TRABAJO FIN DE GRADO

INGENIERÍA INFORMÁTICA Y MATEMÁTICAS

Rankings bayesianos en sistemas de recomendación

Autor
Luis Castro Martín

Director
Fernando Berzal Galiano

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación y Facultad de Ciencias

—

Granada, junio de 2018

Rankings bayesianos en sistemas de recomendación

Autor
Luis Castro Martín

Director
Fernando Berzal Galiano

Rankings bayesianos en sistemas de recomendación

Luis Castro Martín (alumno)

Palabras clave: sistema de recomendación, estadística bayesiana, rankings, recomendador, valoraciones.

Resumen

Los sistemas de recomendación sirven para predecir los gustos de un usuario en base a sus valoraciones anteriores. En este trabajo se estudiarán las propiedades da la estadística bayesiana y su posible aplicación en dichos sitemas.

Se implementarán además tres aplicaciones web. Una servirá para visualizar las conclusiones extraídas mediante métodos bayesianos. Otra será una aplicación con usuarios abierta al público para que cualquiera pueda participar. La última nos permitirá hacer pruebas aplicando el algoritmo final propuesto sobre un conjunto masivo de datos.

Bayesian rankings in recommendation systems

Luis Castro Martín (student)

Keywords: recommendation system, bayesian statistics, rankings, recommender, ratings

Abstract

Recommendation systems aim to predict which products an user may like. They are extensively used nowadays, specially by companies like Netflix or Amazon (for obvious reasons). But in some way every company offer some services, so their costumer may be more or less interested in them. Finding the ones he will find more interesting is always a desirable functionality. Not only for the company itself to make more profit but for us the users as well.

They are based on previous ratings by the user itself, even though sometimes you might have to "deduce" said rating. For example, an user buying a product shows that he probably likes it. One way or the other, you try to give him recommendations for other products using that information.

Lots of path have been studied in order to achieve on point recommenders. In many ways it is not so different from usual data science problems. With previous data you build a model capable of producing estimated ratings. That is why almost every used algorithm is pretty much an application of a generic one. There are some specific approaches using graphs but they are not the most popular or effective solutions.

There is another issue that motivated the ideas presented in this document. Every algorithm has a lot of parameters. Consequently, they have to be adjusted and optimized for every single use case. That is not a simple task and heavily questions the objectivity of the whole process.

By using bayesian statistics, each "parameter" or formula is deeply justified by a mathematical model. The bayesian tools have been proven and validated through the past fifty years. They are not just something that seems to work, they are supported by thourough theorems and strong coherent theories. All this means the results will depend on the data and the model, of course, but the model will not depend (as it should not) a priori on

the data. Therefore it will be an universally valid algorithm with consistent results among every use case.

We will be talking extensively about the foundation and properties of that kind of statistics. One quality we may want to highlight is its ease and accuracy, specially when taking uncertainty into account. Well, at least for the binomial which is the distribution we will be using.

Uncertainty will take a very important role in the formulas and algorithms presented. Usually scorers try to be as close as possible from the real rating. So the smaller the mean of the errors is, the better an algorithm is. While this methodology is true for most of the problems in data science, thinking that way for recommenders may be quite misleading.

Take this for example. Imagine I am recommending you movies. I predict that you are going to enjoy movies 1, 2, 3, 4 and 5 but you are not going to enjoy 6, 7 and 8. Another recommender predicts that you are going to like 2 and 3 but you are going to dislike the rest. Now let's say that in reality you only enjoy 2, 3, 4 and 5. Ok, technically speaking I was right more times than the other recommender. But because of me, you have lost your time and possibly your money watching 1. That is pretty bad. If you had listened to the other suggestions, not only you would have watched two great movies straight away; also that would have given it extra information to give you another good movie title. It would have given you then another safe choice, up until there is enough data to be sure enough of 4 and 5 too.

With that in mind, bayesian statistics and its accurate uncertainty analisis serves us perfectly for recommendarion systems. Therefore it is the only tool we are going to need in the definition of our algorithm. This will differenciate it a lot from the other approaches. It is not a complex algorithm using simple models and ideas, it is a simple algorithm using complex models and ideas. But, then again, they come from solid mathematical theories so they are extremely reliable.

Nonetheless, defining it is not enough. I have implemented some tools in the form of web applications in order to test all the ideas exposed in this document. For starters, you can check the bayesian results for the binomial distribution we are gonna be using in the link `https://luiscastro193.github.io/inferencia-binomial/`. It requires some computations but, for efficiency purposes, everything is programmed in the client so it scales perfectly without problem.

The main application is a portal open to the public where anyone can post issues and answers: `https://rankings-bayesianos.herokuapp.com/`. It is just an excuse to have some user votes (since everyone can vote on everything too). That way, it can build rankings using the techniques developed. Please note that those rankings are completely objective, which means they are the same for every user. Producing custom rankings for such a dynamic enviroment would be too expensive. But do not worry because the third application will show the recommendation system potential even better.

Rankings bayesianos (bayesian rankings) has been online for 7 months and anonymous people have been using it at their will. So every reader is more than welcomed to participate with issues, answers and/or votes. This document will provide some examples of interesting use cases but it is always better checking it yourself. Again, the app is real so the changes are applied dinamically. Also, all the data submitted by the users is somewhere there so I take no responsability for its content, only for its design and implementation. Anyway, that is what the voting system is supposed to be for.

All in all, it shows how users can have all the power in a democratic but fair and reliable system. You will find three kinds of rankings: promising, best and consolidated. They stand for the level of uncertainty we talked about before. Promising score means that is how good it could be. It is the default score so every post has a fair chance of reaching more reliable scores. Otherwise, the first answers would hog all the visibility. You can always go later to the best results or even the safe (consolidated) ones. This last option is specially useful when you use the app as an open voting system for some kind of decision. People can voice their opinions and the one proven (not only expected) best is chosen.

Last but not least, the third application lets anyone test the personalized recommender system on the movieLens database: `https://grouplens.org/datasets/movielens/`. This one is not published online yet because for now it is too specific. Although after seeing the spectacular results I will build a generic API for it. The full database size in this case is more than half a gigabyte though so for the academic purposes of this project the source submitted is better suited.

Personally, I would create a custom profile rating some movies, add it to the original database and run the program with it. For this app I will also show an specific example but, then again, it is much better (and funnier) if you play a little with the app yourself.

It only needs one parameter, which is (at least in my opinion) a great improvement from current alternatives. This parameter (named threshold) allows you to adjust the level of personalization. Any value between 0 and 1 is perfectly valid. My favorite value for it is 0.75 (meaning another user has to prove, at least, a 75 % affinity with you in order for the program to take his opinion into account).

I hope all the considerations and work detailed along this report is enough to convice you about the benefits of bayesian analisis for problems related with probabilities, like the recommenders case. I found the results really interesting and more than satisfactory enough. More generally, I hope it shows the benefits of considering purely mathematical models for advanced data science problems.

Yo, **Luis Castro Martín**, alumno de la titulación Grado en Ingeniería Informática y Matemáticas de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación y Facultad de Ciencias de la Universidad de Granada**, con DNI 77145867A, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Luis Castro Martín

Granada a 12 de junio de 2018.

D. **Fernando Berzal Galiano**, Profesor del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Rankings bayesianos en sistemas de recomendación***, ha sido realizado bajo su supervisión por **Luis Castro Martín**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de junio de 2018.

El director:

Fernando Berzal Galiano

Agradecimientos

Me gustaría agradecer a todos los que han hecho posible alcanzar los objetivos de este trabajo. A los profesores por proporcionarme conocimientos necesarios, para esto y para tantas otras cosas. A mi tutor, Fernando Berzal, por ayudarme durante todo el proceso en todas las iniciativas que proponía. Y a mi familia y amigos por apoyarme siempre. Gracias.

Índice general

Capítulo 1

Introducción

1.1. Contexto

El objetivo final de este trabajo es desarrollar un buen sistema de recomendación. Este tipo de sistemas pretenden predecir los gustos de un usuario en base a los intereses que haya mostrado en el pasado.

Cualquier servicio podría beneficiarse de una funcionalidad así, ya que todos quieren ofrecerle a los usuarios aquello que más les interese. Han invertido en ello con especial interés grandes empresas como Amazon o Netflix [1].

El problema es que las estrategias utilizadas en la actualidad son puramente heurísticas [2]. Aunque algunas se basan en ideas de la estadística bayesiana, en la práctica los modelos no se parecen en absoluto. Además, dependen de parámetros arbitrarios que se deben ajustar y optimizar para cada caso concreto.

Otro problema es que dichas optimizaciones no tienen en cuenta la incertidumbre. Es decir, pretenden (porque en el fondo siguen siendo algoritmos de ciencia de datos genéricos) ajustarse lo máximo posible al valor real. Sin embargo, en un sistema de recomendación los falsos positivos son mucho más graves que los falsos negativos.

Tenemos por tanto algoritmos extremadamente complejos y computacionalmente costosos que dan resultados bastante mejorables.

1.2. Solución

Para paliar estos problemas, la solución propuesta intentará ajustarse lo máximo posible a modelos reales de la estadística bayesiana. Previamente repasaremos las propiedades y beneficios de este tipo de estadística para justificar su adecuación al contexto de los sistemas de recomendación. Por tanto, la validez de los algoritmos se basará en conceptos y teorías puramente matemáticos.

Una vez propuesta, se presentarán 3 aplicaciones web implementadas para ilustrar el funcionamiento de las ideas desarrolladas a lo largo del trabajo. Incluyendo un portal con usuarios abierto al público: `https://rankings-bayesianos.herokuapp.com/`.

