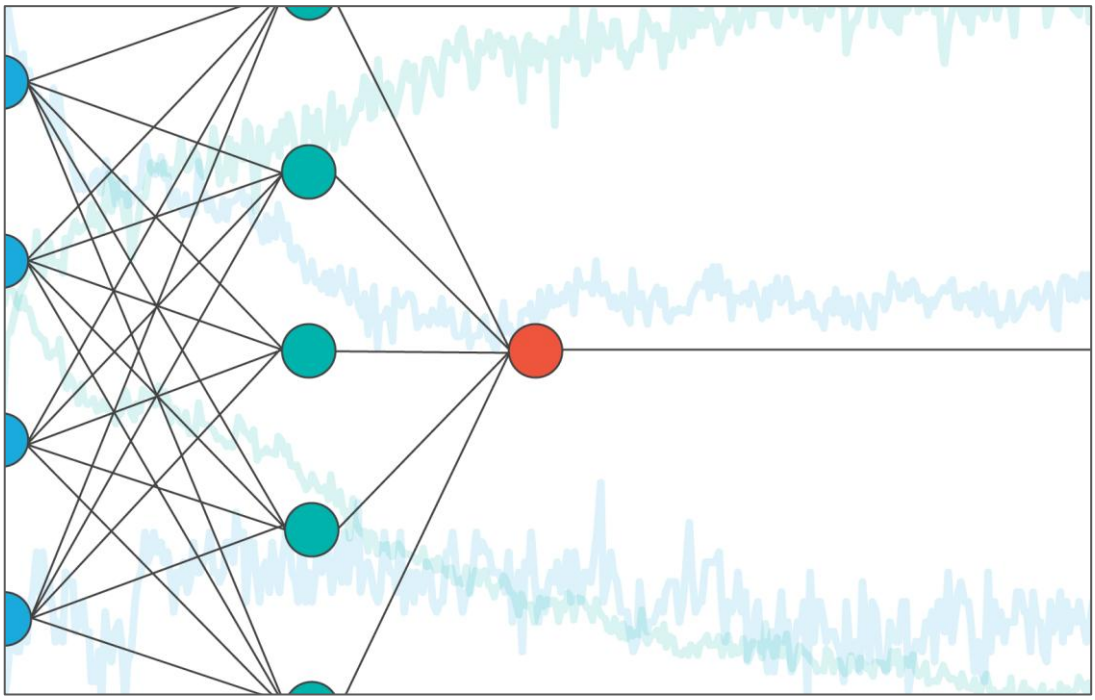
CHALMERS | GÖTEBORGS UNIVERSITET



Predicted Future

- att förutsäga aktiekurser med artificiella neuronnät

Kandidatarbete inom Data- och informationsteknik

LINUS FÄRNSTRAND
OSCAR SÖDERLUND
NIKLAS LÖNNERFORS
EMIL BERNERSKOG
TOBIAS AXELL

Institutionen för Data- och informationsteknik

CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg, Sverige 2012
Kandidatarbete/rapport nr 2012:12

Sammanfattning

Olika modeller har genom åren tagits fram för att förenkla analys av aktiemarknaden. En relativt ny teknik är så kallade artificiella neuronnät. Dessa modeller har visat sig lovande när det gäller att förutsäga aktiekurser. Projektets syfte är att hitta artificiella neuronnät som ger bra förutsägelser av framtida aktiekurser. Under projektet har ett mjukvarusystem som genererar och tränar artificiella neuronnät av typen feedforward utvecklats för att kunna testa neuronnät med olika indata, struktur och träningsinställningar. För att komma fram till vilka konfigurationer på neuronnäten som ger bra resultat har näten testats genom simulerad aktiehandel där köp- och säljbeslut grundats på neuronnätens förutsägelser. Vi har framställt ett system med funktionalitet för att generera samt utvärdera neuronnät. Med detta har vi även funnit neuronnät som lyckas förutsäga aktiemarknaden, om än under en begränsad period.

Abstract

A plethora of models have throughout the years been developed with the purpose of aiding analysis of the stock market. A relatively new such model, one that has shown promise in predicting future stock quotes, is artificial neural networks. The purpose of the project is to construct artificial neural networks that yield high-quality predictions of future stock quotes. The project has resulted in a software system with the capability of training a specific subset of artificial neural networks, namely feed-forward networks, with varying input data, structure and training parameters. In order to identify the specific network configurations that consistently lead to positive results, the networks have been tested by making simulated trading decisions based on their output. We have successfully constructed networks with the ability to accurately predict stock quotes, albeit within a limited time frame.

Ordlista

AJAX Asynchronous Javascript and XML, en teknik som möjliggör transparent uppdatering av sidor.

Artificiellt neuronnät En förenklad modell av ett biologiskt nervsystem som används för att approximera funktioner och hitta mönster i datamängder.

Backpropagation En typ av inlärningsalgoritm för feed forward-nätverk.

Blankning En strategi för att tjäna pengar vid börsnedgång.

Courtage Avgiften som börsmäklaren tar ut för varje transkation.

EMH Hypotesen om den effektiva marknaden, en teori som beskriver att marknaden återger all tillgänglig information i aktiernas pris och att rörelserna följer ett slumpmässigt mönster.

Träningsepok En passering i en upplärningsalgoritm genom all träningsdata ett visst neuronnät använder sig av.

Feature extraction En typ av förbehandling av data för att förbättra artificiella neuronnäts förutsägelser.

Feed forward-nätverk En typ av artificiella neuronnät.

Fundamental analys En analysmetod som utvärderar företags tillväxt- och vinstpotential.

JDBC Java DataBase Connectivity, ett API för hantering av SQL-databaser.

JSON JavaScript Object Notation, ett textbaserat format för överföring av data mellan datorer.

jQuery Ett javascriptbibliotek som bland annat förenklar AJAX-funktionalitet.

Large cap De största bolagen på Stockholmsbörsen.

LMS Least mean squared-algoritmen är en inlärningsalgoritm för artificiella neuronnät.

Nasdaq OMX Stockholm Stockholmsbörsen.

Nyemission Då ett bolag utfärdar nya aktier.

Orderdjup Antalet ej genomförda ordrar vid en viss budnivå för en given aktie.

Perceptron Ett neuronnät bestående av endast en neuron.

SSH Secure Shell, ett protokoll som möjliggör en säker anslutning mellan datorer över ett nätverk.

Teknisk analys En analysmetod som används för att hitta trender i aktiekurser.

UNIX timestamp Standard för tidsangivelser som anges i förfluten tid sedan 00:00:00 1 jan 1970.

Upplärningsalgoritm Algoritm som givet en mängd träningsdata och ett neuronnät justerar vikterna hos neuronnätet i syfte att förbättra dessa approximation den underliggande funktion träningsdatan representerar.

Volym Antal värdepapper som för tillfället omsätts.

Öppnings- och stängningscall Öppnings- respektive stängningsprocedur som används på Stockholmsbörsen för att samla ihop bud och ge ett så korrekt öppnings- respektive stängningspris som möjligt.

Överanpassning Ett fenomen som innebär att ett neuronnät memorerar träningsdatan den visats istället för att approximera den underliggande funktion datan representerar.

Innehåll

1 Inledning

Att spekulera i aktier kan vara lukrativt men också svårt då aktiemarknaden är svår att förutse. Genom åren har många modeller utvecklats för att analysera aktier och förutsäga dess utveckling. En del modeller analyserar företaget ifråga medan andra modeller är oberoende av företaget och endast analyserar trender i aktiens prisutveckling.

Analysmodellen ovan gjordes till en början manuellt där investerare satt och observerade grafer och beräknade nyckelvärden på dem och uppskattade deras framtida utveckling. Senare insågs att de uträkningar man gjorde och de beslut man tog utefter dem var statiska och kunde utföras av datorer. Den datoriserade tekniska analysen var född. Sedan dess har många metoder använts för att försöka lära datorer förutsäga framtida värden av värdepapper. En av dessa är med hjälp av artificiella neuronnät.

Artificiella neuronnät är approximatorer för godtyckliga funktioner och kan användas för försöka hitta mönster i tidsserier. Det kan därför vara intressant att använda neuronnät till att hitta mönster i aktiers prisutveckling och därigenom, om något mönster finns, förutsäga aktiernas pris [1].

1.1 Syfte

Syftet med detta projekt är att utveckla ett system som genererar och testar artificiella neuronnät samt att med detta system hitta neuronnät som framgångsrikt förutsäger aktiekurser.

1.2 Mål

Målet är att systemets förutsägelser skall vara av högre kvalitet än slumpmässiga gissningar.

1.3 Problem

Den övergripande uppgiften är att utifrån statistisk data kunna avgöra, i någon mån, en given akties trend. Detta innefattar flera deluppgifter:

1.3.1 Datainsamling

Nödvändiga förutsättningar för projektet är att hitta och samla in bra aktiedata och avgöra ett bra sätt att lagra samt tillgängliggöra den på. För att kunna genomföra testkörningar behöver ett ramverk för hanteringen av datan byggas upp.

1.3.2 Att skapa ett användargränssnitt för träning av neuronnät

För att kunna genomföra testkörningar behöver ett ramverk för hanteringen av datan byggas upp. Då gruppen saknar kunskap om vilka typer av nätverksarkitekturer och datadimensioner som ger bra förutsättningar för neuronnät att producera positiva resultat, krävs det att ramverket tillåter enkel och snabb konstruktion av olika typer av neuronnät med olika typer av indata. Detta medför att systemet på ett enkelt sätt skall tillåta utvärdering av hur

olika nät presterar och huruvida de är värda att vidareutveckla eller inte. Systemet behöver även funktionalitet för att visualisera neuronnätens förutsägelser, exempelvis med hjälp av grafer och tabeller.

1.3.3 Att utvärdera de artificiella neuronnäten

För att kunna avgöra hur väl ett neuronnät presterar behöver dess förutsägelser utvärderas. Någon form av simulerad handel som utgår från nätens förutsägelser behöver därför genomföras för att få fram tydliga och mätbara resultat.

1.3.4 Identifiering av relevant data

I takt med att mängden insamlad data ökar, ökar även behovet av att kunna sålla ut den data som är relevant och leder till bra resultat. I syfte att välja ut relevant aktiedata behöver den filtreras och förbehandlas. Främst behöver datan avgränsas i tidsavseende, men även onödiga parametrar (sådana som ej påverkar förutsägelser eller påverkar dem negativt) kan behöva tas bort.

Även hur statistiken bör användas för att göra vinst behöver identifieras, det vill säga vilka aktiedata som neuronnäten kan hitta mönster i. En del aktier kan vara korrelerade, till exempel underleverantörer till en större aktör vars aktiekurser kan tänkas följa varandra någorlunda. Det är också tänkbart att valutor kan påverka vissa aktiers kurser.

1.3.5 Identifiering av parameterinställningar som ger bra resultat

Eftersom de fria parametrarna i ett neuronnät kan varieras i ett obegränsat antal kombinationer, är det viktigt att hitta de kombinationer som upprepande ger bra resultat. Det är även viktigt att komma fram till varför vissa parameterkonfigurationer ger bättre resultat än andra.

1.4 Avgränsningar

- Systemet skall inte kunna genomföra faktiska transaktioner utan är enbart tänkt att användas som rådgivning för manuella transaktioner.
- Systemet skall i första hand använda artificiella neuronnät för sina analyser. Att inkludera metoder från fundamental och teknisk analys ligger utanför projektets ramar.
- Systemet skall inte utgöra en komplett lösning för en aktiehandlare. Detta innebär att det inte skall innefatta hjälpmedel för till exempel portföljhantering eller riskhantering.
- Systemet kommer endast att behandla aktier tillgängliga på börsen Nordic OMX Stockholm.

1.5 Metod

Projektet har innefattat teoretiska studier i syfte att öka förståelsen för neuronnät, aktiemarknaden och deras koppling till varandra. Det har dessutom involverat utvecklingen av ett system för att bedriva experiment med neuronnät.

1.5.1 Källsökning och litteraturstudier

Projektet inleddes med en mindre litteraturstudie då det krävdes kunskap både inom artificiella neuronnät och aktiehandel. Parallellt med den inledande litteraturstudien letades även källor till aktie- och valutadata för senare användning vid träning av neuronnäten. Studierna fortsatte sedan med att undersöka hur neuronnät kan användas för att förutse aktiemarknanden och hur projekt med liknande mål har gått tillväga, bland annat Nygren 2004 [2] och McCluskey 1993 [3]. Mer djupgående teoristudier av artificiella neuronnät och aktiehandel genomfördes parallellt med systemutvecklingen.

1.5.2 Systemutveckling

Utvecklingen av mjukvaran följde ingen specifik utvecklingsmetod. Den utvecklingsmetod som användes var av agil karaktär, där ändringar av programmets specifikationer kunde göras även efter att implementation påbörjats. Testning utfördes i samband med utveckling.

För versionshantering av programvaran användes Mercurial med en central server. Projektet delades upp i olika delprojekt som var och en versionhanterades separat.

För att hantera kompilering och körning av de olika delprojekten användes Maven, som automatiskt hanterar beroenden hos koden.

1.5.3 Testning och utvärdering av neuronnät

För att hitta bra neuronnät användes systemet för att generera en mängd nät med olika strukturer, parametrar och indata. Sökandet efter bra nät bedrevs i en iterativ process. I processens början genererades nät med stora variationer i struktur och parameterinställningar. När en generering- och träningsomgång blivit klar studerades de resulterande neuronnäten och i nästa omgång kunde de parametervärden som gett dåliga resultat sorteras bort och sökandet fokuserades på de parameterinställningar som gett bra resultat.

För att kunna avgöra huruvida ett nät ger bra förutsägelser eller ej har aktiehandel med neuronnätens förutsägelser som grund för köp- och säljbeslut simulerats.

