Stochastic Approximation:
Sturen in een veranderende wereld

Rianne Lurvink

BWI werkstuk

Begeleider: Sandjai Bhulai

vrije Universiteit *amsterdam*

Faculteit der Exacte Wetenschappen

Studierichting Bedrijfswiskunde en Informatica

De Boelelaan 1081a

1081 HV Amsterdam

Voorwoord

Een afsluitend onderdeel van de studie Bedrijfswiskunde en Informatica is het BWI werkstuk. Dit werkstuk vindt plaats in de periode vlak voor de bedrijfstage. Het doel van dit werkstuk is om een literatuuronderzoek te doen naar een onderwerp dat minimaal twee van de drie BWI componenten bezit (Bedrijfskunde, Wiskunde en Informatica).

In dit werkstuk zal een literatuuronderzoek worden gedaan naar Stochastic Approximation. Dit is een manier om via een recursieve formule een onbekende parameter van een model te schatten wanneer nieuwe data beschikbaar komt. Bovendien kan de methode ook gebruikt worden om systemen te sturen wanneer de parameters variëren in de tijd. Een verdere probleemstelling zal worden beschreven in de inleiding van dit werkstuk.

Ik wil Sandjai Bhulai bedanken voor zijn bijdrage aan het kiezen van een onderwerp en zijn begeleiding tijdens het schrijven van dit werkstuk, en ik wens iedereen veel plezier bij het lezen van dit verslag.

Rianne Lurvink
Januari 2007

Samenvatting

Het optimaliseren van bedrijfsproblemen gaat vaak met behulp van wiskundige modellen. Deze modellen gaan veelal uit van een constante input parameter die bekend is. Op basis van deze input parameter en het model wordt dan de prestatie van het systeem berekend. In de praktijk hoeft de input parameter natuurlijk lang niet altijd constant te zijn of bekend te zijn.

Er zijn veel methoden om een variabele of parameter te benaderen of te schatten. Hiervoor kan bijvoorbeeld het gemiddelde worden gebruikt. Het grootste nadeel hiervan is dat dit niet wil zeggen dat de *gemiddelde prestatie* van het systeem goed is. Ook zijn er analytische methoden om een reeks van benaderingen te geven van een bepaalde variabele of parameter. Newton's methode geeft een reeks van benaderingen voor de oplossing van de vergelijking $g(x) = 0$. Gauss-Newton geeft een reeks van benaderingen voor het niet-lineaire kleinste kwadraten probleem. Beide methoden convergeren naar de oplossing en maken gebruik van een recursieve formule. Deze methoden hebben echter een nadeel: er wordt vanuit gegaan dat de functie $g(x)$ bekend is en continu differentieerbaar is.
Stochastic Approximation gaat er echter vanuit dat de functie $g(x)$ niet bekend is. Wel kunnen er observaties gedaan worden bij een bepaalde waarde van x om een schatting te krijgen van de functiewaarde. Deze observaties kunnen worden gedaan door middel van simulaties of experimenten. Twee bekende algoritmes binnen Stochastic Approximation zijn het Robbins-Monro algoritme en het Kiefer-Wolfowitz algoritme. Deze algoritmes benaderen respectievelijk de oplossing van de vergelijking $g(x) = v$ en het maximum van de functie $g(x)$, zonder deze functie bekend te veronderstellen en met behulp van een recursieve formule. Het voordeel van een recursieve formule is de kleine rekentijd en het gebruik van beperkte geheugenruimte. Echter, het algoritme hangt af van de beginwaarde en een stapgrootte; wanneer deze niet goed gekozen worden, kan het aantal iteraties gigantisch toenemen.

Met behulp van Stochastic Approximation kan de drempelwaarde van een systeem worden bepaald tot waar klanten het systeem mogen betreden (admission control). Het aantal aankomende klanten per tijdseenheid is niet constant maar stochastisch. Van te voren is het aantal aankomende klanten dus niet bekend, en dit aantal verandert telkens.
Bij iedere drempelwaarde wordt bekeken wat de prestatie van het systeem is met behulp van simulatie, en aan de hand daarvan wordt een nieuwe drempelwaarde ingesteld met behulp van een recursieve formule. Een tweede voorbeeld is het instellen van het aantal servers s om een

goede Service Level te behalen. Dit gaat op dezelfde manier als in het eerste voorbeeld met behulp van Stochastic Approximation. Het blijkt dat het instellen van een parameter met behulp van Stochastic Approximation erg goed werkt, de Service Level tijdens de gehele simulatie schommelt rond de norm dat 80% van de klanten minder wacht dan 20 seconden bij beide voorbeelden. De parameter wordt gestuurd door veranderingen in het aantal klanten.

Inhoudsopgave

Inleiding

Wiskundige modellen veronderstellen vaak een constante input parameter die bekend is. In de praktijk is dit natuurlijk lang niet altijd het geval. Daarom is er een behoefte aan methoden die de parameter van een systeem kunnen bepalen zonder dat er vanuit wordt gegaan dat deze parameter constant of bekend is. Vaak is het doel op zich niet om de parameter te schatten, maar om een bepaald systeem te sturen, zonder dat hierbij de parameter geschat wordt. Dit is Stochastic Approximation. Hierbij wordt gebruik gemaakt van een recursieve formule.
In dit werkstuk zal Stochastic Approximation worden behandeld. De probleemstelling die in dit verslag centraal staat is de volgende:

Wat houdt Stochastic Approximation precies in, en hoe kan deze methode gebruikt worden om de parameter of direct de sturingsparameter van een model te schatten?

Het sturen van een systeem houdt het kiezen van een bepaalde parameter in waardoor de prestatie van een systeem constant blijft rond een gestelde norm. In dit werkstuk zullen twee voorbeelden worden gegeven voor het sturen van een wachtrijsysteem.

Hoofdstuk 1 zal een inleiding geven in de theoretische modellen en er wordt een voorbeeld gegeven waarin een betere schatting van een parameter vereist is. In hoofdstuk 2 zullen allereerst een aantal alternatieve methoden besproken worden om een parameter te voorspellen, en de nadelen van deze methodes zullen worden besproken. Daarna zal Stochastic Approximation worden beschreven. De voor- en nadelen van deze methode zullen worden belicht en er zullen enkele voorbeelden worden gegeven. Ook worden issues in de praktijk behandeld.

In hoofdstuk 3 zullen twee praktijkvoorbeelden worden gegeven voor het sturen van een wachtrijsysteem waarvoor een simulatiemodel is gemaakt. Hierbij wordt gebruik gemaakt van Stochastic Approximation. Het eerste voorbeeld betreft het instellen van een drempelwaarde van een systeem. Als het aantal klanten in het systeem onder de drempelwaarde ligt, dan worden er nog nieuwe klanten geaccepteerd. Als het aantal klanten in het systeem gelijk is aan de drempelwaarde, dan worden nieuwe klanten geweigerd. Het doel is dat deze drempelwaarde zo wordt ingesteld dat er aan een bepaalde Service Level wordt voldaan. Het

tweede voorbeeld betreft het instellen van het aantal servers s in de $M|M|s$ queue. Hier kan geen gebruik worden gemaakt van de Erlang-C formule, omdat λ niet constant is. Hier is het doel dat het aantal servers zo gekozen wordt, dat aan een bepaalde Service Level wordt voldaan. In hoofdstuk 4 zal uiteindelijk de conclusie worden gegeven.

Hoofdstuk 1: Modellen in de praktijk

Het optimaliseren van bedrijfsproblemen gaat vaak met behulp van wiskundige modellen. Het nut van deze wiskundige modellen is het weergeven van de prestatie van het systeem op basis van input parameters.