1.3. Herramientas

Para la parte matemática, evidentemente toda la teoría viene del área de la estadística bayesiana. Por ello se hará un repaso desde su origen hasta sus resultados útiles para nuestro problema, pasando por propiedades y justificaciones de interés.

La parte informática consiste principalmente en aplicar dichos resultas para definir algoritmos coherentes que den buenos resultados. Las aplicaciones de ejemplo están todas orientadas a un ámbito web e implementadas en lenguaje JavaScript (incluyendo Node.js para el lado del servidor).

1.4. Fuentes

Las principales referencias de estadística bayesiana han sido [3], [4] y [5]. El primero destaca por su extenso y profundo análisis de los conceptos desde un punto de vista teórico. El segundo destaca por sus detalladas aplicaciones técnicas y por la claridad de sus explicaciones. También pueden repasarse conceptos básicos de estadística en [6].

En cuanto a las implementaciones, ha sido de enorme utilidad la principal referencia universal en lo que a JavaScript y desarrollo web se refiere: MDN

Web Docs. Así como la documentación oficial de Expressjs y PostgreSQL para el lado del servidor.

El resto de referencias corresponden a justificaciones de afirmaciones concretas que se mencionan a lo largo del trabajo.

Capítulo 2

Objetivos

2.1. Justificar el uso de la estadística bayesiana

El primer objetivo de este trabajo era exponer razonadamente por qué la estadística bayesiana es útil tanto para la estadística en general como para los sistemas de recomendación en particular. En la asignatura de Inferencia Estadística ya estudiamos los conceptos básicos pero había que profundizar en sus razonamientos y metodologías.

A ello se dedica prácticamente todo el aspecto matemático. Todas las afirmaciones están debidamente citadas y suponen una base más que adecuada para los objetivos posteriores.

2.2. Resolver los problemas de los rankings actuales

Un objetivo interesante era superar las dificultades que ofrecen los sistemas de rankings de portales populares, como por ejemplo Stack Overflow. El problema se describe detalladamente pero, en resumidas cuentas, hay que elegir entre limitar los mejores puestos a las primeras propuestas o sobrevalorar soluciones con pocos votos. En la práctica, nunca se trata de rankings

fiables y la mejor respuesta puede estar en cualquier posición.

Considero que la alternativa mostrada en `https://rankings-bayesianos.herokuapp.com/` soluciona satisfactoriamente esos problemas. Llegando a ofrecer hasta 3 tipos de rankings para adaptarse a cada tipo de situación.

2.3. Definir un sistema de recomendación fiable y consistente

Finalmente, quería conseguir un sistema de recomendación basado en conceptos puramente matemáticos. Con resultados que, al menos, fueran decentes como punto de partida.

Lo cierto es que los resultados han superado cualquier tipo de expectativa en este aspecto. Se aporta una implementación concreta y cómoda de usar para una base de datos particular, como fue previsto en un principio. Pero ha sido tan satisfactorio que está en proceso la construcción de una API genérica que permita consultas sobre cualquier tipo de base de datos de valoraciones.

Capítulo 3

Aspecto matemático

3.1. El enfoque bayesiano

La estadística descriptiva se dedica a crear modelos matemáticos en los que enmarcar datos experimentales, pertenecientes a distintos contextos. Contextos provenientes de problemas reales que pueden ser de cualquier tipo: el resultado de tirar una moneda, la efectividad de un medicamento en un paciente, los resultados de unas elecciones, los tiempos de espera en una cola...

A cada tipo se le asigna una distribución de probabilidad como representante de su modelo correspondiente: la distribución de Bernoulli, la distribución Binomial, la distribución Normal, la distribución de Cauchy... Pero todas ellas se rigen por unos axiomas comunes que sientan las bases de la estadística. Y sobre ellos se ha desarrollado una amplia teoría, profundamente asentada y demostrada a día de hoy. Es la que se nos explica en la primera asignatura de la rama y nadie la pone en duda.

Lo interesante viene cuando pretendemos predecir parámetros (y en consecuencia, datos) que no tenemos. A ello se dedica la estadística inferencial. La estadística inferencial clásica define para este propósito nuevos conceptos: verosimilitud, confianza, p-valores... Sin embargo, el rigor y la validez de estos conceptos con respecto a aquello que se pretende estudiar no están tan exentos de ponerse en entredicho [3, apartados 1.6.1 y 1.6.2].

Ante esa situación, los bayesianos proponen lo siguiente: ¿sería posible

expresar los parámetros a inferir en función, a su vez, de alguna distribución de probabilidad (que tan bien entendemos y manejamos)? Con este objetivo en mente se propone el principio de dualidad [4, apartado 1.2]. Es decir: de igual manera que los descriptivos trabajan con datos en función de parámetros, nosotros podemos trabajar con parámetros en función de los datos; obteniendo resultados equivalentes. Y si en un caso se obtenían distribuciones de probabilidad, en este obtendremos, efectivamente, distribuciones de probabilidad.

Sobre probabilidades invertidas ya había trabajado en su momento Bayes, obteniendo su famoso teorema. Como veremos a continuación, de él derivan las bases de la estadística bayesiana y, evidentemente, su nombre.

3.2. Bases de la estadística bayesiana

3.2.1. El teorema de Bayes (probabilidad condicionada)

Si tenemos dos sucesos, A y B, la probabilidad de que ocurra A condicionado a que ocurra B se define como:

Definición 3.1 *Probabilidad condicionada*

$$Pr(A \mid B) = \frac{Pr(AB)}{P(B)}$$

El teorema de Bayes nos proporciona una forma de invertir la condicionalidad:

Teorema 3.1 *Teorema de Bayes*

$$Pr(A \mid B) = \frac{Pr(B \mid A) \cdot Pr(A)}{P(B)}$$

Si bien lo común será particionar el espacio para expresar $Pr(B)$ como la suma de las probabilidades:

Teorema 3.2 *Teorema de Bayes para una partición*

Sea $\{A_i\}$ *una partición del espacio de sucesos con* $Pr(A_i) > 0 \quad \forall i = 1, 2, \ldots$.

$$Pr(A_i \mid B) = \frac{Pr(B \mid A_i) \cdot Pr(A_i)}{\sum_{j=1}^{\infty} Pr(B \mid A_j) \cdot Pr(A_j)} \quad \forall i = 1, 2, \ldots$$

[5, apartado 2.4]

3.2.2. Generalizando el teorema de Bayes

Partiendo de esa idea, supongamos ahora que A representa que el parámetro a inferir tome cierto valor y B representa los datos obtenidos hasta el momento. Veamos que nos encontramos entonces en la situación que buscábamos.

Originalmente tenemos la función de probabilidad asociada la distribución de probabilidad de nuestro problema: $f(x \mid \theta)$. En ella la probabilidad de un posible dato depende del valor del parámetro. No tenemos dicho valor, es lo que intentamos inferir. Pero sí que conocemos esa función "inversa" a la que queremos. También tenemos unos datos concretos, obtenidos experimentalmente, que queremos tomar como fundamento de los resultados que obtengamos.

Para poder aplicar el teorema de Bayes e invertir la dependencia, nos hace falta $Pr(\theta)$. Es decir, una función de probabilidad $\Pi(\theta)$ que le asignemos a priori al parámetro. Antes incluso de tener en cuenta ningún dato. Sobre la elección y validez de esta distribución a priori que damos discutiremos en posteriores capítulos. Por ahora, supongamos que le hemos asignado una (por alguna razón justificada).

Tenemos por tanto todos los elementos para definir la función de probabilidad a posteriori del parámetro θ condicionado a unos datos experimentales $x_1, \ldots, x_n$:

Definición 3.2 *Probabilidad a posteriori*

$$\Pi(\theta \mid x_1, ..., x_n) = \frac{(\prod_{i=1}^{n} f(x_i \mid \theta)) \cdot \Pi(\theta)}{\int_\theta (\prod_{i=1}^{n} f(x_i \mid \theta)) \cdot \Pi(\theta) d\theta}$$

[5, apartado 3.8] y [3, apartado 4.2.1].

3.3. Argumentos a favor

Esta sección está basada en [3, apartado 4.1].

3.3.1. Uso de la información a priori

En muchos casos los experimentos empíricos no parten de cero. Por ejemplo, el caso de una empresa que se encargue a criar peces para después venderlos. Será común hacer estudios sobre las distintas poblaciones criadas independientemente. Sin embargo, es de esperar que poblaciones similares criadas en circunstancias similares tengan características similares.

Mientras que la estadística clásica recurre a herramientas primitivas para aprovechar esta información previa, en bayesiana cuentan con toda una distribución de probabilidad a priori para modelizarla. De manera que además las consecuencias se obtienen de manera "automática", porque el papel que dicha distribución tiene en la distribución a posteriori está definido de antemano.

Cuando estudiemos la forma en que muchos algoritmos actuales de ranking bayesiano trabajan, se hará patente la importancia de este punto. Es natural considerar las valoraciones de productos pasados como indicador de lo que podemos esperar en productos futuros.

3.3.2. Coherencia

La estadística inferencial pretende sacar conclusiones sobre parámetros inciertos. El concepto de incertidumbre va íntimamente ligado al concepto de probabilidad. Si la incertidumbre se mide mediante distribuciones de probabilidad, ¿no es cohente que en este caso se mida de la misma manera?

El ejemplo más claro de esto son los intervalos de confianza. Cualquiera podría esperar que un intervalo con una confianza del 95 % tenga, efectivamente, un 95 % de probabilidad de contener al valor exacto. Sin embargo, entender eso es un error típico cuando se trabaja con estadística clásica.

Es decir, se mencionan conceptos probabilísticos (porque intuitivamente lo son) que no se corresponden con probabilidades reales. Lo que se pretende

al asignarle una función de probabilidad al parámetro es que, si tan natural es hablar en esos términos, los métodos sean coherentes con ellos.