1.6 Disposition

I kapitel 2 kommer den teoretiska bakgrunden till projektet gås igenom. Kapitel 3 kommer att handla om designen av det system som beskrivs i syftet. I kapitel 4 presenteras resultat, i kapitel 5 kommer resultaten diskuteras och i kapitel 6 presenteras slutsatser.

2 Teoretisk bakgrund

Avsnitt 2.1 förklarar hur artificiella neuronnät är uppbyggda och fungerar, avsnitt 2.2 behandlar aktiemarknaden och grundläggande aktieanalys, avsnitt 2.3 sammankopplar avsnitt 2.1 och 2.2 och förklarar hur artificiella neuronnät kan användas till aktieanalys med referenser till tidigare liknande projekt.

2.1 Artificiella neuronnät

Artificiella neuronnät är förenklade, matematiska modeller av de neurala nätverk som en hjärna är uppbyggd av [4, s. 31]. De har förmågan att lära sig av, hitta samband i och klassificera olika typer av data [4, s. 32].

Artificiella neuronnät utgör fortfarande ett aktivt forskningsområde [5][6][7], och neuronnät som kan tillämpas i praktiken är ofta utvecklade från enklare neuronnät, där lager på lager av komplexitet medför ökad praktisk användbarhet [8, s. 346].

Detta kapitel tar ursprungligen upp teorin bakom ett enkelt neuronnät, perceptronen. Definitionerna för perceptronen används sedan för att förklara en mer avancerad typ av neuronnät, feed forward-nätverktet.

2.1.1 Perceptronen

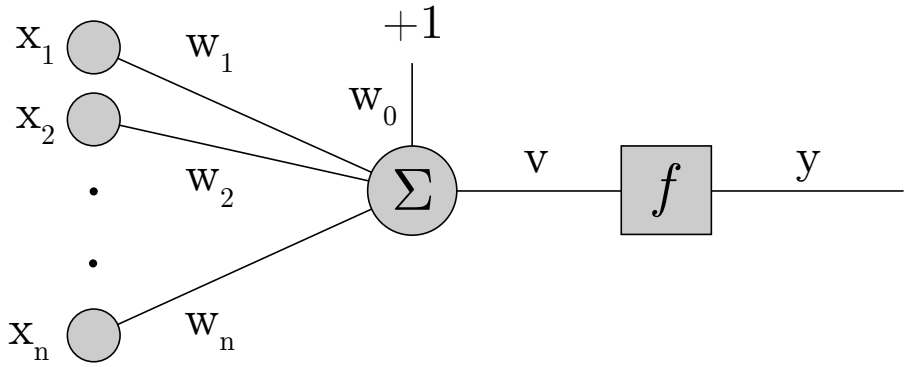
Den enklaste typen av neuronnät består av endast en neuron. En känd implementation av denna typ av nät är perceptronen, som skapades 1958 av Frank Rosenblatt [9, s. 130]. Rosenblatts perceptron bygger på en modell för arcificiella neuroner som tidigare utvecklats av två andra forskare vid namn Warren McCulloch och Walter Pitts [4, s. 78].

McCulloch och Pitts ackrediteras som upphovsmännen till det första neuronnätet som hade praktiska, om än enkla, tillämpningar. Detta neuronnät, som numera kallas MP-neuronen, bestod av en ensam neuron som kunde approximera de allra enklaste booleska funktionerna, AND och OR. Den saknade dock inlärningsförmåga. [10, s. XXXIX]

Även Rosenblatts perceptron består av en ensam neuron, men den skiljer sig från MP-neuronen på en viktig punkt; perceptronen har inlärningsförmåga. Tack vare detta kan perceptronen bland annat användas till grundläggande dataklassificering [4, s. 78].

Neuronen i Rosenblatts perceptron är en enkel beräkningsenhet, som transformerar indata till utdata i ett antal väldefinerade och enkla steg. Modellen av neuronen kan avgränsas i följande delar [4, s. 80]:

1. Ett eller flera ingångsvärden: $x_1 \dots x_n$
2. Ett bias: $+1$
3. Vikter: $w_0 \dots w_n$
4. En kombinerare
5. En aktiveringspotential: v
6. En aktiveringsfunktion: f
7. Ett utgångsvärde: y



Figur 1: Figuren illustrerar modellen av Rosenblatts perceptron, som består av en ensam neuron.

Perceptronen används genom att numeriska värden matas in på dess ingångar, som benämns $x_1 \dots x_n$. Perceptronen har även en särskild ingång med det konstanta värdet $+1$. Denna ingång benämns som perceptronens bias [4, s. 80].

Varje ingång $x_1 \dots x_n$ är associerad med ett eget viktvärde, $w_1 \dots w_n$. Även perceptronens bias har en sådan vikt, w_0. Dessa viktvärden används av perceptronens kombinerare för att bilda en sammantagen aktiveringspotential, v [4, s. 80].

Tillsammans kan perceptronens ingångsvärden och vikter representeras på vektorform:

$$[x_0, x_1, \dots, x_n] = \vec{x} \tag{1}$$

$$[w_0, w_1, \dots, w_n] = \vec{w} \tag{2}$$

Kombineraren beräknar aktiveringspotentialen genom att summera produkten av varje ingångsvärde och dess associerade vikt [4, s. 80]. Denna operation kan beskrivas som att bilda skalärprodukten av neuronens inputvektor och dess viktvektor:

$$v = \sum_{i=0}^{n} x_n w_n = \vec{x} \cdot \vec{w} \tag{3}$$

Perceptronens bias kan beskrivas som en geometrisk translation av aktiveringspotentialen. Dess syfte är analogt med konstanttermen i en linjär ekvation. Att värdet av x_0 är konstant $+1$ innebär att olika värden på w_0 påverkar värdet av aktiveringspotentialen på samma sätt som olika m-värden förflyttar linjen $y = kx + m$ upp och ned i ett kartesiskt koordinatsystem.

Neuronens slutgiltiga utgångsvärde, y, bestäms genom att applicera neuronens aktiveringsfunktion, f, på dess aktiveringspotential [4, s. 80]:

$$y = f(v) \tag{4}$$

Aktiveringsfunktionen transformerar neuronens aktiveringspotential till en meningsfull utsignal. Följande aktiveringsfunktioner är vanligt förekommande [4, s. 43][10, s.151]:

Stegfunktionen

$$g(v) = \begin{cases} 1 & v >= 0 \\ 0 & v < 0 \end{cases} \tag{5}$$

Sigmoidfunktionen

$$g(v) = \frac{1}{1 + e^{-av}} \tag{6}$$

Tangens hyperbolicus

$$g(v) = tanh(av) \tag{7}$$

Signumfunktionen

$$g(v) = \begin{cases} 1 & v > 0 \\ 0 & v = 0 \\ -1 & v < 0 \end{cases} \tag{8}$$

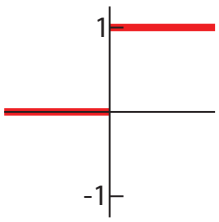
Linjära funktionen

$$g(v) = v \tag{9}$$

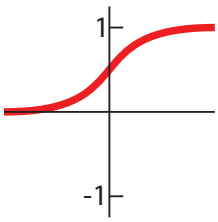
Styckvis linjära funktionen

$$g(v) = \begin{cases} v & a < v < b \\ a & v <= a \\ b & v >= b \end{cases} \tag{10}$$

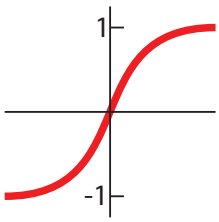
I figurerna 2 till 7 visas grafer för aktiveringsfunktionerna.



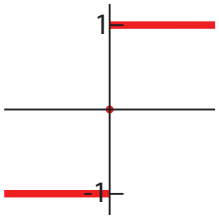
Figur 2: Steg.



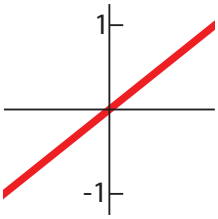
Figur 3: Sigmoid.



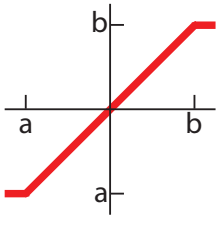
Figur 4: Tanh.



Figur 5: Signum.



Figur 6: Linjära.



Figur 7: Styckvis linjära.

Vilken aktiveringsfunktion som används är avgörande för vilken typ av funktion som perceptronen skall approximera [10, ss. 152-157]. Som exempel använde McCulloch och Pitts stegfunktionen i sin MP-neuron [11, s. 3]. Stegfunktionen används således när en neurons utdata skall modellera booleska eller binära värden. På samma sätt kan signumfunktionens användningsområde härledas till när neuronens utgångsvärde även skall kunna anta det negativa värdet -1.

Utöver att approximera booleska funktioner kan neuroner med steg- eller signumfunktionen även användas till att klassificera datamängder [12, s. 3]. Att klassificera en datamängd innebär att elementen i mängden delas in i ett antal klasser. En perceptron med stegfunktionen som aktiveringsfunktion kan således potentiellt dela in en datamängd i två klasser, eftersom den endast har ett utgångsvärde. Neuronnät som skall klassificera datamängder med större antal klasser kräver fler än ett utgångsvärde.

Sigmoidfunktionen kan ses som en kontinuerlig variant av stegfunktionen. Tangens hyperbolicus kan på samma sätt ses som en kontinuerlig variant av signumfunktionen. Dessa funktioner har som gemensam egenskap att de är kontinuerliga och icke-linjära. De är båda s-formade och har en extra fri parameter a, som bestämmer hur skarpt kurvorna stiger kring $v = 0$. När sluttningsparametern a går mot oändligheten närmar egenskaperna för sig dessa kurvor sina icke-kontinuerliga motsvarigheter.

Eftersom värdemängderna för de kontinuerliga, icke linjära kurvorna inte sträcker sig utanför $(-1, 1)$ oavsett värde på den fria parametern a, används med fördel den linjära funktionen för att approximera funktioner som har en obegränsad värdemängd.

Med endast en neuron kan man alltså bilda ett fullvärdigt neuronnät. Denna typ av nät är dock mycket begränsad i sin approximationsförmåga. Rosenblatt, som använde perceptronen till att klassificera data, lyckades endast träna perceptronen att klassificera datapunkter som är linjärt separerbara, det vill säga att klasserna är åtskiljbara med en linje, ett plan eller ett hyperplan [10, s. 158].

Förutom att klassificera linjärt separerbara punkter kan en perceptron, givet rätt aktiveringsfunktion, även tränas att approximera godtyckliga linjära funktioner. Detta bevisade Bernard Widrow och Ted Hoff när de utvecklade sin upplärningsalgoritm för perceptronen. Denna algoritm är känd som LMS; least mean square-algoritmen [13].

2.1.2 Least mean square-algoritmen

LMS är en grundläggande upplärningsalgoritm för neuronnät. Trots att den endast är applicerbar på enkla perceptroner använder den sig av en strategi som går att generalisera till upplärningsalgoritmer för mer komplicerade neuronnät. Den är också populär tack vare att den har låg beräkningskomplexitet och är lätt att implementera i ett mjukvarusystem [4, s. 122].

LMS är ett exempel på övervakad inlärning, vilket innebär att träningsdatan förutom ingångsvärden även innehåller nätets önskade utgångsvärden [9, s. 86]. Träningsdatan, T, ser ut på följande sätt [4, s. 123]:

$$T = [(\vec{x}_1, d_1), \ldots, (\vec{x}_p, d_p)] \tag{11}$$

Således skall varje träningselement innehålla en vektor med perceptronens ingångsvärden, $\vec{x}_i$, samt en motsvarande vektor med perceptronens önskade utgångsvärde, d_i.

Det önskade utgångsvärdet används under träningen för att beräkna hur nära perceptronen approximerar träningsdatan. Perceptronens lokala fel, e, symboliserar hur stor perceptronens felmarginal är vid approximering av ett enskilt träningselement.

$$e = d - y \tag{12}$$

Nätets globala felvärde, E, är ett normaliserat mått på nätets approximationer [4, s. 132].

$$E = 0.5 \cdot (d - y)^2 \tag{13}$$

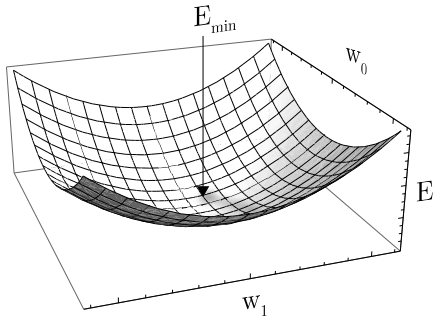
I LMS beräknas det globala felvärdet för ett ensamt element i träningsmängden. Detta tillvägagångssätt kallas även för online-upplärning, eftersom hela träningsmängden inte måste vara tillgänglig samtidigt. Det globala felet kan även beräknas som en summa av felvärdena för alla element i hela träningsmängden, vilket kallas för offline- eller batchupplärning [9, s. 178].

En slutsats av ekvationen för felvärdet är att om $E = 0$ så approximerar perceptronen träningsdatan perfekt. Upplärningsstrategin i LMS bygger på att hitta en viktvektor $\vec{w}$ som om möjligt uppnår detta, eller åtminstone leder till att felvärdet blir så litet som möjligt. Detta uppnås genom att iterativt justera nätets vikter, där varje iteration leder till att E minskas. När $E = 0$, eller när dess värde inte längre minskar, avslutas LMS och perceptronen anses färdigtränad [4, s. 124].

Tillvägagångssättet för att justera vikterna i LMS kan illustreras genom att se det globala felet som en funktion av perceptronens viktvektor och dess ingångsvektor:

$$E = f(\vec{w}, \vec{x}) \tag{14}$$

Givet en perceptron med en bestämd ingångsvektor, som innehåller ett ingångsvärde utöver perceptronens bias, vars viktvektor således har två komponenter, kan det globala felet plottas som en funktion i tre dimensioner med avseende på vikterna:



Figur 8: Figuren visar en plot av perceptronens globala fel E som en funktion av dess viktvektor, $[w_0, w_1]$. Plotten ger upphov till en yta i tre dimensioner.