De meeste van deze wiskundige modellen veronderstellen een constante parameter die bekend is. Met deze parameter als input en het specifieke model kunnen dan berekeningen worden gemaakt. Een voorbeeld van een input parameter kan zijn het aantal klanten dat een bepaald systeem binnenkomt tijdens een tijdsinterval van één uur. De meeste modellen veronderstellen dat dit getal constant en bekend is gedurende het uur, bijvoorbeeld een vast aantal van tien klanten per uur. Wanneer we spreken van een wachtrijmodel waarbij het aantal servers, de verdeling van de servicetijd en het aankomstproces zijn gegeven (bijvoorbeeld Poisson), kan nu bijvoorbeeld de verwachte wachttijd worden bepaald.

Er zijn ook modellen die de parameter niet constant veronderstellen, maar die ervan uitgaan dat de parameter tijdvariërend is. Een voorbeeld hiervan is een call center; het aantal klanten dat tijdens de lunch belt zal een stuk lager zijn dan het aantal klanten dat belt tussen drie en vier uur ’s middags. Er is dan bijvoorbeeld een functie opgesteld die het aantal klanten in een tijdsinterval bepaalt. Maar ook hier wordt er meestal vanuit gegaan dat het aantal klanten in een bepaald tijdsinterval constant is.

Het is duidelijk dat deze methoden (het veronderstellen van een bekende constante parameter of een bekende tijdvariërende parameter), in de praktijk niet goed bruikbaar zijn aangezien het aantal klanten dat binnenkomt in een tijdsinterval nu eenmaal nooit precies gelijk is aan een bekend vast getal λ. In de volgende paragraaf zal een voorbeeld worden gegeven dat aantoont wat voor gevolgen het kiezen van een vaste parameter kan hebben op de wachttijd binnen een systeem.

1.1 Voorbeeld Erlang-C formule

In dit voorbeeld wordt gebruik gemaakt van de Erlang-C formule. De Erlang-C formule geeft bij een gegeven aankomstrate λ, een gegeven verwachte servicetijd β en een gegeven aantal

servers *s*, de gemiddelde wachttijd en de Service Level (bijvoorbeeld: 80% van de klanten wacht minder dan 20 seconden).

De Erlang-C formule wordt gebruikt om de wachttijd te bepalen in de bekende $M|M|s$ queue. Hier staat de eerste *M* voor de aankomsten (een Poisson proces), de tweede *M* voor de servicetijd (exponentieel verdeeld) en *s* voor het aantal servers in het systeem.
We nemen aan dat $\lambda = 4$ in het eerste deel van de periode en $\lambda = 6$ in het tweede deel van de periode. Stel dat we het gemiddelde gebruiken om het aantal servers te bepalen in de hele periode. Er geldt dan dat $\lambda_{gemiddeld} = 5$. Wanneer we met behulp van de Erlang-C calculator [5] bepalen hoeveel servers *s* er nodig zijn om de norm te behalen (80% van de klanten wacht minder dan 20 seconden), geldt bij $\lambda_{gemiddeld} = 5$ en een servicetijd $\beta = 3$ dat $s = 19$. Gemiddeld genomen gaan we er dus vanuit dat er 19 servers nodig zijn om een goede norm te behalen tijdens de hele periode.

Wanneer we nu echter de performance van het systeem gaan bekijken bij $s = 19$, blijkt dat in het eerste deel van de periode (bij $\lambda = 4$) de Service Level erg hoog is, 98% van de klanten wacht minder dan 20 seconden. In het tweede deel van de periode (bij $\lambda = 6$) is de Service Level echter erg slecht, 32.91% van de klanten wacht minder dan 20 seconden. De gemiddelde Service Level gedurende de periode blijkt dus (98% + 32.91%)/2 = 65.46% te zijn. Echter, sommige call centers wegen de periodes met λ, dan zou de gemiddelde Serivce Level neerkomen op (4*98% + 6*32.91%) / 10 = 58.95%. Dit is lang niet de norm waarbij 80% van de klanten minder dan 20 seconden wacht! Het blijkt dus dat wanneer het aantal servers bepaald wordt aan de hand van de gemiddelde aankomstrate, de gemiddelde Service Level niet goed genoeg is om aan de norm te voldoen. Het is dus duidelijk dat er niet zomaar gebruik kan worden gemaakt van gemiddeldes, aangezien deze lang niet altijd voor een goede gemiddelde performance zorgen!
Er is dus een manier nodig die het mogelijk maakt om de parameter nauwkeurig te voorspellen. Wanneer het mogelijk is om parameters nauwkeurig te voorspellen, zullen fouten minder vaak voorkomen dan wanneer er vanuit wordt gegaan dat de parameter constant is. Een methode om parameters te voorspellen is met behulp van Stochastic Approximation.

Stochastic Approximation kan echter ook gebruikt worden om control parameters in te stellen. Een voorbeeld hiervan is een wachtrijmodel met admission control. Hierbij kunnen aankomende klanten geweigerd worden. Er wordt een bepaalde level gekozen tot waar

klanten worden toegelaten tot het systeem, de drempelwaarde. Boven deze waarde worden klanten niet langer toegelaten tot het systeem. Stochastic Approximation kan nu gebruikt worden om deze level te voorspellen wanneer bijvoorbeeld de aankomstintensiteit varieert in de tijd en onbekend is. Een praktijkvoorbeeld hiervan zal worden gegeven in hoofdstuk 3.

Hoofdstuk 2: Stochastic Approximation

In dit hoofdstuk zal Stochastic Approximation in het algemeen worden behandeld. Er zullen enkele voorbeelden gegeven worden waarin deze methode kan worden toegepast. Allereerst zullen er een aantal alternatieve methoden worden besproken om parameters te voorspellen, en de nadelen van deze methodes.

2.1 Alternatieve methodes voor het voorspellen van parameters

Het voorspellen van parameters kan op veel verschillende manieren. In deze paragraaf zullen twee van deze manieren worden besproken, te weten het gemiddelde en Newton's methode en Gauss-Newton. Tevens zullen de nadelen van deze methodes worden belicht.

2.1.1 Parameters voorspellen aan de hand van het gemiddelde

Een parameter kan worden voorspeld aan de hand van de gemiddelde waarde. Een voorbeeld hiervan is reeds gegeven in hoofdstuk 1. Wanneer bijvoorbeeld de parameter in de eerste helft van een periode x_1 is en in de tweede helft van de periode x_2, kan als schatting het gemiddelde $\frac{x_1 + x_2}{2}$ worden gebruikt. Deze waarde kan dan worden gebruikt als input voor het wiskundige model.
Het nadeel hiervan is dat de gemiddelde waarde lang niet altijd voor een goede gemiddelde performance hoeft te zorgen. Dit is het geval wanneer bijvoorbeeld de performance in het eerste deel van de periode heel goed is en in het tweede deel van de periode heel erg slecht. Gemiddeld wordt dan geen goede performance behaald, terwijl wel een goede performance zou worden gehaald als de werkelijke parameter tijdens de hele periode gemiddeld was geweest. Daarom moet niet blindelings het gemiddelde worden gebruikt als voorspelling.

2.1.2 Newton's methode en Gauss-Newton

Newton's methode is een iteratieve manier om een reeks van benaderingen $x_1, x_2, x_3, \ldots$ te genereren voor de oplossing $x = r$ van de vergelijking $g(x) = 0$ [3]. Deze reeks van benaderingen wordt dan gegeven door

$$x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)}.$$

Het doel is dat deze reeks convergeert naar de uiteindelijke oplossing r zodanig dat geldt $|x_n - r| < \varepsilon$ voor alle $n \geq N$. Een goede startwaarde x_0 in de buurt van de oplossing is nodig voor een snelle convergentie.

Gauss-Newton is een gewijzigde versie van Newton's methode en is beschreven door Carl Friedrich Gauss. Het verschil tussen beide methodes is dat Gauss-Newton gebruik maakt van de tweede afgeleide, waar Newton's methode alleen gebruik maakt van de eerste afgeleide.