3.3.3. El punto de vista condicional

La distribución a posteriori hace patente que las conclusiones dependen directamente de los datos obtenidos, por definición.

Puede parecer una trivialidad pero la perspectiva frecuentista, por contra, se basa en que una cantidad infinita de datos darán, en media, los resultados esperados. Si los datos no son infinitos, mala suerte, habrá que trabajar con los que haya.

Si bien daremos resultados que garanticen también esa propiedad, será una conclusión. No es algo de lo que se parta para hacer simplificaciones conforme nos falten datos.

3.3.4. Racionalidad

El enfoque clásico y el enfoque bayesiano parten de distintos axiomas que se asumen ciertos. El sistema axiomático bayesiano es considerado más evidentemente "cierto" que el sistema axiomático clásico. [3, apartado IV del 4.1].

3.3.5. Equivalencia

Por una parte, se han demostrado resultados equivalentes a las conclusiones clásicas fundamentales. Por ejemplo, como ya se ha mencionado y se detallará más adelante, tomar una cantidad infinita de datos sigue convergiendo al valor exacto.

Por otra, resultados clásicos llegan a conclusiones bayesianas. [3, apartado V del 4.1]. Los estadísticos no bayesianos suelen calificarlo como una mera casualidad matemática. En cualquier caso, no deja de ser un indicador interesante.

El ejemplo más sencillo es el estimador máximo verosímil, el estimador

clásico por antonomasia. Casualmente, coincide por definición con la moda de la distribución a posteriori tomando la uniforme como distribución a priori. Sin embargo, tanto la moda como la distribución uniforme son considerados, como veremos, métodos demasiado primitivos de estimación bayesiana.

3.3.6. Ventajas operativas

En un principio, podría parecer que trabajar con estadística bayesiana es mucho más complicado. Determinar la distribución a priori, calcular la distribución a posteriori, trabajar con sus momentos...

De hecho, en muchos casos es más cómodo e incluso recomendado por los propios bayesianos trabajar con análisis clásicos.

Sin embargo, a la hora de obtener resultados significativos en problemas complejos la cosa cambia. El mejor ejemplo es el uso de la estadística en minería de datos. El modelo puramente estadístico que ha dado mejores resultados (para un coste de cálculo razonable) han sido las redes bayesianas.

O, sin ir más lejos, la realización de rankings en base a valoraciones. Cuando veamos ejemplos se hará patente por qué ese esfuerzo extra proporciona resultados mucho mejores. Con propiedades que costaría bastante más alcanzar mediante métodos clásicos.

3.3.7. Objetividad

Hay dos razones por las que he escogido este enfoque para diseñar un algoritmo de valoración. La primera son los extraordinarios resultados empíricos, que se discutirán cuando lleguemos a la aplicación concreta que utiliza. La segunda es la objetividad.

La elección de una distribución a priori, que en teoría podría ser cualquiera, da una fuerte impresión de subjetividad. Es cierto que se puede abusar de ello, pero eso se estudiará en la sección de críticas.

Por contra, si se escoge una justificadamente, la objetividad es mayor incluso que con métodos clásicos. El mejor ejemplo es el principio de invarianza, que aparecerá en el siguiente capítulo. Será tan objetivo que cualquier

reparametrización del problema dará lugar a exactamente las mismas conclusiones.

Esa propiedad es otra de aquellas que deberían ser básicas pero, por desgracia, en la práctica no son tan comunes.

3.4. Elección de la distribución a priori

Este tema es sin duda el más complejo y polémico del enfoque bayesiano. Es por ello que vamos a sentar una serie de bases que nos llevarán directamente al caso concreto que vamos a utilizar. Evidentemente podrían considerarse muchas otras pero no son interesantes para el ámbito de este trabajo.

3.4.1. Objetividad

En la sección anterior se han presentado dos argumentos opuestos. El marco teórico con el que trabajamos nos permite tanto introducir subjetividad de manera totalmente arbitraria para adaptar nuestro problema; como considerar una objetividad absolutamente indiferente a cualquier problema teóricamente equivalente.

En la parte informática comentaremos los trucos que usan algunas empresas para aprovechar lo primero. Pero si queremos diseñar un algoritmo de ranking universalmente válido y creíble, tiene que ser objetivo.

Por ello nos limitaremos a considerar distribuciones a priori no informativas. Es decir, que aporten la mínima información previa a los datos posible. El ejemplo más básico podría ser, aunque no terminará siendo el que usemos, una distribución uniforme.

3.4.2. La distribución uniforme

¿Puede haber algo más objetivo e imparcial que darle a todos los valores, inicialmente, la misma probabilidad? Sorprendentemente, sí.

La propuesta de $\Pi(\theta) = cte$ fue inicialmente considerada por Laplace. Se basó en el "principio de razón insuficiente": si no sé nada sobre el parámetro, ¿qué argumento podría tener para darle más probabilidad a un valor que a otro?

Visto así, no parece lógico siquiera considerar otra propuesta "objetiva". Sin embargo, presenta un grave problema: reparametrizaciones del mismo problema pueden dar lugar a conclusiones distintas. Sería incluso posible reparametrizar malintencionadamente para obtener los resultados deseados.

[4, apartado 3.5.1]

A pesar de todo, es utilizada a veces en la actualidad por lo mucho que simplifica los cálculos. De ahí que se planteara el principio de invarianza [5, apartado 10.2].

3.4.3. El principio de invarianza

El principio de invarianza afirma que dos formas equivalentes de expresar un mismo problema deben llevar a resultados también equivalentes. [3, apartado III de 1.5.2].

El análisis de este principio fue el principal impulsor para que Jeffreys propusiera en 1961 una distribución a priori (no informativa) alternativa. Si bien no fue la única, pues le siguieron las de Novick y Hall (1965), Zellner (1971, 1977), Box y Tiao (1973), Akaike (1978), Bernardo (1979b), y Geisser (1984a). [3, apartado 3.3.3].

¿Por qué tantas para algo teóricamente "objetivo"? Sobre ello hablaremos en el apartado de críticas pero, para ilustrar las ideas que allí se expondrán, vamos a comentar previamente la de Jeffreys y la de Bernardo. Aparte de por ser las dos más importantes, porque son las que usaremos en los algoritmos de este trabajo.

3.4.4. Distribución a priori de Jeffreys

Formalmente, está basada en la función "información de Fisher" de un parámetro, dada por la fórmula:

Definición 3.3 *Información de Fisher*

$$I(\theta) = E_\theta \left[\left(\frac{\partial log(f(X \mid \theta))}{\partial \theta} \right)^2 \right]$$

Entonces, la distribución a priori de Jeffreys viene dada por:

Definición 3.4 *Distribución a priori de Jeffreys*

$$\Pi(\theta) = \sqrt{I(\theta)}$$

Esta propuesta tiene dos ventajas [4, apartado 3.5.3], una práctica y otra algo más filosófica.

La primera es que, como ya hemos comentado que fue su motivación, cumple el principio de invarianza para reparametrizaciones del problema inicial.

La segunda viene de que la información de Fischer está ampliamente aceptada como el indicador de cuánta información aporta un parámetro al modelo. Por tanto, esta definición minimiza la influencia que la distribución a priori tiene en la distribución a posteriori, maximizando así el impacto de los datos.

Parece deseable que en un enfoque objetivo sean los datos los que aporten lo máximo posible a las conclusiones. Por esta razón, la distribución a priori de Jeffreys es la más utilizada a nivel teórico [3, apartado 3.3.3] entre las no informativas. Como es fácil de calcular en el caso de este trabajo, será también la que utilizará en la práctica.

3.4.5. Distribución a priori de Bernardo

Si bien esta propuesta se sale del ámbito de las apliciones que vamos a considerar en los algoritmo, la menciono con el único propósito de ilustrar por qué otras alternativas a la de Jeffreys fueron propuestas.

La información de Fischer está definida como un número real para un parámetro. ¿Qué ocurre cuando hay más de un parámetro desconocido sobre el que pretendemos hacer inferencia? Entonces la información de Fischer sobre un vector es una matriz, no un número. Jeffreys intentó solucionarlo de una manera sencilla:

Definición 3.5 *Distribución a priori de Jeffreys para el caso multidimensional*

$$\Pi(\theta) = \sqrt{|I(\theta)|}$$

Es decir, tomamos el determinante como representante real de la matriz. Sin embargo, esta simplificación no siempre es satisfactoria: [4, ejemplo 3.5.9].

Es por ello que Bernardo propuso otra generalización, mucho más compleja, pero que también coincide con la de Jeffreys en el caso unidimensional [4, apartado 3.5.4].

Esta filosofía de corregir posibles críticas para casos problemáticos se repite a menudo. Eso explica que haya tantas propuestas para un mismo objetivo. Dependiendo del problema, estará "claro" cuál nos interesa más.

3.5. Críticas

Esta sección está basada en [3, apartado 3.7].

Lo cierto es que las críticas más razonables hacia la estadística bayesiana vienen debido a la distribución a priori. Si hubiera una única posibilidad, pocos estadísticos pondrían objeciones. Por desgracia no es el caso, así que hay que justificar las críticas más comunes.

3.5.1. Objetividad

Como de costumbre empezamos por una de las piedras angulares del trabajo. Sin duda la posible arbitrariedad de la distribución a priori escogida podría despertar suspicacias.

La respuesta que dan los bayesianos a esta crítica es cuanto menos curiosa. No niegan que ciertos aspectos puedan ser considerados subjetivos. En su lugar, argumentan que esos mismos aspectos cuestionarían, no en menor medida, la objetividad de la estadística en general.

Por poner un ejemplo, los análisis de decisión tienen que escoger (sea cual sea el enfoque) una función de "pérdida" que es igual de subjetiva. Pero es algo que, simplemente, tiene que hacerse.

