Bayes'sche Datenanalyse (bda)
Cédric Huwyler

Fachhochschule Nordwestschweiz

Minichallenge: Bayes'sche Suchtheorie

1 Ausgangslage

Es hat sich herausgestellt, dass sich die Bayes'sche Art mit Daten umzugehen (kontinuierliches Updaten - Today's posterior is tomorrow's prior) sehr gut für Suchprobleme eignet. Oft ist die Anzahl von durchgeführten Suchläufen teuer oder limitiert und Methoden, die das gesuchte Objekt schneller finden, sind zu bevorzugen. Die Bayes'sche Statistik eignet sich hier insbesondere, weil sie verfügbare Prior-Information manifest berücksichtigt (und Prior-Information ist für Suchprobleme essentiell) und weil sie durch ihren ebenfalls manifesten Umgang mit Unsicherheit auch gut mit kleinen Datenmengen (erst wenig Suchläufe) umgehen kann. In dieser Minichallenge möchten wir Suchprobleme exemplarisch dazu benutzen, um besser an diese Key-Features der Bayes'schen Statistik heranzukommen.

1.1 Die Suche nach der USS Scorpion

Bayes'sche Statistik wurde zum ersten Mal offiziell für ein Suchproblem eingesetzt, als am 22. Mai 1968 im kalten Krieg das US-amerikanische U-Boot *USS Scorpion* irgendwo zwischen Norfolk (Virginia, USA) und den Azoren während einer geheimen Mission verschollen ging und eine grosse Suchaktion gestartet wurde. Da das U-Boot überall im Umkreis seiner geplanten Route von über 4000 km Länge liegen konnte, war diese Suche enorm schwierig. Als Leiter des Suchteams wurde John Craven ernannt, ein Spezialist in der Auffindung verlorener Objekte. Er hatte sich bereits in anderen Suchproblemen (z.B. eine 1966 von einem Flugzeug an der Küste Spaniens verlorene Wasserstoffbombe) bewährt und hatte bereits dort mit mässiger Akzeptanz versucht, einige Ideen aus der Bayes'schen Statistik einzubringen.

Craven befragte eine grosse Zahl von Experten verschiedener Spezialgebiete und erstellte daraus eine Prior-Wahrscheinlichkeitsverteilung für den Aufenthaltsort der USS Scorpion. Dazu nahm er unter anderem natürlich die geplante Route der USS Scorpion zu Hilfe und konnte glücklicherweise auf ein Netzwerk von im Nordatlantik verteilten Unterwasser-Mikrofonen zurückgreifen, auf denen auch mögliche Explosionsgeräusche zu hören waren, die er dazu benutzte, den möglichen Aufenthaltsort der USS Scorpion weiter einzukreisen. Die resultierende Prior-Wahrscheinlichkeitsverteilung war das eigentliche Erfolgsrezept hinter der Suche, die Bayes'sche Statistik konnte dann diesen Prior einbeziehen und nach jedem Suchlauf mit den neuen Daten updaten.

Am 26. Oktober 1968 wurde das Wrack der USS Scorpion schliesslich von der USS Mizar gefunden, indem deren Crew von Anfang Oktober an Stück für Stück jeweils den wahrscheinlichsten Ort auf der Karte untersuchte und nach einer ergebnislosen Suche die aktuelle Wahrscheinlichkeitsverteilung entsprechend aktualisierte. Das war ein grosser Erfolg für den Bayes'schen Ansatz, am Schluss konnte der Aufenthaltsort der USS Scorpion bis auf 240m genau angegeben werden - eine unglaublich kleine Zahl für eine Suchstrecke von 4000 km Länge!

Die Bayes'sche Statistik wird weiterhin für Suchprobleme benutzt: Unter anderem wurden so zum Beispiel die Überreste des 2009 abgestürzten Air France-Flugs 447 gefunden - die speziell auf Bayes'sche Suchprobleme ausgerichtete Firma *Metron* wurde erst 2011 damit beauftragt und hatte schnell Erfolg. Auch die amerikanische Küstenwache benutzt Bayes'sche Suchtheorie zum Auffinden verschollener Personen und Objekte[1] und sogar professionelle Schatzsucherfirmen suchen z.B. versunkene Schiffe mit Bayes'scher Suchtheorie.