Het doel van het Gauss-Newton algoritme is het oplossen van een niet-lineair kleinste kwadraten probleem [7]. Er wordt uitgegaan van m functies $f_1, f_2, \ldots, f_m$ ieder met n variabelen $x_1, \ldots, x_n$. We willen minimaliseren

$$S(x) = \sum_{i=1}^{m} (f_i(x))^2,$$

waarbij x de vector $(x_1, \ldots, x_n)$ voorstelt. De recursieve formule voor x wordt gegeven door

$$x^{k+1} = x^k - (J_f(x^k)^{\mathrm{T}} J_f(x^k))^{-1} J_f(x^k)^{\mathrm{T}} f(x^k),$$

waar $f = (f_1, f_2, \ldots, f_m)$ en $J_f(x)$ de Jacobiaan matrix (de matrix met de eerste afgeleiden $\frac{\partial f}{\partial x}$) is.

Newton's methode en Gauss-Newton kunnen dus worden gebruikt om de 'ware' parameter te voorspellen of te schatten, net zoals Stochastic Approximation een manier is om de 'ware' parameter te schatten.

Het nadeel van deze methodes is dat er vanuit wordt gegaan dat de onderliggende functie bekend is en continu differentieerbaar is. Dit is in de realiteit natuurlijk lang niet altijd het geval. Later in dit hoofdstuk zal een manier worden besproken om met behulp van Stochastic Approximation een reeks van benaderingen te geven voor de oplossing, zonder dat de onderliggende functie bekend is verondersteld.

2.2 Algemene vorm van Stochastic Approximation

Stochastic Approximation houdt zich bezig met het voorspellen van parameters door middel van recursieve algoritmes en de convergentie van deze algoritmes. In andere woorden zijn dit algoritmes die gebruikt worden in 'noisy' stochastische omgevingen. Hierbij is de onderliggende functie van de observaties niet bekend. Het is dus alleen mogelijk om een

observatie te doen bij een bepaalde parameterwaarde, het is niet mogelijk om precies te berekenen wat het effect is van een bepaalde parameter aangezien de onderliggende functie niet bekend is. Dit in tegenstelling tot Newton's methode: het is daar mogelijk om te berekenen wat de functiewaarde is bij een bepaalde waarde van x_n. Nu willen we echter ook de ware parameter voorspellen, alleen kunnen we niet de precieze functiewaarde berekenen. Het is dus nodig om observaties te doen bij deze x_n, in de vorm van experimenten of simulaties.

Stochastic Approximation is een methode om de parameter θ van een model te schatten. Hierbij kan deze parameter θ een input parameter zijn, maar ook een control parameter (zoals wordt besproken in hoofdstuk 3). Er kan een schatting worden gemaakt aan de hand van de observatie van de parameter θ in een eerder tijdsinterval en de schatting van de parameter θ in een eerder tijdsinterval. De standaard notatie voor de recursieve formule is

$$\theta_{n+1} = \theta_n + \varepsilon_n [y_n - \theta_n].$$

Hierbij is $[y_n - \theta_n]$ de afwijking tussen de geobserveerde waarde van θ_n (deze werkelijk geobserveerde waarde wordt genoteerd als y_n) en de geschatte waarde van θ_n, en ε_n is de stapgrootte waarvoor geldt $\varepsilon_n > 0$. Er geldt dat ε_n naar 0 kan gaan wanneer $n \to \infty$.

2.3 Geschiedenis van Stochastic Approximation

Stochastic Approximation is geïntroduceerd door Robbins en Monro in 1951. Kiefer en Wolfowitz hebben daar begin jaren vijftig verder onderzoek naar verricht. In recentere jaren zijn veel Stochastic Approximation algoritmes gebruikt in nieuwe en zeer diverse richtingen. Er zijn nieuwe technieken bedacht om de convergentie van deze algoritmes te bewijzen. Er zijn vele toepassingen waarin Stochastic Approximation gebruikt wordt, onder andere in de communicatie en signaal processing.

Het bekendste algoritme binnen Stochastic Approximation is het Robbins-Monro algoritme. Dit algoritme, zoals de naam al aangeeft, is uitgevonden door Robbins en Monro in 1951. Aangezien dit het eerste algoritme is dat gebruikt maakt van Stochastic Approximation, is dit het bekendste algoritme en in vrijwel ieder boek of artikel over Stochastic Approximation wordt het als eerste beschreven.

2.4 Het Robbins-Monro algoritme

Om een intuïtief gevoel te krijgen bij deze aanpak, kunnen we het volgende voorbeeld geven. Stel dat we een parameter θ hebben die we willen schatten. Hierbij is $x = \theta$ de oplossing van de vergelijking $M(x) = v$. Van de functie $M(x)$ is alleen bekend dat hij stijgend is in x. Verder is er niks over bekend. We willen nu de waarde x vinden, waarvoor geldt dat $M(x) = v$. Als $M(x)$ wel bekend en continu differentieerbaar zou zijn, dan zou Newton's methode gebruikt kunnen worden om dit probleem op te lossen.

Aangezien de functie $M(x)$ nu niet bekend is, kan er geen gebruik worden gemaakt van Newton's methode. Wel kunnen we nu gebruik maken van het Robbins-Monro algoritme om tot een goede schatting van θ te komen. De functie zelf is dan niet bekend, maar het is wel mogelijk om bij een bepaalde parameter observaties te doen (dit kan aan de hand van experimenten of simulaties). Hierbij wordt gebruik gemaakt van de volgende recursieve formule

$$\theta_{n+1} = \theta_n + \varepsilon_n [v - \text{observatie bij parameter } \theta_n] = \theta_n + \varepsilon_n [v - y_n].$$

Hierbij is θ_n de n-de schatting van θ, en y_n stelt de observatie bij parameter θ_n voor. Het is dus de bedoeling om de unieke oplossing θ te bepalen van de vergelijking $M(x) = E[y_n| \theta_n] = v$. De stapgrootte ε_n is veelal van het type $1/n$ [2].

In [1] is dit voorbeeld toegespitst op de dosering θ van een bepaald medicijn. Bij een dosering van θ_n kan dan worden bepaald wat de overlevingskans is bij deze specifieke dosering. Om te bepalen wat deze overlevingskans is, moet een groep met patiënten deze dosering θ_n krijgen, en vanuit deze groep kan worden gekeken hoeveel mensen overleven. Dit is dan dus de observatie bij parameter θ_n.
Er is een gewenste overlevingskans v. Wanneer de overlevingskans bij parameter θ_n ver van deze gewenste overlevingskans v afligt, dan is de verandering in θ_n dus groot en wanneer de observatie dicht bij de gewenste waarde ligt zal de verandering in θ_n dus klein zijn waardoor de schatting convergeert naar de ware parameter θ. Een snelle convergentie hangt in grote mate af van de gekozen startwaarde θ_0 en de stapgrootte ε_n.

De belangrijkste gedachtengang achter het Robbins-Monro algoritme is het feit dat de precieze functie van de input variabele naar de performance van deze variabele niet bekend is. Deze performance kan worden bepaald aan de hand van observaties, welke in de vorm van

een experiment of een simulatie bepaald kunnen worden [2]. Het Robbins-Monro algoritme kent vele toepassingen.

2.4.1 Convergentie van het Robbins-Monro algoritme

Het recursieve Robbins-Monro algoritme zal convergeren naar de ware parameter θ (dus $\theta_n \rightarrow \theta$) in mean square. Dit wil zeggen dat voor een reeks van benaderingen θ_n geldt [8]

$$\lim_{n\to\infty} E(|\theta_n - \theta|^2) = 0.$$

Het algoritme zal convergeren als voldaan wordt aan een aantal voorwaarden. Deze voorwaarden worden gegeven door [2]

1. Er bestaat een C zodanig dat de observaties met kans 1 binnen het interval $(-C, C)$ liggen. De kans is dus 0 dat een observatie buiten het interval valt;
2. Er wordt verwacht van de functie $M(x)$ dat hij niet-dalend is, dat hij een unieke oplossing θ bezit en dat de functie strikt stijgend is bij de oplossing ($M'(\theta) > 0$);
3. De stapgroottes zijn positief en er geldt dat $\varepsilon_n \geq 0$ en $\Sigma\varepsilon^2_n < \infty$;
4. De stapgroottes ε_n zijn van het type $1/n$.