Att minimera det globala felet E är ekvivalent med att hitta koordinaterna för $E_{\min}$. Plotten visar tydligt hur dessa koordinater innehåller värden på w_0

och w_1, för det lägsta möjliga värdet på E; $E_{\min}$. LMS hittar dessa värden genom en metod som kallas gradient descent [13].

Gradient descent bygger på att hitta ett minimivärde till en funktion, i detta fall E, med hjälp av funktionens lutning [4, s. 133]. Tillvägagångssättet är iterativt, och varje iteration förutsätter något värde på funktionens ingångsvärde, i detta fall $\vec{w}$ [4, s. 133].

För det aktuella ingångsvärdet beräknas den partiella derivatan av det globala felet med avseende på viktvektorn [4, s. 133]. Denna derivata är i plotten av E ekvivalent med lutningen av ytan, i den punkt som bestäms av den nuvarande viktvektorn $\vec{w}_t$. $\vec{w}_t$ justeras sedan med hjälp av den partiella derivatan för att erhålla nästa viktvektor, $\vec{w}_{t+1}$, som kommer ligga geometriskt närmare $E_{\min}$ än $\vec{w}_t$: [4, s. 133]

$$\vec{w}_{t+1} = \vec{w}_t - \eta * \frac{\delta E}{\delta \vec{w}} \tag{15}$$

Vektorn med det globala felets derivator, $\frac{\delta E}{\delta \vec{w}}$ brukar även uttryckas som ∇E. Ekvation 16 visar att ∇ är en vektor av partiella derivator med avseende på de olika viktvärdena i $\vec{w}$ [4, s. 125].

$$\frac{\delta E}{\delta \vec{w}} = \nabla E = [\frac{\delta}{\delta w_0}, \frac{\delta}{\delta w_1}, \ldots, \frac{\delta}{\delta w_n}] * E = [\frac{\delta E}{\delta w_0}, \frac{\delta E}{\delta w_1}, \ldots, \frac{\delta E}{\delta w_n}] \tag{16}$$

Genom att analytiskt utveckla derivatan $\frac{\delta E}{\delta \vec{w}}$ kan ∇E förenklas enligt ekvation 17 [4, s. 132].

$$\nabla E = -\vec{x} \cdot (d - y) \tag{17}$$

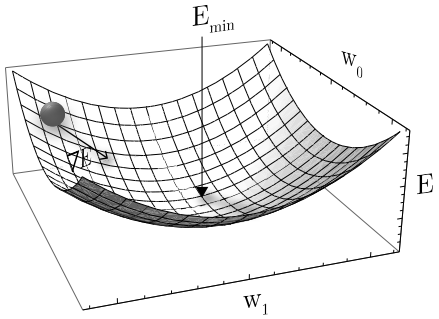
Komponenterna i ∇E talar om hur mycket varje individuell komponent i viktvektorn, det vill säga varje separat vikt, skall justeras. Eftersom justeringen sker med hjälp det globala felets derivata, kommer det globala felet att vara lägre vid nästkommande iteration. När derivatan är 0, liten eller oföränderlig är gradient descent-metoden slutförd. Ingångsvärdet för den första iterationen kan antingen väljas slumpmässigt eller vara $\vec{0}$ [4, s. 133].

Gradient descent liknar således Newton-Rhapsonmetoden som i korthet kan beskrivas som en metod för att hitta rötter till reella funktioner. Likheten med gradient descent ligger i att båda metoder utgår från godtyckliga ingångsvärden till sin målfunktion, för att sedan iterativt justera dessa med hjälp av målfunktionens derivata tills dess att dess önskade värde är funnet. För gradient descent är det önskade värdet målfunktionens minimum, medan Newton-Rhapsonmetoden hittar målfunktionens rot [14].

Parametern η i ekvation 15 används för att bestämma hur stora steg varje iteration skall ta. Ett högre värde på η leder till större steg i varje iteration, och således också till att antalet iterationer potentiellt blir mindre. När gradient descent utnyttjas i LMS-algoritmen, eller andra träningsalgoritmer, kallas η-parametern för upplärningshastighet [4, s. 133].

Ett sätt att illustrera gradient descent-metoden är att tänka sig att en kula placeras på ytan i plotten av E, på de koordinater som bestäms av den aktuella viktvektorn $\vec{w}$. I varje iteration, då viktvektorn justeras och koordinaterna förflyttas mot det lokala minimat, så "rullar" kulan nedåt mot ytans botten.

I de fall där LMS inte konvergerar mot $E_{\min}$, exempelvis på grund av att koordinaterna oscillerar kring minimat [4, s. 133], kan träningen även avbrytas



Figur 9: Figuren visar hur ∇E är ekvivalent med riktningen som en kula färdas när den placeras på ytan som är ett resultat av att plotta den partiella derivatan av E med avseende på viktvektorn.

efter ett bestämt antal iterationer. När alla träningselement använts i en av algoritmens iterationer har en träningsepok passerat. Det är vanligt att träningen sker över flera träningsepoker, beroende på hur många element som finns i träningsmängden [10, s. 44].

En formulering av LMS-algoritmen i pseudokod kan nu uttryckas som ett antal väldefinerade steg.

Algoritm 1 LMS

Require: $T = [(\vec{x}_1, d_1), \ldots, (\vec{x}_n, d_n)]$
Require: $\eta = \hat{\eta}$
  $\vec{w} := \vec{0}$
  $i := 0$
  repeat
    $e := d_i - (\vec{w} \cdot \vec{x}_i)$
    $\vec{w} = \vec{w} + \eta \cdot \vec{x} \cdot e$
    $i := (i + 1) \bmod |T|$
  until *done*

Algoritm 1 förutsätter ingen specifik metod för när träningen skall avbrytas. En faktisk implementation skulle till exempel kunna beräkna E för alla element i träningsmängden T efter varje träningsepok och avbryta träningen när det globala felet inte längre minskar, vilket vore en förenklad variant av en generell metod vid namn early stopping [4, s. 203].

Med hjälp av LMS är det således möjligt att hitta optimala vikter för en perceptron. Den enda riktiga begränsningen är att neuronen måste ha en deriverbar aktiveringsfunktion, i och med att aktiveringsfunktionen måste kunna deriveras för att sambandet i ekvation 17 ska gälla. I annat fall skulle beräkningen av den nya viktvektorn ej vara giltig i algoritm 1.

Trots att algoritmen här demonstrerats för ett neuronnät med endast ett ingångsvärde och ett bias, så är det bevisat att gradient descent-metoden är applicerbar för att hitta funktionsminimum i godtyckliga dimensioner. Således fungerar LMS på alla typer av perceptroner, oavsett hur många ingångsvärden de har. Algoritmen blir dock långsammare ju fler ingångsvärden perceptronen har [10, s. 145].

Perceptronen består som tidigare nämnts av endast en neuron, och på grund av detta är dess användningsområden begränsade. För att approximera data med samband av högre än linjär komplexitet krävs större och mer komplicerade neuronnät [8, s. 346]. Dessa nät består av flera neuroner och kallas för flerlagersperceptroner.

2.1.3 Flerlagersperceptroner

Den konceptuella övergången mellan en perceptron och flerlagersperceptroner är naturlig; en flerlagersperceptron består helt enkelt av fler än en neuron. Dessa neuroner kan vara sammankopplade med varandra i olika konfigurationer, men allra vanligast är att neuronerna är indelade i separata lager, vilket också har gett upphov till flerlagerperceptronens namn [4, s. 153].

Varje lager i en flerlagersperceptron kan innehålla ett godtyckligt antal neuroner. Det första lagret innehåller nätets inputneuroner, som var och en har ett ingångsvärde. Dessa ingångsvärden bildar tillsammans nätets totala indata, som i likhet med perceptronen benämns $\vec{x}$ [4, s. 52].

En stor skillnad mellan en flerlagersperceptron och en enkel perceptron, som följer av att alla lager kan innehålla ett godtyckligt antal neuroner, är att flerlagersperceptronen kan ha flera utgångsvärden. Dessa utgångsvärden bildas av neuronerna i nätets sista lager, outputlagret. En flerlagersperceptron kan således potentiellt approximera funktioner som avbildar vektorer på vektorer, med godtycklig dimensionalitet i båda ändar, vilket är en betydande generalisering av perceptronen [4, s. 52].

Mellan input- och outputlagret ligger flerlagersperceptronens gömda lager. Dessa lager kan vara godtyckligt många, men fler än två gömda lager tillför inte något till neuronnätets potentiella approximationsförmåga [10, s. 158].

Hur lagren i en flerlagersperceptron är konstruerade refereras till som nätverkets topologi [9, s. 451]. Topologin för ett nät brukar ofta skrivas som en sifferserie, där varje siffra anger antalet neuroner i ett av neuronnätets lager. Till exempel beskriver topologin 5-10-1 en flerlagersperceptron med 5 ingångsvärden, 10 gömda neuroner och ett utgångsvärde [4, s. 52].

Flerlagersperceptroner är indelade i många olika typer av underkategorier, där varje underkategori har särskiljande egenskaper, såsom en viss typ av topologi eller en viss träningsalgoritm. En av dessa kategorier är feed forward-nätverk.

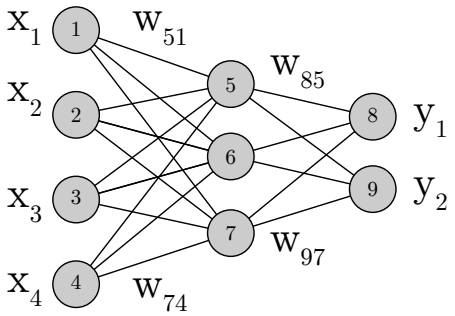
2.1.4 Feed forward-nätverk

Feed forward-nätverk är en underkategori av flerlagersperceptroner där signalerna i nätverket hela tiden matas framåt, aldrig bakåt eller i sidled [4, s. 51]. Topologin för ett feed forward-nätverk karaktäriseras därför av att den efterliknar en riktad, acyklisk graf.

Trots att det existerar betydligt mer komplexa klasser av flerlagersperceptroner [9, s. 181], är feed forward-nätverk ett populärt val i många olika typer av tillämpningar. De har visats besitta en kraftfull förmåga att hitta komplicerade samband i stora datamängder [10, s. 158].

I likhet med en generell flerlagersperceptron är neuronerna i ett feed forward-nätverk indelade i lager; ett inputlager, ett outputlager, samt ett antal gömda lager. Utgångsvärdena för neuronerna i ett visst lager är sammankopplade med ingångsvärdena för neuronerna i nästkommande lager. I de flesta fall är alla

neuroner i två angränsande lager parvis sammankopplade, det vill säga att två angränsande lager tillsammans bildar en komplett, bipartit graf. Om alla lager i neuronnätet har denna egenskap sägs nätet vara fullständigt sammankopplat, om så ej är fallet är nätet partiellt sammankopplat [4, s. 52].



Figur 10: Figuren visar ett feed forward-nätverk med topologin 4-3-2. Vikterna i figuren indexeras med index för viktens in- respektive utgångsvärde.

Vikter fyller samma funktion i ett feed forward-nätverk som i perceptronen. Varje koppling mellan utgångsvärdet för en neuron till ingångsvärdet för en annan neuron är associerat med en vikt.

Varje enskild neuron i nätet följer i princip samma modell som för perceptronens neuron, med endast ett undantag. Neuronerna i inputlagret är något förenklade, då de saknar bias och aktiveringsfunktion. Detta beror på att de endast har ett ingångsvärde vardera, och detta ingångsvärde bildar direkt inputneuronens utgångsvärde som överförs till neuronerna i nästa lager. I vissa modeller kallas inputneuronerna för inputnoder, för att markera att de inte fungerar på samma sätt som resten av neuronerna i nätet [4, s. 51].

Alla neuroner i ett feed forward-nätverk behöver inte ha samma aktiveringsfunktion. Däremot bör som tidigare nämnts den linjära funktionen används för neuronerna i outputlagret, förutsatt att nätet skall kunna tränas att approximera en godtycklig reell funktion.

För kunna referera till varje unik neuron och varje unik vikt i nätet används följande indexeringsmetod.

- En unik neuron n kan indexeras n_{ab}, där a är index för ett av neuronnätets lager, och b är neuronens index inuti sitt lager.
- y_{ab} betecknar utgångsvärdet för neuron b i lager a.
- w_{abcd} betecknar vikten associerad med kopplingen mellan neuron b i lager a och neuron d i lager c.
- En nätverkstopologi kan anges på formen $L = [l_1, \ldots, l_n]$, där l_n betecknar antalet neuroner i nätets n:te lager.

Det är nu möjligt att definera stegen i feed forward-algoritmen, som används för att beräkna utgångsvektorn, $\vec{y}$, för ett feed forward-nät.

Algoritmen använder samma metod som i ekvationer 3 och 4 för perceptronen, men för varje neuron i nätet. Algoritmen förutsätter att nätet är fullständigt

Algoritm 2 Feed forward

Require: $\vec{x} = [x_0, \ldots, x_m]$
Require: $L = [l_1, \ldots, l_n]$
Require: $\vec{w}$
  for $i := 0$ to m **do**
    $y_{0i} := x_i$
  end for
  for $c := 1$ to n **do**
    $a := c - 1$
    for $d := 0$ to l_c **do**
      $temp := 0$
      for $b := 0$ to l_a **do**
        $temp := temp + y_{ab} \cdot w_{abcd}$
      end for
      $y_{cd} := temp$
    end for
  end for

sammankopplat. Om nätet är partiellt sammankopplat kan värdet av w_{abcd} låtas vara 0 för de neuroner n_{ab} och n_{cd} som ej är sammanlänkade.

