Deep Neural Network & Recommendation Engine on TensorFlow

Gary Chen

Outline

- Recommedation Engine in Matrix Factorization
- Variant of Matrix Factorization

Outline

- Recommedation Engine in Matrix Factorization
- Variant of Matrix Factorization

Recommendation Engine in Matrix Factorization

Movielens Data

User Movie Rating

	userId	movieId	rating	timestamp
0	0	30	2.5	1260759144
1	0	833	3.0	1260759179
2	0	859	3.0	1260759182
3	0	906	2.0	1260759185
4	0	931	4.0	1260759205

User Movie Tag

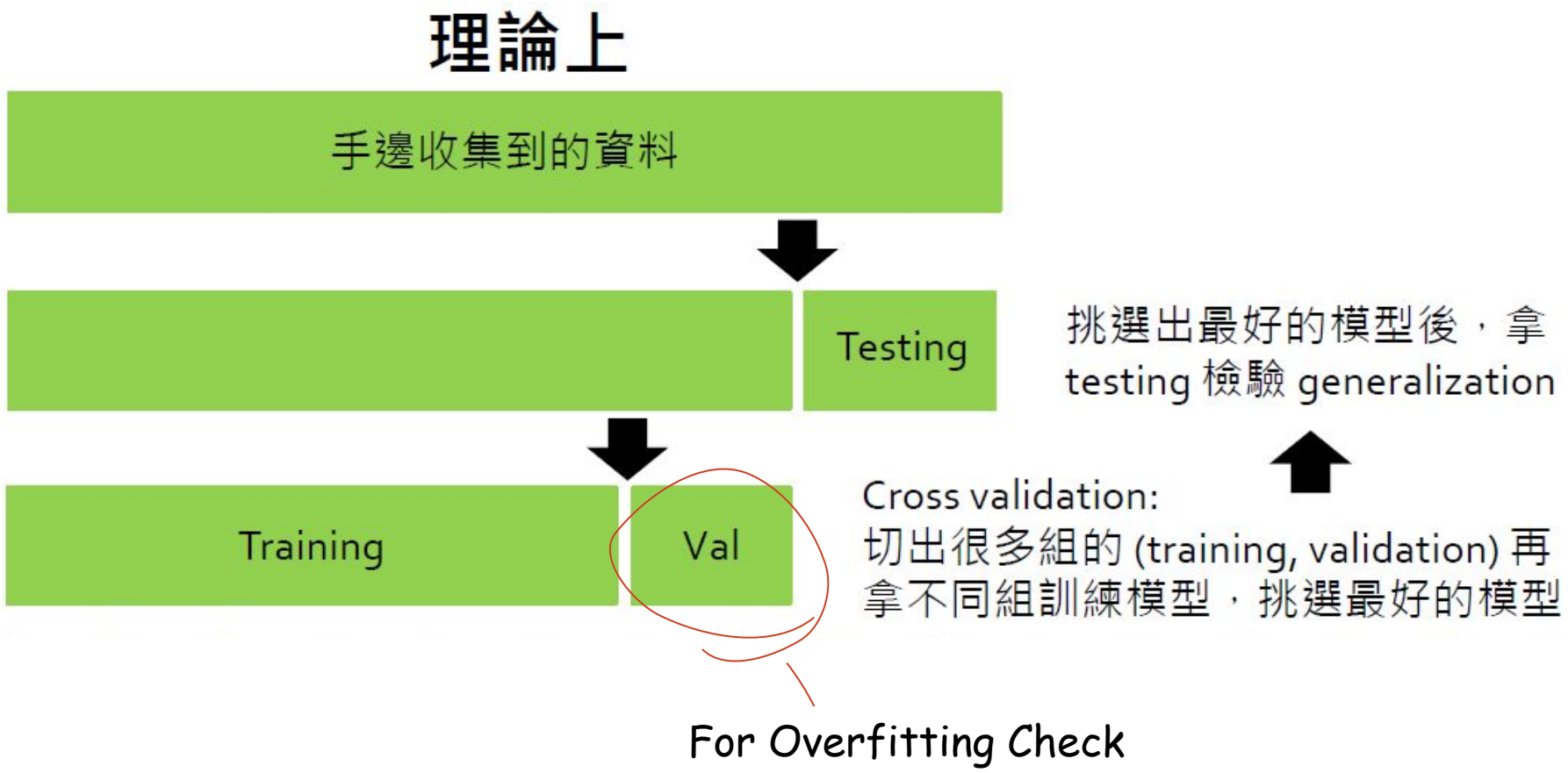
	userId	movieId	tag	timestamp
0	14	304	sandra 'boring' bullock	1138537770
1	14	1517	dentist	1193435061
2	14	5166	Cambodia	1170560997
3	14	6118	Russian	1170626366
4	14	6178	forgettable	1141391765