Het bewijs hiervan wordt gegeven in [2].

Zoals wordt besproken in [2], is het niet meteen duidelijk waarom een stapgrootte $1/n$ leidt tot convergentie naar de ware oplossing. Een mogelijke verklaring die genoemd wordt is dat de stapgroottes tot oneindig moeten sommeren, zodat de hele oplossingsruimte wordt onderzocht. Het algoritme blijft dan niet hangen in een sub-optimaal punt.
Een andere verklaring die genoemd wordt is dat dalende stapgroottes vereist zijn, zodat wanneer we in de buurt van de oplossing komen, de reeks van benaderingen ook convergeert naar de oplossing.

2.5 Het Kiefer-Wolfowitz algoritme

Nadat in 1951 het algoritme van Robbins en Monro gepubliceerd was, hebben Kiefer en Wolfowitz hier verder onderzoek naar verricht in de jaren vijftig. Het Kiefer-Wolfowitz algoritme is een manier om het maximum $x = \theta$ van een functie $M(x)$ te bepalen. De

onderliggende functie $M(x)$ is weer niet bekend verondersteld, waardoor het niet mogelijk is om de afgeleide van de functie te nemen om zo tot een maximum of minimum te komen.

Kiefer-Wolfowitz maakt gebruik van de volgende recursieve formule [2]

$$\theta_{n+1} = \theta_n + \varepsilon_n \left[\frac{(y_{2n} - y_{2n-1})}{c_n} \right].$$

Hierbij is y_{2n} de observatie die gedaan is bij $[\theta_n + c_n]$ en y_{2n-1} de observatie die gedaan is bij $[\theta_n - c_n]$. Het verschil tussen deze twee observaties wordt gebruikt om een nieuwe waarde θ_n te bepalen.
De afgeleide van de functie $M(x)$ is dus niet bekend, maar er wordt een benadering van de afgeleide gemaakt door observaties te doen in de buurt van de parameter θ_n.

Het Kiefer-Wolfowitz algoritme is een makkelijke manier om tot het maximum of het minimum van een functie te komen zonder dat kennis van de afgeleide nodig is. Ook bij dit algoritme is de convergentie naar de oplossing gegarandeerd in mean square, de voorwaarden zullen worden gegeven in de volgende paragraaf.

2.5.1 Convergentie van het Kiefer-Wolfowitz algoritme

Net als bij het Robbins-Monro algoritme is de convergentie van het Kiefer-Wolfowitz algoritme gegarandeerd (dat wil zeggen dat $\theta_n \rightarrow \theta$) in mean square als er voldaan wordt aan een aantal voorwaarden. Deze voorwaarden worden gegeven in [2]

1. Er moet gelden dat $\int_{-\infty}^{\infty} (y - M(x))^2 \, dp(Y \mid x) < \infty$, dus een eindige variantie;
2. Voor $M(x)$ geldt dat $M(x)$ strikt stijgend is voor $x < \theta$ en $M(x)$ is strikt dalend voor $x > \theta$;
3. Voor de stapgrootte $\{\varepsilon_n\}$ en voor $\{c_n\}$ geldt dat $\varepsilon_n, c_n > 0$, $c_n \rightarrow 0$, $\Sigma\varepsilon_n = \infty$, $\Sigma\varepsilon_n c_n < \infty$, $\Sigma\varepsilon^2_n c_n^{-2} < \infty$.

Voor de functie $M(x)$ moeten de volgende condities gelden [2]

1. De afgeleide van $M(x)$ moet klein zijn in de omgeving van θ, dus er bestaat een β en een B zodanig dat

$|x' - \theta| + |x'' - \theta| < \beta \Rightarrow |M(x) - M(x')| < B|x' - x''|$;

2. Voor iedere $\delta > 0$ is er een $\pi(\delta)$ zodanig dat

   $|\theta_n - \theta| > \delta \Rightarrow inf_{\delta/2>\upsilon>0} \frac{|M(\theta_n + \upsilon) - M(\theta_n - \upsilon)|}{\upsilon} > \pi(\delta)$.

   Dit is omdat wanneer het algoritme ver verwijderd is van θ, de afgeleide naar beneden moet worden begrensd. Anders is de kans groot dat het algoritme blijft hangen in een ander punt dan θ. Hierbij geeft *inf* de ondergrens aan;
3. Er bestaat een ρ en een *R* zodanig dat er geen sprong van ∞ naar $-\infty$ kan plaatsvinden, dus $|x' - x''| < \rho \Rightarrow |M(x') - M(x'')| < R$.

Als aan al deze voorwaarden worden voldaan, convergeert het algoritme in mean square. Het bewijs hiervan wordt gegeven in [2].

2.6 Issues in de praktijk

Zoals reeds genoemd in hoofdstuk 1, kan Stochastic Approximation niet alleen gebruikt worden om de input parameters van een systeem te schatten, maar het kan ook gebruikt worden om direct de control parameters te in te stellen. Hierbij is θ_n een control parameter, bijvoorbeeld de drempelwaarde tot waar klanten in het systeem worden toegelaten. y_n geeft dan de performance van het systeem aan bij de control parameter θ_n. De input van het systeem (bijvoorbeeld het aantal klanten dat het systeem per tijdseenheid binnenkomt) is niet constant maar bijvoorbeeld stochastisch en onbekend. Een uitgebreid voorbeeld hiervan wordt besproken in hoofdstuk 3.

Een voorbeeld van het gebruik van Stochastic Approximation zijn toepassingen in neurale netwerken [1]. In de leerfase wordt een random input serie gepresenteerd aan het netwerk, en er is een gewenste response op iedere input. Het probleem is dan het aanpassen van de gewichten in het netwerk om de gemiddelde afstand tussen de werkelijke en gewenste response te minimaliseren. Dit wordt gedaan met behulp van een training procedure, waar de gewichten worden aangepast nadat ieder (input, output) paar is geobserveerd. Het aanpassen van deze gewichten gaat met behulp van de recursieve formule, en dus met behulp van Stochastic Approximation.

Ook zijn er veel toepassingen in de learning theorie. Bijvoorbeeld het aanpassen van een optimale jachtstrategie van een dier, gebaseerd op de historische data van successen en mislukkingen in pogingen zichzelf efficient te voeden [1]. Ook dit kan met behulp van Stochastic Approximation.

2.7 Voor- en nadelen

Een groot voordeel van het gebruiken van een recursief algoritme is dat er relatief weinig rekentijd en geheugencapaciteit bij komt kijken [1]. Na iedere observatie wordt een nieuwe schatting gemaakt die voortkomt uit een simpele formule van de oude schatting en de observatie. Zelfs voor het werk van Robbins en Monro werden recursieve schattingen reeds gebruikt in de control- en communicatie theorie vanwege de simpele berekening.

In de praktijk is Stochastic Approximation zeer goed bruikbaar omdat er geen kennis verondersteld wordt van de onderliggende functie. Alleen de observaties zijn noodzakelijk.

Een nadeel aan een recursieve formule is het aantal iteraties totdat de optimale oplossing wordt bereikt. Wanneer de startwaarde en de stapgrootte niet goed gekozen worden, kan het een gigantisch aantal iteraties duren voordat het algoritme de optimale oplossing bereikt.

Hoofdstuk 3: Een praktijkvoorbeeld

In dit hoofdstuk zullen twee praktijkvoorbeelden worden besproken waarin Stochastic Approximation wordt gebruikt.