Att feed forward-nätverk har kapacitet att approximera godtyckliga, reella funktioner visade George Cybenko 1989 genom att formulera ett bevis för det universella approximationsteoremet [15].

$$F(x_1, x_2, \ldots, x_m) = \sum_{i=1}^{m_1} \alpha_i \phi \left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i \right) \tag{18}$$

Det universella approximationsteoremet är ett matematiskt teorem som säger att vilken reell funktion F som helst kan approximeras av ekvationen på höger sida om likhetstecknet i ekvation 18. Även om Cybenkos bevis inte involverar neuronnät så har Haykin visat att teoremet är direkt applicerbart på feed forward-nätverk [4, s. 197].

Givet en reell funktion $F(x_1, x_2, \ldots, x_m)$, samt värden på de fria variablerna i ekvation 18 så att ekvationen håller, kan ett feed forward-nätverk skapas som approximerar F på följande sätt.

1. Låt antalet inputneuroner vara m_0. Nätets ingångsvärde blir $[x_1, \ldots, x_{m_0}]$.
2. Låt nätet ha ett gömt lager med m_1 neuroner.
3. Låt $\alpha_1, \ldots, \alpha_m$ vara vikterna i det gömda lagret.
4. Låt ϕ vara nätets aktiveringsfunktion. Cybenko bevisade teoremet för sigmoidfunktionen [15, s. 10].

Således är det bevisat att ett feed forward-nätverk i teorin kan approximera villken reell funktion som helst. Utmaningen är att hitta en topologi för nätverket som passar just den funktion som skall approximeras, och att sedan lyckas träna nätet.

Att träna ett feed forward-nätverk involverar en högre nivå av komplexitet än för den enklare perceptronen. För att förstå varför är det viktigt att känna till att nätets träningsalgoritm behöver hitta optimala vikter mellan alla sammankopplade neuroner i nätet.

Anledningen till att LMS inte kan appliceras på feed forward-nätverk, eller någon annan typ av flerlagersperceptron, är att algoritmen förutsätter ett önskat utgångsvärde för alla neuroner vars vikter den justerar. Eftersom LMS erhåller detta önskade utgångsvärde ur träningselementen saknar algoritmen förmåga att justera vikterna för några neuroner förutom i outputlagret.

Att träna feed forward-nätverk och andra typer av flerlagerspereptroner var länge ett olöst problem som medförde att utvecklingen av neuronnät stagnerade. Detta problem löste Arthur Bryson och Yu-Chi Ho när de 1969 publicerade en upplärningsalgoritm kallad backpropagation-algoritmen [16].

2.1.5 Backpropagation-algoritmen

Backprogagation-algoritmen är en upplärningsalgoritm som kan användas för att träna feed forward-nätverk. Likt LMS är den en övervakad upplärnigsalgoritm, alltså använder den sig av träningselement som innehåller ett värde till nätets inputneuroner, samt ett önskat utgångsvärde från nätets outputneuroner. Den avgörande skillnaden från LMS är att backpropagation-algoritmen löser problemet att träna ett neuronnät med fler än en neuron [9, s. 159].

Backpropagation-algoritmen använder samma gradient descent-strategi som LMS, men den lägger till en extra teknik för att beräkna ett felvärde även för nätets gömda neuroner, vars önskade utgångsvärden inte är kända.

För en perceptron visades ∇E vara den partiella derivatan av nätets globala fel, E, med avseende på viktvektorn. ∇E kunde således användas för att justera hela nätets viktvektor samtidigt. Så är inte fallet för ett feed forward-nätverk, eftersom viktvektorn nu innehåller vikter för flera olika neuroner.

Beräkningen av ∇E är mer komplicerad för feed forward-nätverk än för perceptroner. För en perceptron innehåller den förändringen för nätets totala viktvektor: $[\frac{\delta E}{\delta w_0}, \ldots, \frac{\delta E}{\delta w_n}]$. I en flerlagersperceptron med många sammankopplade neuroner är det lättare att fokusera på enskilda komponenter i viktvektorn istället för att försöka träna hela nätet i ett svep. För ett feed forward-nät är det intressant att beräkna $\dfrac{\delta E}{\delta w_{ji}}$ [4, s. 160].

Vikten w_{ji} viktar ett utgångsvärde y_i skickat från neuron i till neuron j. På samma sätt som i LMS går det att givet ett felvärde justera vikten med hjälp av felvärdets partiella derivata. Denna partiella derivata kan analytiskt bestämmas till resultatet av ekvation 19 [4, s. 161].

$$\frac{\delta E}{\delta w_{ji}} = -e_j \cdot f'_j(v_j) \cdot y_i \tag{19}$$

I likhet med LMS kan upplärningstakten η användas för att styra viktförändringens storlek.

$$\Delta w_{ji} = -\eta \cdot \frac{\delta E}{\delta w_{ji}} \tag{20}$$

Ekvation 19 är mycket viktig, då den talar om hur en individuell vikt w_{ji} kan justeras, givet att neuronens aktiveringspotential v_j och lokala fel e_j är känt. Eftersom feed forward-fasen sker innan backpropagation-fasen finns aktiveringspotentialen redan "lagrad", och även utgångsvärdet från den föregående neuronen, y_i, är känt. Det lokala felet å andra sidan, är inte känt för de gömda neuronerna [4, s. 161].

För outputneuronerna kan det lokala felet beräknas enligt ekvation 12. Detta är möjligt tack vare att det önskade utgångsvärdet d är känt för var och en av dessa neuroner. Backpropagation-algoritmen löser avsaknaden av lokala fel i de gömda neuronerna genom att helt enkelt skicka outputneuronernas felvärden baklänges genom nätet, vilket också har gett upphov till algoritmens namn [4, s. 160].

För att kunna skicka neuronernas lokala fel bakåt i nätet på ett korrekt sätt måste ytterligare ett värde tas i beaktning; neuronens lokala förändringstakt. Den lokala förändringstakten benämns δ. Dess värde erhålls genom att räkna bort det inkommande ingångsvärdet ur ekvation 19 [4, s. 161].

$$\delta_j = e_j \cdot f'_j(v_j) \tag{21}$$

Förändringstakten för en gömd neuron j kan beräknas med hjälp av förändringstakterna för neuronerna i det framförliggande lagret enligt ekvation 22.

$$\delta_j = f'_j(v_j) \cdot \sum_k (\delta_k \cdot w_{kj}) \tag{22}$$

När δ är känt för en gömd neuron kan dess vikt justeras, tack vare att ekvation 19 kan formuleras om med avseende på den lokala förändringstakten [4, s. 161].

$$\Delta w_{ji} = \eta \cdot \delta_j \cdot y_i \tag{23}$$

Backpropagation-algoritmen kan nu beskrivas i pseudokod som ett antal avgränsade steg.

I algoritmen ovan körs en iteration av backpropagation. I likhet med LMS talar inte algoritmen om när träningen bör avslutas, utan detta lämnas som ett beslut till implementeraren [4, s. 169].

Värt att notera är att alla δ måste beräknas innan vikterna kan uppdateras. Att göra båda parallellt skulle leda till felaktiga värden, på grund utav att δ beräknas med hjälp av de aktuella vikterna.

Ett sätt att veta när det är dags att avbryta träningen är att beräkna nätets globala fel som summan av felen hos varje outputneuron och avbryta när det globala felet är noll, litet eller oföränderligt. Problemet med detta är att det globala felet är en funktion av nätets alla vikter, vilket för större nät blir en funktion i många dimensioner som antagligen har flera lokala minimum och andra egenskaper som försvårar träningen [9, s. 178].

För att lösa problemet med att avsluta träningen, samt öka chanserna att träningen får ett lyckat resultat, finns det ett antal tekniker och metoder man kan använda när backpropagation tillämpas i praktiken.

Algoritm 3 Backpropagation

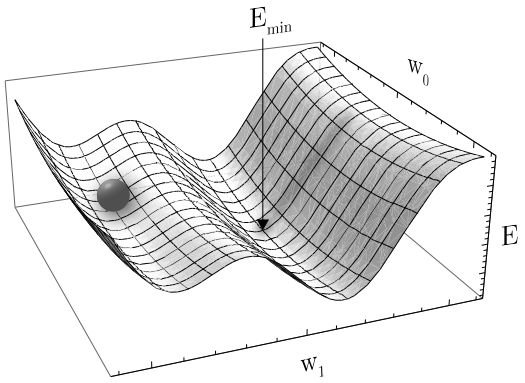
Require: $\vec{v}$
Require: $\vec{y}$
Require: $\vec{w}$
Require: $\vec{d}$
Require: $\vec{f}$
Require: $L = [l_1, \ldots, l_n]$
  for $i := 0$ to l_n **do** // Beräkna δ i outputlagret
    $e_{ni} := d_i - y_{ni}$
    $\delta_{ni} = e_{ni} * f'_{ni}(v_{ni})$
  end for
  for $a := n - 1$ to 1 **do** // Beräkna δ i de gömda lagren
    $c := a + 1$
    for $b := 0$ to l_a **do**
      $temp := 0$
      for $d := 0$ to l_{a+1} **do**
        $temp := temp + w_{abcd} \cdot \delta_{cd}$
      end for
      $\delta_{ab} := temp$
    end for
  end for
  for $a := n$ to 2 **do** // Uppdatera vikter
    $c := a - 1$
    for $b := 0$ to l_a **do**
      $temp := 0$
      for $d := 0$ to l_{a+1} **do**
        $temp := temp + w_{abcd} \cdot \delta_{cd}$
        $w_{abcd} := w_{abcd} + \eta \cdot \delta_{ab} \cdot y_{cd}$
      end for
    end for
  end for

2.1.6 Praktisk tillämpning

Vid praktisk tillämpning av backpropagation finns ett antal hinder på väget till att lyckas träna högkvalitativa feed forward-nätverk. Ett av dessa hinder är problemet med lokala minima.

Ett eller flera lokala minima i funktionen för neuronnätets globala fel kan leda till att träningen slutar ge resultat trots att nätet inte uppnått optimal approximationsförmåga. Detta beror på att gradient-descent metoden helt enkelt konvergerar i ett lokalt minimum ifall viktvektorn korsar det på vägen mot det globala minimat [4, s. 169].

För att minska risken att träningen konvergerar i ett lokalt minimum kan viktvektorn ges en viss rörelsemängd, även kallad momentum, som bevaras mellan iterationerna av träningsalgoritmen. Momentumvärdets påverkan kan beskrivas genom att tänka sig att förändringsvektorn för vikterna i en given iteration pekar mot ett lokalt minimum, men att "kulan" tar sig förbi detta tack vare momentum från tidigare iterationer [9, s. 195].



Figur 11: Figuren visar hur vikterna under upplärningen konvergerat i ett lokalt minimum. Genom att använda momentum hade "kulan" med större sannolikhet färdats förbi det lokala minimat och istället konvergerat i $E_{\min}$.

Ekvation 24 beskriver hur momentum används vid justering av en given vikt [9, s. 195].

$$\Delta w_t' = \Delta w_t + (\alpha \cdot \Delta w_{t-1}) \tag{24}$$

I ekvationen ovan utgör Δw_t justeringen av vikten w utan momentum. Momentum inkorporeras således genom att addera en fraktion av föregående viktjustering till den nuvarande viktjusteringen. Värdet av denna fraktion benämns alpha och bör ligga mellan 0 och 1 [4, s. 168]. Detta värde kan i sig hänvisas till som träningens momentum.

Inte heller genom att använda momentum går det att garantera att träningen konvergerar, även om chanserna ökar [4, s. 168]. Ett problem som måste överkommas under träningen är att stora neuronnät tenderar att överanpassas om de tränas under för många träningsepoker. Med överanpassning menas att neuronnätet i grund och botten "memorerar" träningselementen istället för att approximera den underliggande funktionen [9, s. 153].

Överanpassning kan motverkas genom en metod som kallas early stopping. Early stopping bygger på att en del av elemeneten i träningsmängden T reservereas för att mäta neuronnätets globala fel efter varje träningsepok, elementen som reserveras till dessa benämns därför valideringsdata. Eftersom neuronnätet inte tränas på elementen i valideringsdatan kan den användas för att mäta när träningen inte längre påverkar nätets approximationsförmåga. När detta inträffar kan träningen avbrytas [4, s. 204].

Träningen påverkas även av vilka värden som de fria parametrarna utöver vikterna sätts till, dessa kan till exempel vara upplärningshastigheten η och momentum α. För att uppnå bästa resultat kan samma nät tränas med olika värden på dessa parametrar.

För att mäta skillnaden i approximationsförmåga mellan flera nät som tränats på samma data, kan en del av träningsdatan läggas undan som testdata. Denna testdata används för att mäta nätens globala fel efter att träningen slutförts. Det globala felet på testdatan kan således användas som en indikator för vilket av flera nät som har bäst approximationsförmåga [4, s. 204].

2.2 Aktiehandel

En aktie är en andel i ett aktiebolag [17, s. 164]. Anledningen till att aktiebolaget är en så populär bolagsform för större bolag är dels att varje aktieägare enbart svarar för sitt investerade kapital och således inte kan förlora mer än så vid en konkurs [17, s. 165], men kanske framförallt att handel med aktier är snabb och enkel även för gemene man via exempelvis internet [17, ss. 137-139].

Det kan finnas flera skäl till att handla med aktier. Att aktieägare får beslutsrätt i företaget är sällan det mest attraktiva med aktiehandel, då en eller ett fåtal aktier (och motsvarande röst/er) ofta inte gör någon skillnad då det finns tusentals aktier eller mer. Att få del i den vinst företaget gör kan vara attraktivt på lång sikt, men det som är mest attraktivt är att göra vinst på själva aktiehandeln. När aktier utfärdas (antingen vid nyemission eller bolagsbildning) har de ett värde som svarar mot det kapital de innebär för företaget, men därefter bestäms deras värde av marknaden. Om spekulanterna tror att företaget kommer att gå bra och därmed göra vinst är dess aktier mer attraktiva och kommer således öka i värde vilket innebär att aktieägare har som mål att hitta aktier man tror kommer öka i värde, köpa dem och sedan sälja till ett högre pris efter en tid.