Movie Metadata

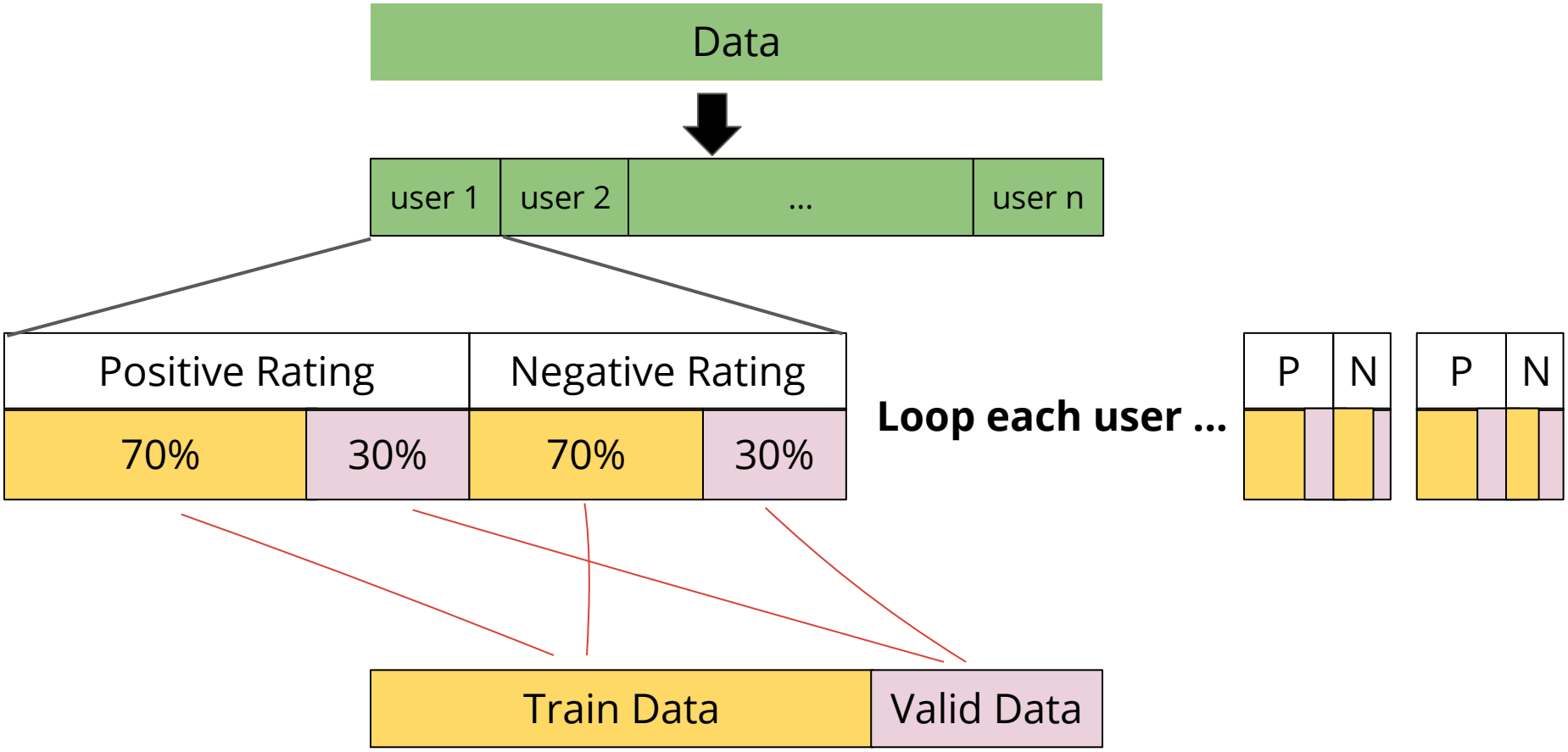
	movieId	title	genres
0	0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	1	Jumanji (1995)	Adventure|Children|Fantasy
2	2	Grumpier Old Men (1995)	Comedy|Romance
3	3	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	4	Father of the Bride Part II (1995)	Comedy