Het eerste voorbeeld betreft het kiezen van een drempelwaarde in een systeem. Wanneer het aantal klanten in het systeem lager ligt dan de drempelwaarde, worden nog nieuwe klanten toegelaten. Wanneer het aantal klanten in het systeem gelijk is aan de drempelwaarde, worden geen nieuwe klanten toegelaten. Het doel is om hierbij de drempelwaarde zo in te stellen, dat aan een bepaalde Service Level wordt voldaan. Wanneer het aantal aankomende klanten λ per tijdseenheid constant zou zijn, is er een analytische methode om deze drempelwaarde te kiezen. Deze methode zal worden besproken in paragraaf 3.1. Wanneer echter het aantal klanten stochastisch is in plaats van constant, kan deze methode niet worden gebruikt. In paragraaf 3.2.1 zal een methode worden besproken waarin deze drempelwaarde kan worden gekozen met behulp van Stochastic Approximation.

Het tweede voorbeeld, dat zal worden besproken in paragraaf 3.2.2, betreft het kiezen van het aantal servers s in een $M|M|s$ queue. Aangezien het aantal aankomende klanten λ per tijdseenheid niet constant is, kan hier geen gebruik worden gemaakt van de Erlang-C formule om het aantal servers te bepalen. Ook hier is het doel om het aantal servers zo in te stellen, dat aan een bepaalde Service Level wordt voldaan.

3.1 De drempelwaarde bepalen bij vaste λ

In dit hoofdstuk zal een praktijkvoorbeeld van Stochastic Approximation worden behandeld. Het gaat hier om een $M|M|1$ wachtrij; dus er zijn aankomende klanten volgens een Poisson proces met rate λ, de servicetijden zijn exponentieel verdeeld met rate μ en er is één server. Er zijn holding kosten $C(x)$ als zich er x klanten in het systeem bevinden. Er is de mogelijkheid om klanten te weigeren. De kosten voor het weigeren van een klant (de rejection kosten) worden gegeven door r.

De relatieve waardefunctie $V_n(x)$ is gedefinieerd als de totale verwachte opbrengst als gestart wordt in staat x bij $n = 0$. In het geval van de $M|M|1$ wachtrij geeft de relatieve waardefunctie

dus de verwachte kosten als gestart wordt met x klanten in het systeem bij $n = 0$. Het is duidelijk dat het doel is om de kosten te minimaliseren.

Volgens [6] wordt de relatieve waardefunctie van deze $M|M|1$ wachtrij nu gegeven door

$$V_{n+1}(x) = C(x) + \lambda \min\{r + V_n(x),\ V_n(x+1)\} + \mu V_n((x-1)^+) + (1 - \lambda - \mu)V_n(x).$$

Hierbij moet gelden dat $\lambda + \mu < 1$.
Aangezien in iedere staat x holding kosten $C(x)$ betaald moeten worden, is deze een onderdeel van de relatieve waardefunctie. Verder komen er klanten binnen met rate λ. Er zijn dan twee mogelijkheden: de klant wordt geweigerd met rejection kosten r en we blijven in staat x, met de relatieve waardefunctie $\lambda\ (r + V_n(x))$, of de klant wordt toegelaten in het systeem en het aantal klanten in het systeem wordt met één opgehoogd met de relatieve waardefunctie $\lambda\ V_n(x + 1)$. De optimale actie wordt gegeven door het minimum te bepalen van deze twee mogelijkheden wanneer er een klant binnenkomt met rate λ.
Ook verlaten klanten het systeem met rate μ. Op dat moment wordt het aantal klanten in het systeem één minder met relatieve waardefunctie $\mu V_n(x - 1)$, waar natuurlijk geldt dat het aantal klanten in het systeem niet negatief kan worden. Er is ook nog de mogelijkheid dat er in een tijdsperiode geen nieuwe klanten binnenkomen of dat er klanten vertrekken, in dat geval blijft het systeem in dezelfde staat met relatieve waardefunctie $(1 - \lambda - \mu)V_n(x)$.

Met behulp van deze relatieve waardefunctie kan nu de optimale strategie bepaald worden voor verschillende parameters. Er kan zo dus worden bepaald bij welke waarden van x er geen klanten meer worden toegelaten tot het systeem. Volgens [6] geldt dat een drempelstrategie optimaal is als de relatieve waardefunctie functie convex is en de functie van de holding kosten $C(x)$ convex en stijgend is.

Deze methode werkt echter alleen waneer een constante λ verondersteld wordt. Als λ niet constant is kunnen we de bovenstaande formule ook niet gebruiken om de drempelwaarde te bepalen. Bij het analytische voorbeeld van hierboven wordt de optimale drempelwaarde gekozen aan de hand van de relatieve waardefunctie. In het volgende voorbeeld wordt de drempelwaarde niet gekozen op basis van de relatieve waardefunctie, maar op basis van de performance van een simulatierun. Als indicatie van de performance wordt dan de Service Level van een systeem gebruikt bij een bepaalde geschatte parameter.

3.2 Het bepalen van een parameter van een systeem met behulp van SA

Met behulp van Stochastic Approximation kan door middel van simulatie een control parameter van een systeem worden bepaald, bijvoorbeeld zoals in paragraaf 3.1 de drempelwaarde tot waar klanten worden toegelaten tot het systeem. Er wordt dan geen constante λ verondersteld, maar de aankomsten zijn stochastisch. Een uitwerking van deze simulatie en de resultaten worden beschreven in paragraaf 3.2.1.

Het is ook mogelijk om in de $M|M|s$ queue het aantal servers s te bepalen met behulp van Stochastic Approximation door middel van simulatie. Een uitwerking van deze simulatie en de resultaten hiervan worden beschreven in paragraaf 3.2.2.

3.2.1 Het bepalen van de optimale drempelwaarde θ met behulp van SA

Als we er vanuit gaan dat het aantal aankomende klanten per tijdseenheid niet constant is maar stochastisch, kunnen we Stochastic Approximation gebruiken om op deze manier de drempelwaarde x te bepalen waarboven geen klanten meer worden toegelaten tot het systeem (in dit geval bij een $M|M|s$ queue). Aangezien we werken met Stochastic Approximation, noemen we deze drempelwaarde θ. We gaan nu dus niet de aankomende klanten λ_{t+1} voorspellen, maar de drempelwaarde θ_{t+1} instellen. De recursieve formule ziet er dan als volgt uit

$$\theta_{t+1} = \theta_t \pm \varepsilon_t [\alpha - perf(\theta_t)].$$

Hierbij is α een gewenste target waarde en deze gewenste waarde wordt vergeleken met de performance van het systeem bij de geschatte drempelwaarde θ_t. Deze performance kan worden bepaald door een simulatie uit te voeren. Wanneer deze performance erg goed was (de waarde van de performance ligt dicht bij de gewenste target waarde), zal de volgende voorspelling van de drempelwaarde relatief dicht bij de vorige geschatte waarde liggen dan wanneer de performance ver van de gewenste target waarde ligt.

Aangezien we λ_t niet constant veronderstellen, kan hier echte data van bijvoorbeeld een call center worden gebruikt om de simulaties uit te voeren. Ook is het mogelijk om het aankomstproces te simuleren wanneer wordt uitgegaan van bijvoorbeeld een Poisson proces met een bepaalde parameter λ_t. De tussenaankomsttijden van de klanten is een getal getrokken uit de exponentiële verdeling met parameter λ_t.

