Exact Sampling: Der Propp-Wilson-Algorithmus

Markus Gerstel

Proseminar: Markovketten in der Algorithmik
Technische Universität München
gerstel@in.tum.de

Zusammenfassung Der Propp-Wilson-Algorithmus liefert eine Lösung für das Sampling-Problem bei Markovketten. Im Gegensatz zu dem bekannten MCMC-Sampling kann der Propp-Wilson-Algorithmus perfekte Ergebnisse liefern und terminiert i.A. selbständig. Im Folgenden wird der Aufbau und die Funktionsweise des Propp-Wilson-Algorithmus dargelegt. Anschliessend wird kurz auf mögliche Erweiterungen des Algorithmus eingegangen

1 Hintergrund: Probleme bei Markov Chain Monte Carlo-Sampling

Das Sampling-Problem bei Markovketten beschreibt das Problem, dass zu einer gegebenen Markovkette zufällig einen Zustand so herausgegriffen wird, dass die Wahrscheinlichkeit diesen Zustand zu erhalten mit seiner Auftrittswahrscheinlichkeit in der stationären Verteilung π der Markovkette übereinstimmt.

Ein möglicher Lösungsansatz, das Markov Chain Monte Carlo-Sampling, wurde bereits vorgestellt: Man betrachtet eine irreduzible aperiodische Markovkette und eine beliebige Startverteilung $\mu^{(0)}$. Anschließend durchläuft man diese Markovkette n mal. Das Konvergenzkriterium bei Markovketten garantiert nun, dass $\lim_{n\to\infty} \mu^{(n)} \to \pi$.

Leider besagt das Konvergenzkriterium nicht, dass es ein n gibt, für das $\mu^{(n)} = \pi$ gilt. In den meisten Fällen ist sogar das Gegenteil der Fall. Dies bedeutet aber, dass in der Regel ein Fehler $\epsilon > 0$ zwischen der Verteilung der MCMC-Ausgabe und π vorhanden ist. Dieser Fehler kann natürlich klein gehalten werden, indem man die Anzahl der Simulationsschritte n erhöht. Will man den Fehler jedoch unter einer gewählten Fehlerschranke $\epsilon' \geq \epsilon$ halten, so kann je nach Konvergenzgeschwindigkeit der gegebenen Markovkette die Zahl der notwendigen Simulationsschritte n' sehr schnell ansteigen. So könnte für ein $\epsilon' = 0.01$ bei einer bestimmten Markovkette das n' z.B. in einer unpraktischen Größenordnung von 10^{100} liegen. Man erreicht mit dieser Strategie also schnell die Grenzen des technisch Machbaren.

Üblicherweise löst man dieses Problem, indem man n optimistisch wählt und hofft, dass μ so schnell konvergiert, dass $\epsilon \leq \epsilon'$ ist. Hier stellt sich sogleich die Frage wie schnell μ gegen π konvergiert. Leider ist die Bestimmung der Konvergenzgeschwindigkeit meist sehr aufwendig.

2 Der Propp-Wilson-Algorithmus

Jim Propp und David Wilson entwickelten 1996 am Massachusetts Institute of Technology einen anderen Ansatz, der die oben genannten Probleme elegant umgeht: Der Propp-Wilson-Algorithmus (auch *Coupling From The Past* genannt) liefert als Ausgabeverteilung exakt $\mu^{(n)} = \pi$ und kann zusätzlich selbständig erkennen wenn π erreicht ist. Die Konvergenzgeschwindigkeit muss also nicht extra bestimmt werden.

2.1 Funktionsweise des Propp-Wilson-Algorithmus anhand eines Beispiels

Gegeben sei eine reversible, irreduzible und aperiodische Markovkette mit der Zustandsmenge $S = \{s_1, .., s_k\}$, einer Übergangsmatrix P und einer gültigen Updatefunktion $\phi : S \times [0, 1] \to S$. Aus den Eigenschaften der Markovkette folgt automatisch, dass eine stationäre Verteilung π existiert. Nun definiert man noch zwei Folgen: $N_1, N_2, ... \in \mathbb{N}$ sei eine monoton steigende Folge positiver ganzer Zahlen, und $U_0,\ U_{-1},\ U_{-2}, ... \in [0, 1]$ sei eine Folge gleichverteilter, voneinander unabhängiger Zufallszahlen.