How to Treat Your Data?

- Split **train, valid, test** data



How Did We Split?



Concept of Latent Factor

U: User, M: Movie, R: Rating(1 ~ 5)

	M1	M2	M3	M4	M5
U1	3	5			1
U2			2	4	
U3	5		1		
U4		3		3	3

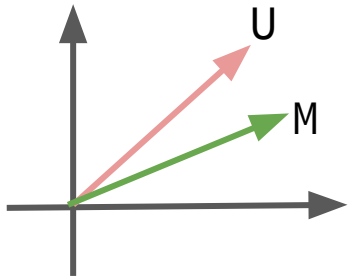
≈

U1	
U2	
U3	
U4	

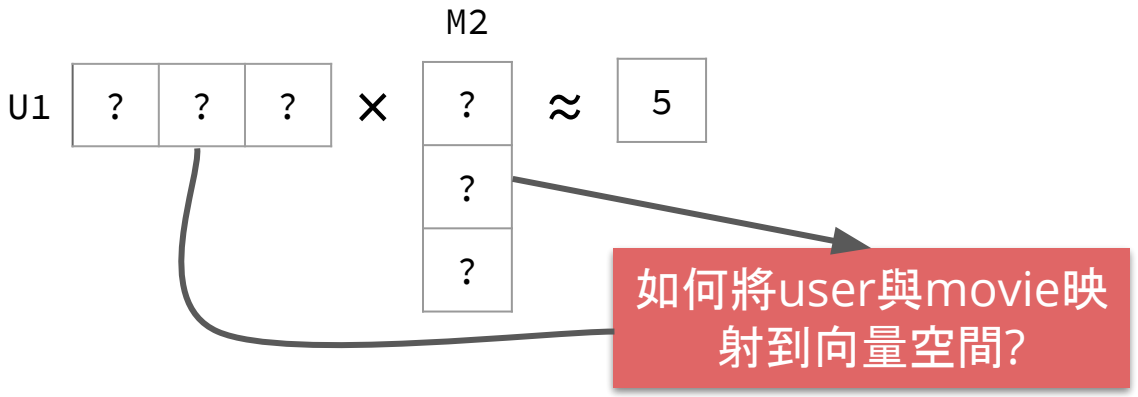
X

M1	M2	M3	M4	M5

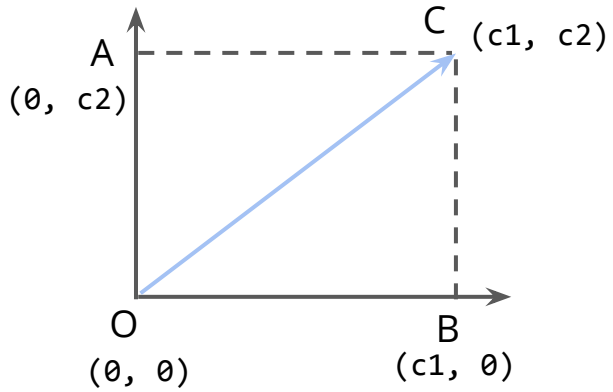
latent factor



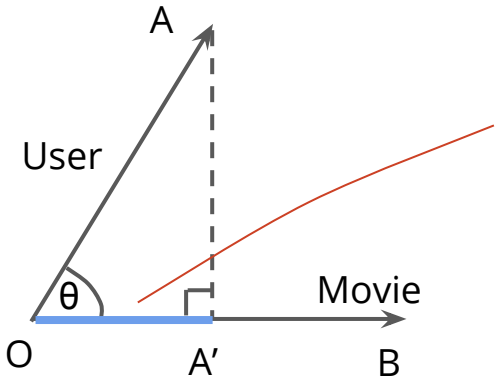
User與Movie的距離



The Vector and Inner Dot



