

Matrix Factorization

Otakus v.s. No. of Figures

A	
B	
C	
D	
E	

There are some common *factors* behind otakus and characters.

<http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/>

r^i		r^j	r^1	r^2	r^3	r^4
						
r^A	A		5	3	?	1
r^B	B		4	3	?	1
r^C	C		1	1	?	5
r^D	D		1	?	4	4
r^E	E		?	1	5	4

$$r^A \cdot r^1 \approx 5$$

$$r^B \cdot r^1 \approx 4$$

$$r^C \cdot r^1 \approx 1$$

⋮

Minimizing

$$L = \sum_{(i,j)} (r^i \cdot r^j - n_{ij})^2$$

Find r^i and r^j by gradient descent

Only considering the defined value

Matrix Factorization：有兩種東西，有兩種 object，它們之間是受到某種共通的 latent factor 去操控的。

舉例

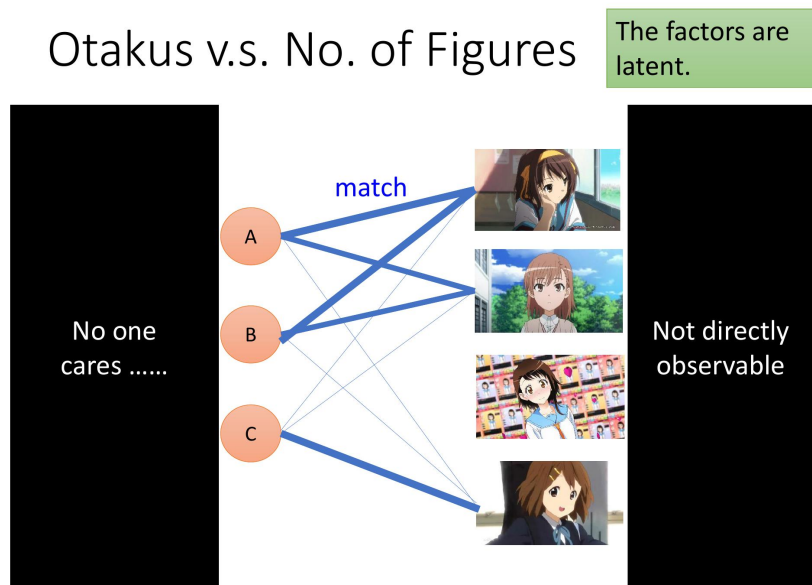
A B C D E 代表 5 個人，調查一下，每個人手上有的公仔數目。

A 有 5 個涼宮春日的公仔; B 有 4個; C 有 1 個; D有 1 個。然後這個是御坂美琴，A有 3 個，B有 3 個，C有 1 個.....

