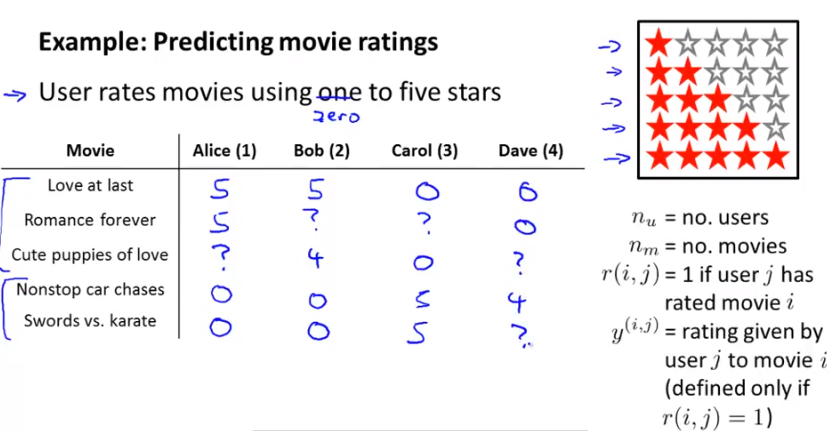
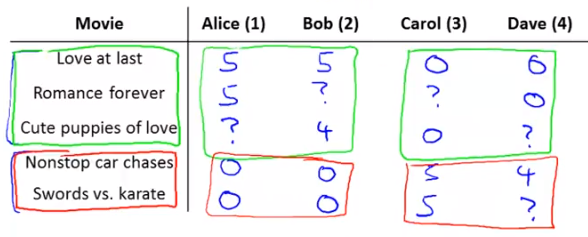
Sistemas de recomendación

# Formulación del problema

En el siguiente grupo de vídeos, me gustaría hablarles acerca de los sistemas de recomendación. Hay dos razones que tuve, dos motivaciones, por las que quería hablarles de los sistemas de recomendación.   
  
La primera es que simplemente es una aplicación importante del aprendizaje automático. En los últimos años, en algunas ocasiones he visitado distintas empresas de tecnología aquí en Silicon Valley y a menudo platico con personas que trabajan en las aplicaciones del aprendizaje automático allí, y les he preguntado cuáles son las aplicaciones de aprendizaje automático más importantes de las cuales desearían que se mejorara su desempeño. Y una de las respuestas que recibo con mayor frecuencia es que existen muchos grupos en la actualidad en Silicon Valley, intentando construir mejores sistemas de recomendación. Así es que si piensan en lo que los sitios web como Amazon, o lo que Netflix o lo que eBay, o lo que iTunes Genius, fabricado por Apple hacen, hay muchos sitios web o sistemas que tratan de recomendar nuevos productos para utilizar. Así, Amazon les recomienda libros nuevos Netflix intenta recomendarlos películas nuevas, y así sucesivamente. Y este tipo de sistemas de recomendación, que observan qué libros han comprado en el pasado, o cuáles películas han clasificado en el pasado, pero estos también son los sistemas que son responsables por hoy, una fracción sustancial de los ingresos de Amazon y por una empresa como Netflix, las recomendaciones que le hacen a los usuarios es responsable también por una fracción sustancial de las películas que ven sus usuarios. De manera que una mejora en el desempeño de un sistema de recomendación puede tener un impacto sustancial e inmediato en la parte fundamental de muchas de estas empresas. Los sistemas de recomendación son un problema un divertido dentro del aprendizaje automático académico de manera que podríamos asistir a una conferencia de aprendizaje automático académico, el problema de los sistemas de recomendación, recibe en realidad relativamente poca atención, o por lo menos en una fracción menor a lo que sucede dentro del mundo académico. Pero si observan lo que está pasando, muchas empresas de tecnología, la capacidad para construir estos sistemas parece ser una gran prioridad para muchas empresas. Y esa es una de las razones por las que quiero hablar de ellos en esta clase.   
  
La segunda razón por la que deseo hablarles acerca de los sistemas de recomendación es que a medida que nos acercamos a los últimos conjuntos de vídeos de esta clase, quiero hablar acerca de algunas de las grandes ideas en el aprendizaje automático y compartir con ustedes, ya saben, algunas de las grandes ideas en el aprendizaje automático. Y ya hemos visto en esta clase que las variables son importantes para el aprendizaje automático, las variables que elijan tendrán un gran efecto en el desempeño de su algoritmo de aprendizaje. Entonces existe esta gran idea en aprendizaje automático, que para algunos problemas, tal vez no para todos los problemas, sino para algunos problemas, hay algoritmos que pueden intentar aprender de forma automática un buen conjunto de variables por ti. Entonces, en lugar de intentar diseñar a mano o de codificar a mano las variables, que es lo que hemos estado haciendo hasta ahora, existen algunas configuraciones en las que uno podría tener un algoritmo, sólo para aprender qué variable usar y los sistemas de recomendación son solo un ejemplo de ese tipo de escenario. Existen muchos otros, pero grabados a través de sistemas de recomendación, se podrá ir un poco más dentro de esta idea de aprender las variables y podrán ver al menos un ejemplo de esta, creo yo, gran idea del aprendizaje automático también. Así que, sin más preámbulos, vamos a empezar y hablaremos de la formulación del problema del sistema de recomendación.   
  
Como mi ejemplo de ejecución, voy a utilizar el problema moderno de la predicción de las clasificaciones de películas. Por lo tanto, aquí tenemos un problema. Imaginen que son un sitio web o una empresa que vende o alquila películas o lo que sea. Así es que, ya saben, Amazon y Netflix, y creo que iTunes, son ejemplos de empresas que hacen esto, y digamos que permiten que sus usuarios califiquen diferentes películas utilizando una calificación de 1 a 5 estrellas. De modo que los usuarios pueden, ya saben, usar una, dos, tres, cuatro o cinco estrellas.   
  
Con el fin de hacer que este ejemplo sea sólo un poco más agradable, voy a permitir el 0 hasta 5 estrellas también porque eso hace que algunos de los cálculos sean más agradables, aunque la mayoría de estos sitios web utilizan la escala de 1 a 5 estrellas.   
  
Así que aquí tengo 5 películas. Amor por fin, Romance para siempre, lindos cachorros de amor, persecuciones sin paradas, y Swords vs. Karate. Y tenemos 4 usuarios, que vamos a llamar Alice, Bob, Carol y Dave, con las iniciales A, B, C y D; los llamaremos usuarios 1, 2, 3 y 4. Así que, digamos que a Alice realmente le gustó Love at last y la califica con 5 estrellas; le gustó Romance Forever y la califica con 5 estrellas. No vio Cute puppies of love y no la calificó, así es que no tenemos una calificación para esa, y a Alice realmente no le gustó Nonstop car chases ni Swords vs Karate. Y un usuario diferente, Bob, el usuario dos, tal vez calificó un conjunto diferente de películas, tal vez a él le gustó Love at last, no vio Romance forever, sólo tiene una calificación de 4, un 0, un 0; y tal vez nuestro tercer usuario califica esta con 0, no vio esa, 0, 5, 5, y, ya saben, vamos a llenar algunos de los números.

  
  
De modo que, sólo para introducir un poco de notación, esta notación que usaremos a lo largo del cálculo, voya usar nu para denotar el número de usuarios. Así, en este ejemplo,

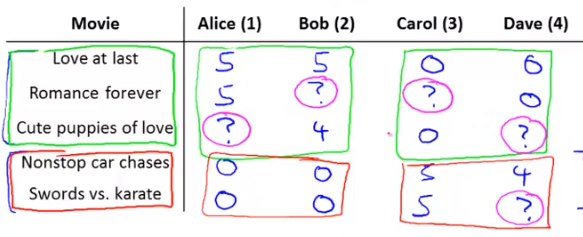
* nu será igual a 4. De manera que subíndice u significa los usuarios
* nm lo voy a usar para denotar el número de películas, así es que aquí tengo cinco películas, de modo que nm es igual a 5. Y para este ejemplo, tengo aproximadamente 3, tal vez películas románticas o películas de comedia romántica y 2 películas de acción y si observan este pequeño ejemplo, parece que Alice y Bob están dando calificaciones altas a estas comedias o películas románticas sobre el amor, y le dan calificaciones muy bajas a las películas sobre acción, y para Carol y Dave, es todo lo contrario, ¿cierto? A Carol y David, los usuarios tres y cuatro, realmente les gustan las películas de acción y les dan calificaciones altas, pero no les gustan las películas sobre romance y amor tanto.



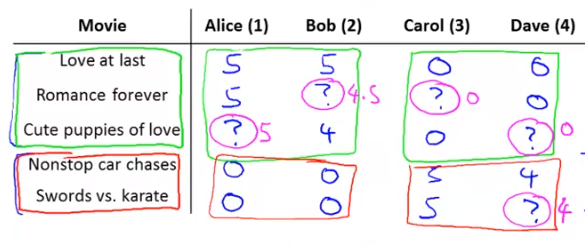
Específicamente, en el problema del sistema de recomendación se nos dan los siguientes datos. Nuestros datos comprende lo siguiente: tenemos estos valores:

* r(i, j), y r(i, j) es 1 si el usuario j ha calificado la película i. Así que nuestros usuarios calificaron sólo algunas de las películas. Y siempre que r(i, j) sea igual a 1, siempre que el usuario j haya calificado la película i, también obtenemos este número y(i, j), que es la calificación otorgada por el usuario j a la película i. De modo que y(i, j) sería un número de cero a cinco, dependiendo de la calificación con estrellas, de cero a cinco estrellas que el usuario dio a esa película en particular.