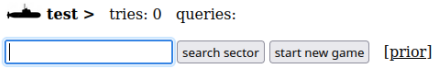
1.2 Ein Suchsimulator

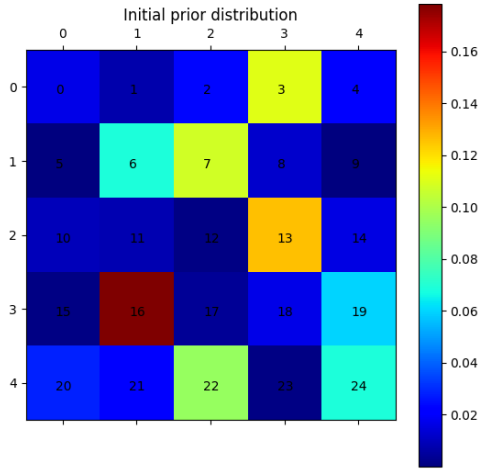
Ein Team der Universität Bilbao hat zu pädagogischen Zwecken einen stark vereinfachten Suchsimulator[2] entwickelt, der die Suche nach der USS Scorpion simuliert, unter der vereinfachenden Annahme, dass nur in neun Feldern (3x3) gesucht werden muss. Leider scheint er nicht (mehr?) korrekt zu funktionieren.

Einen weniger schön anzuschauenden, dafür (hoffentlich) korrekt funktionierenden Simulator findest du auf

`http://chuwyler.pythonanywhere.com/`

Der Suchsimulator ist eine Python-Flask-App und bietet folgendes Interface:





Oben findest du eine Statuszeile mit deinem Nickname (aktuell ohne wirklichen Nutzen), der Anzahl bereits getätigter Suchläufe (tries) und der bereits durchsuchten Sektoren (queries, im Beispiel aktuell noch kein Sektor durchsucht). Im Eingabefeld kannst du die Nummer des Sektors eingeben, den du als nächstes mit 'search sector' durchsuchen möchtest. Die initiale Prior-Verteilung ist auf der Karte farbcodiert (siehe Colorbar). Du bekommst die flachgedrückte Prior-Verteilung als numerische Werte, wenn du oben auf 'prior' klickst.

Natürlich würdest du im obigen konkreten Beispiel mit Sektor 16 mit der höchsten Prior-Wahrscheinlichkeit starten, wie es weitergeht musst du dann mit den hergeleiteten Formeln aus der Aufgabenstellung herausfinden. Wenn du das Objekt gefunden hast, wirst du über ein Popup-Fenster darüber informiert und bekommst nachher auf der Karte den genauen Ort angezeigt. Falls du das aktuelle Spiel abbrechen und ein neues Spiel starten möchtest, kannst du auf 'start new game' drücken.

[1] Search and Rescue Optimal Planning System (Sarops)
[2] `http://www.et.bs.ehu.es/bayes/EN/`

Dieser Simulator wurde extra für diese Minichallenge 'entwickelt'. Er kann nur das nötigste und könnte bestimmt noch beliebig erweitert werden - hier ist mir aber vor allem wichtig, dass du dich sofort bei mir meldest, falls du einen Fehler im System findest oder vermutest. Der Suchsimulator-Code ist auch auf Github[3] einsehbar.

1.3 Vertiefende Informationen

Mehr Informationen als die Ausgangslage und die Aufgabenstellung sind für diese Minichallenge nicht nötig, hier findest du aber noch weiteres Informationsmaterial, falls du mehr über die oben erwähnten Fallgeschichten erfahren möchtest (Text enthält Hyperlinks):

- Vertiefende Informationen von Jarred S. Murray im Rahmen seiner Statistikvorlesung zur Fallgeschichte der USS Scorpion und dem damit verbundenen Bayes-Suchmodell
- Wissenschaftlicher Report von Metron zur Suche nach dem Flug AF 447
- Slides von Metron zur Suche nach dem Flug AF 447 mit Einführung zur Suche nach der USS Scorpion
- Buch über die Suche nach MH370
- Buch: Sharon Bertsch McGrayne - The Theory That Would Not Die - kann gerne bei mir ausgeliehen werden.

Eine ähnliche Methode wird zum Beispiel für die effiziente Optimierung von Hyperparametern benutzt.

[3] `https://github.com/chuwyler/bayesian_search_websimulator`

2 Aufgabenstellung

Aufgabe 1: Bayes- oder Frequentist-Statistik?

Wir möchten die Suche nach der USS Scorpion mit Wahrscheinlichkeiten formalisieren. Dazu müssen wir zuerst einmal klären, was wir unter dem Begriff 'Wahrscheinlichkeit' verstehen:

Wenn wir sagen, die USS Scorpion liege mit einer Wahrscheinlichkeit von $p = 0.05$ in Sektor i, benutzen wir das Wort 'Wahrscheinlichkeit' im frequentistischen oder Bayes'schen Sinne? Argumentiere in Worten.