In dit specifieke voorbeeld gebruiken we als indicatie van de performance van de simulatie de Service Level. Hierbij nemen we als norm de gewenste Service Level zoals reeds gegeven in hoofdstuk 1, namelijk 80% van de klanten wacht minder dan 20 seconden. Dit percentage wordt met α weergegeven, dus $\alpha = 80\%$. Het percentage klanten dat minder wacht dan 20 seconden nemen we als maat voor de performance van de simulatie met de drempelwaarde θ_t. De performance van de simulatie met drempelwaarde θ_t noteren we als $perf(\theta_t)$. Een voorbeeld hiervan zou dus kunnen zijn $perf(\theta_t) = 40\%$, hierbij wachtte in de uitgevoerde simulatie 40% van de klanten minder dan 20 seconden.

Om de simulatie van de $M|M|s$ queue uit te voeren, maken we gebruik van de simulatiecode zoals beschreven in [4]. Hierin wordt de $M|M|s$ queue gesimuleerd in C++. Deze code kan worden aangepast zodanig dat hij bruikbaar is om Stochastic Approximation toe te passen en een drempelstrategie te implementeren. We gaan ervan uit dat alleen gehele waarden een drempel kunnen zijn, aangezien het onmogelijk is om halve klanten te weigeren.

Er wordt een initiële waarde θ_t gekozen, afhankelijk van de parameters van de simulatie. Aan de hand van deze gekozen drempel wordt een simulatie uitgevoerd. Er wordt gebruik gemaakt van discrete-event simulatie. Events zijn gebeurtenissen die de toestand van het model wijzigen. De toestandsvariabelen in dit systeem zijn: het aantal servers dat bezet is en het tijdstip van vrijkomen; het aantal klanten in de wachtrij en de aankomsttijden van de klanten in de wachtrij. Aan de hand hiervan kan de wachttijd van een klant bepaald worden, welke nodig is om de totale Service Level van een simulatie te bepalen.
Events (een vertrek of aankomst van een klant) kunnen alleen op discrete tijdstippen plaatsvinden, dus de toestand van het systeem verandert niet continu. Deze aanname leidt ertoe dat de simulatieklok alleen wordt verzet op het moment dat een event plaatsvindt. Ook verandert de toestand van het systeem alleen als een event plaatsvindt [4].

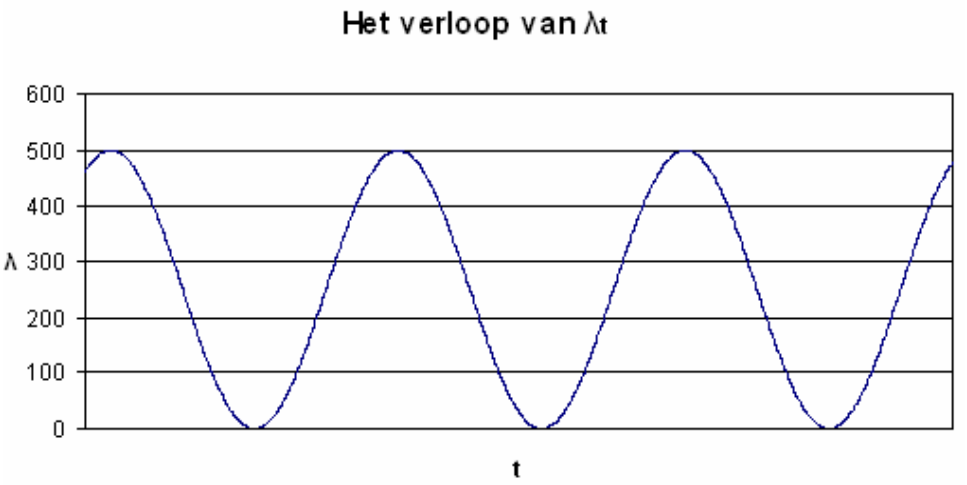
De recursieve formule bepaalt een drempelwaarde die waarschijnlijk niet-geheeltallig is. Om toch een geheeltallige drempelwaarde te verkrijgen, wordt in het simulatieprogramma de optimale waarde van de drempel naar beneden afgerond. Het is duidelijk dat de drempelwaarde naar beneden moet worden afgerond en niet naar boven. Wanneer de optimale drempelwaarde naar boven zou worden afgerond, is de gewenste Service Level niet langer 80% maar lager. Des te hoger de drempel komt te liggen, des te lager de Service Level wordt.

Dit is natuurlijk logisch, hoe meer klanten worden toegelaten des te langer ze moeten wachten en dat leidt dus ook tot een lagere Service Level.

Ook wordt de drempelwaarde naar beneden afgerond op het moment dat er wordt besloten of een aankomende klant het systeem mag betreden. Er wordt dan gekeken naar het aantal klanten in de wachtrij plus het aantal servers dat bezet is, en wanneer dit getal kleiner is dan de (naar beneden afgeronde) drempelwaarde, mag de klant het systeem betreden. Wanneer de drempel hier niet naar beneden wordt afgerond, zal op het moment dat er zich bijvoorbeeld in totaal drie klanten in het systeem bevinden, en de drempelwaarde 3.5 bedraagt, er nog steeds klanten worden toegelaten in het systeem. Dit is natuurlijk niet de bedoeling, omdat er dan uiteindelijk een klant wordt toegelaten en er dus vier klanten in het systeem zijn terwijl dit niet is toegestaan.

3.2.1.1 Resultaten

In deze paragraaf worden de resultaten van de simulatie gegeven. In het programma wordt een Poisson proces gesimuleerd met rate λ_t. Er wordt verondersteld dat functie van λ_t eruit ziet zoals weergegeven in grafiek 1.



Grafiek 1: Het verloop van λ_t

λ_t wordt gegeven door

$$\lambda_t = 250 \cdot (\sin(t) + 1) .$$

Hier zijn λ_t het aantal klanten dat het systeem binnenkomt per uur. De tussenaankomsttijden van de klanten zijn dan exponentieel verdeeld met parameter λ_t / 60 (per minuut). Hierbij is *t* de simulatietijd op het moment dat de klant het systeem binnenkomt. Iedere keer dat een

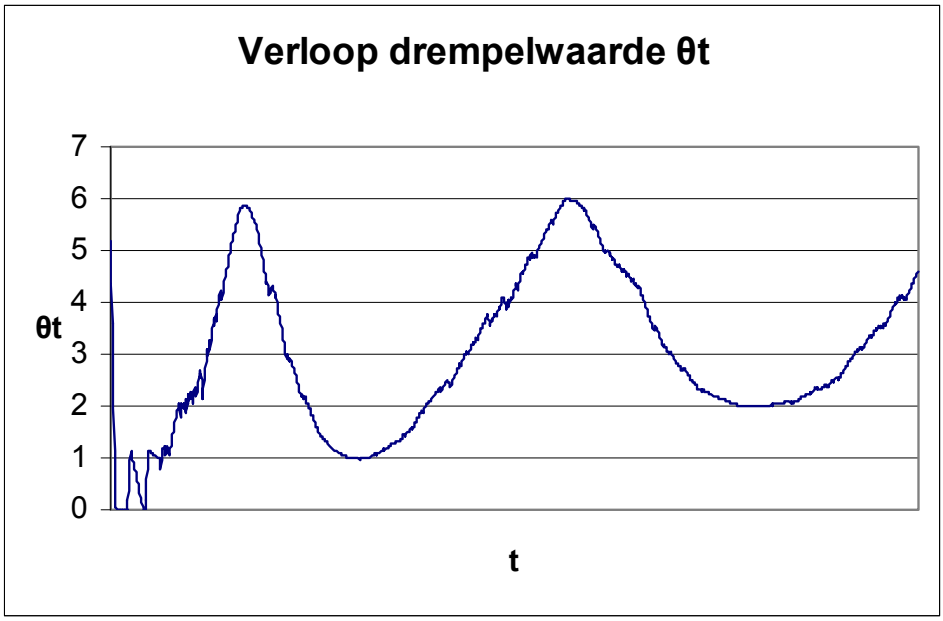
nieuwe klant het systeem binnenkomt, wordt de nieuwe drempelwaarde ingesteld. Aan de hand van deze drempelwaarde wordt bepaald of de klant het syteem mag betreden. Hoe kan deze drempelwaarde zo worden ingesteld, dat de norm α wordt behaald?