De modo que el problema del sistema de recomendación es, dados estos datos que son estos conjuntos de datos r(i, j) y los y(i, j), observar todos los datos y ver todas las calificaciones de las películas faltantes y tratar de predecir cuáles deberían ser los valores para los signos de interrogación.



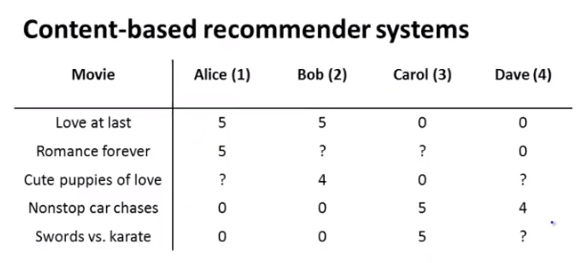
En este ejemplo particular, tengo un número muy reducido de películas y un número muy reducido de usuarios, así es que la mayoría de los usuarios han calificado la mayoría de las películas pero en escenarios reales, sus usuarios, cada uno de sus usuarios podrían haber calificado sólo una fracción minúscula de sus películas; pero al observar estos datos, si a Alice y Bob les gustan las películas románticas tal vez pensemos que Alice le habría dado un cinco a esta. Tal vez pensamos que Bob le daría a esta un 4.5 o algún valor alto, así como pensaríamos que quizás Carol y Dave darían estas calificaciones muy bajas. Y Dave, bueno, si a Dave realmente le gustan las películas de acción, tal vez le habría dado a Swords y Karate una calificación de 4 o tal vez una calificación de 5, ¿cierto?



De manera que nuestro trabajo al desarrollar un sistema de recomendación es proponer un algoritmo de aprendizaje que pueda, de forma automática, llenar esos valores faltantes por nosotros para que podamos ver, por ejemplo, las películas que el usuario no haya visto aún y recomendar nuevas películas a ese usuario para que las vea. Ustedes intentan predecir qué otra cosa podría ser interesante para un usuario. Así que ese es el formalismo del problema en el sistema de recomendación. En el siguiente vídeo empezaremos a desarrollar un algoritmo de aprendizaje para abordar este problema.

# Recomendaciones basadas en contenido

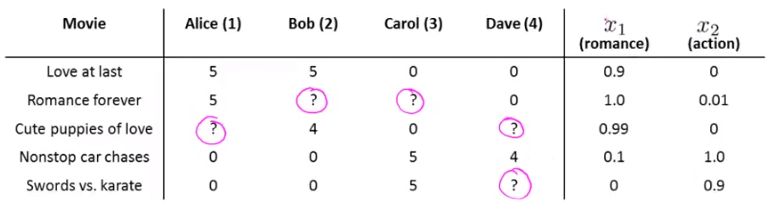
En el último vídeo hablamos acerca del problema del sistema de recomendación en el que, por ejemplo, pudieran tener un conjunto de películas y pudiera tener un conjunto de usuarios, cada uno de los cuales ha calificado algún subconjunto de las películas, calificando las películas con 1 a 5 estrellas o de 0 a 5 estrellas, y lo que me gustaría hacer, es ver a estos usuarios y predecir cómo habrían calificado otras películas que aún no han calificado. En este video, me gustaría hablar de nuestro primer planteamiento para construir un sistema de recomendación; este enfoque se denomina «recomendaciones basadas en el contenido.»   
  
Aquí tenemos el conjunto de datos que establecimos anteriormente



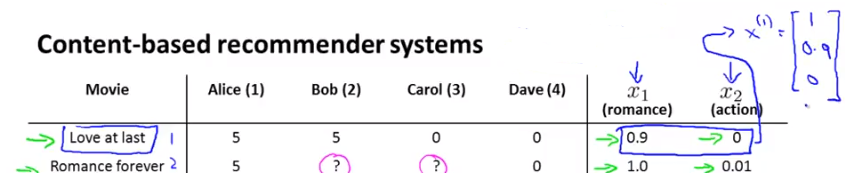
Y, sólo para recordarles un poco de la notación, estuve usando nu para denotar el número de usuarios, así es que ese es igual a 4, y nm para indicar el número de películas; tengo cinco películas.



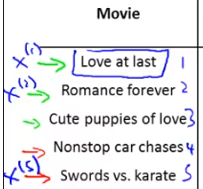
Así que, ¿cómo puedo predecir cuales serían estos valores faltantes correspondientes al signo de interrogación?   
  
Supongamos que para cada una de estas películas, tengo un conjunto de variables para ellas. En particular, digamos que para cada una de las películas, tengo dos variables que voy a llamar x1 y x2,



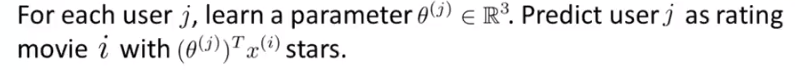
En donde x1 mide hasta qué grado una película es una película romántica y x2 mide hasta qué una película es una película de acción. Así que si tomamos la película Love at last, la calificación de 0.9 en la escala de romance, es una película muy romántica, pero cero en la escala de la acción, de modo que casi no hay acción en esa película. Romance forever tuvo 1.0, mucho romance, y 0.01 de acción, no sé, tal vez hay un accidente de tráfico menor en esa película o algo parecido, así que poca acción. Nos vamos a saltar una; veamos Swords vs. Karate; tal vez esa tiene una calificación de cero en romance y sin romance en absoluto en ella, pero mucha acción y, ya saben, persecuciones de coches sin parar. Tal vez haya un poco de romance en esa película, pero sobre todo acción, y en Cute puppies of love nuevamente, es sobre todo una película de romance sin acción en absoluto.   
  
Así que si tenemos variables como éstas, entonces cada película se puede representar con un vector de dirección. Tomemos la película 1, sólo las llamaremos películas 1 2, 3, 4 y 5. Para mi primera película, Love at last, tengo mis dos variables, 0.9 y 0, por lo que estas son las variables x1 y x2, y agreguemos una variable extra, como de costumbre, que es mi variable interceptor x0, que es igual a 1 y así, al unirlas, tendría entonces una variable x(1), el superíndice 1 indica que es el vector de dirección para mi primera película, y este vector de dirección es igual a uno. El primero de ellos es el interceptor, y luego mis dos variables 0.9, 0, de esta manera. De modo que, para Love at last, tendría un vector de dirección x(1)



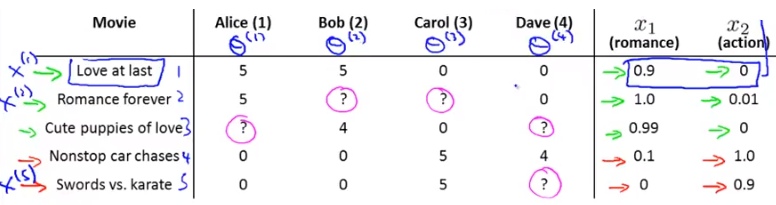
para la película Romance forever, tenemos un vector de dirección separado x(2), y así sucesivamente, y para Swords vs karate tendría un vector de dirección diferente x superíndice 5.



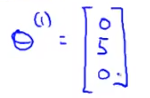
También, consistente con nuestra notación anterior que estábamos utilizando, vamos a establecer n para que sea el número de variables, sin contar este término del interceptor x0 de modo que n es igual a dos, porque tenemos dos variables x1 y x2 que capturan el grado de romance y el grado de acción en cada película. Ahora bien, a fin de hacer las predicciones, aquí está algo que podríamos hacer,



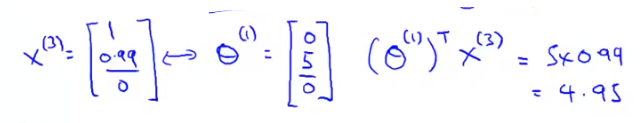
Y es que podríamos tratar de predecir las calificaciones de cada usuario como un problema de regresión lineal separado. Así es que digamos específicamente que para cada usuario j vamos a conocer un vector de parámetros teta j, que en este caso sería R3, de manera más general teta j estaría en R n+1, en donde n es el número de variables, sin contar el término de intercepción, y vamos a predecir al usuario j como si estuviera calificando la película i, sólo con el producto interno entre los parámetros vector teta y las variables "xii".   
  
Así es que vamos a tomar un ejemplo concreto. Tomemos al usuario uno.   
  
Esa sería Alice y, asociado con Alice, estaría algún vector de parámetros, teta1, y nuestro segundo usuario, Bob, estará asociado con un vector de parámetros diferente, teta2. Carol se asociará con un vector de parámetros diferente, teta3, y Dave con otro vector de parámetros, teta4.



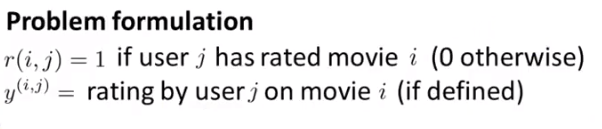
Así es que digamos que queremos hacer una predicción de lo que Alice pensará de la película, Cute puppies of love. Bueno, esa película va a tener un vector de parámetros x(3), en donde tenemos que X(3) va a ser igual a 1, que es mi término de intersepción, y después 0.99, y después 0.   
  
Y digamos que para este ejemplo, digamos que, de alguna manera, ya tenemos un vector de parámetros, teta 1 para Alice - más adelante diremos exactamente cómo encontramos este vector de parámetros- pero digamos simplemente por ahora que algún algoritmo de aprendizaje no especificado ha aprendido el vector de parámetros teta1, y es igual a 0 5 0.