El estudio de distribuciones a priori no informativas asume esa necesidad inapelable de decisiones subjetivas para, a partir de ahí, intentar ser lo más objetivo posible.

En cualquier caso, la relevancia que tiene una alternativa aceptable en los resultados no es mayor que el error asumido por métodos frecuentistas [3, apartado 3.3.4]. Este hecho se hará patente cuando estudiemos las propiedades del caso binomial.

Sencillamente, da mejores resultados objetivos a pesar de su presunta mayor subjetividad [7].

3.5.2. Uso irresponsable de las distribuciones a priori

De acuerdo, supongamos que hemos escogido la distribución a priori adecuada y el estudio es cuanto menos tan correcto, objetivo y preciso como el que más. Pero, ¿qué te impide haber escogido una inadecuada? Nada. De hecho, podrías buscarla malintencionadamente para obtener los resultados que desees.

Ante esto, argumentan que la respuesta al posible uso inadecuado de la ciencia no debe ser de rechazo, sino una mejor educación.

3.5.3. Robustez

Pequeños cambios en los datos deben causar pequeños cambios en las conclusiones.

Esto no está garantizado para cualquier distribución a priori. Será necesario tener en cuenta resultados que nos proporcionen esta propiedad [3, apartado 4.7].

3.5.4. Dependencia con los datos o el modelo

Esta ha sido una de las grandes preocupaciones a superar del trabajo (como se comentará al compararlo con otras técnicas utilizadas en la actualidad). También es la principal razón de que haya hecho tanto hincapié en el tema de la objetividad.

El problema consiste en que la distribución a priori, por definición, no debe depender de los datos obtenidos. En la práctica, en muchos casos, esto no es realista.

Pongamos de ejemplo los mismos rankings bayesianos. Supongamos que estamos intentando hacer valoraciones para las series de Netflix y más del 90 % de las ofertadas tienen calificaciones muy altas. Por la sencilla razón de que sólo se recuerda al usuario que vote al terminar de verla; y no vas a verte entera una serie que no te gusta. Sería muy tentador definir una distribución a priori que se adapte a este hecho.

Si bien eso sería aceptable (puede que incluso recomendable para ciertas aplicaciones), lo que está claro es que los resultados obtenidos no tendrán credibilidad ninguna fuera de su contexto. Aunque para esos datos, en esa situación concreta, "acierten" mucho. Lo hacen porque has preparado las hipótesis después de ver los datos y no antes, como teóricamente sería lo correcto.

Es un dilema similar al del sobreaprendizaje o sobreajuste. Para evitarlo, el modelo del trabajo no partirá de ningún caso concreto. Serán los datos los que lo adapten a cada caso a posteriori.

3.6. Validez

No podemos terminar un repaso general del enfoque bayesiano sin justificar su validez. De nada sirve explicar su origen, justificar sus definiciones, argumentar todas las posibles críticas... si al final nada nos garantiza que los resultados sean, al menos, tan válidos como los que nos proporcionaría un análisis clásico.

3.6.1. Teorema Central del Límite

El Teorema Central del Límite (o Teorema del Límite Central) es el resultado fundamental de la Estadística en el que se basa la validez del enfoque clásico. Afirma lo siguiente:

Teorema 3.3 *Teorema Central del Límite*

Sea $X_1, X_2, \ldots, X_N$ un conjunto de variables aleatorias, independientes e idénticamente distribuidas de una distribución con media μ y varianza finita σ^2. Entonces, para N suficientemente grande, la media muestral

$$X = \frac{1}{N}\sum_{i=1}^{N} x_i$$

tiene aproximadamente una distribución normal con media μ y varianza $\frac{\sigma^2}{N}$ (convergente a 0).

[8]

3.6.2. El caso bayesiano

Pues bien, los bayesianos tienen un resultado similar que asegura que, al menos para una cantidad de datos suficientemente grande (como en el caso frecuentista), la distribución a posteriori se comporta como una normal centrada en el valor exacto del parámetro.

Teorema 3.4 *Sea $X_1, X_2, \ldots, X_N$ un conjunto de variables aleatorias, independientes e idénticamente distribuidas de una distribución con un parámetro θ del que existe estimador máximo verosímil $\hat{\theta}$. Supongamos $\Pi(\theta)$ y $f(x \mid \theta)$ positivas y dos veces diferenciables entorno a $\hat{\theta}$.*

Entonces, para N suficientemente grande y bajo condiciones comúnmente satisfechas, la distribución a posteriori tiene aproximadamente una distribución normal con media $\hat{\theta}$ y varianza convergente a 0.

[3, Stable Estimation del apartado 4.7.8]

3.6.3. Robustez

Pequeños cambios en los datos deben causar pequeños cambios en las conclusiones.

Como ya dijimos en la sección de críticas, esto no está automáticamente garantizado. Sin embargo, para el ámbito de este trabajo nos basta saber que la distribución a priori de Jeffreys lo cumple en el caso unidimensional [3, Noninformative Priors del apartado 4.7.9].

3.7. El caso binomial

Una vez argumentado el uso de la estadística bayesiana, es hora de aplicarla al caso que nos interesa: la distribución binomial.

3.7.1. La distribución binomial

Esta distribución modela el caso en el que un experimento con dos posibles resultados (comúnmente denominados "éxito" y "fracaso") que se repite un número determinado de veces.

Sus parámetros son el número n de repeticiones y la probabilidad p de éxito. Mide la probabilidad de que se produzcan exactamente x éxitos.

$Pr(x; n, p) = \binom{n}{x} p^x (1-p)^{n-x}$, su media es np y su varianza $np(1-p)$ [9].

3.7.2. Inferencia bayesiana

En nuestro problema tendremos datos provenientes de una distribución binomial de la que desconocemos la probabilidad de éxito. Buena parte de la solución consistirá en inferir el valor de ese parámetro.

Generalmente hablando, hacer cálculos de inferencia bayesiana es complejo y requiere de herramientas especializadas como http://mc-stan.org/ que ni siquiera recomiendan usar distribuciones a priori avanzadas [10].

Por suerte, casos estándar como el de la binomial ya están ampliamente estudiados. De hecho, su distribución a priori de Jeffreys coincide con una conocida: la beta de parámetros $\alpha = 1/2$ y $\beta = 1/2$. [4, ejemplo 3.5.4].

Por si eso fuera poco, la distribución a posteriori que deriva de ésta sigue siendo una conocida: la beta de parámetros $\alpha = 1/2 + x$ y $\beta = 1/2 + n - x$. [4, ejemplo 1.4.1].

Esto nos permitirá dar resultados exactos usando los métodos bayesianos más avanzados a nuestra disposición, pues consiste simplemente en trabajar con una distribución clásica. Prácticamente se convierte en un problema de estadística descriptiva.

Por ejemplo, ¿qué estimación podemos dar del parámetro? Está claro, su media. ¿Y qué intervalo de confianza? Pues su media ± su desviación típica. ¿Conocemos la probabilidad de dicho intervalo? Por supuesto, la probabilidad que contiene su gráfica.

¿Y si quisiéramos que el intervalo de confianza tuviera la probabilidad de equivocarse (tanto por alto como por bajo) igual a una dada? Sin problema, también conocemos su función de distribución. ¿Y si queremos que contenga a los valores más probables? De acuerdo, aquellos que queden por encima de un valor de la función de densidad que contenga la confianza deseada.

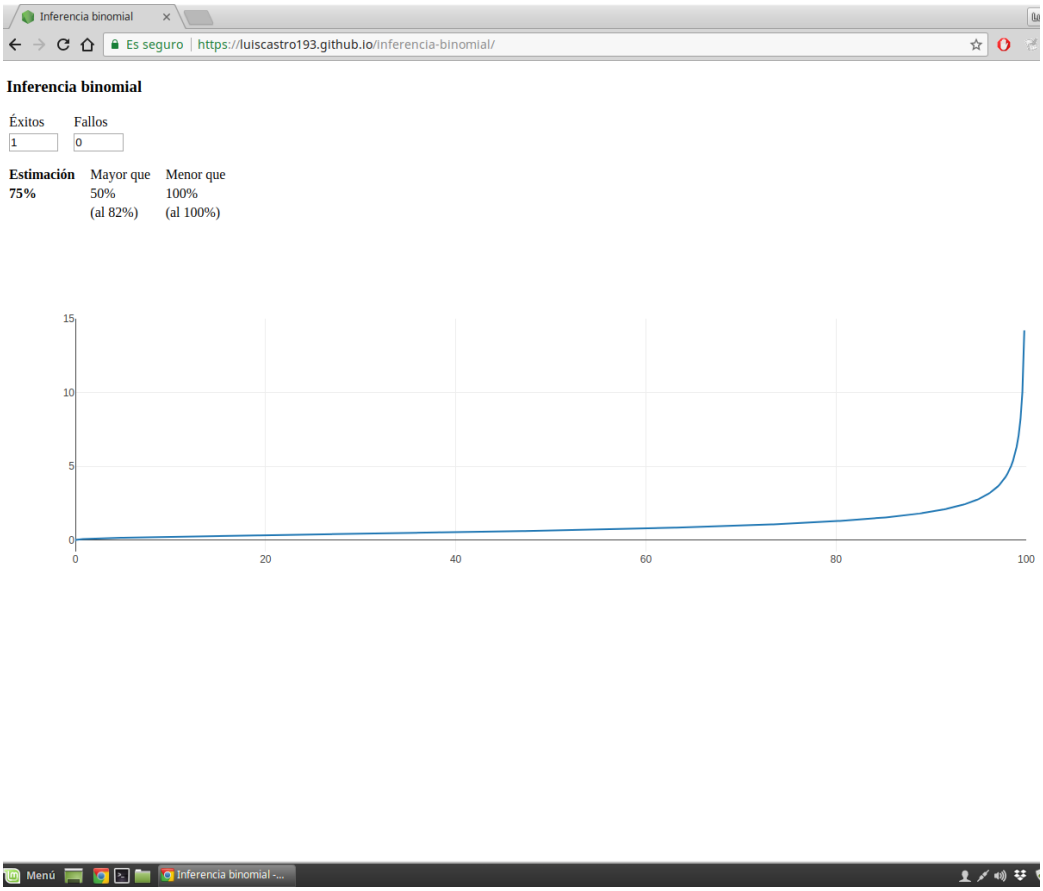
Problemas que con el enfoque clásico eran auténticos quebraderos de cabeza, parecen ahora triviales.