注意：在這個 matrix 上面，每一個 table 裡面的 block並不是隨機出現。

如果有買涼宮春日公仔的人，就比較有可能會有御坂美琴的公仔，有這個姐寺公仔的人，就比較有可能有小唯的公仔。

因為每一個人跟每一個人跟角色背後是有一些共同的特性，有一些共同的 factor 來操控這些事情發生。



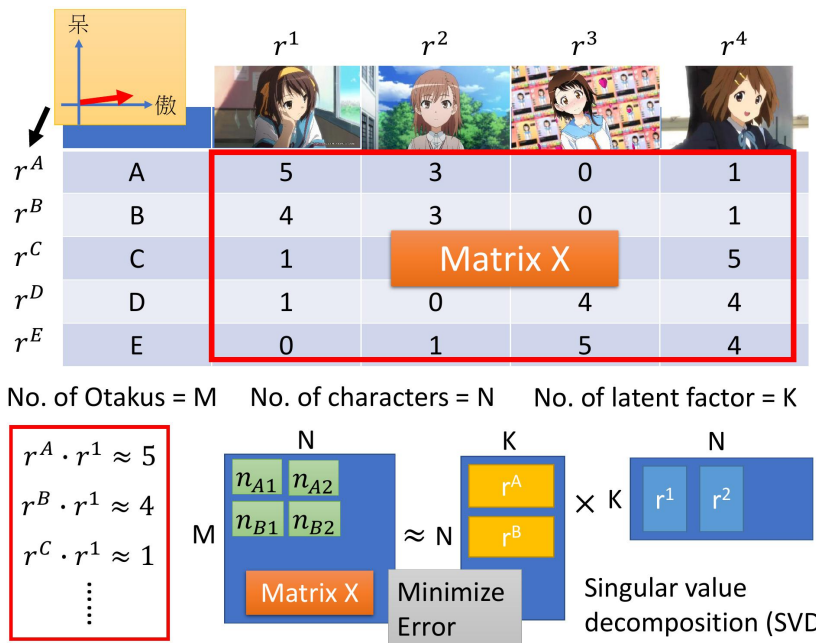
如果將動漫宅分成兩種，有一種是萌傲嬌的，有一種是萌天然呆的，就可以假設成平面上的點。舉例來說如果比較偏A，就是比較萌傲嬌的類型。

每一個角色，可能是有傲嬌屬性，或者是有天然呆的屬性，所以每一個角色，也都是平面上的一個點，並且都可以用一個 vector 來描述它。

如果某一個人萌的屬性，跟某一個角色，他本身所具有的屬性是 match 的話，背後的 vector 在做 inner product 的時候值很大。那這個人就可能會買很多涼宮春日的公仔。

所以他們這個匹配的程度就取決於他們背後的這個的 latent factor 是不是匹配的。

如果他們的屬性和它們背後的屬性 match 的話，那這一個人，就會買這個公仔。



目前有的是，這個動漫人物跟阿宅中間的關係，也就是他手上有的公仔的數目，我們要憑著這個關係去推論出每一個人，跟每一個動漫人物他們背後的 latent factor。

- 設每一個人背後都有一個二維的 vector 分別代表他萌傲嬌或是萌天然呆的程度，都用一個 vector 來表示。
- 設每一個人物，他背後也都有一個 vector，代表他傲嬌的屬性和天然呆的屬性。
- 則每一個阿宅手上的這個公仔的數目就可以看作是一個 matrix, X 。

(那這一個 matrix, X ，它的 row 的數目就是宅宅的數目，它的 column 的數目，就是動漫角色的數目)

Matrix Factorization的假設

每一個這個 matrix 裡面的 element，都來自於兩個 vector 的 inner product。

(Ex. A 會有 5 個涼宮春日的公仔，是因為 r^A 跟 r^1 的 inner product 很大，它們的 inner product 是 5。所以他就會買，他們就會有 5 個涼宮春日的公仔; 如果 r^B 跟 r^1 ，它的 inner product 是 4 的話，那 B 就會有 4 個涼宮春日的公仔，以此類推.....)

如果用數學式來表示它的話：

- 可以把 r^A 到 r^E 排成一排，把 r^A 到 r^E 排成一排，把 r^1 到 r^4 也當作是另外一個 matrix 的 row 把它排起來。
- K 是 latent factor 的數目。

(實際上要有多少 factor 這件事情，必須要試出來，就像 principal component 的數目，或者是這個 neural network 的陳述一樣。把人分成只有萌傲嬌和萌天然呆，是一個不精確的分析方式，如果有更多的 data 的話應該可以更準確知道應該要有多少 factor)

現在假設 latent factor 的數目就是 K

因此 r^A 到 r^E 當作是 matrix row，這邊就有一個 $N \times K$ 的 matrix，把 r^1 到 r^4 當作 column，那就是 $K \times N$ 的 matrix。把這個 $N \times K$ matrix、 $K \times N$ matrix 乘起來就會得到一個 $M \times N$ 的 matrix。

那我們要做的就是找一組 r^A 到 r^E ，找一組 r^1 到 r^4 把這兩個 matrix 相乘以後跟這個 matrix, X 越接近越好。

也就是minimize 這兩個 matrix 相乘以後跟這個 matrix, X 之間的 Reconstruction error。

這個東西就可以用 SVD 來解。

What about missing value?