Der Algorithmus sieht nun folgendermaßen aus:

1. Setze $m = 1$.
2. Simuliere für alle $s \in S$ eine Markovkette mit Startzustand s.
   Diese Markovkette läuft von der Zeit $-N_m$ bis zur Zeit 0.
   Verwende ϕ mit Eingaben $U_{-N_m+1}, U_{-N_m+2}, .., U_{-1}, U_0$
3. Wenn alle k Markovketten aus Schritt 2 im gleichen Zustand s' enden, gib s' aus und brich ab. Sonst weiter bei Schritt 4.
4. Erhöhe m um 1 und fahre bei Schritt 2 fort.

$$P = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} \qquad \phi = \begin{cases} & x \in [0, \frac{1}{2}[& [\frac{1}{2}, 1] \\ (s_1, x): & s_2 & s_3 \\ (s_2, x): & s_1 & s_3 \\ (s_3, x): & s_1 & s_2 \end{cases}$$

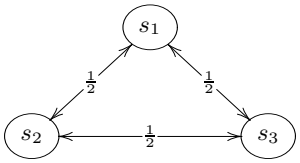
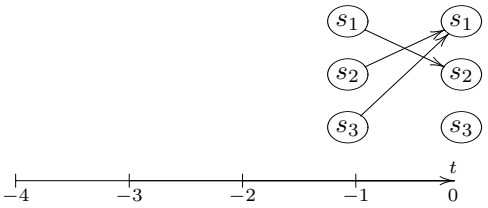


Abbildung 1. Random Walk auf drei Zuständen

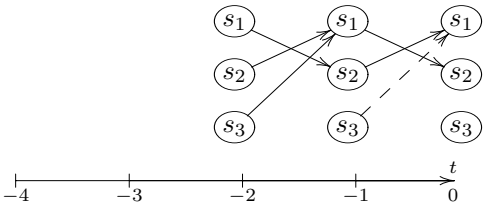
Betrachten wir einen Random Walk auf $S = \{s_1, s_2, s_3\}$.
Die zugehörige Markovkette ist offensichtlich irreduzibel, reversibel und aperiodisch. Die Übergangsmatrix P und Übergangsfunktion ϕ lassen sich einfach konstruieren. Für die Folge N wählt man $N_x = x\ \forall x \in \mathbb{N}$, und als „Zufallszahlen“ werden $U_0 = 0,48, U_{-1} = 0,11, U_{-2} = 0,72$ vorgegeben.

Jetzt kann man mit dem Algorithmus beginnen und setzt $m = 1$. Anschließend simuliert man ab der Zeit $-N_m = -1$ bis 0 für alle Startzustände $s \in S$

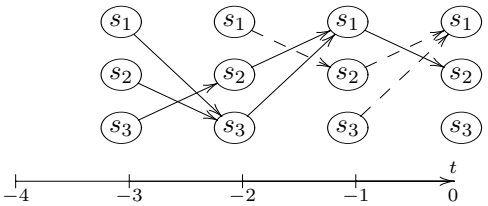
die entsprechende Markovkette und verwendet den Übergang aus $\phi(s, U_0)$ mit $U_0 = 0,48$. Zeichnet man die Zustandsübergänge zur Zeit t in ein Diagramm ergibt sich folgendes Bild:



Die beiden Markovketten, die bei s_2 und s_3 starten, gehen nach einem Simulationsschritt in den Zustand s_1. Die MarkovKette die bei s_1 startet, geht nach einem Simulationsschritt in den Zustand s_2. Da die drei Markovketten nicht im gleichen Zustand enden, erhöhen wir m auf 2, und simulieren wieder ab der Zeit $-N_m = -2$ bis 0 die drei Markovketten. Für den ersten Simulationsschritt verwendet man den Übergang aus $\phi(s, U_{-1})$ und für den zweiten Schritt den (bereits ermittelten) Übergang $\phi(\phi(s, U_{-1}), 0)$. Für $U_{-1} = 0,11$ ergibt sich nun folgendes Bild:



Wieder enden die drei Markovketten nicht im gleichen Zustand. Man setzt $m = 3$ und simuliert ab dem früheren Zeitpunkt $-N_m = -3$ bis 0. Der erste Simulationsschritt ergibt sich diesmal aus $\phi(s, U_{-2})$, die weiteren Schritte sind bereits bekannt. Für $U_{-2} = 0,72$ ergibt sich nun folgendes Bild:



Nun treffen sich alle drei Markovketten im Endzustand s_2. Daher gibt der Algorithmus s_2 als Ergebnis aus und terminiert.

Betrachtet man nun kurz das letzte Diagramm, so kann man feststellen, dass für weitere Simulationen mit $m \geq 4$ unabhängig von weiteren Zufallszahlen das Ergebnis immer s_2 lauten würde.