2.2.1 Börsen

I Sverige finns tre börser, varav två aktiebörser. Nasdaq OMX Stockholm för etablerade bolag är den största och mest kända, medan NGM Equity är en mindre börs avsedd för tillväxtbolag. Den tredje, Burgundy, erbjuder inte handel med aktier.

Ordet "börs" är i Sverige en skyddad beteckning och får bara användas av de handelsplatser som fått tillstånd av Finansinspektionen att bedriva börsverksamhet. Syftet med tillståndet är att verifiera att allmänheten kan lita på börsen och dess ingående företag, med avseende på redovisning exempelvis, vilket uppmuntrar handel. Att ett bolag är börsnoterat innebär således att det genomgått de granskningar och uppfyller de krav som ställs på dem av respektive börs.

På Nasdaq OMX Stockholm återfinns därför huvudsakligen bolag med mycket aktiv aktiehandel [18].

2.2.2 Courtage

Privatpersoner får inte handla direkt på börserna, utan måste bedriva sin handel genom en mäklare. Denna tar vanligtvis ut en avgift, ett courtage, för varje transaktion. Courtaget är ofta en procentsats av hela beloppet i affären men med ett minimicourtage för att även små affärer skall generera en märkbar inkomst för mäklaren [17, s. 364].

2.2.3 Öppnings- och stängningscall

Varje handelsdag påbörjas och avslutas med ett så kallat call, där handel inte sker på samma sätt som under resten av handelsdagen. En call är till för att samla ihop köp- och säljbud, för att sedan räkna ut ett jämviktspris till vilket flest antal transaktioner kan ske. Öppningscallet startar 08:45 och då är det möjligt att lägga ordrar för samtliga aktier, men själva handeln börjar genomföras först klockan 09:00. Stängningscallet startar vid 17:25 och därefter avslutas handeln för kvarliggande ordrar i bolagens storleksordning fram till klockan 17:30. De jämviktspriser (ett för varje aktie) som räknas fram av morgoncallen används som öppningspriser och de jämviktspriser som räknas fram av stängningscallen används som stängningspriser [19].

2.2.4 Volym

Antal värdepapper som omsätts. Det vill säga antalet aktier som bytt ägare under en tidsperiod. Ofta är tiden så länge som börsen varit öppen nuvarande dag. Vanligtvis menar man alltså antalet aktier som bytt ägare under börsdagen [17, s. 411].

2.2.5 Orderdjup

Orderdjupet anger antalet köp- respektive säljbud som finns på varje budnivå för en given aktie. Till exempel kan det finnas 1000 aktier med säljbud på 14.10 kr och 800 aktier med köpbud på 14.00 kr. I detta fallet finns då orderdjupen 1000 för 14.10 kr och 800 för 14.00 kr [20].

2.2.6 Blankning

Som spekulant är det även möjligt att göra vinst vid nedgångar på börsen. En spekulant som tror att en aktie kommer att minska i värde kan låna ett antal av dessa aktier av någon som innehar dem. Tanken är att denne säljer aktierna omedelbart för att vid ett senare tillfälle, när aktien minskat i värde, köpa tillbaka dem och returnera till långivaren. Om aktien faktiskt minskar i värde kommer spekulanten att vid köp av den behöva lägga ut mindre pengar än denne fick in vid försäljning och således ha gjort vinst. Denna proccess kallas blankning [17, s. 360].

2.2.7 Prissättning

När ett aktiebolag bildas eller beslutar att utfärda nya aktier (så kallad nyemission) har aktierna ett värde som bestämts av bolaget, det nominella värdet. Transaktioner vid nyemission och bolagsbildning utgör dock en minoritet av alla transaktioner. Den vanligaste transaktionen är överlåtelse av aktier mellan aktieägare. Vid sådana transaktioner bestäms priset av utbud och efterfrågan vilket betyder att priset kan vara helt orelaterat till det verkliga värdet på bolaget.

Detta kan självklart leda till enorma överpriser och i slutändan ekonomiska kollapser. Ett tydligt exempel på detta är den så kallade tulpanhysterin i Holland under 1600-talet.

Något så underligt som tulpanlökar blev extremt populärt på kort tid och detta resulterade i en enorm "tulpanhysteri" men efter månader utav uppgång så började priset sjunka i snabb takt, på en månad sjönk priset ifrån 5500 cent till 50 cent. Detta visar att prissättningen inte är knuten till det faktiska värdet [17, ss. 20-23].

2.2.8 Psykologi

Den ideala investeraren beter sig rationellt - denne verkar oberoende av andra och omvärderar alltid sin analys av marknaden då ny information görs tillgänglig. Exempelvis det sista är något som inte nödvändigtvis är sant i verkligheten, då faktumet att folk tenderar att värdera kortsiktig data mer än långsiktig observerats i studier [21, s. 17]. Även det första, att investerare verkar oberoende av varandra, är inte nödvändigtvis sant, vilket demonstreras med exempelvis tulpanhysterin i Holland som nämndes ovan.

Flera andra exempel på övertro och flockbeteenden finns, exempelvis "The South Sea Bubble" i England på 1700-taglet [17, ss. 23-25] och IT-bubblan på 90-talet [17, ss. 41-63]. Ytterligare ett "irrationellt" beteende som observerats är att människor tenderar att ha svårare för att sälja än att köpa [17, s. 304].

2.3 Aktieanalys

Det finns ett flertal metoder för att analysera aktiemarknaden och utifrån dessa analyser spekulera i hur marknaden kommer se ut i framtiden.

2.3.1 Fundamental analys

Målet med fundamental analys är att utvärdera företags framtida tillväxt- och vinstpotential. Detta görs med exempelvis års- och kvartalsrapporter som grund. Även att följa nyhetsflödet och informera sig om företagens kommande investeringar, är en del av denna metod [17, ss. 192-193].

2.3.2 Teknisk analys

Teknisk analys tar till skillnad från den fundamentala analysen inte hänsyn till respektive företags vinstpotential utan använder sig utav tidigare aktiedata för att förutse framtida kurser.

Teknisk analys bygger på att man identifierar mönster i aktiehandeln i förväg, antar att de kommer gälla även i framtiden och baserar sina förutsägelser på matematiska modeller som konstruerats utifrån dessa mönster.

Att analysera börsen med hjälp av algoritmer är ett omdebatterat ämne då man är oense om huruvida historisk marknadsdata överhuvudtaget kan användas för att förutsäga dess framtida beteende [17, ss. 244-245].

Det finns dock exempel på tekniska strategier som under avgränsade tidsperioder visats hålla vilket talar för att teknisk analys kan fungera [17, s.246].

2.3.3 Aktieanalys med hjälp av artificiella neuronnät

Artificiella neuronnät har visat sig mycket lämpliga att approximera funktioner och modeller genom att enbart analysera tidsserier [1][22].

Nackdelen med tekniska modeller är att de vilken dag som helst kan sluta fungera om förutsättningarna på marknaden förändras. Här har neuronnät en klar fördel i och med dess förmåga att identifiera godtyckliga mönster i datan utan de antaganden som existerar inom klassisk teknisk analys [22]. Neuronnät lämpar sig därför bra för teknisk analys och har också använts flitigt för detta ändamål i snart två decennier [23][24][25].

2.3.4 Hypotesen om den effektiva marknanden

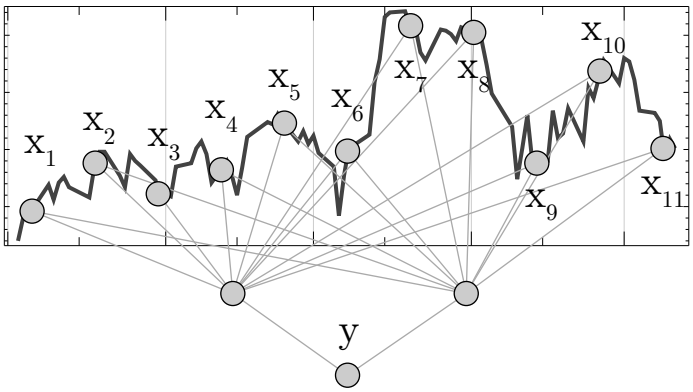
Hypotesen om den effektiva marknaden, Effecient Market Hypothesis [26] (hädanefter EMH) beskriver marknaden som effektiv, vilket innebär att all tillgänglig information redan återges i aktiernas pris och att marknadens rörelser följer ett slumpmässigt mönster, vilket Maurice Kendall påvisat år 1953 [17, ss. 323-325].

Något som talar emot att marknaden är effektiv, det vill säga att den i någon grad är förutsägbar, är historiskt framgångsrika aktiehandlare som beskriver hur de baserat sina förutsägelser på ett antal enkla system. Dessa system är huvudsakligen baserade på att börsen påverkas av flockbeteenden och liknande psykologiska fenomen [27].

EMH-förespråkare förklarar dock detta med att även om det för individen är osannolikt, måste någon göra vinst även i det långa loppet. Paralleller kan dras till en tänkt turnering i myntkastning, där ett godtyckligt stort antal deltagare möter varandra en mot en och slår ut varandra genom att gissa vilken sida av myntet som kommer att hamna uppåt. Även om chansen att vinna turneringen är försvinnande liten för individen, kommer någon att vara vinnare i slutändan [17, ss. 323-325].

Werneryd [21, s. 21] nämner några av de viktigaste punkterna i kritiken mot EMH: 1. All information är inte publik och all information är inte omedelbart tillgänglig på grund av olika fördröjningar. 2. Investerare är inte nödvändigtvis rationella. 3. Investerare är inte nödvändigtvis oberoende av varandra, det kan finnas flockbeteende. 4. Tidsperspektivet; För längre tidsperioder kan utvecklingen för en given aktie vara närmare slumpen än för korta tidsperioder.

Även om ett neuronnät lyckas finna trender i data betyder det inte att trenderna är relevanta eller att dessa fungerar för data som neuronnätet inte tränats på [2]. För att förbättra chanserna att hitta relevanta trender i en komplex modell bör man behandla datan i förväg genom att bland annat filtrera ut onödig information [28].



Figur 12: Figuren visar ett möjligt sätt att konstruera ett feed forward-nätverk som approximerar tidsserien för en aktie. Som ingångsvärden ges på varandra följande samplingar ur tidsserien. Det önskade utgångsvärdet defineras som nästkommande sampling. Om den sista indatasamplingen är tagen direkt från aktuell börsdata kan nätet potentiellt approximera tidsseriens framtida värde.

3 Systemdesign och implementation

I korthet kan systemet beskrivas som att det tillhandahåller följande funktionalitet för användaren:

- Träning av neuronnät.
- Utvärdering av pågående och planerade träningar.
- Utvärdering av färdigtränade neuronnät.
- Simulerad handel med hjälp av färdigtränade nät

För att tillhandahålla denna funktionalitet för användaren utför även systemet kontinuerlig insamling och lagring av aktiedata, vilket är dolt för användaren.

Systemets samtliga funktioner är fördelade över fem systemmoduler, som även kan ses i figur 13:

- Datamodul
- Beräkningsmodul
- Webbservermodul
- Gränssnittsmodul
- Simuleringsmodul

Var och en av dessa moduler är därtill indelad i mindre submoduler.

Tillsammans bildar modulerna ett system som körs på en centraliserad server. Samtliga moduler förutom gränssnittmodulen är utvecklade i programspråket Java. Gränssnittsmodulen använder en kombination av HTML, CSS och Javascript.

3.1 Datamodul

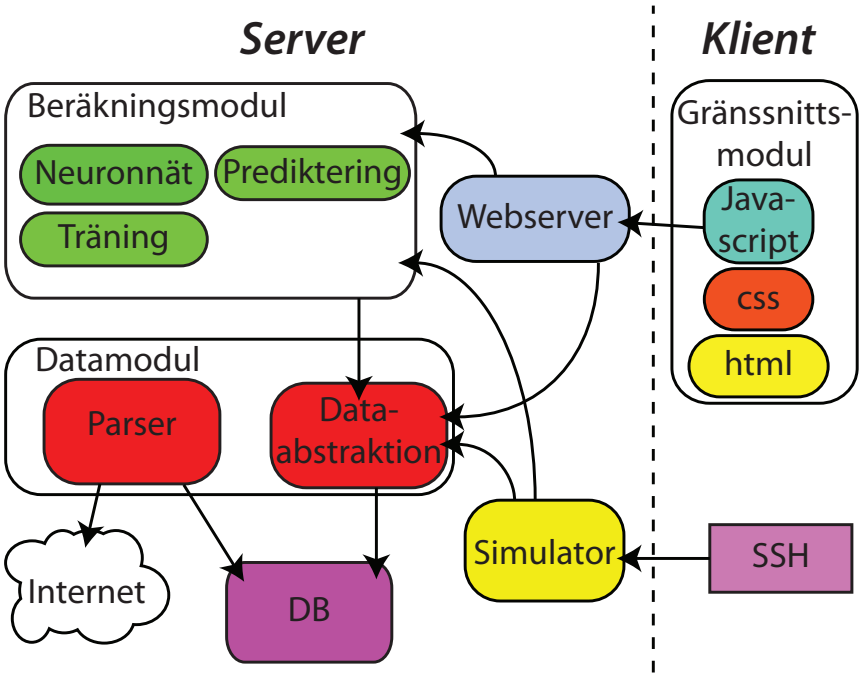
Systemets datamodul inkapslar och tillhandahåller all funktionalitet som involverar insamling, lagring och åtkomst av aktiedata.

3.1.1 Insamling

Submodulen som sköter insamling av data innehåller ett antal webbspindlar, även kallade webcrawlers. Dessa webbspindlar parsar med jämna intervall ett antal externa webbsidor som tillhandahåller kontinuerligt uppdaterad data.

Aktiedatan samlas in från Nasdaq OMX Nordic [29], som tillhandahåller realtidsdata från Stockholmsbörsen med 15 minuters fördröjning. Börsen har öppet från klockan 09:00 till 17:30 och hemsida. Webbspindeln hämtar data för samtliga large cap-aktier som finns tillgängliga via Nasdaq OMX Nordic.