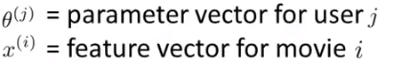


De manera que nuestra predicción para esta entrada va a ser igual a teta1, que es el vector de parámetros de Alice, multipicado por la transpuesta de x3, que es el vector de dirección para la película número 3 Cute Puppies of Love. De modo que el producto interior entre estos dos vectores va a ser 5 x 0.99, que es igual a 4.95. Así que mi predicción para evaluar esto aquí, será 4.95.



Y tal vez eso me parece un valor razonable, si es que este es mi vector de parámetros teta 1. Así que todo lo que estamos haciendo aquí es aplicar una copia diferente de la regresión esencialmente lineal para cada usuario, y estamos diciendo que lo que hace Alice, es que Alice tiene algún vector de parámetros teta1 que ella usa, que utilizamos para predecir sus calificaciones como una función de cuán romántica y cuán llena de acción está la película, y Bob, y Carol, y David tiene, cada uno, una función lineal diferente del romanticismo o la acción, o el grado de romance y el grado de acción en una película, y así es como vamos a predecir sus calificaciones con estrellas.   
  
De manera más formal, aquí está cómo podemos escribir el problema. Nuestra notación es que r(i,j) es igual a uno, si el usuario j ha calificado la película i; y(i,j) es la calificación de esa película sí existe esa calificación.

  
Esto es, si ese usuario ha calificado en realidad esa película. Y en la diapositiva anterior también definimos teta j, que es un parámetro para el usuario j, x(1) que es un vector de dirección para la película específica



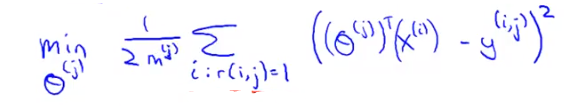
Y para cada usuario y cada película podríamos predecir esas calificaciones, de la siguiente manera.



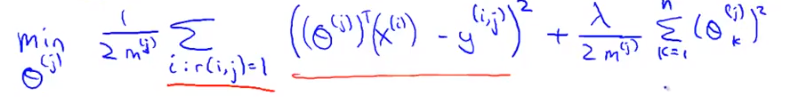
Permítanme presentarles, sólo temporalmente, una notación extra, mj; vamos a utilizar mj para indicar el número de usuarios que calificaron por película j, vamos a necesitar esta notación sólo para esta diapositiva.



Ahora, a fin de conocer el vector de parámetros para teta j, bueno, ¿cómo podemos hacerlo? Esto es básicamente un problema de regresión lineal. Así que lo que podemos hacer es simplemente elegir un vector de parámetros, tetaj, de manera que el valor predicho aquí  esté lo más cerca posible a los valores que observamos en nuestro conjunto de entrenamiento, los valores que observamos en nuestros datos.   
  
Así que vamos a escribir eso. A fin de conocer el vector de parámetros teta j, vamos a minimizar sobre mi vector de parámetros, teta j, de la suma - y quiero sumar sobre todas las películas que el usuario j haya calificado - por lo que escribimos esto como suma sobre todos los valores de i, --esto es un signo de dos puntos-- r(i,j) igual a 1. Así que la manera de leer este índice de suma es esta es la suma sobre todos los los valores de i, de manera que r(i,j) es igual a 1. Así que esto se va a sumar sobre todas las películas que el usuario j ha calificado. Y luego voy a calcular teta j multiplicada por la traspuesta xi así que esa es la predicción de la calificación del usuario j sobre la película i, menos y(i,j), así que eso es la calificación real observada, al cuadrado, y luego, voy a dividir entre el número de películas que el usuario j, haya calificado en realidad, así es que sólo se divide entre 1 sobre 2mj.



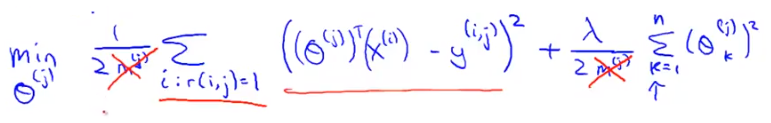
Y entonces esto es como la regresión de mínimos cuadrados, es como una regresión lineal en donde queremos elegir el vector de parámetros teta j, para minimizar este tipo de término de error al cuadrado. Y si lo desean, pueden añadir también un término de regularización de modo que más «lambda» sobre 2 m, y esto es realmente 2 m(j) porque esto como si tuviéramos ejemplos de m(j) ¿correcto? Porque si el usuario j ha calificado tantas películas, es como si tuviéramos tantos puntos de datos con los cuales ajustar los parámetros teta j, y después permítanme añadir mi habitual término de regularización aquí de teta j, k cuadrada.



Como de costumbre, esta suma es de k igual a 1 hasta n, por lo que aquí teta j va a ser una n más 1 vector de dimensiones en donde, en nuestro ejemplo anterior, n era igual a dos, pero de manera más general, n es el número de variables que tenemos por película. Así que, como de costumbre, no regularizamos sobre teta 0. No regularizamos sobre el término de oscilación, porque la suma es de k1 hasta n.

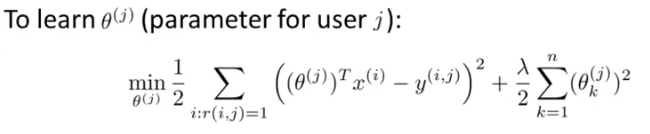
Si minimizan esto como una función de teta j obtienen una buena solución, obtienen un estimado bastante bueno de un vector de parámetros teta j con el que hacer las predicciones para las calificaciones de películas del usuario j.

Para los sistemas de recomendación, vamos a cambiar esta notación un poco. Así que para simplificar las matemáticas subsecuentes, de hecho me voy a deshacer de este término mj.

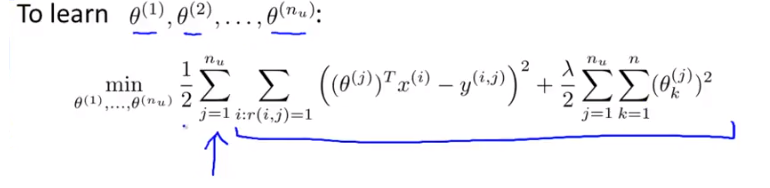


Así que eso es sólo una constante, de modo que puedo borrarlo sin cambiar el valor de teta j que obtengo de esta optimización, así que si se imaginan tomando esta ecuación completa, tomando esta expresión completa y multiplicándola por mj y se deshacen de esa constante, y cuando minimizo ésta, todavía debería obtener el mismo valor de teta j como antes.

Así, sólo para repetir lo que escribimos en la diapositiva anterior, aquí está nuestro objetivo de optimización.



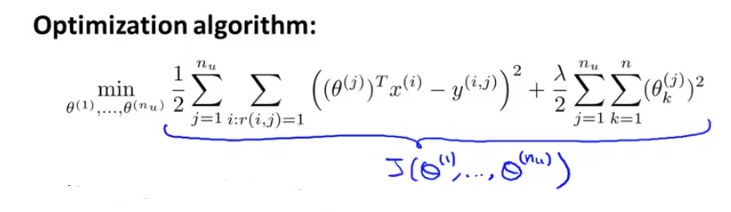
A fin de conocer teta j, que es un parámetro para el usuario j, vamos a minimizar sobre teta j de este objetivo de optimización. Así es que este es nuestro término de error cuadrado habitual y después este es nuestro término de regularización.   
  
Por supuesto, cuando estamos construyendo un sistema de recomendación, no sólo deseamos conocer los parámetros para un solo usuario, queremos conocer los parámetros para todos nuestros usuarios; tengo usuarios n subíndice u así que quiero saber todos estos parámetros y lo que voy a hacer es tomar esta minimización, tomar este objetivo de optimización y sólo agregar una suma adicional allí. Así que, esta expresión aquí con la mitad en la parte superior de nuevo, así que es exactamente lo mismo como lo que tenemos en la parte superior, excepto que ahora, en lugar de sólo hacer esto para un usuario específico teta J, voy a sumar mi objetivo sobre todos mis usuarios y luego voy a minimizar este objetivo de optimización general minizando esta función de costos general.



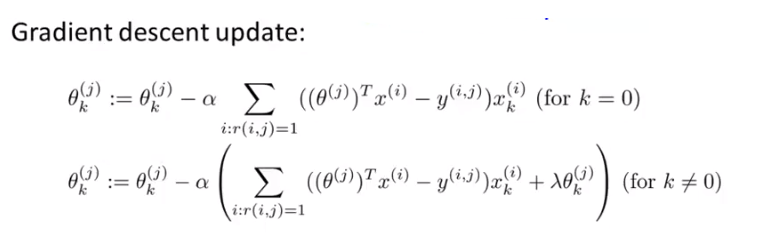
Y cuando minimizo esto como una función de teta 1, teta 2, hasta teta nu, obtendré un vector de parámetros independiente por cada usuario y entonces puedo usar éste para hacer predicciones para todos mis usuarios para todos mis usuarios n subíndice u.

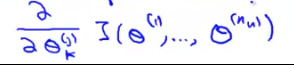


Así que al poner todo junto, este fue nuestro objetivo de optimización en la parte superior, y para darle un nombre a esto, voy a llamar a esta j de teta 1, punto, punto, punto teta nu. De manera que, como de costumbre, j es mi objetivo de optimización que estoy tratando de minimizar.