2.2 Folgerungen

Mit diesem kleinen Experiment sollten einige Besonderheiten des Propp-Wilson-Algorithmus erkennbar geworden sein: Der Algorithmus verfolgt den Zustands-

wechsel der k Markovketten in die Vergangenheit. Durch diese Eigenschaft erklärt sich der Name *Coupling From The Past (CFTP)*. Ab einem bestimmten Zeitpunkt - hier ab einem $m \geq 4$ - wird die Ausgabe immer s_2 sein. Dies bedeutet, dass man auch eine Markovkette betrachten kann, die schon beliebig lange vorher gelaufen ist - zum Zeitpunkt 0 wird sie in s_2 sein. Man kann insbesondere auch eine Kette betrachten, die schon unendlich lange gelaufen ist. Das Konvergenzkriterium besagt jedoch dass eine Markovkette für eine Laufzeit $n \to \infty$ die stationäre Verteilung π erreicht. Dies wiederum würde bedeuten, dass zum Zeitpunkt 0 eine stationäre Verteilung vorliegt.

Diese zunächst interessanten Vermutungen sollen nun im Folgenden formal betrachtet werden. Außerdem stellt sich die Frage, ob dieser Algorithmus auch für nicht-triviale Markovketten terminiert.

3 Formale Betrachtung des Propp-Wilson-Algorithmus

3.1 Beweis: Korrektes Ergebnis des Propp-Wilson-Algorithmus

Benennt man die Ausgabe des Algorithmus Y, dann ergibt sich die zu beweisende Behauptung:

$$P(Y = s_i) = \pi_i \tag{1}$$

Wählt man einen Zustand $s_i \in S$ fest, so wird diese Behauptung durch

$$\forall \epsilon > 0 : |P(Y = s_i) - \pi_i| \leq \epsilon \tag{2}$$

bewiesen. Wählt man auch ein ϵ fest, so ist sicher, dass

$$\exists M : P(\textit{Algorithmus muss nicht Ketten vor} - N_M \textit{testen}) \geq 1 - \epsilon \tag{3}$$

Wir gehen dabei davon aus, dass der Algorithmus terminiert.

Man wähle nun ein M und lasse eine weitere Markovkette von dem Zeitpunkt $-N_M$ bis 0 laufen. Diese soll dasselbe ϕ und dieselben U verwenden, aber bereits zum Zeitpunkt $-N_M$ die stationäre Verteilung π haben. Den Zustand dieser neuen Kette zum Zeitpunkt 0 benennt man $\tilde{Y}$.

Da π eine stationäre Verteilung ist, folgt, dass $\tilde{Y}$ ebenfalls stationär verteilt ist. Falls wir nicht Ketten vor $-N_M$ testen müssen, gilt weiter: $\tilde{Y} = Y$. Die Wahrscheinlichkeit für diesen Fall ist laut (2) mindestens $1 - \epsilon$:

$$P(Y = \tilde{Y}) \geq 1 - \epsilon \quad \Longrightarrow \quad P(Y \neq \tilde{Y}) \leq \epsilon \tag{4}$$

Nun gilt wegen der Stationärverteilung der neuen Kette:

$$P(Y = s_i) - \pi_i \quad = \quad P(Y = s_i) - P(\tilde{Y} = s_i) \tag{5}$$

$$\leq P(Y = s_i, \tilde{Y} \neq s_i) \quad \leq \quad P(Y \neq \tilde{Y}) \leq \epsilon \tag{6}$$

Zu zeigen war: $|P(Y = s_i) - \pi_i| \leq \epsilon$, aber mit umgedrehten Vorzeichen geht die Beweiskette ähnlich:

$$\pi_i - P(Y = s_i) \quad = \quad P(\tilde{Y} = s_i) - P(Y = s_i) \tag{7}$$

$$\leq P(\tilde{Y} = s_i, Y \neq s_i) \quad \leq \quad P(Y \neq \tilde{Y}) \leq \epsilon \qquad \square \tag{8}$$

Der Algorithmus liefert also das gewünschte Ergebnis - wenn er terminiert.

3.2 Terminiert der Propp-Wilson-Algorithmus ?

$$\phi^* = \begin{cases} & x \in [0, \frac{1}{2}[& [\frac{1}{2}, 1] \\ \hline (s_1, x): & s_2 & s_3 \\ (s_2, x): & s_3 & s_1 \\ (s_3, x): & s_1 & s_2 \end{cases}$$

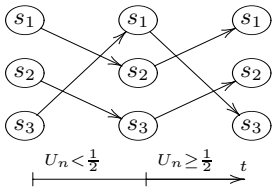


Abbildung 2. Übergangsfunktion ϕ^*: Algorithmus terminiert nicht mehr

Betrachtet man dieselbe Markovkette wie im letzen Beispiel mit einer geringfügig geänderten Übergangsfunktion ϕ^*, so kann man schnell sehen, dass unabhängig von den Zufallsvariablen U_n die Ketten sich niemals in einem Endzustand treffen können. Der Algorithmus terminiert nicht mehr. Er versagt bei ungünstigen ϕ.