Varje insamlad datasampling innehåller information om aktuellt försäljningspris och orderdjup. Orderdjupet innehåller de fem mest populära offererade priserna för köp och sälj och för var och en av dessa; antalet ej genomförda ordrar.



Figur 13: Översikt av systemets moduler.

En annan webbspindel samlar in valutadata. Denna data hämtas från en webbsida vid namn CurrencyRates [30] och består av de relativa priserna för brittiska pund (GBP), amerikanska dollar (USD), Euro (EUR) och svenska kronor (SEK).

Webbspindeln för aktier hämtar samplingspunkter med 10-sekunders intervall, under de tider då Stockholmsbörsen är öppen. Spindeln för valutadata hämtar samplingspunkter med 10-sekunders intervall dygnet runt.

3.1.2 Lagring

Varje insamlad samplingspunkt lagras i en relationsdatabas. Databasschemat innehåller en tabell per aktie eller valuta som samlas in och varje rad i tabellerna innehåller en samplingspunkt tillsammans med en tidsstämpel som talar om exakt när samplingen samlades in. Tidsstämpeln representeras på det generiska UNIX timestamp-formatet.

Förutom insamlad data så hanterar datamodulen även lagring av data relaterad till systemets neuronnät. Denna data inkluderar främst parameterkonfigurationer för färdigtränade nät, men även deras resulterande förutsägelser.

3.1.3 Åtkomst

Datamodulen innehåller logik som förenklar åtkomst av den lagrade datan för övriga delar av systemet. Internt används JDBC (Java Database Connectivity) för att kommunicera med databasen via SQL, och datamodulen kapslar helt in direktkommunikationen med databasen.

Systemet kan returnera alla tidpunkter mellan klockan 09:00 och 17:25 då börsen annars är stängd eller är i en call (se avsnitt 2.2.3).

Gränssnittet som tillgängliggör de insamlade datapunkterna använder sig av interpolering för att möjliggöra dataförfrågningar med godtyckliga start- och slutdatum, samt med godtyckliga samplingsintervall. Detta åstadkoms genom att interpolera linjärt mellan de insamlade samplingspunkterna, vilket medför att data för vilken tidpunkt som helst kan returneras.

Om en önskad tidpunkt inte finns bland de insamlade samplingarna databasen så interpolerar gränssnittet linjärt mellan de två kringliggande punkterna. Detta skapar en enkelhet hos resten av systemet som inte behöver ta hänsyn till vilken data som faktiskt finns lagrad.

Utöver åtkomst av faktiska samplingvärden stöder datamodulen även grundläggande förbehandling av datapunkterna, även kallad feature extraction. Följande förbehandlingsmetoder är implementerade:

- Differens
- Signumdifferens
- Promilleförändring

Differens är en förbehandling som returnerar, istället för de faktiska samplingsvärdena, differensen mellan samplingsvärdet för den önskade tidpunkten och samplingsvärdet för den omedelbart föregående tidpunkten.

Signumdifferensen använder datan för samplingspunktens differens och normaliserar den ytterliggare med hjälp av signumfunktionen. Signumdifferensen är således antingen 1, -1 eller 0.

Promilleförändringen är ett mått på hur många promille samplingsvärdet förändrats sedan föregående samplingspunkt. Promille används istället för procent för att få en lagom numerisk skala för många olika längder på samplingsintervallet.

3.2 Beräkningsmodul

3.2.1 Generering

Neuronnät skapas med hjälp av en intern genereringsmodul, som möjliggör för systemet att generera en stor mängd neuronnät med så lite ansträngning från användaren som möjligt. Gränssnittet mot genereringsmodulen är enkelt; givet en genereringsspecifikation genereras en uppsätting konfigurationer.

En konfiguration är en unik parameteruppsättning avsedd att användas för att bygga ett neuronnät. En konfiguration innehåller således neuronnätets inputvärden, outputvärden, upplärningshastighet, aktiveringsfunktioner, träningsdata, interna struktur med mera.

En genereringsspecifikation är en generaliserad konfiguration, där de numeriska parametrarna kan anges i intervall, och icke-numeriska värden kan anges i listor. Varje intervall innehåller ett startvärde, ett slutvärde, en interpoleringsmetod och antalet önskade punkter. De tillgängliga interpoleringsmetoderna är linjär interpolering, där avståndet mellan punkterna är konstant, samt logaritmisk interpolering, där avståndet mellan punkterna ökar exponentiellt.

Genereringsmodulen sätter sedan samman alla möjliga kombinationer av elementen i listorna och de interpolerade numeriska värdena till en uppsättning konfigurationer.

Exempelvis kan en genereringsspecifikation innehålla en lista av två separata aktiveringsfunktioner till neuronnätens gömda lager och ett numeriskt tal med med tre logaritmiskt interpolerade samplingar för upplärningshastigheten. Utifrån detta bildar genereringsmodulen sex separata konfigurationer.

Genereringsmodulen bör användas med försiktighet i och med att antalet konfigurationer som en generationsspecifikation ger upphov till är exponentiellt korrellerat till antalet element i listorna och interpoleringar som ställts in för de numeriska värdena.

3.2.2 Träning

När ett nätverk skall tränas har det tillgång till sin egen konfiguration och således till vilken data den behöver. Den har även tillgång till databasgränssnittet och börjar med att hämta all data till lokala arrayer för snabb åtkomst. All data delas sedan in i tre grupper, träningsdata, valideringsdata och testdata, utefter kvoterna definierade i konfigurationen.

Träningsdatan används för att lära upp nätverket, det är denna datan den lär sig att approximera en funktion för. I jämna intervall körs även ett test som mäter hur bra nätverket presterar på valideringsdatan. Detta används för att avgöra om nätverket är överträngt och hur bra det anpassar sig till data den inte tränat på. När algoritmen avgör att det är dags att sluta träna, det vill säga när nätverket ej anpassar sig märkbart längre, görs en körning mot testdatan. Detta för att få ett mätvärde på hur bra nätverket är på att förutsäga data det aldrig har sett förut.

3.3 Webbservermodul

Webbservermodulens uppgift är att tillgängliggöra de övriga modulernas funktionalitet över HTTPS-protokollet. För att webbservermodulen ska kunna tillhandahålla informationen krävs det att den har kännedom om de övriga modulerna. I och med detta kan webbservermodulen både leverera data till användaren samt ta emot data, vilket sker genom GET respektive POST förfrågningar (så kallade requests).

3.4 Gränssnittsmodul

Gränssnittsmodulen använder sig bland annat utav javascriptbiblioteket jQuery, som förenklar modifiering utav HTML, händelsehantering och AJAX-funktionalitet.

3.4.1 Träningsvyn

Träningsvyn utgör ett gränssnitt mellan användaren och systemets genereringsmodul för neuronnät. Denna vy utgörs således av ett omfattande webbformulär, där användaren kan mata in all nödvändig information för att bilda en genereringsspecifikation till genereringsmodulen.

Eftersom de fria parametrarna i neuronnät i många fall är korrelerade till varandra, till exempel aktiveringsfunktioner och deras lutningar, använder

träningsvyn Javascript och jQuery för att dynamiskt generera formulärets delar progressivt, där nästa del av formuläret kan genereras med hänsyn till användarens val i tidigare delar av formuläret.

3.4.2 Statusvyn

Konfigurationsuppsättningen som skapas i beräkningsmodulen efter att användaren genomgått träningsvyn kan innehålla en mängd neuronnät som påbörjar träningsfasen i omgångar. Dessa neuronnät, både de som påbörjat träning och även de som väntar på att påbörja träning finns under statusvyn, bland annat i tabellform. Användaren kan sortera eller söka efter neuronnäten beroende på dess parametrar och även radera dem.

I och med att funktionaliteten för den ursprungliga HTML-tabellen är relativt enkel så använder sig gränssnittsmodulen utav ett jQuery-bibliotek, datatables.js, som ger tabellen egenskaper som till exempel sortering och sökning av rader med mera.

Användaren har möjlighet att välja ett neuronnät i tabellen och utvärdera prestationen genom att undersöka dess utveckling som finns tillgänglig i form av två grafer som beskriver hur nätet anpassar sig.

Den ena grafen visualiserar ett mått på nätets fel i förhållande till det korrekta förväntade värdet. Algoritmen som mäter felet är root mean squared [31].

Den andra grafen är baserad på ökning/minskning och ger högre fel om nätet förutser värdeökning när värdet sjunker eller värdeminskning när värdet ökar. Den andra grafen ser alltså inte till det faktiska utvärdet av nätverket utan endast ökning/minskning gentemot nuvarande värde. Att denna graf visar ett fel lägre än 0.5 innebär att nätverket gissar rätt riktning oftare än det gissar fel.

I de fall där användaren upptäcker att ett neuronnät presterar dåligt, finns det möjlighet att manuellt stoppa träningen.

Graf-funktionaliteten är implementerad med hjälp av en modifierad version utav g.raphael, som är en utbyggnad av javascript-biblioteket Raphael.js, vars uppgift är att skapa SVG-bilder direkt i webbläsaren. Grafbiblioketet utnyttjar detta till att rita kurvor i form utav SVG-bilder.

3.4.3 Administreringsvyn

Administreringsvyn följer en liknande princip som statusvyn, men istället för att visa neuronnät som för tillfället är under träning visas här istället neuronnät som är färdigtränade.

3.5 Simuleringsmodul

Modulen simulerar handel med olika aktier. Simulatorn kan använda sig av godtyckliga prediktorer för att avgöra hur den ska agera. En prediktor kan till exempel vara ett färdigtränat neuronnät.

En simulator startas med en portfölj som initialt innehåller en miljon obundna kronor och noll aktier.

Simulatorn genomför transaktioner i samma intervall som prediktorn ger förutsägelser. Varje handelstillfälle börjar med att prediktorn förutsäger värdet på aktien vid nästa handelstillfälle. Om det förutsagda värdet skiljer sig med

mer än ett öre från nuvarande värde på aktien kommer simulatorn antingen att försöka köpa aktier (vid uppgång) eller sälja aktier (vid nedgång). Simulatorn investerar om möjligt alla tillgängliga medel vid transaktionerna.

Simulatorn kan köras med flera olika alternativ. Det är möjligt att handla med/utan courtage, transkationsmarginal och blankning. Courtage är satt till 0,055 % av transaktionssumman eller 99 kr minimum. Transaktionsmarginal innebär att simulatorn endast genomför affärer med potentialen att generera en vinst större än en viss (variabel) andel av transaktionssumman.

Blankning innebär att simulatorn har möjligheten att blanka genom att låna aktier till ett värde av dess tillgångar. Det antas här att det alltid finns tillgång till aktier att låna, omedelbart och utan utlåningsränta eller andra avgifter.

Köp sker genom att simulatorn itererar igenom de olika nivåer av säljbud som finns. På varje budnivå avgör simulatorn hur många aktier den kan köpa med tillgängliga medel (det vill säga obundna pengar), om möjligt genomförs transaktionen. Om blankning är aktiverat och simulatorn har lånat aktier betalas dessa tillbaka omedelbart. Kan inte simulatorn genomföra någon transaktion på budnivån eller om alla budnivåer understigande aktiens förutsagda värde vid nästa handelstillfälle har gåtts igenom vilar den tills nästa handelstillfälle.

Vid försäljning gås på motsvarande sätt som vid köp de olika nivåerna av köpbud igenom. På varje budnivå avgör simulatorn hur många aktier den kan sälja och om möjligt genomförs transaktionen. Om blankning är aktiverat lånar (om möjligt) simulatorn aktier att sälja då den sålt slut på egna aktier och fortsätter sedan sälja. Kan inte simulatorn genomföra någon transaktion på budnivån eller om alla budnivåer överstigande aktiens förutsagda värde vid nästa handelstillfälle har gåtts igenom vilar simulatorn tills nästa handelstillfälle.

Observera att courtage utgår på varje transaktion (det vill säga ej endast vid varje handelstillfälle). Med en transaktion avses här samtliga genomförda köp eller försäljningar vid en viss budnivå och ett visst handelstillfälle.

Simulatorn använder sig av dataabstraktionslagret för att komma åt data om aktier, dock med ett mellanliggande transaktionshanteringslager. Lagret håller reda på vilka köp eller försäljningar som genomförts vid ett handelstillfälle så att det inte är möjligt att sälja/köpa flera gånger till samma köpare/säljare vid samma handelstillfälle.

Simulatorn skriver ut status om simulationen på skärmen och generar diagram i form av länkar till Google Charts.

Simulatorn körs på servern, och för att kunna köra den behöver man vara inloggad (exempelvis via SSH). Simulationer körs ej per automatik för färdigtränade nät utan måste startas manuellt.

Simulatorn kan även utnyttja en slumpmässig prediktor. Simulatorn kommer då istället för att använda ett neuronnäts förutsägelser till grund för sina beslut slumpa fram förutsägelser och använda dessa som grund vid beslut. Denna slumpprediktor kommer att ange priset vid nästa handelstillfälle till 1% över, 1% under eller samma som det nuvarande priset med $\frac{1}{3}$ sannolikhet för samtliga alternativ.

4 Resultat

Resultaten är naturligt indelade i två större delavsnitt, 4.1 som avhandlar det system som tagits fram för experimentering med neuronnät och 4.2 som avhandlar de neuronnät som tagits fram och deras resultat vid träning och simulering.

4.1 Slutgiltigt system

Den slutgiltiga versionen av systemet är ett mjukvarusystem vars ändamål är att generera och träna artificiella neuronnät för att förutsäga aktiekurser samt att testa dessa genom att simulera aktiehandel baserad på nätens förutsägelser. Med systemet kan en användare generera artificiella neuronnät av typen feed-forward, se de nät som redan finns i systemet och deras egenskaper, administrera näten samt starta simulationer.