3.7.3. Resultado

Siguiendo ese criterio, he desarrollado una pequeña aplicación que muestra online los resultados para unos datos concretos:

`https://luiscastro193.github.io/inferencia-binomial/`

Un ejemplo interesante es el caso en el que ha habido un único intento exitoso:



El estimador máximo verosímil clásico es conocido por todos: éxitos / intentos. Por tanto, eso nos daría una probabilidad de éxito estimada de un 100 %. Sin embargo, el enfoque bayesiano nos da un 75 %.

Ahora bien, supongamos que hemos llevado acabo un experimento del que no sabemos nada y queremos dar una estimación objetiva. Por ejemplo, la valoración de un producto; el único cliente encuestado está satisfecho con él. ¿Qué estimación parece más razonable: un 75 % de éxito o un 100 % de éxito? Puedes pensar que has encontrado el producto definitivo que le gusta a todo el mundo pero la realidad es que, en media, vas a acertar más esperando un 75 %.

Basta considerar que si esa fuera realmente la tasa de éxito, seguiría siendo lo más probable que un único ensayo fuera positivo. Equivalentemente para cualquier valor del intervalo $(50, 100]$. Incluso siendo relativamente cercano a un 50 % tampoco sería de extrañar que hubiera ocurrido. Irte directamente al extremo parece un poco excesivo.

Es más, según esa lógica un producto con 999 valoraciones a favor y 1 en contra es, presumiblemente, peor que otro con 1 valoración a favor y ninguna en contra. Con el estimador bayesiano, en cambio, ocurre al revés. De nuevo, ¿qué parece más razonable?

Más allá de la intuición, estudios empíricos han demostrado que los resultados de este método son objetivamente precisos. ¿Cómo? Simulando computacionalmente un experimento con probabilidad fija pero oculta. El intervalo al 90 % de confianza acierta en media, efectivamente, un 90 % de las veces. Mejorando así al intervalo clásico. [7].

3.8. Modelando las valoraciones de usuarios

Ahora que manejamos todo lo que vamos a necesitar, podemos usarlo para definir el modelo matemático de nuestro problema.

3.8.1. El modelo básico

El problema básico consiste en que ante una serie de propuestas, cada usuario puede votar "me gusta" o "no me gusta". Sobre eso querremos hacer un ranking.

La correspondencia con la distribución binomial es directa. Cada "me gusta" será un éxito y cada "no me gusta" un fracaso. Así de sencillo.

Con eso tendremos tres valores que se pueden interpretar de la siguiente manera: la valoración objetiva (la estimación, su media), la valoración "optimista" (el extremo superior del intervalo de confianza, su media + su desviación típica) y la valoración "pesimista" (el extremo superior del intervalo de confianza, su media - su desviación típica).

A cada ranking correspondiente lo llamaremos: "mejores", "prometedoras" y "consolidadas"; respectivamente.

3.8.2. Valoraciones continuas

En algunos casos, las valoraciones no son duales. Pueden ser, por ejemplo, una nota del 0 al 10 o un número de estrellas del 1 al 5.

Por suerte, si bien la binomial es discreta, la beta (que es la distribución con la que realmente trabajamos) es continua. Por tanto, la "extensión" continua del problema es tan natural que ni siquiera hay que cambiar nada. Un 8 sobre 10, por ejemplo, supondría un 0.8 de "éxito" y un 0.2 de "fracaso".

Nota: Hay quien podría razonar que un número de estrellas corresponde a una multinomial. Aquí entra la subjetividad inherente de la estadística a la que aludíamos en la sección de críticas. Sin embargo, considerar que la diferencia entre 5 estrellas y 1 estrella es la misma que entre 5 estrellas y 4 estrellas es evidentemente absurdo.

3.8.3. Afinidad entre usuarios

Cualquier sistema de recomendación que se precie no puede dar el mismo ranking a todos los usuarios. Debe ser subjetivo a los gustos de cada usuario. El algoritmo que resuelve esa cuestión se detallará en el aspecto informático pero necesitará de una medida matemática sobre la afinidad entre dos usuarios.

De nuevo, nos encontramos ante una binomial. Los únicos datos que tenemos para medir la afinidad son las valoraciones comunes entre ambos usuarios. Cada vez que coincidan en la misma opinión, eso supondrá un éxito. Y cada vez que difieran, un fracaso.

Por tanto, todo lo considerado en las valoraciones se sigue aplicando de la misma forma para las afinidades.

Capítulo 4

Aspecto informático

4.1. Sistemas de recomendación

En esta sección determinaremos el verdadero objetivo de este trabajo: un sistema de recomendación.

4.1.1. ¿Qué son?

Un sistema de recomendación, definido de manera genérica, es un proceso que selecciona de entre una lista de productos aquellos que puedan ser de mayor interés para un cliente concreto.

En este contexto estaríamos hablando, evidentemente, de un algoritmo. Los datos de entrada serán las valoraciones previas que los clientes hayan dado sobre los productos y el identificador del cliente en cuestión. La salida será un ranking de los productos más recomendados para dicho cliente y sus valoraciones estimadas.

Esto le interesa, por ejemplo, a Netflix. Encontrar las series y películas que más le interese a cada usuario aumenta en gran medida el interés del servicio y, por tanto, la probabilidad de que los usuarios prorroguen su subscripción. Tanto es así que en 2007 organizaron un concurso para dar 1 millón de dólares al primer equipo que superase en al menos un 10 % los resultados de su recomendador [1].

Cabe destacar que no sólo es interesante para servicios masivos en los que cada usuario tenga un ranking personalizado a sus gustos. También páginas como Stack Overflow necesitan un algoritmo que ordenen las soluciones de cada problema de manera lo más inteligente posible.

4.1.2. El problema de los rankings

Para ilustrar el problema principal de intentar realizar un ranking vamos a partir del ejemplo más sencillo: Stack Overflow.

Supongamos que alguien hace una pregunta y distintos usuarios proponen distintas soluciones. La comunidad vota sobre la utilidad de cada propuesta y en función de eso se mostrarán los resultados a futuros navegantes.

Si nos fijamos en lo que nos recomienda la estadística clásica, la mejor estimación para cada propuesta sería "número de votos positivos" / "número de votos". A día de hoy se sabe que es una mala idea. ¿Por qué? Porque entonces una solución que ha tenido 999 votos a favor y 1 voto en contra es, presumiblemente, mejor que una con 1 voto a favor y ninguno en contra. Eso obviamente no es razonable y provoca que nuevas propuestas aparezcan consistentemente en la primera posición sin merecerlo.

En la mayoría de los casos se llega a una solución extremadamente sencilla y elegante, pero que también deja bastante que desear. Consiste en que la valoración sea "votos positivos" - "votos negativos". Si antes estábamos siendo demasiado permisivos con las nuevas propuestas, ahora estamos siendo tremendamente injustos con ellas.

Supongamos un caso más que típico en el caso de Stack Overflow. Hay una pregunta que era difícil de resolver con la tecnología del día en que se hizo. La solución que se propuso era compleja y poco elegante. Unos meses más tarde, aparece una solución directa y sencilla (por una nueva librería, un cambio en el estándar, una nueva versión...), sin duda la mejor respuesta posible.

Ahora bien, tras tanto tiempo de valoraciones la respuesta farragosa tiene ya +54 puntos. Por si fuera poco, todas las otras respuestas medianamente aceptables tienen también una puntuación positiva considerable. Así que la verdadera mejor respuesta queda automáticamente enterrada por ellas y apenas será vista por futuros usuarios.

Ha sido un ejemplo de algunos meses pero esa misma situación puede darse fácilmente en cuestión de horas o incluso minutos, dependiendo del contexto. Esto obliga al usuario experimentado a revisar minuciosamente casi todas las respuestas porque el algoritmo de ranking no es de suficiente utilidad.

Por tanto, las empresas con un mayor interés en este aspecto (como Amazon) se ven obligadas a recurrir a métodos más avanzados. Si bien estos métodos son llamados "bayesianos" por estar basados en conceptos de la estadística bayesiana, en realidad son más bien heurísticos. Básicamente, definen un modelo más o menos sencillo que depende de manera arbitraria de unos parámetros a optimizar [2, capítulo 5]. No obstante, ni la definición de dicho modelo ni la optimización de dichos parámetros siguen un rigor matemático. Simplemente tienen sentido y, hasta cierto punto, funcionan.

El objetivo de este trabajo es definir un algoritmo (tanto para el caso común de Stack Overflow como para el caso personalizado de Netflix) que esté fundamentado en las sólidas bases definidas en el aspecto matemático. Un algoritmo con el mínimo número posible de parámetros a optimizar heurísticamente y basado en un modelo debidamente justificado.

4.1.3. Uso de las herramientas matemáticas

Vamos a hacer por tanto un rápido repaso de las herramientas obtenidas en el aspecto matemático. Todos los detalles sobre su justificación, objetividad y validez se encuentran en ese capítulo.

Tenemos dos modelos matemáticos: uno sobre la estimación de un producto o propuesta (basado en sus valoraciones) y otro sobre la afinidad entre dos usuarios (basado en las valoraciones comunes de productos o propuestas).