r^i	r^j	r^1	r^2	r^3	r^4
r^A	A	5	3	?	1
r^B	B	4	3	?	1
r^C	C	1	1	?	5
r^D	D	1	?	4	4
r^E	E	?	1	5	4

$r^A \cdot r^1 \approx 5$
 $r^B \cdot r^1 \approx 4$
 $r^C \cdot r^1 \approx 1$
 \vdots

Minimizing

Only considering the defined value

$$L = \sum_{(i,j)} (r^i \cdot r^j - n_{ij})^2$$

Find r^i and r^j by gradient descent

有時候可能會遇到這個問題就是有一些 information，是 missing，是不知道的。

比如說你並不知道 ABC 手上有沒有小野寺的公仔（有可能就只是在他所在的地區，沒有發行這個公仔而已）。也就是假設這個 table 上有一些問號的話，怎麼辦呢？

則用剛才那個 SVD 的方法，就會有點問題。


所以，如果在 matrix 上面有一些 missing value 的話，就用 Gradient Descent 的方法來做。

就寫一個 loss function，要讓 r^i (指的是每一個 Otaku 背後的 latent factor)、 r^j (指的是每一個動漫角色背後的 latent factor)，要讓 i 這個人，他的 latent factor 乘上 j 這個角色的 inner product 跟他購買的數量， n_{ij} 越接近越好。

在 summation over 這些 element 的時候，可以避開這一些 missing 的 data，如果今天這個值是沒有的就不算它，只算有定義的部分

接下來，就用 Gradient Descent 運算即可。

Gradient Descent Example

		r^1	r^2	r^3	r^4
					
r^A	A	5	3	-0.4	1
r^B	B	4	3	-0.3	1
r^C	C	1	1	2.2	5
r^D	D	1	0.6	4	4
r^E	E	0.1	1	5	4

Assume the dimensions of r are all 2 (there are two factors)

A	0.2	2.1
B	0.2	1.8
C	1.3	0.7
D	1.9	0.2
E	2.2	0.0

1 (春日)	0.0	2.2
2 (炮姐)	0.1	1.5
3 (姐寺)	1.9	-0.3
4 (小唯)	2.2	0.5

根據例子，就可以實際地用 Gradient Descent 來做。

假設 latent factor 的數目等於 2，那每一個這個 A 到 E 都會對應到一個二維的 vector，每一個這個角色也都可以對應到一個 vector。每一個角色，也都得到一個 latent factor，代表他的屬性，這個每一個人的 latent factor 就代表他萌哪一種屬性。

所以，如果把每個人在兩個維度的結果裡面，比較大的維度挑出來的話，就會發現，A 跟 B 是萌同一組屬性的，C，D，E 是萌同一組屬性的。

接下來，有這些 data 以後，就可以預測 missing value。

(已知 r^3 ，已知道 r^a ，購買的數量就是這個動漫角色背後的 latent factor 那個人背後的 latent factor 做 inner product 的結果。所以，只要把 r^3 跟 r^a 做 inner product，就可以預測這個人會買多少的特定公仔。)

這個方法常用在這種推薦系統裡面，如果今天把動漫角色換成電影，把中間的數值換成 rating 的話，就可以預測某一個人會不會喜歡某一部電影。線上推薦系統，很多都會使用這樣的技術。

More about Matrix Factorization

More about Matrix Factorization

- Considering the individual characteristics

$$r^A \cdot r^1 \approx 5 \quad \longrightarrow \quad r^A \cdot r^1 + b_A + b_1 \approx 5$$

b_A : otaku A likes to buy figures

b_1 : how popular character 1 is

$$\text{Minimizing } L = \sum_{(i,j)} (r^i \cdot r^j + b_i + b_j - n_{ij})^2$$

Find r^i, r^j, b_i, b_j by gradient descent (can add regularization)