A continuación, con el fin de realmente hacer la minimización, si van a derivar las actualizaciones en gradiente descendente, estas son las ecuaciones que obtendrían,

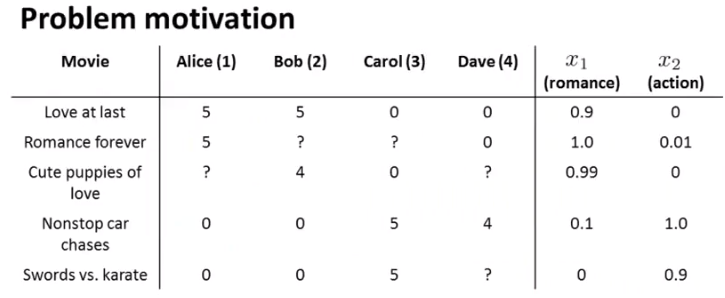
  
Así que tomaría teta jk y le restaría «alfa», que es la tasa de aprendizaje multiplicada por estos términos, a la derecha. Así que tenemos casos ligeramente diferentes, de modo que cuando k es igual a 0 y cuando k no es igual a 0, debido a nuestro término de regularización aquí regulariza sólo los valores de teta jk sólo si k no es igual a 0. Así es que no regularizamos teta 0, de modo que las ligeramente diferentes actualizaciones para k son igual a cero, y k no es igual a 0. Y este término, por aquí, por ejemplo, es sólo una derivada parcial con respecto a su parámetro, el de su objetivo de optimización, ¿correcto? Así es que, esta es sólo una gradiente descendiente y ya he calculado las derivadas y las conecté aquí.



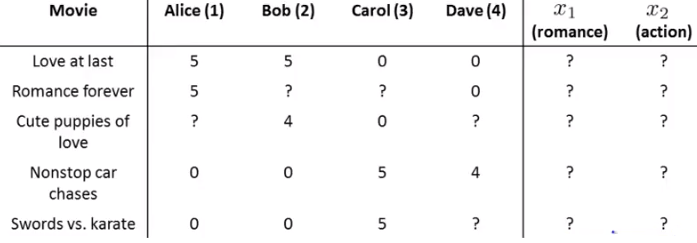
Si estas actualizaciones en gradiente descendente son muy parecidas a las que tuvimos para la regresión lineal, esto es debido a que estas son esencialmente las mismas que las de la regresión lineal. La única diferencia menor es que para la regresión lineal tenemos estos 1 sobre los términos m - es realmente 1 sobre mj - pero debido a que antes, cuando estábamos derivando el objetivo de optimización nos libramos de esto, es por eso que no tenemos este 1 sobre el término m. Pero por lo demás, es realmente la suma de mis ejemplos de entrenamiento de, ya saben, el error multiplicado por xk más ese término de regularización que contribuye a la derivada.   
  
De manera que si están usando el gradiente descendiente, aquí está cómo pueden minimizar la función de costo j, para conocer todos los parámetros y el uso de estas fórmulas para las derivadas, si lo desean, también pueden conectarlas en un algoritmo de optimización más avanzada como un gradiente de clúster o LBFGS o lo que sea, y usar eso para tratar de minimizar la función de costo j también.   
  
Así que espero que ahora sepan cómo pueden aplicar esencialmente una variación en la regresión lineal a fin de predecir las diferentes calificaciones de las películas por parte de diferentes usuarios. Este algoritmo particular se llama recomendaciones basadas en el contenido, o enfoque basado en contenidos, ya que suponemos que tenemos disponibles variables para las diferentes películas. De manera que tenemos variables que capturan lo que es el contenido de estas películas. ¿Qué tan romántica es esta película? ¿Cuánta acción hay en esta película? Y en realidad estamos usando las variables del contenido de las películas para hacer nuestras predicciones. Pero para muchas películas, no tenemos en realidad dichas variables, o pudiera ser muy difícil conseguir tales variables para todas nuestras películas, para todos los artículos que sea que estemos tratando de vender.   
  
De modo que en el siguiente vídeo empezaremos a hablar de un enfoque para los sistemas de recomendación que no se basa en el contenido, y no supone que tenemos disponible a alguien más que nos dé todas estas variables para todas las películas en nuestro conjunto de datos.

# Filtrado colaborativo

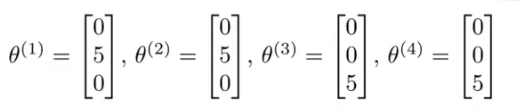
En este video definiré cuál es posible que sea el problema más común un enfoque para construir un sistema de recomendación llamado filtrado colaborativo.   
  
El algoritmo del que estamos hablando tiene una propiedad muy interesante que hace lo que se llama aprendizaje de variables y con esto quiero decir que este será un algoritmo que puede empezar a aprender por sí mismo qué variables usar.   
  
Aquí tenemos el conjunto de datos que teníamos y habíamos supuesto que para cada película, alguien había venido y dicho qué tan romántica era esa película era y cuánta acción hubo en esa película.



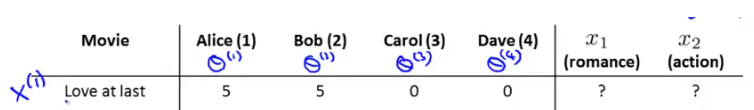
Pero como se pueden imaginar, puede ser muy difícil y se consume tiempo y dinero si se trata en realidad que alguien, ya saben, vea cada película y les diga que tan romántica es cada película y cuánta acción tiene cada película, y muchas veces ustedes querrán aún más variables que sólo estas dos. Así que, ¿de dónde obtienen estas variables?   
  
Cambiemos un poco el problema y supongamos que tenemos un conjunto de datos en donde no sabemos los valores de estas variables. Así que tenemos el conjunto de datos de las películas y de qué forma las calificaron los usuarios, pero no tenemos idea de lo romántica que es cada película y no tenemos idea de cuánta acción tiene cada película, por lo que reemplacé todas estos asuntos con signos de interrogación.



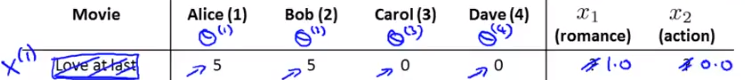
Pero ahora vamos a hacer una suposición un poco diferente.   
  
Digamos que hemos ido con cada uno de nuestros usuarios, y que cada uno de nuestros usuarios nos ha dicho cuánto le gustan las películas románticas y cuánto le gustan las películas llenas de acción.



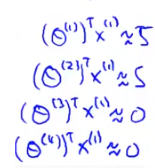
Así que Alice está asociada con un vector de parámetros teta1, Bob, teta2, Carol, teta3, Dave, teta4. Y digamos que también usamos esta y que Alice nos dice que a ella realmente le gustan las películas románticas, así que hay un cinco aquí, que es el multiplicador asociado con x1, y digamos que Alice nos dice que en realidad no le gustan las películas de acción, así que hay un 0 para x2. Bob nos dice algo similar, mientras Carol nos dice que a ella realmente le gustan las películas de acción, por lo que hay un 5 allí, es el multiplicador asociado con x2, y recuerden que también hay x0 que es igual a 1, y digamos que Carol nos dice que no le gustan las películas románticas y así sucesivamente, de manera similar para Dave. Así que supongamos que de alguna manera, podemos ir a los usuarios y cada usuario j nos dice cuál es el valor de teta j para ellos. Así que básicamente nos especifica lo mucho que les gustan los diferentes tipos de películas.   
  
Si podemos obtener estos parámetros teta de nuestros usuarios, entonces resulta que se hace posible tratar de inferir cuáles son los valores de x1 y x2 para cada película.   
  
Veamos un ejemplo. Veamos la película 1. Así que esa película 1 está asociada con un vector de dirección x(1)



Y ya saben, esta película se llama Love at last, pero vamos a ignorar eso. Hagamos de cuenta que no sabemos de qué se trata esta película, así que vamos a ignorar el título de esta película. Todo lo que sabemos es que a Alice le encantó esta película. A Bob le encantó esta película. Carol y Dave odiaron esta película. Así que, ¿qué podemos deducir? Bueno, sabemos por los vectores de variables, que a Alice y a Bob les encantan las películas románticas porque nos dijeron que hay un 5 aquí. En cuanto a Carol y Dave, sabemos que ellos odian las películas románticas y que les encantan las películas de acción. De modo que debido a que esos son los vectores de parámetros que los usuarios 3 y 4, Carol y Dave, nos dieron. Así que basados en el hecho de que la película 1 les encantó a Alice y a Bob, y Carol y Dave la odiaron, podríamos concluir de manera razonable que esta es probablemente una película romántica, probablemente no sea una película con mucha acción.



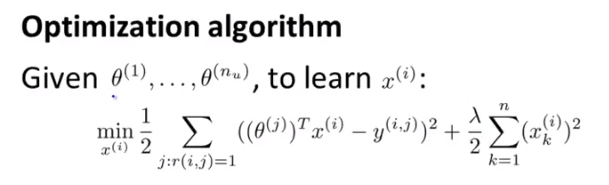
Este ejemplo es un poco matemáticamente simplificado, pero lo que realmente estamos preguntando es qué vector de dirección debería ser x1 de manera que la transpuesta de teta 1 por x(1) es aproximadamente igual a 5; esa es la calificación de Alice, y teta 2transpuesta por x(1) también es aproximadamente igual a 5, y teta 3 transpuesta por x(1) es aproximadamente igual a 0, por lo que esta sería la calificación de Carol, y teta 4 transpuesta por x(1) es aproximadamente igual a 0.