- ➔ 具有**大小(長度)**和**方向**的意義
- ➔ 長度 ⇒ ||C|| = 和原點的距離, 通常使用歐幾里得定義(L2 norm)
- ➔ 可拆成各分量相加 ⇒ OC = OA + OB



A projection on B ⇒ ‖OA'‖

‖OA'‖ = ‖OA‖·cosθ

= ‖OA‖·(OA·OB)/(‖OA‖·‖OB‖)

= (OA·OB) / ‖OB‖

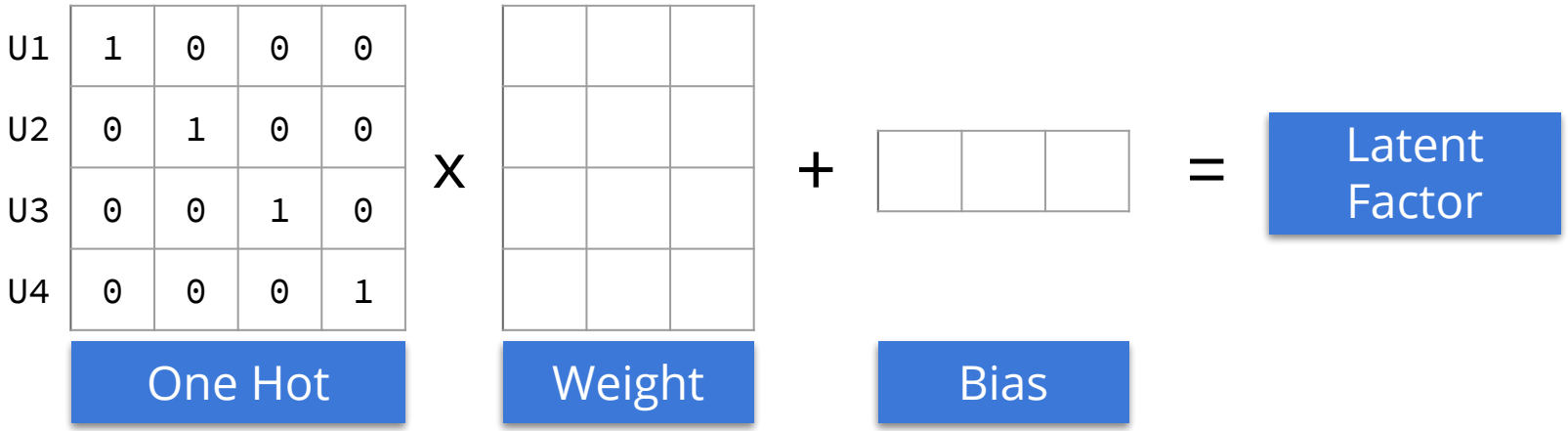
cosθ = (OA·OB)/(‖OA‖·‖OB‖)

|cosθ| <= 1

Reflections(Transformation)

user id ⇒ 1263, 1080, 87...
movie id ⇒ 103, 554, 187...

- 把user和movie看成是categorical variable ⇒ One Hot Encoding
- **Add linear feature transformation (Linear hidden layer)**