El primero se define como la probabilidad de éxito de una distribución binomial donde los éxitos son las valoraciones positivas y los fracasos las valoraciones negativas.

El segundo se define como la probabilidad de éxito de una distribución binomial donde los éxitos son las valoraciones que coinciden y los fracasos son las valoraciones que difieren.

En ambos casos se permiten valoraciones proporcionales (por ejemplo un

8/10 corresponde a un 0.8 de éxito y un 0.2 de fracaso). Los resultados mediante inferencia bayesiana pueden visualizarse online mediante esta sencilla aplicación que he desarrollado:

`https://luiscastro193.github.io/inferencia-binomial/`

Nos da 3 valores principales: la estimación objetiva (que nos dará el ranking de los "mejores"), cómo de bueno ha demostrado ser (que nos dará el ranking de "consolidados") y cómo de bueno podría ser (que nos dará el ranking de "prometedores").

4.1.4. Algoritmo propuesto

El ranking común

Lo bueno de haber invertido tanto esfuerzo en unas herramientas tan útiles e inequívocamente determinadas es que, de entrada, el algoritmo para el caso común es directo. Tenemos unas valoraciones y unas fórmulas que nos proporcionan 3 valores con significado. Ordenamos en función de dichos valores y listo. El algoritmo formal para el caso del ranking de mejores productos o propuestas sería:

```
Para cada elemento:
   elemento.valoración =
      (1 + 2 * elemento.votos_a_favor) / (2 + 2 * elemento.núm_votos)

Ordenar el conjunto de elementos de mayor valoración a menor
```

No difiere tanto del utilizado por Stack Overflow. Sin embargo, se libra de los problemas que planteábamos en la sección anterior. Además, ahora dispones de rankings auxiliares con propuestas "prometedoras" y "consolidadas" para asegurar a todas una oportunidad o asegurar al usuario resultados fiables, respectivamente.

El ranking personalizado

Algo más complejo es el caso persalizado, en el que queramos tener en cuenta los gustos y particularidades de cada usuario. Para ello, en cada voto tendremos en cuenta quién lo ha emitido, ponderando mejor aquellos realizados por personas con las que tengamos una mayor afinidad.

Nota: Un desconocido tiene una afinidad a priori de 1/2.

El problema es que cuando el número de votos sea demasiado grande, una cantidad masiva de votos pobremente ponderados superarán a una cantidad reducida de votos altamente ponderados. Por ello será necesario definir un umbral a partir del cuál los votos son tenidos en cuenta. Este parámetro regula el nivel de "personalización" de la recomendación.

El algoritmo quedaría:

```
Para cada voto de usuario.votos
   Para cada otro_voto de voto.elemento.votos
      afinidades[otro_voto.usuario].suma +=
         máx_puntuación - abs(voto.valor - otro_voto.valor)
      afinidades[otro_voto.usuario].núm_votos += 1

Para cada otro_usuario
   afinidad = afinidades[otro_usuario]
   afinidad.valoración =
      (1 + 2 * afinidad.suma / máx_puntuacion) / (2 + 2 * afinidad.núm_votos)
   if (afinidades.valoración > umbral)
      Para cada voto de otro_usuario.votos
         elementos[voto.elemento].suma += afinidad * voto.valor
         elementos[voto.elemento].núm_votos += afinidad

Aplicar el algoritmo común para el conjunto de elementos obtenido
```

4.2. Aplicación web

En esta sección pondremos a prueba la versión para rankings comunes implementando un "Stack Overflow" genérico que use este nuevo algoritmo. El resultado final se encuentra públicamente accesible en el siguiente enlace:

`https://rankings-bayesianos.herokuapp.com/`

Disclaimer: Esta aplicación no es una simulación. Es una página web pública que incorpora tanto un front-end dinámico como un back-end para almacenar los datos de los usuarios. Por tanto, no me hago cargo de su contenido, sólo de su diseño e implementación.

Dicho esto, invito a cualquier posible lector a probarla y participar.

4.2.1. Diseño

Para la arquitectura de la aplicación, he optado por un modelo REST (Representational State Transfer) [11, capítulo 5]. El objetivo es definir una API intermedia que desacople totalmente el front-end del back-end. El trabajo del back-end consiste exclusivamente en manejar internamente los datos de los usuarios. El trabajo del front-end consiste exclusivamente en mostrar dichos datos a los usuarios. La comunicación entre ambos se realiza, también exclusivamente, mediante dicha API.

Diseño de la API

La API determina la funcionalidad de la aplicación. En este caso, necesitamos métodos para las siguientes operaciones:

- Registrar un nuevo usuario.

- Obtener el nombre de usuario del usuario actual.

- Obtener la lista de cuestiones propuestas para un orden dado.

- Enviar una nueva cuestión.

- Emitir un voto para una cuestión.

- Obtener la lista de respuestas propuestas para una cuestión y un orden dados.

- Enviar una nueva respuesta para una cuestión dada.

- Emitir un voto para una respuesta.

- Obtener los datos de una cuestión dado su identificador.

Diseño del front-end

El front-end se basa en 3 factores. Primero, un esqueleto HTML que determina la interfaz de la aplicación.

Segundo, un modelo de carga dinámica que, para cada petición de cada url, inyecta dinámicamente los datos proporcionados por la API en el esqueleto predefinido.

Tercero, un sistema de eventos que maneja la transición entre estados. Por ejemplo, hacer clic en una de las opciones de voto lanza el evento que realiza la petición mediante la API y resalta en la interfaz la opción escogida.

Diseño del back-end

Por tanto, lo único que quedar por tratar y de lo que se encarga el back-end consiste en hacer las consultas correspondientes a la base de datos para cada posible petición de la API. Además, tiene que devolver información sobre los errores en caso de producirse.

4.2.2. Implementación

El diseño es tan estándar y sencillo que la implementación es bastante directa. No obstante, vamos a comentar los aspectos relevantes de la misma.

JavaScript

Todo el código va a estar escrito en JavaScript. ¿Por qué?

En el front-end porque es el lenguaje que entienden los navegadores, que serán los que lo ejecuten. De todas formas, si queremos implementar un sistema de eventos como hemos decidido en el diseño, su modelo basado en un bucle de eventos [12] se ajusta a la perfección.

En el back-end usaremos Node.js. También aquí nos viene genial trabajar con un bucle de eventos no bloqueante [12] que nos evite tener que crear una hebra para cada petición. Hay que tener en cuenta que todo lo que tiene que hacer es enviar la consulta correspondiente a la base de datos y devolver el resultado. Sería absurdo estar lanzando constantemente hebras, con el overhead que ello conlleva, para algo tan trivial. El trabajo real lo hará la base de datos y es a ella a quien habrá que reservarle los recursos.

Heroku + PostgreSQL

La plataforma en la que se despliega la aplicación es Heroku. He escogido Heroku porque es un PaaS (Platform as a Service) que nos permite desplegar directamente código escrito para Node.js estándar. Además, su capa gratuita no caduca y es más que suficiente para lo que necesitamos.

La base de datos está alojada en su servicio basado en PostgreSQL. También se trata de tecnología totalmente estándar que no habría problema en trasladar a otro servicio en caso de interesarnos en el futuro.

Gestión de usuarios

Para el registro y login de usuarios, la aplicación usa Google Sign-in for Websites: `https://developers.google.com/identity/sign-in/web/`.

De esa manera, los usuarios pueden crear una cuenta directamente sin tener que pasar por el proceso de crear una nueva contraseña ni de confirmar su correo. Pero al mismo tiempo se controla la creación de cuentas spam y se mantiene el más alto nivel de seguridad.

Promesas

Tanto las llamadas a la API por parte del front-end como las consultas a la base de datos por parte del back-end se ejecutan mediante promesas [13].

En términos generales, una promesa es una función que no bloquea al proceso. Es decir, la llamas e inmediatamente después la ejecución del programa continúa (encargándose de otros eventos). Cuando el proceso asociado a la promesa haya terminado, se activará el código que continúa trabajando con los resultados del mismo (como otro evento independiente).

De esta manera, todas las llamadas y consultas son tratadas en paralelo. De nuevo, sin necesidad de hebras adicionales, como debería ser lógico teniendo en cuenta que lo único que hacen es esperar a que recursos externos estén listos.