Y de esto, parece que x(1) es igual a uno, que es el término de intercepción, y luego 1.0, 0.0:

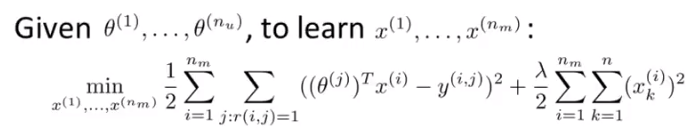


eso tiene sentido ya que sabemos las preferencias de Alice, Bob, Carol, y Dave por las películas, y la forma en la que calificaron esa película.   
  
Así, de manera más general, podemos revisar esta lista y tratar de averiguar cuáles podrían ser variables razonables para estas otras películas también.   
  
Vamos a formular este problema para conocer las variables x(1). Digamos que nuestros usuarios nos han dado sus preferencias. Así que digamos que nuestros usuarios vinieron y nos dijeron estos valores para teta 1 hasta teta nu y queremos saber el vector de dirección x(1) para la película número i. Lo que podemos hacer es, por lo tanto, plantear el siguiente problema de optimización.

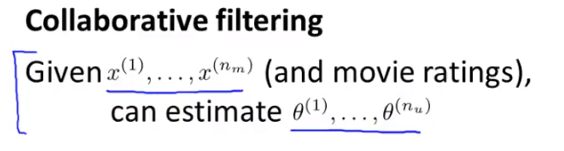


Así que queremos sumar sobre todos los índices de j para los que tenemos una calificación para la película i porque estamos tratando de conocer las variables para la película i que es este vector de dirección x(1).   
  
Así que a continuación, lo que queremos es minimizar el error al cuadrado, así que queremos elegir las variables x(i), de manera que, el valor predictivo de cómo el usuario j califica la película i será similar, no estará muy alejado en el sentido del error cuadrado del valor real y(i,j) que observamos en realidad en la calificación del usuario j sobre la película i.

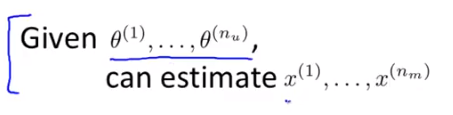
Así, sólo para resumir lo que hace este término, es que trata de elegir las variables x(i) de modo que para todos los usuarios j que calificado esa película, el algoritmo también predice un valor para cómo habría calificado ese usuario esa película; eso no está demasiado alejado, en el sentido del error cuadrado, del valor real con el que el usuario calificó esa película.   
Así que ese es el término de error al cuadrado. Como de costumbre, podemos añadir también el término de regularización para prevenir que las variables se hagan demasiado grandes.   
  
De modo que así es cómo conoceríamos las variables para una película específica, pero lo que queremos hacer es conocer todas las variables para todas las películas, así que lo que voy a hacer es añadir esta suma adicional,



Así voy a sumar sobre todas las películas nm, películas n subíndice m, y voy a minimizar este objetivo en la parte superior que suma todas las películas. Y si si hacen esto, terminan con el siguiente problema de optimización.   
  
Y si minimizan esto, con suerte tendrán un conjunto razonable de variables para todas sus películas.   
  
Así que juntando todo, el algoritmo del que hablamos en el vídeo anterior y el algoritmo del que acabamos de hablar en este video, en el vídeo anterior, lo que mostramos es que, si tienen un conjunto de calificaciones de películas, --sí tienen los datos r(i,j) y tienen los datos y(i,j)-- así que tenemos las calificaciones de películas, entonces, dadas las variables para sus diferentes películas, podemos conocer estos parámetros teta. Así que si conocieran las variables podrían saber los parámetros teta para sus diferentes usuarios.



Y lo que mostramos anteriormente en este vídeo es que si los usuarios están dispuestos a darles los parámetros, entonces ustedes pueden calcular las variables para las diferentes películas.



Así que esto es una especie de problema del huevo y la gallina. ¿Cuál fue primero? Ya saben, si podemos obtener los tetas, podemos conocer las x’s. Si tenemos las x’s, podemos conocer los tetas.

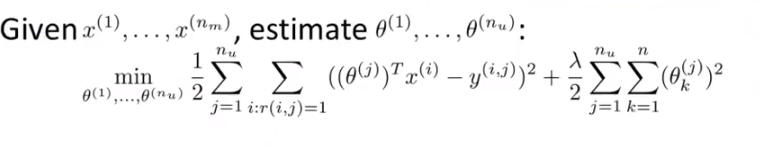
Lo que pueden hacer es, y esto realmente funciona, lo que pueden hacer es, de hecho, adivinar al azar algún valor para las tetas. Con base en su conjetura aleatoria inicial para las tetas, pueden seguir adelante y usar el procedimiento del que acabamos de hablar a fin de conocer las variables para sus diferentes películas.   
  
Ahora, dado algún conjunto inicial de variables para sus películas, pueden usar entonces este primer método del que hablamos en el vídeo anterior, para tratar de obtener una estimación aún mejor para sus parámetros teta. Ahora que tienen una mejor configuración de los parámetros teta para sus usuarios, podemos usar eso para, tal vez, incluso obtener un mejor conjunto de variables y así sucesivamente. Podemos seguir iterando, yendo y viniendo y optimizando teta, x, teta, x, teta, y esto realmente funciona, y si hacen esto, esto en realidad va a causar que su algoritmo converja en un conjunto razonable de variables para sus películas y un conjunto razonable de parámetros para sus diferentes usuarios.



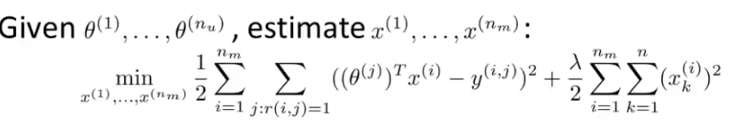
Así que este es un algoritmo básico de filtrado colaborativo. Esto no es en realidad el algoritmo final que vamos a usar. En el siguiente vídeo, vamos a poder mejorar en este algoritmo y hacerlo un poco más computacionalmente eficaz. Pero espero que éste les haya dado una idea sobre cómo pueden formular un problema en el que pueden conocer de manera simultánea los parámetros y al mismo tiempo conocer las variables de las diferentes películas.   
  
Y para este problema, para el el sistema de recomendación, esto es posible sólo porque cada usuario califica varias películas y, con suerte, cada película es calificada por varios usuarios. Así que pueden hacer esto un proceso de ida y venida para calcular teta y x. De modo que para resumir, en este vídeo hemos visto un algoritmo de filtrado colaborativo inicial. El término filtrado colaborativo se refiere a la observación de que, cuando ejecutan este algoritmo con un gran conjunto de usuarios, lo que todos estos usuarios están haciendo efectivamente es una especie de colaboración - o colaboran para obtener mejores calificaciones de películas para todos, porque con cada usuario que califica un subconjunto de películas, cada usuario le está ayudando al el algoritmo un poco a conocer mejores variables,   
y luego al ayudar - al calificar algunas películas yo mismo, estaré ayudando al sistema a conocer mejores variables, y entonces estas variables pueden ser usadas por el sistema para hacer mejores predicciones de películas para todos los demás. Así que hay un sentido de colaboración en el que cada usuario está ayudando al sistema a conocer mejores variables para el bien común. Esto es filtrado colaborativo. En el siguiente vídeo, lo que vamos a hacer es tomar las ideas que han funcionado, y tratar de desarrollar un mejor algoritmo, una técnica un poco mejor para el filtrado colaborativo.

# Algoritmo de filtrado colaborativo

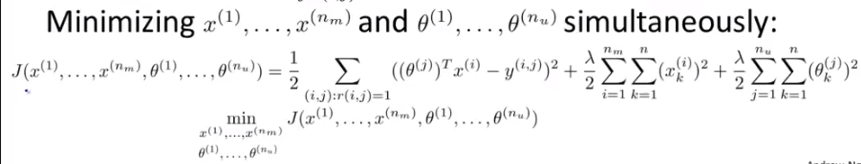
En los últimos dos vídeos, hablamos sobre las ideas de cómo, en primer lugar, si se les han proporcionado variables para sus películas, pueden usar eso para saber los parámetros teta para los usuarios. Y en segundo lugar, si se les proporcionaron los parámetros para los usuarios, pueden usar eso para saber las variables para las películas. En este vídeo, vamos a tomar esas ideas y las vamos a unir para presentar un algoritmo de filtrado colaborativo.   
  
Así es que una de las cosas con las que trabajamos anteriormente, es que si tienen las variables para las películas, entonces pueden resolver este problema de minimización para encontrar la parámetros teta para sus usuarios.



Y luego también resolvimos que si se les dan los parámetros teta, también pueden usar eso para estimar las variables x, y pueden hacer eso solucionando este problema de minimización.

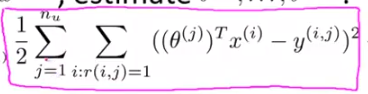


Así que algo que podrían hacer es en realidad ir y venir. Quizás inicializar aleatoriamente los parámetros y luego despejar para teta, despejar para x, despejar para teta, despejar para x. Pero resulta que existe un algoritmo más eficiente que no necesita ir y venir entre las x’s y las teta, pero que puede resolver para teta y x de manera simultánea. Y aquí está.

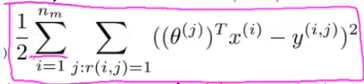


Lo que haremos es básicamente tomar ambos objetivos de optimización y colocarlos en el mismo objetivo. Así es que voy a definir el nuevo objetivo de optimización j, el cual es una función de costos, que es una función de mis variables x y una función de mi parámetros teta. Y son básicamente los dos objetivos de optimización que tenía en la parte superior, pero los puse juntos.   
  
Entonces, con el fin de explicar esto, primero quiero señalar que los términos de error recogidos en las dos primeras ecuaciones que mide la diferencia entre los valores estimados y reales al cuadrado, es el mismo en ambas, y las sumatorias se ven un poco diferente, pero vamos a ver lo que las sumas están haciendo en realidad.