Recommendation Engine in Matrix Factorization

User ID to Embedding

U1	1	0	0	0
U2	0	1	0	0
U3	0	0	1	0
U4	0	0	0	1

One Hot

x

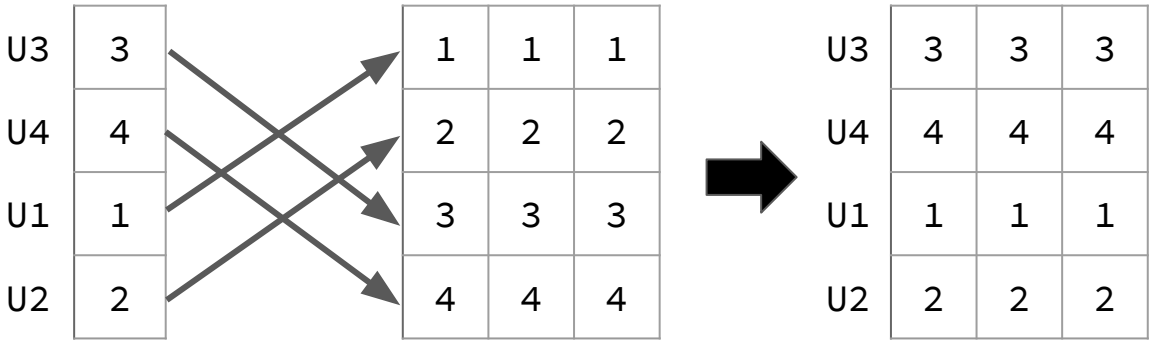
1	1	1
2	2	2
3	3	3
4	4	4

Weights

=

1	1	1
2	2	2
3	3	3
4	4	4

第i號user的latent factor等價於取Weight的第i列 ⇒ **Embedding**



User inputs *Lookup from weights* *User id embedding*

- movie id embedding 也是如法炮製
- 實際都會省略 bias

Model Formulation and Loss Function

U, M ⇒ 透過embedding取得的latent factor

	M1	M2	M3	M4	M5
U1	3	5			1
U2			2	4	
U3	5		1		
U4		3		3	3

$r_{12} \Rightarrow u_1 \cdot m_2 \approx 5$

$r_{33} \Rightarrow u_3 \cdot m_3 \approx 1$

$r_{23} \Rightarrow u_2 \cdot m_3 \approx 2$

$\vdots$

Regression

$$L = \sum_{(i,j)} (u_i \cdot m_j - rij)^2$$

model function

- User rating behavior and movie being rated are highly **biased**
- Recall linear combination ⇒ **wx + b**

$$u_i \cdot m_j + b_u + b_m + b_{global}$$

- 不是U和M的外在影響因素
- Model bias

Minimizing $L = \sum_{(i,j)} (u_i \cdot m_j + b_u + b_m + b_{global} - r_{ij})^2$