Flexbox

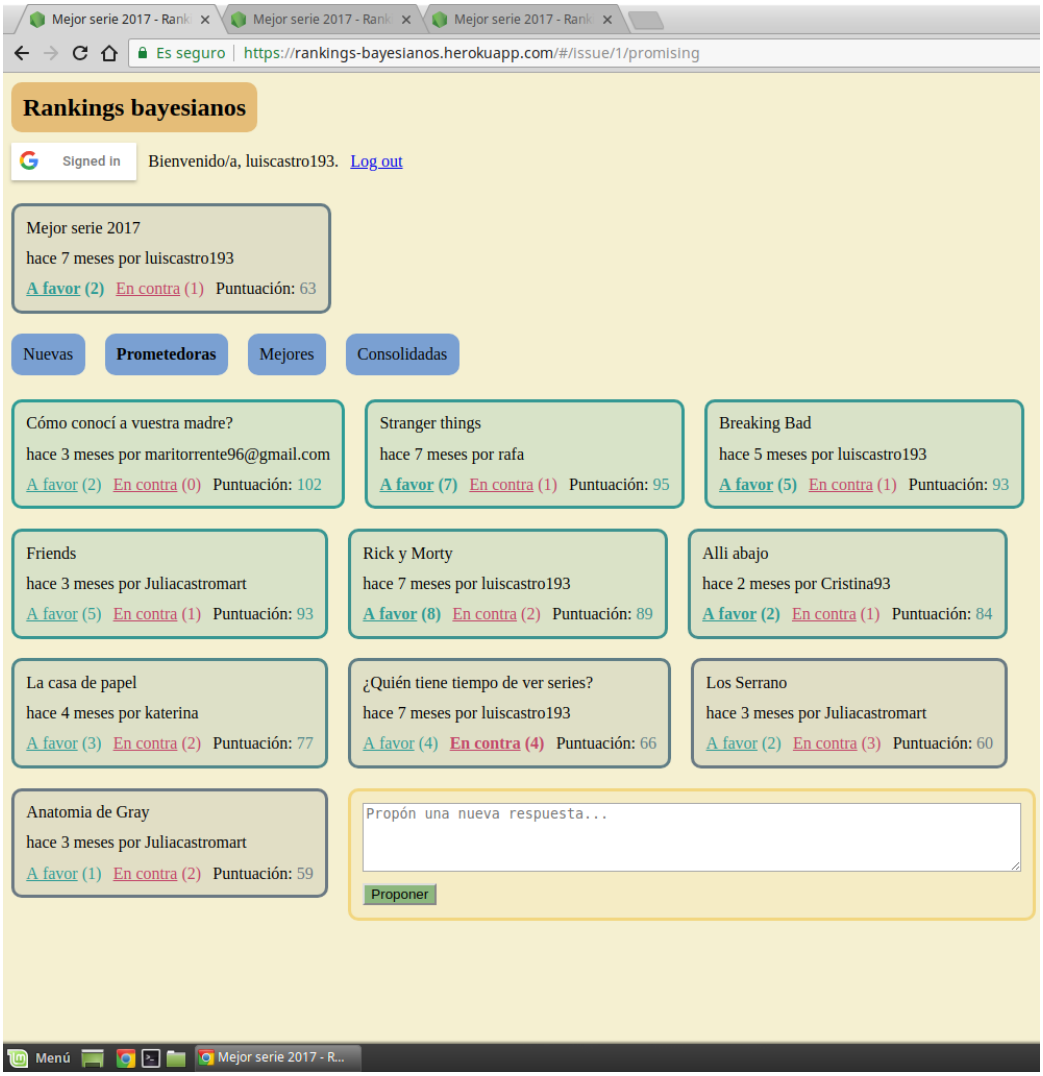
Suele hablarse mucho de que el diseño de una página web debe ser adaptable. Es decir, que pueda visualizarse cómodamente en cualquier tipo de pantalla (incluso en móviles).

Las "cajas flexibles" (flexbox) fueron introducidas en HTML5 para proporcionar una manera elegante y efectiva de conseguir eso. Consisten en contenedores cuyos elementos se extienden a lo ancho de la pantalla hasta que no quede más espacio, momento en el que empiezan a extenderse a lo largo. Algo parecido a lo que ocurre cuando escribimos en una hoja de papel.

Todos los elementos de la aplicación están implementados dentro de estas cajas flexibles para adaptarse correctamente a cualquier dispositivo.

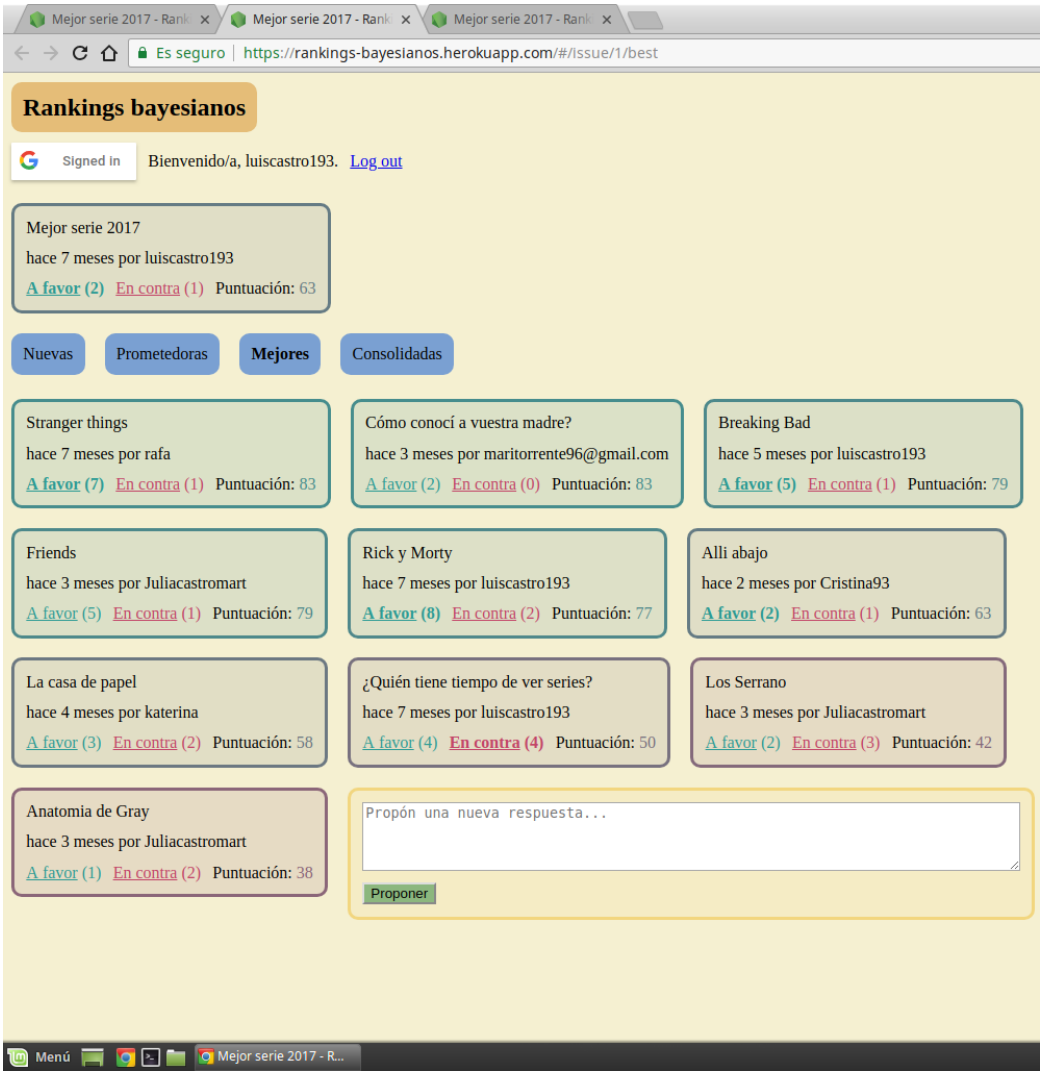
4.2.3. Ejemplo

A continuación vamos a ver un ejemplo tipo del algoritmo en acción, generado de manera natural a través de la aplicación web.



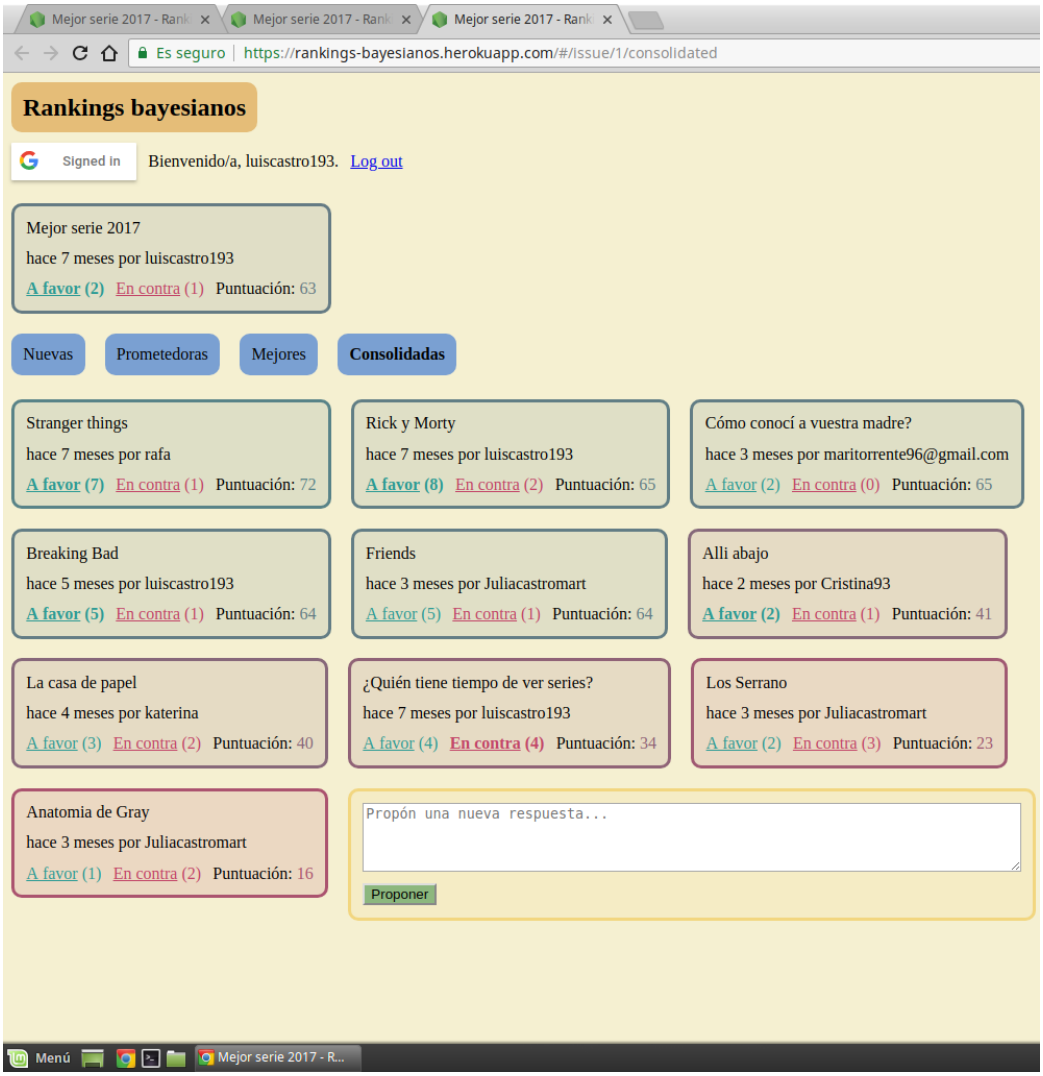
La propuesta que aparece en primera posición es “Cómo Conocí a Vuestra Madre”, con 2 votos a favor y 0 en contra. En segundo puesto está “Stranger Things” con 7 votos a favor y 1 en contra.