La primera suma es la suma sobre todos los usuarios j y después, la suma sobre todas las películas calificadas por ese usuario.



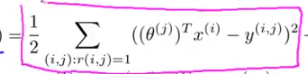
Por lo tanto, esto está realmente sumando sobre todos los pares ij, que corresponden a una película que fue calificada por un usuario. La suma sobre j dice, para cada usuario, la suma de todas las películas calificadas por ese usuario.   
  
El sumatorio de la segunda ecuación simplemente hace las cosas en el orden inverso:



Esto dice que para cada película i, suma sobre todos los los usuarios j que han calificado esa película, así que estas sumas, ambas son sólo sumas sobre todos los pares ij para las que r de i,j es igual a 1. Es solamente sumar sobre todas los pares de películas del usuario para las que tienen una calificación,

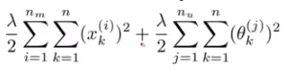


De modo que estos dos términos allí son sólo exactamente este primer término:



Y acabo de escribir la suma aquí de forma explícita, en donde sólo digo que la suma de todos los pares ij, de tal forma que r(i,j) es igual a 1. Así que lo que vamos hacer es definir un objetivo de optimización combinado que queremos minimizar a fin de resolver de forma simultánea para x y teta.

Los otros términos en el objetivo de optimización son estos:



Que es una regularización en función de teta y de x que nos la hemos traido al combinar las dos ecuaciones.

Y este objetivo de optimización. j en realidad tiene una propiedad interesante, que si mantuvieran la constante de x y simplemente minimizaran con respecto a teta, entonces estarían resolviendo exactamente la primera ecuación de minimización, mientras que si hicieran lo contrario, si fueran a mantener la constante de teta y minimizar j sólo con respecto a las x, entonces se convertiría en equivalente a la segunda ecuación de minimización, ya sea porque en mi ecuación resultante de combinar las otras dos tanto el primer término de regularización o el segundo término es una constante si están minimizando sólo respecto a las x , o sólo respecto a las tetas. Así que aquí está un objetivo de optimización objetivo que reúne mis funciones de costo en términos de x, y en términos de teta.

Y con el fin de llegar a un solo problema de optimización, lo que vamos a hacer, es tratar esta función de costos como una función de mis variables x y de los parámetros de mi usuario teta y simplemente minimizar todo esto como una función tanto de las x’s, como de los tetas.



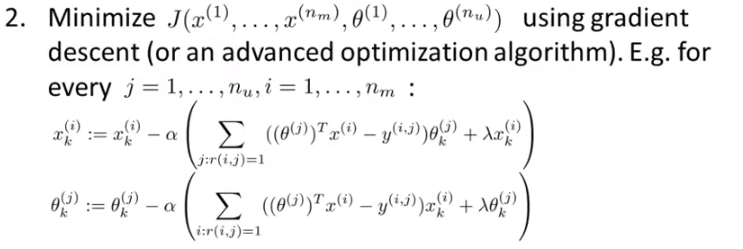
Y realmente la única diferencia entre éste y el algoritmo anterior es que, en lugar de ir y venir, ya saben, anteriormente hablamos de minimizar con respecto a teta y después minimizar con respecto a x, mientras minimizamos con respecto a teta, minimizamos con respecto a x, y así sucesivamente. En esta nueva versión, en lugar de ir de manera secuencial entre los 2 grupos de parámetros x y teta, lo que vamos a hacer es minimizar con respecto a ambos conjuntos de parámetros de manera simultánea.   
  
Finalmente, un último detalle, es que cuando estamos aprendiendo las variables de esta forma, anteriormente habíamos usado esta convención de que tenemos una variable x0 igual a x0 = 1, que corresponde a un término de interceptor. Cuando usamos esta especie de formulación en el que estamos realmente conociendo las variables, realmente vamos a eliminar esta convención y así las variables que vamos a aprender, x, estarán en Rn, mientras que anteriormente teníamos las variables x y Rn + 1, incluyendo el término de intercepción al deshacernos de x0, ahora sólo tenemos x en Rn.   
  
Y de manera similar, debido a que los parámetros teta están en la misma dimensión, ahora también tenemos a teta en Rn porque, si no hay x0, entonces no hay necesidad del parámetro teta0 tampoco. Y la razón por la que eliminamos esta convención es porque ahora estamos conociendo todas las variables, ¿verdad? De modo que no hay necesidad para codificar la variable que siempre es igual a uno. Porque si el algoritmo realmente desea una variable que sea siempre igual a 1, puede optar por aprender una por sí mismo. Así que si el algoritmo lo elige, puede establecer la variable x1 es igual a 1. de modo que no es necesario codificar la variable de 001, el algoritmo ahora tiene la flexibilidad para simplemente aprender por sí mismo.

Así, uniendo todo, aquí está nuestro algoritmo de filtrado colaborativo.

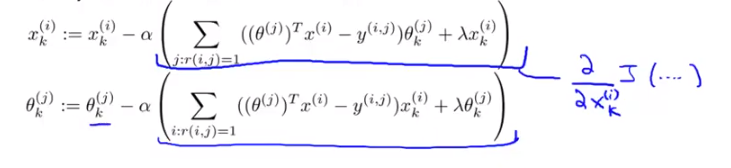
1. Primero, vamos a inicializar x y teta a pequeños valores aleatorios.Y esto es un poco como el entrenamiento de la red neuronal, en donde también inicializamos todos los parámetros de una red neuronal a pequeños valores aleatorios.



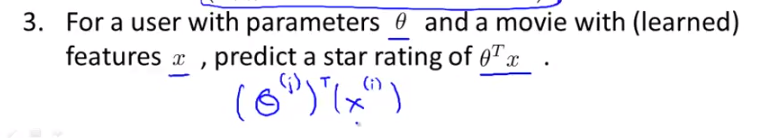
1. Después, vamos a minimizar la función de costos usando descensos de gradiente o uno de los algoritmos de optimización avanzados. Por lo tanto, si toman las derivadas, encontrarán las actualizaciones de gradiente en descenso como estas



y por tanto el primer término aquí es la derivada parcial de la función de costos con respecto al valor de variable x(i)k y, de manera similar, el segundo término aquí también es un valor de la derivada parcial de la función de costos con respecto al parámetro teta que estamos minimizando.



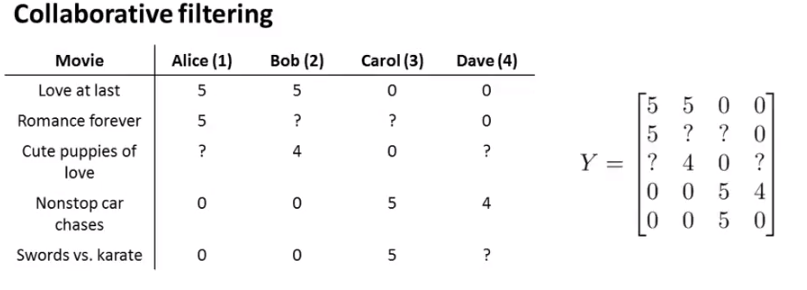
Y sólo como recordatorio, en esta fórmula ya no tenemos esta x0 que es igual a 1 y tenemos que x está en Rn y teta es una Rn. En esta nueva formulación, estamos regularizando cada uno de nuestros parámetros teta, ya saben, cada uno de nuestros parámetros x. Ya no existe el caso especial teta cero, que se regularizó de manera diferente, o que no se regularizó en comparación con la parámetros teta1 hasta teta n. Así que ahora ya no hay una teta 0, razón por la que, en estas actualizaciones, no desglosé un paréntesis especial para cuando k es igual a 0. De modo que entonces usamos el gradiente de descenso minimizar la función de costos j con respecto a las variables x y con respecto a los parámetros teta.   
  
Finalmente, dado un usuario, si un usuario tiene algunos parámetros, teta, y si hay una película con algún tipo de variables aprendidas x, entonces podríamos predecir que a esa película se le daría una calificación de y estrellas por parte de ese usuario usando teta transpuesta por x . O sólo para llenarlos, entonces diríamos que si el usuario j aún no ha calificado la película i, entonces lo que hacemos es predecir que el usuario j va a calificar la película i de acuerdo con teta transpuesta por xi.



Así que ese es el algoritmo de filtrado colaborativo. Si implementan este algoritmo, obtienen en realidad un algoritmo muy decente que aprenderá de manera simultánea buenas variables para, con suerte, todas las películas, así como conocer los parámetros para todos los usuarios y dar muy buenas predicciones sobre cómo los diferentes usuarios calificarán diferentes películas que aún no hayan calificado

# Vectorización

En los últimos vídeos, hemos hablado de un algoritmo de filtrado colaborativo. En este vídeo voy a hablar un poco acerca de la implementación de vectorización de este algoritmo. Y también hablaré un poco acerca de otras cosas que pueden hacer con este algoritmo. Por ejemplo, uno de las cosas que puedes hacer es, dado un producto, ¿se pueden encontrar otros productos relacionados a éste? de modo que, por ejemplo, si un usuario ha estado observando recientemente un producto, ¿hay otros productos relacionados que le pudieran recomendar a este usuario? Así que veamos lo que podemos hacer al respecto.   
  
Lo que me gustaría hacer es elaborar una manera alternativa de escribir las predicciones del algoritmo de filtrado colaborativo. Para empezar, aquí está nuestro conjunto de datos con nuestras cinco películas. Y lo que voy a hacer es tomar todas las calificaciones por parte de todos los usuarios y agruparlos en una matriz.