Aufgabe 2: Herleitung eines einfachen Modells zur Bayes'schen Suche

Nun möchten wir ein (sehr einfaches) mathematisches Framework zur Bayes'schen Suche aufbauen. Dazu führen wir zuerst einmal entsprechende Zufallsvariablen ein:

Sei Y_i eine Zufallsvariable, die 1 sei, wenn sich das gesuchte Objekt irgendwo im Sektor i befinde und 0 sonst. Weiter sei X_i eine Zufallsvariable die das Resultat einer Suche im Sektor i angebe: sie ist 0, wenn das Objekt bei einer Suche im Sektor i nicht gefunden wird und 1, wenn es im Sektor i gefunden wird.

Also:

- $Y_i = 0$: Gesuchtes Objekt ist nicht in Sektor i.
- $Y_i = 1$: Gesuchtes Objekt ist irgendwo in Sektor i.
- $X_i = 0$: Gesuchtes Objekt wurde nicht in Sektor i gefunden.
- $X_i = 1$: Gesuchtes Objekt wurde in Sektor i gefunden.

Es ist leicht einzusehen, dass $P(X_i = 1 | Y_i = 0) = 0$ und $P(X_i = 0 | Y_i = 0) = 1$.

Ausserdem werde die Annahme getroffen, dass, wenn das gesuchte Objekt tatsächlich in Sektor i liegt und dort gesucht wird, es mit einer Wahrscheinlichkeit

$$P(X_i = 1 | Y_i = 1) = p \tag{1}$$

gefunden werde. Da die Such-Sektoren eine gewisse Grösse haben, ist ein Sucherfolg nicht garantiert, auch wenn das gesuchte Objekt tatsächlich in diesem Sektor ist. In Realität variert p natürlich je nach Sektor, wir nehmen hier aber zur Vereinfachung an, dass p für alle Sektoren gleich ist.

Aufgabe Initial wird jedem Sektor eine Prior-Wahrscheinlichkeit π_i zugewiesen, die das vorhandene Vorwissen möglichst gut zusammenfasst, und dann die Suche gestartet. Jedesmal wenn in einem Sektor gesucht wird, resultiert das in einem Update der aktuellen Wahrscheinlichkeitsverteilung der Sektoren. Im Prior sind nur die Wahrscheinlichkeiten $P(Y_j)$ bekannt, im Posterior die bedingten Wahrscheinlichkeiten $P(Y_j | X_i)$ (in Sektor i wurde gesucht - wie ändert das die Wahrscheinlichkeit dafür, in Sektor j das Objekt zu finden?).

Zeige, dass

$$P(Y_i = 1 | X_i = 0) = \pi_i \frac{1-p}{1-p\pi_i} \quad \text{und} \quad P(Y_j = 1 | X_i = 0) = \pi_j \frac{1}{1-p\pi_i} \quad \text{mit } j \neq i \tag{2}$$

und gib in Worten an, welches Ereignisse diese Wahrscheinlichkeiten bezeichnen und was die Folge einer ergebnislosen Suche für die Posterior-Verteilung ist. Benutze dazu den Bayes'schen Satz. Argumentiere, dass mit Kenntnis dieser zwei Wahrscheinlichkeiten die Suche nun zielgerichtet mit kontinuierlichen Updates nach jeder Suche durchgeführt werden kann.

Aufgabe 3: Suche mit Bayes

Nun möchtest du die in Aufgabe 2 gefundenen Formeln in ein Bayes'sches Update-Framework in Python oder R implementieren. Dieses soll dir erlauben, Objekte im Suchsimulator schneller als mit einem Brute-Force-Ansatz zu finden. Da der Simulator nur die initiale Prior-Verteilung ausgibt, musst du die aktuelle Posterior-Verteilung jeweils selbst bestimmen. Konkret möchtest du also ein Tool, dass dir mit Kenntnis der initialen Prior-Verteilung einen optimalen Suchplan in Form einer Liste von der Reihe nach zu durchsuchenden Sektoren erstellt und ausgibt.

Aufgabe Entwickle das oben beschriebene Framework in Python oder in R und prüfe, ob du damit die gesuchten Objekte im Simulator in vernünftiger Zeit finden kannst. Der Simulator benutzt $p = 0.6$ und 25 verschiedene Sektoren.

Mögliche freiwillige Zusatzaufgaben

Falls du Lust hast, ein wenig tiefer in die Materie einzudringen, könnte die Beantwortung folgender Fragestellungen interessant sein:

a) Was ist die Rolle von p? Welche Probleme treten auf, wenn p im Simulator zu gross oder zu klein gewählt wird?

b) Wie viele Suchläufe sind im Median für verschiedene zufällig generierte Suchprobleme nötig? Wieviele mit einer Bruto-Force-Methode oder einer zufälligen Suche? Erstelle dazu deinen eigenen Simulator.

c) Visualisiere jeweils die aktuelle Posterior-Verteilung (z.B. indem du die Funktion `show` aus dem Simulator-Code vom oben erwähnten Github-Repository benutzt). Visualisiere einzelne Suchläufe als Gif-Animation.