Verder wordt het aantal servers gegeven door $s = 3$. De verwachte servicetijd van een klant wordt gegeven door $\beta = 2.4$ minuten. Een enkele simulatierun telt gemiddeld 42.000 minuten.

Wanneer een klant het systeem verlaat, wordt bepaald hoe lang deze klant heeft gewacht. Aan de hand hiervan wordt de Service Level bepaald door het aantal klanten te nemen dat korter dan 20 seconden heeft gewacht en dit getal te delen door het totaal aantal klanten dat het systeem mocht betreden.

Telkens als een nieuwe klant het systeem binnenkomt wordt de nieuwe drempelwaarde ingesteld. Hierbij geldt dat wanneer de Service Level bij aankomst van de vorige klant hoger was dan de Service Level bij aankomst van de huidige klant, de drempelwaarde naar beneden gaat (met behulp van de recursieve formule). Aangezien de huidige Service Level lager ligt, betekent dit dat er meer klanten het systeem zijn binnengekomen. Om toch een goede Service Level te blijven behalen moet de drempelwaarde naar beneden (dan komen er dus minder klanten het systeem binnen). Andersom geldt dan dus ook dat wanneer de huidige Service Level hoger ligt dan de vorige, er minder klanten het systeem zijn binnengekomen. De drempelwaarde kan dan dus omhoog, er kunnen meer klanten het systeem betreden en toch kan een goede Service Level worden behaald. Echter, wanneer de huidige Service Level hoger ligt dan de norm α, geldt dat de drempelwaarde altijd naar beneden gaat. Het verschil [α – Service Level] is dan altijd negatief.

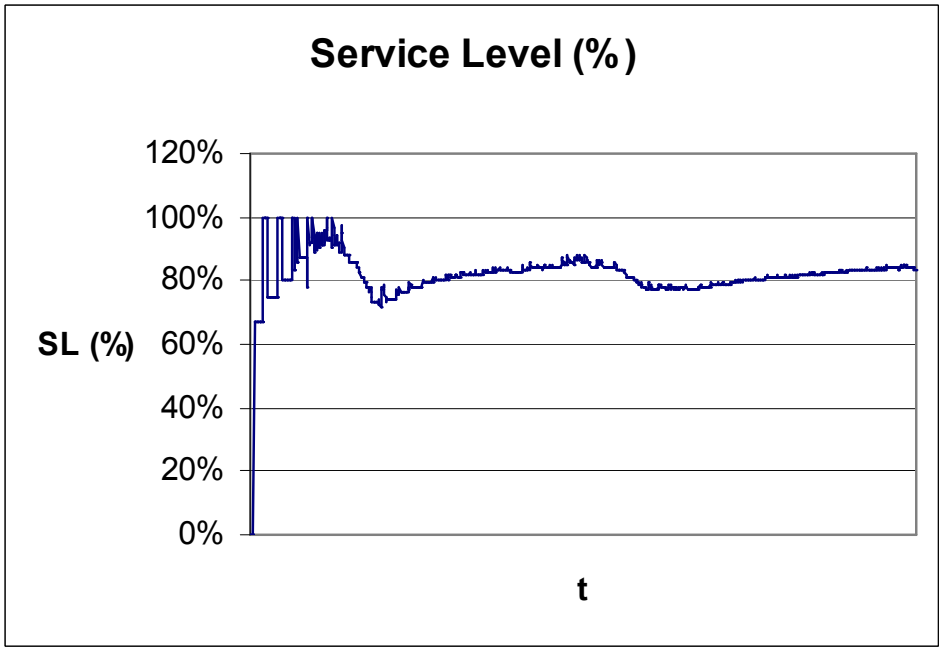
Op deze manier beweegt de drempelwaarde mee met het aantal klanten dat het systeem betreedt. In grafiek 2 wordt het verloop van de drempelwaarde θ_t weergegeven, wanneer als startwaarde $\theta_0 = 6$.



Grafiek 2: Het verloop van de drempelwaarde θ_t

In deze grafiek is dezelfde golvende beweging te zien als in de grafiek van λ_t. Hij beweegt echter niet zo snel als λ_t. Dit is natuurlijk het geval omdat één enkele klant met een hoge λ niet meteen effect heeft op de Service Level van de hele simulatierun. Pas nadat er veel klanten met een hoge λ zijn binnengekomen (dus veel klanten vlak na elkaar), wordt de drempelwaarde naar beneden ingesteld.

In grafiek 3 wordt het verloop van de Service Level uitgezet tegen de tijd.



Grafiek 3: Het verloop van de Service Level gedurende de simulatierun

In grafiek 3 is te zien dat de Service Level gedurende de simulatierun telkens rond de norm α (80% van de klanten wacht minder dan 20 seconden in de wachtrij) schommelt.

3.2.2 Het bepalen van de optimale s met behulp van SA

Een ander voorbeeld waarin Stochastic Approximation kan worden gebruikt om een parameter te schatten is wederom in de $M|M|s$ queue. Aangezien λ_t gesimuleerd wordt en dus niet constant is per uur, is het erg moeilijk om het aantal servers s dat nodig is om een goede Service Level te behalen te bepalen. Wanneer λ constant zou zijn en vast kan het aantal servers s makkelijk worden bepaald met behulp van de Erlang-C formule.

De Stochastic Approximation stap in deze simulatie wordt nu gegeven door

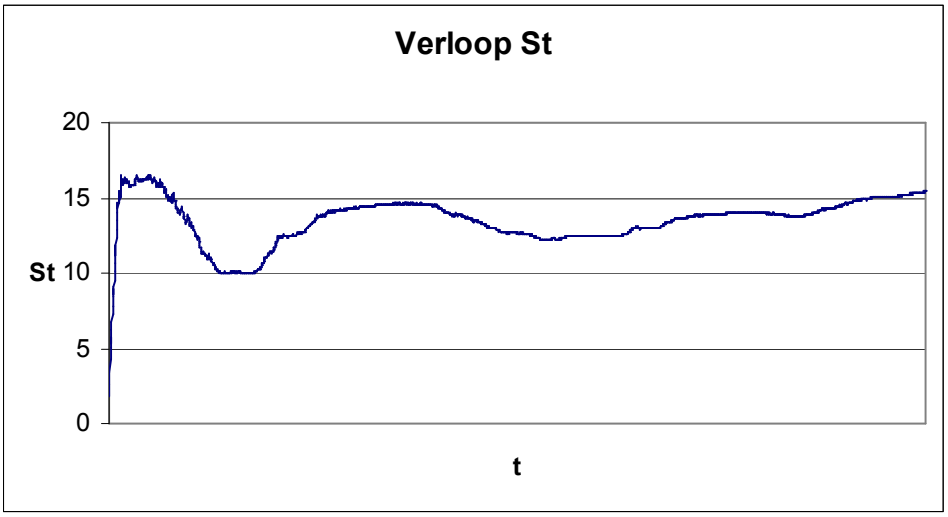
$$s_{t+1} = s_t \pm \varepsilon_t[\alpha - perf(s_t)].$$

Hierbij stelt α weer de gewenste target waarde voor, namelijk de norm dat 80% van de klanten minder dan 20 seconden wacht. Net als in het vorige voorbeeld wordt weer gebruik gemaakt van discrete-event simulatie. Bij de vertrekkende klant wordt weer telkens bijgehouden of hij minder dan 20 seconden heeft gewacht. Bij een nieuwe aankomende klant wordt de nieuwe waarde s_t ingesteld. Hierbij geldt precies het tegenovergestelde als bij het vorige voorbeeld. Hier geldt dat wanneer de Service Level bij aankomst van de vorige klant hoger was de de Service Level bij aankomst van de huidige klant, het aantal servers omhoog gaat met behulp van de recursieve formule. Er zijn dan namelijk meer klanten binnengekomen (wat leidt tot een lagere Service Level), en daarom zijn meer servers nodig om een goede Service Level te blijven behalen. Wanneer de Service Level bij aankomst van de vorige klant lager was dan de Service Level bij aankomst van de huidige klant, gaat het aantal servers omlaag.