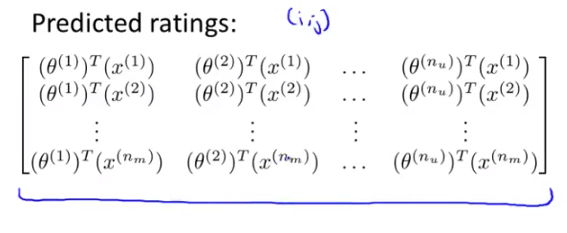


Así pues, aquí tenemos cinco películas y cuatro usuarios, así que esta matriz “Y” será una matriz de 5 por 4. Sólo se toman todos los elementos, todos estos datos, incluyendo los signos de interrogación, y se agrupan en esta matriz. Y por supuesto, los elementos de esta matriz, el elemento (i, j)



De esta matriz es lo que realmente estuvimos escribiendo anteriormente como y superíndice i, j. Es la calificación que se dio a la película i por el usuario j.

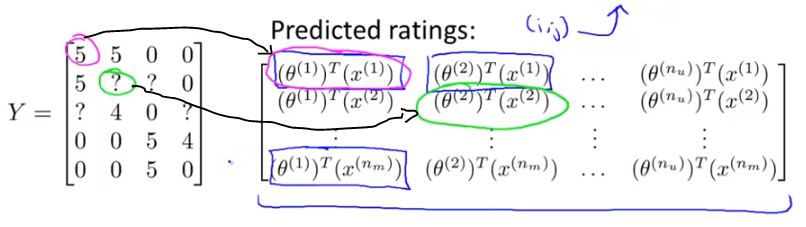
Dada esta matriz Y de todas las calificaciones que tenemos si observan lo que cierto usuario predice sobre alguna película, lo que el usuario j predice sobre la película i se da mediante una matriz de las calificaciones predichas, lo que tendrían es la siguiente matriz en la entrada i, j.:



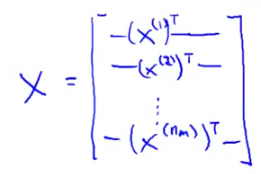
Así que esto corresponde a la calificación que predijimos que el usuario j, le dará a la película i es exactamente igual a esa «thetaj» transpuesta multiplicada por x(I),



y por eso, esta es una matriz en donde EL primer elemento, el elemento uno-uno, es una calificación predictiva del usuario uno o la película uno, y el elemento uno-dos es la calificación predicha del usuario dos para la película uno, y así sucesivamente, y la calificación fila 1 columna n es la calificación predicha del usuario uno sobre la última película.

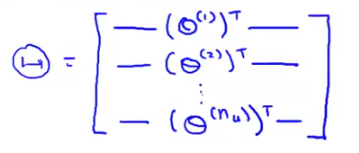


Ahora, dada esta matriz de calificaciones predictivas, existe entonces una manera más sencilla o vectorizada de escribir éstas. En particular, si defino la matriz x, y ésta va a ser justo como la matriz que teníamos anteriormente para la regresión lineal de modo que x1 transpuesta x2 transpuesta hasta x(nm) transpuesta. Así que voy a tomar las variables para mis películas y las voy a apilar en filas:

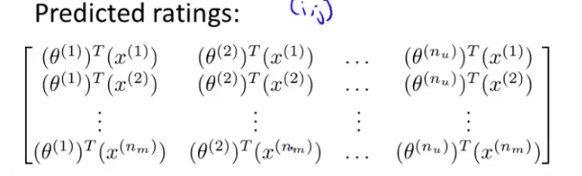


Así es que si piensa en cada película como un ejemplo y apilan todas las variables de las diferentes películas y filas.

También definimos una matriz, «theta» mayúscula, Lo que voy a hacer hacer es tomar cada uno de los vectores de parámetros por usuario y apilarlos en filas, de esta manera.



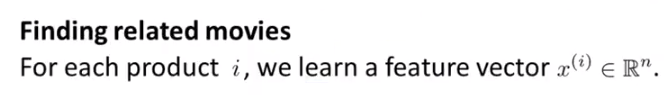
Así que tengo «theta» 1, que es el vector de parámetros para el primer usuario. , «theta» 2, así que deben apilarlos en filas de esta forma, para definir una matriz «theta» mayúscula así que tengo los vectores de parámetros todos apilados en filas.   
  
Ahora, dada esta definición para la matriz x, y esta definición para la matriz «theta», a fin de tener una forma vectorizada para calcular la matriz de todas las predicciones, simplemente pueden calcular x veces la transpuesta de la matriz «theta», y eso les da una manera vectorizada para calcular la matriz de predicciones para cada usuario.

= 

Para darle al algoritmo de filtrado colaborativo que han estado usando, otro nombre, el algoritmo que estamos usando también se llama factorización de matriz de bajo rango.   
  
Así que si escuchan hablar a las personas acerca de la factorización de matriz de bajo rango, ese es en esencia exactamente el algoritmo del que hemos estado hablando. Y este término proviene de la propiedad que esta matriz x multiplicada por theta transpuesta y tiene una propiedad matemática en la álgebra lineal que se llama matriz de bajo rango y así que eso es lo que hizo surgir el nombre, factorización de matriz de bajo rango para estos algoritmos, debido a esta propiedad de bajo rango de esta matriz «theta» transpone a x. En caso de que no sepan qué significa bajo rango, o en caso de que no sepan qué es una matriz de bajo rango, no se preocupen por eso. Realmente no necesitan saber eso para poder utilizar este algoritmo. Pero si son expertos en álgebra lineal, eso es lo que produce este algoritmo, este otro nombre de factorización de matriz de bajo rango.

Por último, después de ejecutar el algoritmo de filtrado colaborativo, aquí está otra cosa que pueden hacer, que es usar las variables aprendidas a fin de encontrar películas relacionadas.

Específicamente, para cada producto i, realmente para cada película i, hemos aprendido un vector de dirección x(i).



Así que, cuando conocen ciertas variables realmente no saben por adelantado cuáles van a ser las diferentes variables, pero si ejecutan el algoritmo y, de manera empírica, las variables tenderán a captar cuáles son los aspectos importantes de estas diferentes películas o diferentes productos o lo que sea. Cuáles son los aspectos importantes que provocan que a algunos usuarios les gusten ciertas películas y que hacen que a algunos usuarios les gusten diferentes grupos de películas. Así que tal vez terminen conociendo una variable, ya saben, en donde x1 es igual a romance, x2 es igual a acción -similar a un video anterior- y tal vez conocieron una variable diferente x3 que se trata de una comedia, después alguna variable x4 que es alguna otra cosa. Y tienen N variables en conjunto, y después de que hayan aprendido variables es en realidad bastante difícil entender las variables aprendidas y llegar a una interpretación humana comprensible de lo que estas variables son en realidad. Pero en la práctica, las variables, aunque esas variables sean difíciles de visualizar, --puede ser difícil de averiguar exactamente que son estas variables-- por lo general, aprenderemos variables que son muy significativas para capturar las que son las propiedades más importantes o más sobresalientes de una película que hace que te guste o te disguste.

Recomendación de productos

Así que ahora digamos que deseamos abordar el siguiente problema. Digamos que tienen alguna película específica i y desean encontrar otras películas j que están relacionados con esa película.



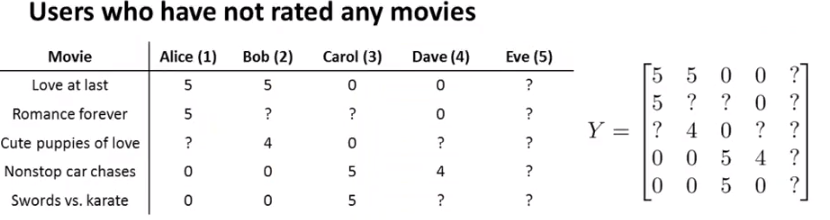
Así es que, ¿por qué querrían hacer esto? Bien, tal vez tienen un usuario que está navegando por las películas, y que actualmente están viendo la película j, entonces ¿cuál es una película razonable para recomendarle que vea después de que haya terminado con la película j? O si alguien ha adquirido recientemente la película j, bueno, ¿cuál sería una película diferente que sería razonable recomendarles para que consideren la compra?.   
  
Así que ahora que han aprendido estos vectores de dirección, esto nos da una forma muy conveniente para medir qué tan parecidas son dos películas. En particular, la película i tiene un vector de dirección x(i) así que si pueden encontrar una película diferente, j, de modo que la distancia entre x(i) y x(j) es pequeña,



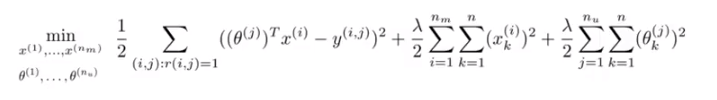
entonces esta es una indicación bastante fuerte de que las películas j e i son de alguna manera similares. Al menos en el sentido de que a algunos de ellos les gusta la película i, tal vez sea más probable que les guste la película j también. Solo por recapitular, si su usuario está viendo alguna película i, y si quieren encontrar las 5 películas más parecidas a esa película a fin de recomendarles 5 películas nuevas, lo que hacen es encontrar las cinco películas j, con la distancia más pequeña entre las variables entre estas películas diferentes. Y esto les podría dar algunas películas diferentes para recomendar a su usuario. Así que con eso, con suerte, ahora saben cómo utilizar una aplicación vectorizada para calcular todas las calificaciones predichas de todos los usuarios y todas las películas, y también cómo hacer cosas, como usar las variables aprendidas para encontrar lo que pudieran ser ya sea películas ó productos que están relacionados entre sí.