4.1.1 Datainsamling och datahantering

De webbspindlar som använts till datainsamling har sedan början av februari samlat in sampel av aktie- och valutakursdata. Insamlingen av valutadata avbröts den 17 april då källan som valutaspindeln hämtade data från upphörde att existera. Valutaspindeln kraschade även ett antal gånger dessförinnan.

Totalt har data för 144 aktier samlats in, dessa aktier utgör samtliga large-cap-bolag på OMX Nordic. Antalet datapunkter varierar mellan aktierna på grund av problem vid insamlingen. 2012-05-14 befinner sig spannet av antalet insamlade datapunkter mellan 400 000 och 500 000.

4.1.2 Generering och träning av neuronnät

Genom systemets webbgränssnitt kan användare skapa nya neuronnät. Användaren får specificera en rad olika inställningar antingen genom en guide, eller genom att manuellt skriva inställningarna på JSON-format i ett textfält. De inställningar som användaren kan göra är:

Utdata Vilken utdata näten ska tränas att ge, det vill säga vilken akties kurs näten skall förutsäga och på vilken form aktiens kurs ska förutsägas; differensen, signumdifferensen, promilleförändringen (se 3.1.3 Åtkomst) eller den faktiska kursen.

Indata Vad näten ska ta som indata, det vill säga vad näten ska basera sina förutsägelser på.

Intervall Tiden mellan varje sampling som plockas ur dataserien och används till indata. Intervallet är även avståndet till den punkt i framtiden nätet försöker approximera data för.

Steg Hur många steg bakåt i tiden som näten ska ta indata. Parametern intervall anger tiden mellan varje steg.

Dolda lager Hur många dolda lager näten kommer ha samt hur många neuroner varje dolt lager skall ha.

Aktiveringsfunktion Vilka aktiveringsfunktioner som neuronerna i de dolda lagren kommer att använda.

Lutning på aktiveringsfunktion Aktiveringsfunktionerna sigmoid och tangens hyperbolicus har en konstant som avgör deras lutning.

Momentum Vilka momentum som skall användas av backpropagation-algoritmen under träning.

Upplärningshastighet Vilka upplärningshastigheter som skall användas av backpropagation-algoritmen under träning.

Träningsdatans start och slut Perioden vars data kommer att användas vid träningen av näten.

Träningsdatafraktion Hur stor andel av datan i träningsperioden som ska användas till träning.

Valideringsdatafraktion Hur stor andel av datan i träningsperioden som ska användas till validering.

Minsta antal träningsepoker Det minsta antalet träningsepoker ett neuronnät ska tränas innan träningen kan avrbrytas genom early stopping.

Stoppfaktor Hur många träningsepoker neuronnätet fortsätter att tränas innan den avbryts om inte resultatet av träningen förbättras i förhållande till hur många träningsepoker som redan körts.

För inställningarna momentum, upplärningshastighet, aktiveringsfunktion och lutning på aktiveringsfunktion kan användaren välja mer än ett värde. Systemet kommer då att generera ett neuronnät för alla möjliga kombinationer av användarens valda värden för dessa inställningar.

4.1.3 Administrering av neuronnät

Systemets webbgränssnitt har två vyer för administration av neuronnät, en för färdigtränade neuronnät och en för neuronnät som ännu inte tränats färdigt. För alla neuronnät visas deras konfiguration, hur länge de har tränat (både i tid och antal träningsepoker) samt hur de har presterat i träningen. Utöver att inspektera existerande neuronnät kan användaren genom systemets webbgränssnitt söka efter neuronnät, avbryta träningar och radera neuronnät.

4.1.4 Visualisering av förutsägelser

Webgränssnittet inehåller en påbörjad vy vars syfte är att visualisera statistik från handelsimulering som kör i bakgrunden. Vyn ska för varje färdigtränat neuronnät visa en graf med den faktiska kursen för aktien som nätet är tränat att förutsäga tilsammans med en graf över nätets förutsägelser. Denna vy är ej färdigställd och kan ej användas.

4.1.5 Simulator

Det är möjligt att utvärdera färdigtränade neuronnät genom att simulera aktiehandel med dessa som grund för beslutsfattande.

Simuleringen är ej tillgänglig via webbinterfacet utan startas manuellt och konfigureras beroende på syftet med simuleringen:

Courtage anges ifall användaren vill testa hur neuronnätet skulle prestera i en mer verklighetsenligt miljö, där courtage utgår för varje transaktion.

Tröskelvärde anger hur riskfyllda köp/säljtransaktioner simulatorn ska genomöra. Ett högre tröskelvärde innebär att det krävs en högre vinstpotential för att simulatorn ska genomföra transaktionen.

Blankning avgör ifall simulatorn ska ha möjlighet att blanka.

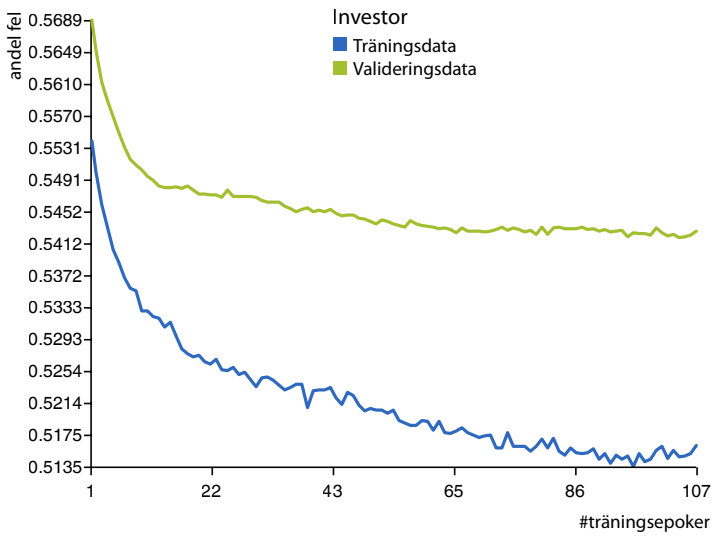
Startdatum anger vilket datum simulationen ska starta på.

Slumpförutsägelser anger att simulatorn ska använda sig av slumpförutsägelser.

4.2 Neuronnät

Nedan presenteras resultaten för ett urval av de nät som togs fram under projektets gång.

4.2.1 Träning



Figur 14: Figuren visar andelen av träningsdatan och valideringsdatan vars utdata förutsägs gå åt rätt håll. Indatan är Investors akties värde och utdatan detsamma.

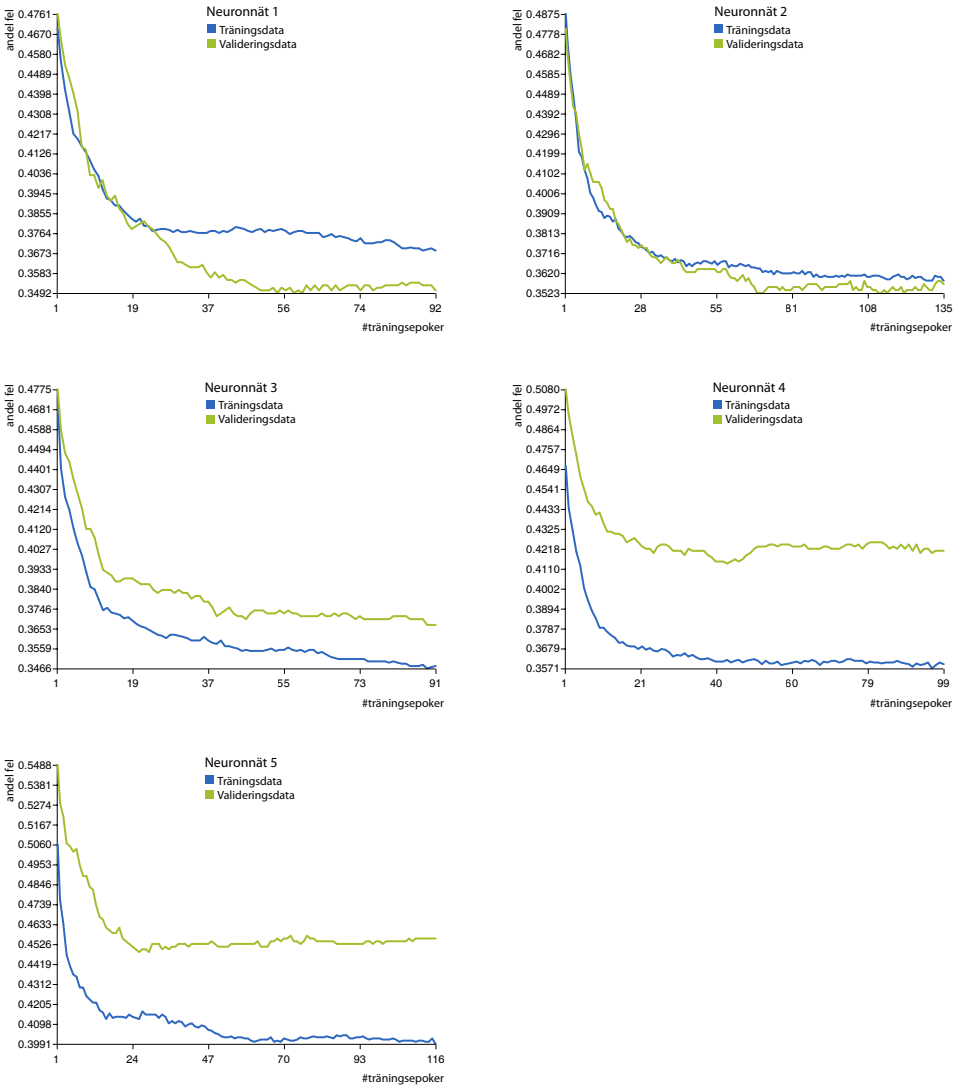
Resterande resultat presenteras för träningar på svenska banker då denna branch var den med flest aktörer och mest komplett data i databasen.

4.2.2 Simulering

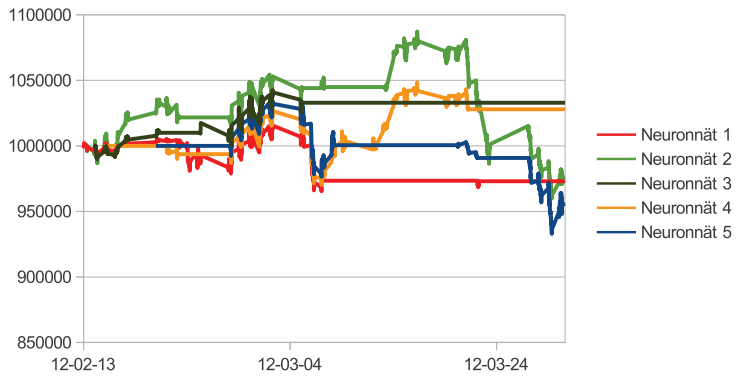
Redovisade simuleringar har körts på neuronnäten som förutsäger Nordeas aktievärde då dessa var de som gav bäst och stabilast resultat under träningen.

Parameter	**Värde**
Upplärningshastighet	0.0005
Aktiveringsfunktionslutning	0.07
Aktiveringsfunktion	Tangens
Momentum	0.9
Intern struktur	Två dolda lager med n/2 neuroner var. Där n är antalet indataneuroner
Tidsintervall	1 minut
Indata	Differens i aktievärdet och köpvolymen på följande bolag: Nordea, SEB, Handelsbanken, Swedbank, Danske bank och Sydbank
Utdata	Differensen för aktievärdet hos Nordea

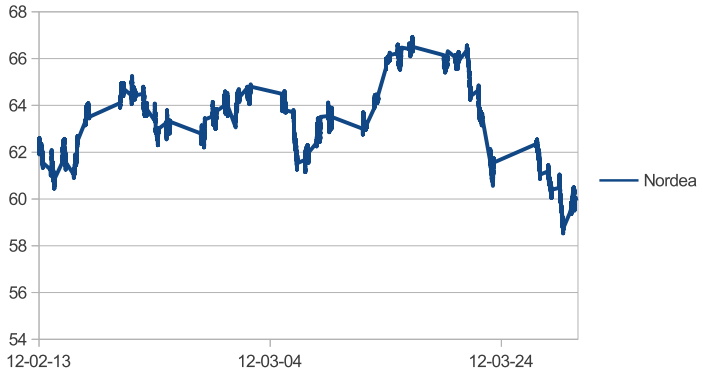
Tabell 1: De exakta parametrarna för nätverken som i simulation gav mest avkastning och vars träning presenteras i figur 15.



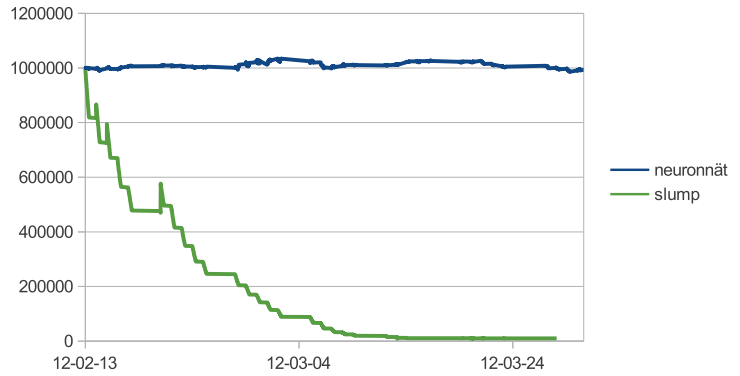
Figur 15: Figuren visar andelen av träningsdatan och valideringsdatan vars utdata förutsägs gå åt rätt håll för de fem nät som tränats med ovanstående parametrar.