En principio podría parecer que 7-1 es mejor que 2-0. Ahora bien, la sección de “Prometedoras” da prioridad a aquellas que aún no han sido suficientemente valoradas. Y no es menos cierto que una alternativa con 2-0 tiene mucho potencial. Si no se le diera esa visibilidad, ella y cualquier otra nueva opción quedarían permanentemente enterradas por las que ya tienen bastantes votos.



Si ahora nos pasamos a la sección de "Mejores", observamos que la opción con 7-1 ocupa su merecido trono. Sin embargo, 2-0 sigue pareciendo mejor que 5-1 o que 8-2. Este ranking es el más equilibrado e intuitivo.

Es decir, es justo con la verdadera valoración esperada de cada una. Independientemente de si tiene más o menos votos.



Por último, tenemos el caso de querer decidirnos por una opción segura. Aquí, en la sección de "Consolidadas", se considera lo buena que una alternativa ha demostrado (hasta cierto punto) ser.

Vemos que entonces 8-2 pasa a darnos más garantías que 2-0. Aunque, curiosamente, un 2-0 sigue siendo mejor (por apenas un 1 %) que un 5-1. A pesar de tener muy en cuenta la cantidad de votos, sigue siendo mucho más coherente con las probabilidades reales que los algoritmos usuales.

4.3. Recomendador de películas

Ahora es el turno de la versión personalizada. Todo sistema de recomendación avanzado debe tener en cuenta los gustos de cada usuario. Para ilustrar el funcionamiento del algoritmo en este caso, qué mejor arquetipo que el de las películas.

4.3.1. MovieLens

MovieLens es un proyecto de GroupLens (`https://grouplens.org/about/what-is-grouplens/`), especializado en la investigación de sistemas de recomendación. Su objetivo es el mismo que el de esta sección, generar buenas recomendaciones personalizadas de películas.

Nos interesa especialmente porque tienen la cortesía de publicar los datasets con los que trabajan. Éstos pueden descargarse en `https://grouplens.org/datasets/movielens/`. En particular, haremos pruebas con un dataset de 26 millones de valoraciones, actualizado en agosto de 2017.

Es un privilegio poder contar de entrada con una base de datos fiable para mostrar las capacidades del algoritmo. Por tanto, partiremos de ella.

4.3.2. Aplicando el algoritmo

Fiabilidad

Estadísticamente hablando, la tasa de acierto debería ser mayor usando la puntuación de afinidad media. Sin embargo, para evitar falsos positivos, en esta versión usaremos la puntuación consolidada. Recordemos que ésta mide lo buena que ha "demostrado" ser, en lugar de lo buena que se espera que sea.

Accesibilidad

Para facilitar el acceso público a las pruebas, lo he implementado en un archivo HTML que puede ejecutar cualquier navegador. De esta forma, no

hay necesidad de instalar previamente programas ni paquetes específicos.

Entrada

Los datos vienen en archivos CSV. Cada película y usuario cuentan con respectivos identificadores. Las valoraciones están comprendidas entre 0.5 y 5 (como era típico en la época en la se votaba mediante estrellas), aunque le haremos una traslación de -0.5 para que partan desde cero.

Parámetros

Los dos únicos parámetros necesarios, como ya comentamos en su momento, son el identificador del usuario para el que queramos optimizar la recomendación y el umbral de afinidad a partir del cuál se tienen en cuenta las valoraciones. Este último (un real entre 0 y 1) nos permite regular el nivel de personalización.

Salida

El programa lee los datos (ratings.csv) en su formato original, aplica el algoritmo descrito en este trabajo y genera los datos en formato similar. La salida son las tres puntuaciones (por mejores, por prometedoras y por consolidadas) resultantes para cada película, ordenados en función de su puntuación consolidada.

Uso de la memoria RAM

El único problema que me encontré fue que, como es lógico, los navegadores ponen un límite por defecto a la cantidad de memoria que una aplicación puede consumir. Ese límite es configurable pero, por si acaso, hay dos versiones del programa.

La versión rápida lee el archivo de datos una sola vez y los carga todos memoria para posteriores accesos. La versión estándar accede secuencialmente al archivo en disco cada vez que sean necesarios los datos (un total de tres lecturas completas).

4.3.3. Ejemplo

Una vez tenemos el programa, podemos hacer pruebas sobre cualquier usuario. En este apartado vamos a crear un perfil muy específico con el que ejecutar el algoritmo, para mostrar como efectivamente se adapta a dicho perfil. Concretamente, el usuario ficticio insertado manualmente ha realizado las siguientes valoraciones:

- Frozen: el reino de hielo. 5 estrellas.

- Mulan. 5 estrellas.

- High School Musical. 5 estrellas.

- High School Musical 2. 4.5 estrellas.

- Barbie en La Princesa y La Costurera. 5 estrellas.

- Zootopia. 4.5 estrellas.

- Monstruos S.A. 4.5 estrellas.

- Camp Rock. 4.5 estrellas.

- Shrek. 4 estrellas.

- Planet Earth. 0.5 estrellas.

Claramente es un perfil interesado básicamente en Disney y películas de animación similares. Pues bien, las recomendaciones devueltas entre 45843 películas de todo tipo son las siguientes:

1º: Enredados. Una película de animación, de Disney, basada en el cuento de Rapunzel. Su protagonista es muy similar a las de Frozen y Mulan pero también tiene una componente romática del estilo de High School Musical o Camp Rock.

2º: Del Revés. Una película de animación de éxito mundial. Personifica a las emociones de alegría, miedo, desagrado, ira y tristeza; conviviendo dentro del cerebro de una chica. Recuerda sin duda a Zootopia y Monstruos SA. Una película infantil pero pensada para que la disfrute toda la familia.

3º: El Rey León. Qué decir de este clásico.

4º: Aladdin. Otro clásico que encaja a la perfección con el perfil completo.

5º: Buscando a Nemo. Efectivamente, otra recomendación que cualquier persona consideraría obvia.

Invito al lector a crear su propio perfil y añadirlo a la base de datos original de movieLens para probar el programa por sí mismo. Todos los experimentos realizados han dado resultados similares: un éxito rotundo a la hora de detectar películas adecuadas para el gusto del usuario concreto.

Nota: El parámetro de personalización (threshold) es importante. Habiendo tantos datos, no hay problema en exigir 0.75 y eso es suficiente para garatinzar un grado alto de personalización.

Capítulo 5

Conclusiones y vías futuras

A lo largo de todo el trabajo hemos podido comprobar el potencial que tiene la estadística bayesiana para tratar problemas relacionados con probabilidades.

En particular, hemos resuelto el dilema de los rankings en base a valoraciones. Proporcionando mecanismos para mantener un equilibrio entre propuestas novedosas y propuestas populares.

Mucho más allá de eso, su aplicación nos ha proporcionado una solución con resultados extraordinariamente fiables para un problema de ciencia de datos avanzado. Concretamente, uno tan complejo como el de los sistemas de recomendación. Hay que tener en cuenta que su interés residía tanto en la enorme utilidad que tiene como en los cuestionables resultados que dan las técnicas actuales.

Todo esto abre la puerta a considerar algoritmos cada vez más basados en complejas teorías matemáticas, antes que en complejas ideas heurísticas.

Por lo pronto, viendo la facilidad con la que se han alcanzado los objetivos, voy a montar una API para que cualquier base de datos con valoraciones pueda generar, de manera directa y eficiente, el sistema de recomendación desarrollado en este trabajo. De forma que cualquier servicio pueda incorporarlo fácilmente como funcionalidad.

Bibliografía

[1] Netflix prize. https://www.netflixprize.com/.

[2] Mung Chiang. *Networked Life*. Cambridge, 2012. ISBN: 978-1-107-02494-6.

[3] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, second edition, 1985. ISBN: 978-1-4419-3074-3.

[4] Christian P. Robert. *The Bayesian Choice*. Springer, second edition, 2007. ISBN: 978-0-387-71598-8.

[5] Morris H. DeGroot. *Optimal Statistical Decisions*. Wiley-Interscience, wiley classics library edition, 2004. ISBN: 0-471-68029-X.

[6] Geoge Casella and Roger L. Berger. *Statistical Inference*. Duxbury, second edition, 2002. ISBN: 0-534-24312-6.

[7] T. Tony Cai. One-sided confidence intervals in discrete distributions. *Statistical Planning and Inference*, 131, 2005. DOI: https://doi.org/10.1016/j.jspi.2004.01.005.

[8] Eric W. Weisstein. Central limit theorem. MathWorld. http://mathworld.wolfram.com/CentralLimitTheorem.html.

[9] Binomial distribution. Wikipedia. https://en.wikipedia.org/wiki/Binomial_distribution.

[10] Prior choice recommendations for stan software. https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations.

[11] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[12] Bucle de eventos de javascript. https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop.

[13] Promesas en javascript. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises.