ofta kallad *sigmoid-funktion*,

$$\sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}} \tag{11}$$

där $x$ är den viktade summan vid ingången till neuronen. På så sätt är utsignalen från varje dold enhet begränsad mellan 0 och 1. I utlagret, eller utenheten om den bara består av en enhet, görs en viktad summation av resultatet från de dolda enheterna. Vikterna i de olika summationerna utgör modellens parametrar.

Det beskrivna nätverket är en funktion $g : \mathbb{R}^n \to \mathbb{R}$. Vi kan uttrycka detta som

$$g(\varphi, \theta) = \sum_{j=1}^{n_h} c_j \sigma(A_j^T \varphi + a_j) + c_0 \tag{12}$$

där $n_h$ är antalet dolda enheter. $A_j$, är en kolumnvektor och $a_j$ är en skalär. De utgör parametrarna till den dolda enheten $j$. Vidare är $c_j$ parametern mellan den dolda enheten $j$ och utenheten. Vektorn $\varphi$ innehåller värdet på insignalen till inlagret. Alla våra parametrar, $A$, $a$ och $c$, kan vi nu lägga i parametervektorn $\theta$.

Man kan visa att (12) kan approximera en godtycklig kontinuerlig funktion godtyckligt bra bara $n_h$ välj tillräckligt stort. Detta ska jämföras med Weierstrass sats för polynom.

Nu ska vi generalisera de linjära konfektionsmodellerna ARX och OE till olinjära

modeller. De linjära modellerna kan uttryckas som

$$\hat{y}(t) = \theta^T \varphi(t) \tag{13}$$

där $\theta$ består av modellens parametrar och

$$\begin{aligned} \varphi(t) = [&-y(t-1) \ -y(t-2) \ \dots \\ &-y(t-n_a) \ u(t-n_k) \dots u(t-n_b-n_k)] \end{aligned} \tag{14}$$

i ARX fallet, och

$$\begin{aligned} \varphi(t) = [&-\hat{y}(t-1) \ -\hat{y}(t-2) \\ &-\hat{y}(t-n_a) \ u(t-n_k) \dots u(t-n_b-n_k)] \end{aligned} \tag{15}$$

i OE fallet.

Dessa två val av regressionsvektor kan vi nu också göra med (12) som olinjär modellstruktur. Dessa kallar vi NARX och NOE, där N står för "Non-linear". Precis som i det linjära fallet har NARX viss möjlighet att modellera bruset medan NOE endast modellerar systemet. I det linjära fallet är det tre designparametrar att bestämma, $[n_a \ n_b \ n_k]$, och nu har det blivit en till, $n_h$, antalet neuroner i det dolda lagret.

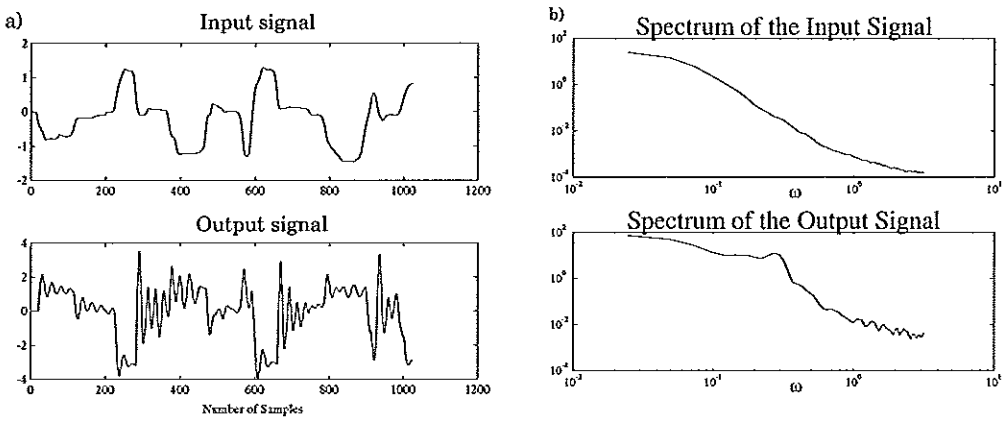
I ett exempel ska vi nu visa hur NARX kan användas för att modellera ett olinjärt system.

**Exempel 5.1** En hydraulisk kran
Vi studerar samma hydrauliska kran som i exempel 10.2 i kursboken. För att beskriva resonansen tillfredsställande med en linjär modell var man tvungen att filtrera data vid den intressanta frekvensen. Som vi ska se så har vi mer frihet med en olinjär model.

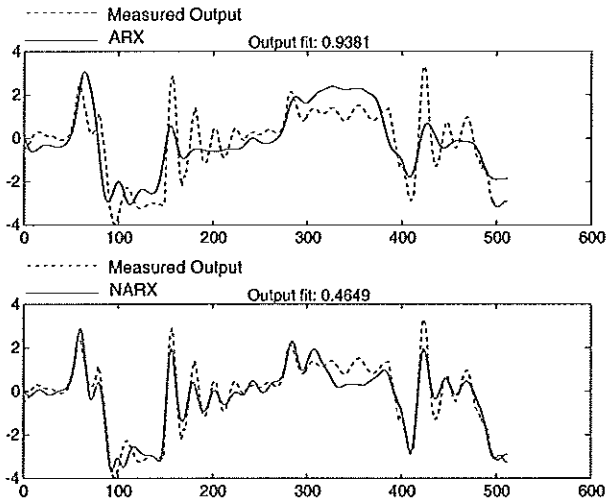
I figur 2 a) ser vi insignal respektive utsignal som motsvarar ventilöppningen till den hydraliska cylindern och trycket i densamma.

Först tar vi en titt på signalernas spektrum i figur 2 b). Vi ser att utsignalen är starkt exicterad inom ett frekvensområde där det i princip saknas insignalenergi. Det svarar mot resonansen i systemet och som det visas i kursboken kan den endast modelleras med en linjär modell om vi först bandpassfiltrerar data kring den frekvensen. Det innebär att modellen bara blir giltig inom ett smalt frekvensband. Med en olinjär modell kommer vi att klara av att beskriva resonansen utan att filtrera våra data och vi får därför en mer "global" model.

Låt oss prova en linjär ARX-modell utan att bandpassfiltrera data. Bästa resultat ger en en andra ordningens model, $n_a = 2$, $n_b = 2$, $n_k = 1$. Sedan testar vi motsvarande NARX modell med åtta dolda enheter (neuroner), $n_a = 2$, $n_b = 2$, $n_k = 1$, $n_h = 8$. I figur 3 ser vi simulerade signaler på valideringsdata jämfört med den uppmätta signalen. Uppenbart har vi lyckats modellera viktig olinjär




Figur 2: a) Insignal och utsignal till kranen. b) Spektrum av signalerna.




Figur 3: Heldragen linje: simulerad signal med ARX respektive NARX. Streckad linje: uppmätt utsignal.