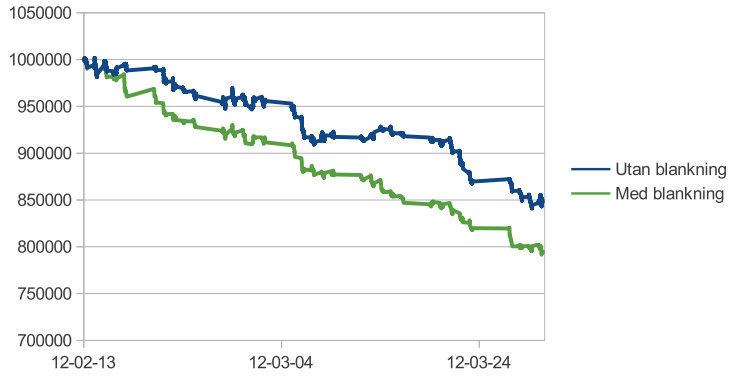
Figur 16: Simulatorns tillgångar över tiden för de fem neuronnäten vars träning kan ses i figur 15. Här körs simuleringen med courtage och ett tröskelvärde på 0.00055.



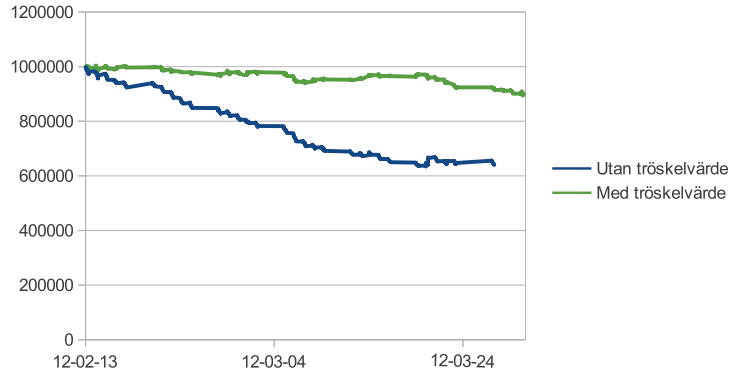
Figur 17: Nordeas aktiepris under samma tidsperiod som simuleringen i figur 16.



Figur 18: Medelvärdet på tillgångarna för prediktorerna i figur 16 och 15 slump-prediktorer (beskrivna i slutet av avsnitt 3.5).



Figur 19: Jämförelse av medelvärdet på tillgångarna för simulatorer som kör prediktorer med och utan blankning. Alla simulatorer använder här courtage.



Figur 20: Jämförelse av medelvärdet på tillgångarna för simulatorer som kör med och utan tröskelvärde på 0.00055.

5 Diskussion

Först kommer systemet som har utvecklats diskuteras och därefter de neuronnät som har producerats av det. Sist diskuteras framtida förbättringar av projektet.

5.1 Slutgiltigt system

Systemet har i stort sett den funktionalitet som planerades ifrån början. Dock har vissa funktioner fått lämnas ute och nya krav har under projektets gång införts då de framkommit.

5.1.1 Skalning av data

Det insågs snabbt att det faktiska värdet på en aktie inte var användbart i näten, vilket kan utläsas ur figur 14. Huruvida en aktie kostar 100 eller 500 kr är irrelevant och bör ej påverka nätet ifråga. Därför användes endast differensen och förändringen vid alla körningar därefter, vilket har resulterat i nätverk som presterar bättre.

Ett potentiellt problem med interpoleringen är att den introducerar värden på kursen som inte finns på riktigt. Men då de underliggande punkterna i vanligtvis har 10 sekunders avstånd mellan varandra kommer felmarginalen för de interpolerade värdena att vara försumbar.

5.1.2 Separering av data

Till en början och under en väsentlig del av projektet fördelades datamängden tillgänglig för träning upp i träningsdata, valideringsdata och testdata slumpmässigt. Detta var något som vållade problem då alla tre dataset var blandade med varandra och starkt korrelererade. Senare gjordes en mindre omimplementering så att träningsdatan togs från början av datamängden, valideringsdatan ur mitten och testdatan ur slutet. På så sätt har datan separerats mer och tillförlitligheten för den statistik man får ut av att mäta felet på valideringsdatan och testdatan har ökats.

5.1.3 Simulator

Simuleringsmodulen kan utföra simulationer med ett antal olika parametrar (med/utan blankning/courtage/transaktionsmarginal). För den begränsade tid som tilläts projektet och den tid varje simulering potentiellt kan ta, i storleksordningen flera timmar, kan antalet parametrar anses vara tillräckliga.

De varierbara parametrarna bör dessutom vara bland de mest intressanta. Att kunna köra simulationer utan courtage är intressant för att utröna huruvida näten åtminstone teoretiskt sett kan nyttjas i aktiehandel framgångsrikt.

En större brist med simulationen är att transaktioner ej påverkar marknaden. De simulerade köp och försäljningar som genomförs påverkar inte utbud och efterfrågan mer än vid det specifika handelstillfället. Det antal köpare och säljare som finns vid nästa handelstillfälle kommer att finnas där oavsett om simulatorn säljer/köper till alla köpare/säljare vid det nuvarande handelstilfället. Om simulatorn köper en större mängd aktier är det rimligt att någon motreaktion kan inträffa på marknaden. Möjligen hade färre säljare funnits vid nästa

handelstillfälle, något simulatorn alltså ej tar hänsyn till. Noterbart är att denna felkälla är större ju kortare predikteringsintervallen är då eventuella effekter transaktioner har på marknaden försvinner med tiden.

Det hade varit önskvärt att implementera fler strategier för simulatorn. En tänkbar förändring är att inte investera samtliga tillgängliga medel vid varje transaktion.

Att anta att aktier alltid finns tillgängliga till utlåning, dessutom utan avgift, vid blankning är en förenkling av verkligheten. Dock verkar blankning påverka simulationens resultat negativt och används inte i de simuleringar som utgör huvudresultaten och redovisas i denna rapport (se figur 16).

5.2 Neuronnät

Huvudobservationer:

- Nästintill inga nät presterade väl vid simulering under längre tidsperioder.
- Blankning verkar påverka simulationens resultat negativt (se figur 19).
- Användning av tröskelvärde påverkar simulationens resultat positivt (se figur 20).
- Nätverken presterar väsentligt bättre än slumpmässiga prediktorer under samma förhållanden (se figur 18).

5.2.1 Föråldring

Att näten presterade bra under en tid för att sedan försämras (se figur 16) var att vänta, då den data näten tränats på blev äldre och utdaterad. Denna utdatering kan bero på att de trender nätet har funnit i träningsdatan relativt snabbt förändras och att marknaden inte längre följer dessa. Det intressanta att notera är efter hur lång tidsperiod detta sker. Det hade varit intressant att se om denna tidsperiod påverkas av neuronnätens förutsägelseintervall. En lösning på detta problem skulle vara att systemet kontinuerligt mäter ett näts kvalitet och automatiskt tränar om det när det når en viss kritisk nivå.

5.2.2 Korrelation av data

Indatan till det nät som visas i figur 14 bestod endast av aktiens värde. Det visade sig under projektets gång att näten presterade bättre då även orderdjup användes som indata (se figur 15).

Det nät som mest framgångsrikt förutsåg Nordea-kursen använde sig av indata även från andra banker och det tycks alltså vara möjligt att dessa aktier i någon mån är korrelerade.

Det var ej möjligt att undersöka huruvida aktiekurser var relaterade till valutakurser då valutadatan upphörde att samlas in under projektet och datan även dessförinnan inte var fullständig eftersom valutaspindeln kraschade ett antal gånger.

5.2.3 Resultat av simulering

Som kan utläsas av resultaten (figur 18) går neuronnäten i snitt med varken vinst eller förlust. Man ska dock ta hänsyn till att courtage utgår vid varje transaktion och alltså försämrar neuronnätens vinst och att varje köp eller försäljning av aktier innebär en direkt förlust då ett köp av en aktie alltid sker till ett högre pris än vad man i det läget kan sälja samma aktie för. Med andra ord förlorar man direkt pengar om man hela tiden skulle köpa och sälja en aktie vars medelpris står stilla. Sammantaget preseterar neuronnäten bättre än simuleringarna med slumpprediktorer.

5.3 Framtida arbete

Den naturliga fortsättningen på projektet vore att fortsätta variera olika konfigurationer för neuronnäten och även att testa strategier för användning av dem.

Även andra tekniker hade kunnat tillämpas. Exempelvis klassificering av förutsägbarhet hos dataserier genom beräkning av Hurstexponenten. Även andra typer av neuronnät såsom Kohonen-nätverk och recurrent-nätverk hade kunnat användas. Träningen och testningen av olika nätverk och parametrar hade kunnats automatiseras ytterligare och det hade varit intressant att använda en genetisk algoritm för att hitta så bra parametrar till neuronnäten som möjligt.

Problemet med föråldrig som har diskuterats i kapitel 5.2.1 borde även avhjälpas med kontinuerlig omträning av neuronnäten i fasta intervaller.

För förbättrat resultat skulle man även kunna förbehandla datan mer genom så kallad feature extraction. En form av feature extraction som redan utforskats i projektet är differensen och promilleförändringen av indatan men det finns många fler avancerade såsom dimensionsreducering, glidande medelvärde, signalutjämning med flera.

6 Slutsatser

Systemet har de flesta av de ursprungligen planerade funktionerna och det har producerat neuronnät som i simuleringar gett resultat avsevärt bättre än simuleringar med slumpmässiga investeringar.

En slutsats som dragits är att det är viktigt med den feature extraction som används i form av förändring snarare än det faktiska värdet på all data. Ytterligare en slutsats är att orderdjup som indata förbättrar neuronnätens förutsägelser, medan endast aktiekurser som indata ger sämre resultat.

De förbättringar av systemet som antagligen skulle förbättra resultaten mest är mer avancerad feature extraction och kontinuerlig omträning av neuronnät.

Referenser

[1] K. H. och M. Stinchcombe och H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 259–366, March 1989.

[2] K. Nygren, "Stock prediction - a neural network approach," Kungliga Tekniska Högskolan, Stockholm, Tech. Rep., 2004.

[3] P. Mccluskey, "Feedforward and recurrent neural networks and genetic programs for stock market and time series forecasting," Brown University, Providence, Rhode Island, Tech. Rep., 1993.

[4] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Upper Saddle River: Pearson Education Inc., 2009.

[5] F. Ozkan, "A comparison of the monetary model and artificial neural networks in exchange rate forecasting," *Business and Economics Research Journal*, vol. 3, no. 1, p. 27, January 2012.

[6] M. Udrescu and C. Ilie, "New techniques applied in economics. artificial neural network," *Annals of Faculty of Economics*, vol. 4, no. 1, pp. 1080–1084, May 2009.

[7] S. Wong, K. K. Wan, and T. N. Lam, "Artificial neural networks for energy analysis of office buildings with daylighting," *Applied Energy*, vol. 87, no. 2, pp. 551–557, February 2010.

[8] T. Szecsi, "Cutting force modeling using artificial neural networks," *Journal of Materials Processing Technology*, vol. 92-93, pp. 344–349, Augusti 1999.

[9] R. Rojas, *Neural Networks: A Systematic Introduction*, 1st ed. New York: Springer, 1996.

[10] J. Heaton, *Introduction to neural networks with Java*. St. Louis: Heaton Research, 2008.

[11] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of mathematical biology*, vol. 5, no. 4, pp. 115–133, 1943.

[12] T. Shinzato and Y. Kabashima, "Perceptron capacity revisited: classification ability for correlated patterns," *Journal of physics A-Mathematical and theoretical*, vol. 41, no. 32, p. 324013, 2008.

[13] B. Widrow, "Thinking about thinking: the discovery of the lms algorithm," *Signal Processing Magazine*, vol. 22, pp. 100–106, 2005.

[14] Wolfram Research Inc, "Newton's method," http://mathworld.wolfram.com/NewtonsMethod.html.

[15] G. Cybenko, "Approximations by superpositions of a sigmoidal function," University of Illinois, Urbana, Illinois, Tech. Rep., 1989.

[16] J. A. E. Bryson and Y.-C. Ho, *Applied Optimal Control: Optimization, Estimation and Control*, 2nd ed. Taylor & Francis, 1975.

[17] J. Bernhardsson, *Tradingguiden*, 2nd ed. Stockholm: Fischer & Co, 2002.

[18] Finansinspektionen, "Börser och aktiehandel," http://www.fi.se/Konsument/Fragor-och-svar/Borser-och-aktiehandel/, April 2012.

[19] Avanza, "Kundhandbok - så här går handeln till," https://www.avanza.se/aza/kunskapscenter/depahandbok.jsp?page=shghter.

[20] G. Wrede, "Visar orderdjupet hur aktien ska gå?" https://www.avanza.se/aza/press/press_article.jsp?article=122343, Maj 2012.

[21] K. Wärneryd, *Stock-Market Psychology*, 1st ed. Cheltenham: Edward Elgar Publishing Ltd., 2001.

[22] B. Q. och K. Rasheed, "Hurst exponent and financial market predictability," in *Proceedings of The 2nd IASTED international conference on financial engineering and applications*. Cambridge, MA, USA: IASTED international, November 2004, pp. 203–209.

[23] A. Refenes, *Neural networks in the capital markets*, 1st ed. New York: Wiley, 1995.

[24] E. Gately, *Neural networks for financial forecasting*, 1st ed. Wiley, 1996.

[25] S. Walchak, "An empirical analysis of data requirements for financial forecasting with neural networks," *Journal of management information systems*, vol. 17, no. 4, pp. 203–222, 2001.

[26] R. Lawrence, "Using neural networks to forecast stock market prices," University of Manitoba: Department of Computer Science, Manitoba, Tech. Rep., 1997.

[27] C. Faith, *Way of the Turtle: The Secret Methods that Turned Ordinary People into Legendary Traders*, 1st ed. McGraw-Hill, 2007.

[28] L. Chulhee, "Decision boundary feature extraction for neural network," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 75–83, January 1997.

[29] NASDAQ OMX Nordic, "Nasdaq omx nordic," http://www.nasdaqomxnordic.com/.

[30] Currencyrates, "Currencyrates," http://www.currencyrates.com.

[31] Wolfram Research Inc, "Root-mean-square," http://mathworld.wolfram.com/Root-Mean-Square.html.