- Ref: Matrix Factorization Techniques For Recommender Systems

這個 model 可以做更精緻。

A 背後的 latent factor，乘上 1 背後的 latent factor 得到的結果，就是 table 上面的數值。但事實上，可能還有別的因素會操控他的數值。

更精確的寫法其實是 r^A 跟 r^1 inner product 加上某一個跟 A 有關的 scalar 再加上某一個跟 1 有關的 scalar 才等於 5。

這跟 A 有關的 scalar, $b(\text{下標A})$ 指的是甚麼呢？

它代表的是 A 他有多喜歡買公仔，所以這個 $b(\text{下標A})$ ，代表他本身有多想要買公仔本身有多想要買公仔；

這個 b_1 代表這個角色，他本質上會有多想讓人家購買，這件事情是跟屬性無關的，就是它本來就會吸引各種人的部分。（比如說，最近涼宮春日慶祝動畫十周年，就出現藍光 DVD，所以大家就可能會比較想要買，涼宮春日的公仔）

因此更改一下 minimize 的式子：

把 r^i 跟 r^j 的 inner product 加上 b_i 再加上 b_j ，同樣希望這個值跟 $n(\text{下標ij})$ 越接近越好。

且一樣可以用 Gradient Descent 硬解，甚至可以在 r^i, r^j, b_i, b_j 上面加上 Regularization。

Other Matrix Factorization Applications

Matrix Factorization for Topic analysis

character→document,
otakus→word

- Latent semantic analysis (LSA)

	Doc 1	Doc 2	Doc 3	Doc 4
投資	5	3	0	1
股票	4	0	0	1
總統	1	1	0	5
選舉	1	0	0	4
立委	0	1	5	4

Number in Table:

Term frequency
(weighted by inverse
document frequency)

Latent factors are topics
(財經、政治)

- Probability latent semantic analysis (PLSA)

- Thomas Hofmann, Probabilistic Latent Semantic Indexing, SIGIR, 1999

- latent Dirichlet allocation (LDA)

- David M. Blei, Andrew Y. Ng, Michael I. Jordan, Latent Dirichlet Allocation, Journal of Machine Learning Research, 2003

Matrix Factorization 還有很多其他的應用，比如說Topic analysis。

如果是把剛才所講的 Matrix Factorization 的方式用在 Topic 的analysis 上面，也叫做 Latent semantic analysis (LSA)。

它的技術基本上是一模一樣的，只是把剛才的動漫人物，通通換成文章，把剛才的人都換成詞彙。

那 table 裡面的值就是 Term frequency，也就是說投資這個 word 在 Doc 1 出現 5 次; 股票這個 word 在 Doc1 出現 4 次，以此類推.....

有時候不只會用 Term frequency，會再乘上一個 weight，代表說這個 term 本身有多重要。

如果把一個 term 乘上比較大的 weight 的話，在做 matrix factorization 的時候，那一個 term 它就比較會被考慮到，就比較想要讓那個 term 的 Reconstruction error 比較小。

怎麼 evaluate 一個 term 重不重要呢？

有很多方法，常用的就是 inverse document frequency。

inverse document frequency 簡單來講，就是計算某一個詞彙，在整個 corpus 裡面，有多少比例的 document 有涵蓋這個詞彙。

比如說，"的" 每一個 document 都有，則它的 inverse document frequency 就很小，代表說這個詞彙的重要性是低的。

Topic Analysis latent factor 是什麼呢?

這個 task 裡面，你的 latent factor 指的是topic，指的是某一個 document，它背後要談的主題。

其實Topic analysis 的方法多如牛毛，但基本的精神是差不多的還有很多各種各樣的變化，常見的是 Probability latent semantic analysis (PLSA)，另外一個 latent Dirichlet allocation (LDA)。

臺灣大學人工智慧中心

科技部人工智慧技術暨全幅健康照護聯合研究中心

<http://aintu.tw>