Wann terminiert nun der Algorithmus? Es gibt für den Propp-Wilson-Algorithmus das sogenannte „0-1 law“:

$$P(\textit{Algorithmus terminiert}) > 0 \quad \Longrightarrow \quad P(\textit{Algorithmus terminiert}) = 1$$

Es genügt also *einen* Fall zu finden, in dem der Algorithmus terminiert.

Über die Laufzeit sagt dies selbstverständlich nichts aus. Die Laufzeit ist eventuell sehr lange, ähnlich wie man bei einer Laplace-Münze auch eventuell über 1000 Würfe auf seine favorisierte Seite warten kann. Dennoch sollte man den Algorithmus wenn er einmal gestartet wurde nicht abbrechen, denn dann würde man durch die Beschränkung auf schnelle Ergebnisse diejenigen Fälle ausser Acht lassen, in denen die Ergebnisse etwas länger auf sich warten lassen. Dies verfälscht natürlich die Ausgabeverteilung.

4 Optimierungen und Ausblick

Wenn man die Idee hinter dem Algorithmus verstanden hat, fallen einem zwei vielleicht in der Praxis unangenehme Eigenschaften beim *Coupling From The Past* auf: Einerseits könnte es vielleicht einfacher oder schneller sein, die Ketten in die Zukunft zu simulieren. Gibt es *Coupling To The Future* ? Andererseits könnte man Speicherplatz sparen, wenn man sich die Zufallszahlen U_n nicht merkt, sondern nach Bedarf neue Zufallszahlen erzeugt.

Zuletzt ist man in der Praxis natürlich immer an einer Laufzeitoptimierung interessiert. Beim Propp-Wilson-Algorithmus lautet daher die zugehörige Frage: Muss man alle $m * k$ Ketten simulieren ?

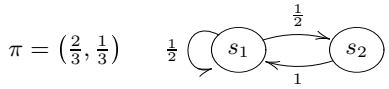


Abbildung 3. Coupling To The Future

4.1 Coupling To The Future

Man kann mit einem Gegenbeispiel schnell und anschaulich zeigen, dass es keinen *Coupling To The Future*-Algorithmus gibt: Man simuliere zwei Kopien der hier angegebenen Markovkette, eine mit Startzustand s_1, die andere mit Startzustand s_2. Die beiden Ketten werden sich zum Zeitpunkt $N > 0$ treffen. Zum Zeitpunkt $N - 1$, kurz bevor sich die beiden Ketten treffen, wird also eine Kette in s_1, die andere in s_2 sein. Die Kette, die zum Zeitpunkt $N - 1$ in s_2 ist, wird zum Zeitpunkt N bei s_1 sein. Daher muss der Algorithmus immer s_1 ausgeben. Das würde eine Verteilung $\pi^* = (1, 0) \neq \pi$ bedeuten - und führt zum Widerspruch.

4.2 Muss man sich Zufallszahlen merken ?

Tatsächlich ist die π-Verteilung der Ausgabe auch von der Wiederverwendung der Zufallszahlen abhängig. Der interessierte Leser kann den Beweis hierzu in [1], Kapitel 10 finden.

4.3 Laufzeitoptimierungen und Ausblick

In den hier verwendeten Beispielen hatten wir die Markovketten immer ab den Zeitpunkten $-N_x = -x$ betrachtet. Mit einer anderen Folge für N_x kann unter Umständen Rechenaufwand gespart werden. Üblicherweise wählt man $N_x = 2^{x-1}$ $(1, 2, 4, ..)$. Dadurch kann man die Zahl der notwendigen Simulationen von $m * k$ auf $\lceil\sqrt{m}\rceil * k$ reduziert werden.

Der Propp-Wilson-Algorithmus ist üblicherweise für komplexere Probleme interessant. Dies bedeutet meist auch sehr große Zustandsmengen $S = (s_1, .., s_k)$. Es gibt Methoden mit sehr großen Zustandsmengen umzugehen, bei denen dann nur $m * k'$ mit $k' \leq k$ Ketten explizit simuliert werden müssen.

Eine dieser Techniken, das sogenannte Sandwiching, wird in dem folgenden Kapitel vorgestellt.

Literatur

1. Olle Häggström. *Finite Markov Chains and Algorithmic Applications*. Cambridge University Press, 2002. Based on lecture notes.