# Detalle de implementación: Normalización de la media

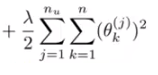
Para ahora, ya han visto todas las piezas principales del algoritmo del sistema de recomendación o el algoritmo de filtrado colaborativo. En este video quiero compartir un último detalle de implementación, la normalización media, que a veces puede simplemente hacer que el algoritmo funcione un poco mejor.   
  
Para motivar la idea de la normalización de la media, vamos a considerar un ejemplo en el que hay un usuario que no ha calificado ninguna película. Así que, además de nuestros cuatro usuarios, Alice, Bob, Carol, y Dave, he agregado un quinto usuario, Eve, que no ha calificado ninguna película.



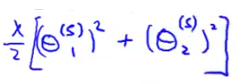
Veamos lo que nuestro algoritmo de filtrado colaborativo hará con este usuario.



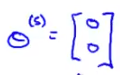
Digamos que n es igual a 2, así que vamos a aprender dos variables y tendremos que conocer un vector de parámetros teta5, que estará en R2 (recuerden que estos son ahora vectores en Rn, no Rn+1). Vamos a conocer el vector de parámetro theta5 para nuestro usuario número 5, Eve.   
  
De modo que si vemos el primer término en este objetivo de optimización, bueno, el usuario Eve no ha calificado ninguna película, por lo que no hay películas para las que r(i,j) sea igual a 1 para el usuario Eve, así que este primer término no desempeña papel alguno en la determinación de «theta» 5 porque no hay películas que haya calificado Eve.   
  
Así que el único término que afecta «theta» 5 es este término.



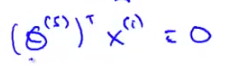
Por tanto, lo que estamos diciendo es que deseamos elegir el vector «theta» 5 de manera que el último término de regularización sea lo más pequeño posible. En otras palabras, si desarrollamos el sumatorio, queremos minimizar:



De modo que ese es el componente del término de regularización que corresponde al usuario 5, y por supuesto, si su objetivo es minimizar este término, entonces con lo que van a terminar es sólo con «theta» 5 es igual a [0 0],

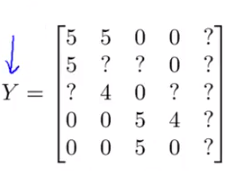


Porque el término de regularización nos anima a establecer parámetros cercanos a 0, y si no hay datos para tratar de alejar los parámetros de 0, debido a que este primer término no afecta «theta» 5, sólo terminamos con «theta» 5 es igual al vector con todos los términos igual a cero. Así que cuando vamos a predecir cómo va a calificar el usuario 5 alguna película, tenemos que «theta» 5 transpuesta multiplicada por 5,

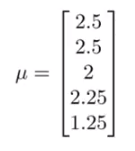


Para cualquier i eso sólo va a ser igual a cero porque «theta» 5 es 0 para cualquier valor de x, este producto interior va a ser igual a 0. Y lo que vamos a tener, por lo tanto, es que vamos a predecir que Eve va a calificar cada una de la películas con cero estrellas.

Pero esto no parece muy útil ¿verdad? Quiero decir, si observamos las diferentes películas, Love at Last, esta primera película, un par de personas la calificaron con 5 estrellas. Y hasta para Swords vs. Karate, alguien la calificó con 5 estrellas. Así que a algunas personas sí les gustan algunas películas. No parece útil sólo predecir que Eve va a calificar todo con 0 estrellas. Y de hecho, si estamos prediciendo que Eve va a calificar todo con 0 estrellas. tampoco tenemos ninguna manera adecuada para recomendarle ninguna película porque, ya saben, todas estas películas están obteniendo exactamente la misma calificación predicha para Eve, así que no hay una película con una calificación predicha más alta que le podamos recomendar a ella, así que eso no es muy bueno.   
  
La idea de la normalización de la media nos permitirá solucionar este problema. Así que aquí está cómo funciona.   
  
Como antes, voy a agrupar todas mis calificaciones de las películas en esta matriz «Y»,

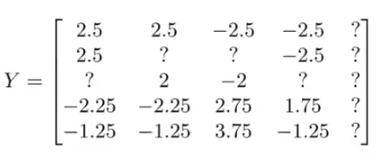


Así que sólo tomamos todas estas calificaciones y las agrupamos en la matriz «Y». Y esta columna por aquí de todos los signos de interrogación, corresponde al hecho de que Eve no ha calificado ninguna película.   
  
Ahora bien, para llevar a cabo la normalización de la media, lo que voy a hacer es calcular la calificación promedio que obtuvo cada película, y voy a almacenar eso en un vector que llamaremos «mu».

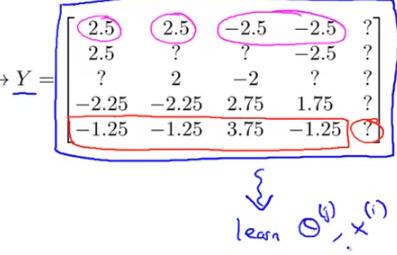


Así que la primera película consiguió dos de 5 estrellas y dos calificaciones de 0 estrellas, de manera que esa es una calificación de 2.5 estrellas. La segunda película tuvo un promedio de 2.5 estrellas, y así sucesivamente. Y la última película tiene 0, 0, 5, 0, y el promedio de 0, 0, eso promedia a una calificación promedio de 1.25. Y lo que voy a hacer, es observar todas las calificaciones de las películas y voy a restar la calificación media.

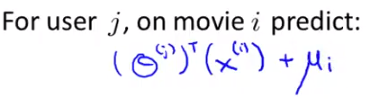
De modo que de este primer elemento 5 voy a restar 2.5 y eso me da 2.5. Y al segundo elemento 5 le resto 2.5, obtengo un 2.5. Y después el 0, 0, le resto 2.5 y obtengo -2.5, -2.5. En otras palabras, lo que voy a hacer es tomar mi matriz de calificaciones de películas, tomar esta matriz «Y» y restar de cada fila la calificación promedio para esa película.



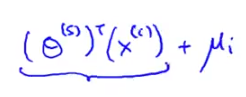
Así que lo que estoy haciendo es simplemente normalizar cada película para tener una calificación promedio de cero.Desde luego, los signos de interrogación se quedan como signos de interrogación. De manera que cada película en esta nueva matriz tiene una calificación con media.   
  
Lo que voy a hacer entonces es tomar este grupo de calificaciones y usarlo con el algoritmo de filtrado colaborativo. Así que voy a pretender que estos eran los datos que había obtenido de mis usuarios, o pretender que éstas son las calificaciones reales que había obtenido de los usuarios, y voy a usar esto como mi conjunto de datos para conocer mis parámetros «theta» j y mis variables xi a partir de estas calificaciones de las películas normalizadas de la media.



Cuando quiero hacer predicciones de las calificaciones de las películas, lo que voy a hacer es lo siguiente: para el usuario j en la película i voy a predecir «theta» j transpone xi, en donde x y «theta» son los parámetros que hemos aprendido de este grupo de datos normalizados de la media. Pero, debido a que en el conjunto de datos resté los promedios a fin de hacer una predicción sobre la película «i», voy a tener que agregar de nuevo la media, así que la voy a añadir de nuevo en mu i. Así que esa va a ser mi predicción,



En mis datos de entrenamiento resté todas las medias así que cuando hacemos predicciones, necesitamos agregar nuevamente estas medias «mu i» para la película «i». Así que específicamente, si su usuario 5, que es Eve -todavía aplica el mismo argumento que en la diapositiva anterior, en el sentido de que Eve no había calificado ninguna película- así que el parámetro aprendido para el usuario 5 todavía va a ser igual a 0, 0. Así que lo que vamos a obtener es que en una película «i» en particular, vamos a predecir para Eve «theta» 5 transpone xi, más, agregar de nuevo «mu i», y este primer componente va a ser igual a cero, si «theta» cinco es igual a cero. Así que en la película «i» vamos a terminar prediciendo «mu i».



Y esto realmente tiene sentido. Significa que en la película 1 vamos a predecir que Eve la califica con 2.5. En la película 2, vamos a predecir que Eve la califica con 2.5. En la película 3, vamos a predecir que Eve la califica con 2., y así sucesivamente. Esto realmente tiene sentido, porque nos dice que si Eve no ha calificado ninguna película y simplemente no sabemos nada acerca de este nuevo usuario Eve; lo que vamos a hacer es sólo predecir para cada una de las películas, que son las calificaciones promedio que obtuvieron esas películas.   
  
Por último, como una nota aparte, en este vídeo hablamos de la normalización de la media, en donde normalizamos cada fila de la matriz «y», para obtener el promedio 0. En caso de que tengan algunas películas sin calificaciones, de modo que sea análogo a un usuario que no ha calificado nada, pero en caso de que tengan algunas películas sin ninguna calificación, también pueden jugar con versiones del algoritmo, en donde normalizan las diferentes columnas para tener media de 0, en lugar de normalizar las filas para tener media de cero, aunque eso es probablemente menos importante, ya que si realmente tienen una película sin calificación, tal vez simplemente no deberían recomendar esa película a nadie, de todas formas. Así que, prestar atención al caso de un usuario que no ha calificado nada, podría ser más importante que cuidar del caso de una película que no ha obtenido una sola clasificación.   
Así que, para resumir, así es como pueden hacer la normalización de la media, como una especie de etapa de pre-procesamiento para el filtrado colaborativo. Dependiendo de su conjunto de datos, esto pudiera hacer que, algunas veces, su implementación funcione un poquito mejor.