3.2.2.1 Resultaten

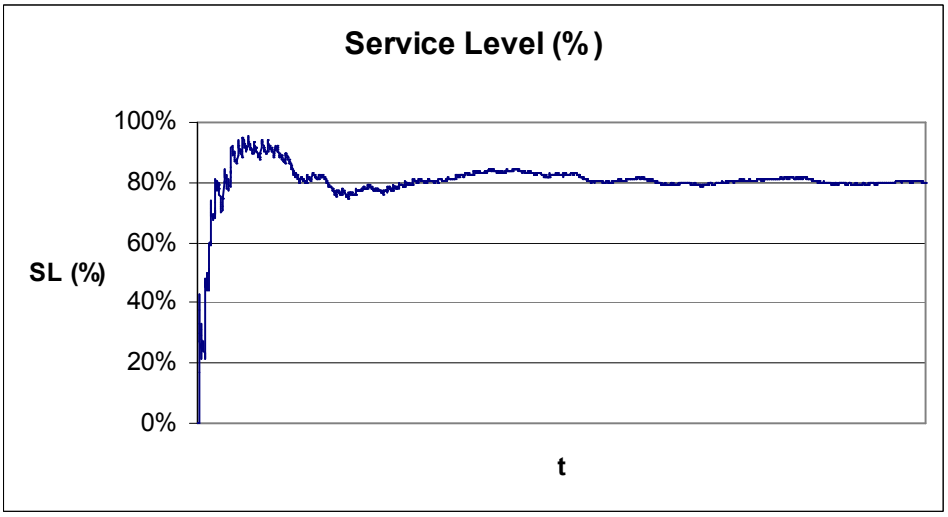
In deze simulatie wordt gebruik gemaakt van dezelfde aanname voor λ_t als in het vorige voorbeeld, dus $\lambda_t = 250 \cdot (\sin(t) + 1)$. Tevens geldt hier weer dat de verwachte servicetijd β = 2.4 minuten, en één enkele simulatierun telt gemiddeld 42.000 minuten. Net als bij het bepalen van de drempelwaarde geldt ook hier weer dat s geheel moet zijn, een niet-geheeltallig aantal servers is natuurlijk niet mogelijk.

In grafiek 4 wordt het verloop van s_t weergegeven, wanneer als startwaarde $s_0 = 1$ wordt genomen.



Grafiek 4: Het verloop van het aantal servers s_t

Ook in grafiek 4 is te zien (net zoals bij het vorige voorbeeld) dat het aantal servers meebeweegt met het aantal klanten dat het systeem betreedt. In grafiek 5 wordt het verloop van de Service Level weergegeven tijdens de simulatierun.



Grafiek 5: Het verloop van de Service Level tijdens de simulatierun

In grafiek 5 is ook weer te zien dat de Service Level tijdens de simulatierun rond de norm α beweegt. Bij beide voorbeelden (zowel het instellen van de drempelwaarde als het instellen

van het aantal servers) is dus te zien dat de parameter goed ingesteld wordt, aangezien de Service Level vrijwel nergens beneden de norm α komt.

3.3 Nadelen en discussie

Wanneer geen goede startwaarde wordt gekozen voor een parameter, neemt de rekentijd erg toe. Ook de keuze van de stapgrootte is erg belangrijk. Tevens is het gebruik van een recursieve formule bij een geheeltallige parameter lastig. Bij de voorgaande twee voorbeelden waren de parameters die geschat moeten worden beide geheeltallig. Aangezien de recursieve formule ook niet-geheeltallige oplossingen beschouwt, worden veel tussengelegen waarden (bijvoorbeeld 5.1, 5.2, etc.) bekeken. Dit is niet efficiënt.

Hoofdstuk 4: Conclusie

Stochastic Approximation is een methode die benaderingen of voorspellingen kan geven van een bepaalde parameter. Met behulp van een recursieve formule wordt een reeks van benaderingen gegeven die convergeert naar de optimale waarde of parameter. De eerste Stochastic Approximation algoritmes zijn het Robbins-Monro algoritme en het Kiefer-Wolfowitz algoritme, welke beide dateren uit begin jaren vijftig van de vorige eeuw. Deze algoritmes benaderen respectievelijk de oplossing van de vergelijking $M(x) = v$ en het maximum van de functie $M(x)$. De onderliggende functie $M(x)$ is echter niet bekend. Dit is een groot voordeel van Stochastic Approximation. Ook is de rekentijd en het gebruik van geheugen relatief klein omdat een recursieve formule een simpele lineaire vergelijking is waarbij de nieuwe waarde bepaald wordt uit de oude waarde. Een nadeel is dat wanneer de startwaarde of de stapgrootte niet goed gekozen wordt, het aantal iteraties totdat de optimale waarde bereikt wordt gigantisch kan toenemen.

Stochastic Approximation is ook te gebruiken voor het sturen van systemen. Een voorbeeld hiervan is admission control in een wachtrij. Hierbij wordt een drempelwaarde ingesteld. Onder deze drempelwaarde mogen klanten het systeem betreden, daarboven worden ze geweigerd. Een drempelstrategie is optimaal.

Het eerste voorbeeld dat besproken is, is het instellen van deze drempelwaarde. Het aantal klanten dat het systeem per tijdseenheid binnenkomt is stochastisch in plaats van constant. De performance van het systeem bij een nieuw ingestelde drempelwaarde wordt dan bepaald met behulp van een simulatieprogramma. Hierbij geeft de Service Level de performance aan. Dit is het percentage klanten dat minder wacht dan twintig seconden.
De tweede simulatie die is uitgevoerd in dit verslag is het bepalen van het aantal servers s dat nodig is om een bepaalde Serivce Level te behalen. Hierbij is het aantal klanten dat het systeem per tijdseenheid binnenkomt niet constant, maar stochastisch waardoor de Erlang-C formule niet nauwkeurig genoeg is.
Aan de hand van het verschil in Service Level tussen de oude ingestelde parameter en de nieuw ingestelde parameter, wordt de nieuwe parameter bepaald. Deze parameter beweegt dus mee met het aantal klanten dat het systeem binnenkomt. Het blijkt dat het instellen van een

parameter met behulp van Stochastic Approximation erg goed werkt, de Service Level schommelt telkens rond de norm dat 80% van de klanten minder wacht dan 20 seconden.

Een bijkomend moeilijkheid van stochastic approximation is dat de recursieve formule ook niet-geheeltallige waarden beschouwt, terwijl een niet-geheeltallige drempelwaarde of aantal servers niet mogelijk is.

Literatuurlijst

[1] Kushner, H.J. en Yin, G.G. (1997) , “Stochastic Approximation and Recursive algorithms and applications”, New York: Springer

[2] Raghunathan, V., “Stochastic Approximation”, pp. 1-11

[3] Edwards, C.H. en Penney, D.E. (2002), “Calculus 6^{e}, Early Transcendentals”, New Jersey: Prentice Hall, pp. 201-202

[4] Tijms, H. (2002), “Operationele analyse, een inleiding in modellen en methoden”, Utrecht: Epsilon Uitgaven, pp. 405-411

Websites

[5] http://www.math.vu.nl/~koole/ccmath/ErlangC/index.php

[6] http://www.cs.vu.nl/obp/edu/so/notes.pdf, pp. 34

[7] http://en.wikipedia.org/wiki/Gauss-Newton_algorithm

[8] http://en.wikipedia.org/wiki/Convergence_of_random_variables