model function with bias

Recommendation Engine in Matrix Factorization

Prediction

以向量運算表示 $u_i \cdot m_j + b_u + b_m + b_{global}$

$$Predict =$$

$u_i \cdot m_j$

	M1	M2	M3	M4	M5
U1	3.5	4.7	0.87	1.87	0.8
U2	4.87	5.6	2.1	3.87	4.57
U3	5.87	4.78	1.87	1.2	3.6
U4	1.7	3.2	2.9	2.87	2.6

+

b_u Row Wise

0.1
0.2
0.3
0.4

+

0.5

b_{global} Element Wise

+

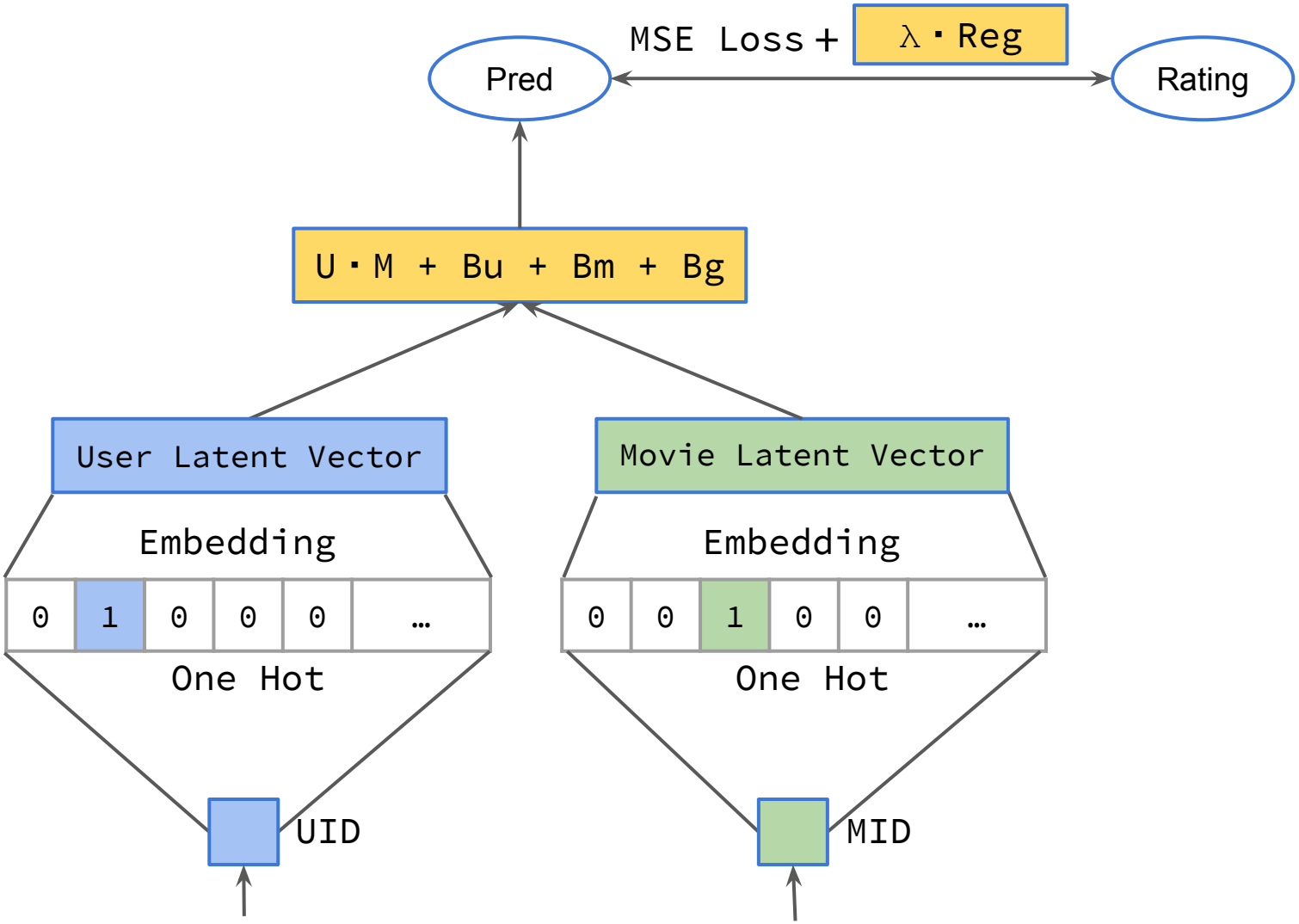
0.8	0.7	0.6	0.5	0.4

b_m Column Wise

```
[[ 4.9   6.    2.07  2.97  1.8 ]
 [ 6.37  7.    3.4   5.07  5.67]
 [ 7.47  6.28  3.27  2.5   4.8 ]
 [ 3.4   4.8   4.4   4.27  3.9 ]]
```

這個例子實際跑出來的

First Model - Basic MF



First Model - Basic MF

- Function set formulation

$$f_{u,m}(x) = u_i \cdot m_j + b_u + b_m + b_{global}$$

- Loss function

$$L = \sum (f_{u,m}(x) - r_{ij})^2 + \lambda \cdot \frac{1}{2}(\sum u^2 + \sum m^2 + \sum b_u^2 + \sum b_m^2)$$

Regularization: L2 Norm

L2 regularization:

$$\|\theta\|_2 = (w_1)^2 + (w_2)^2 + \dots$$

$$L'(\theta) = L(\theta) + \lambda \frac{1}{2}\|\theta\|_2 \quad \text{Gradient:} \quad \frac{\partial L'}{\partial w} = \frac{\partial L}{\partial w} + \lambda w$$

Update:

$$w^{t+1} \rightarrow w^t - \eta \frac{\partial L'}{\partial w} = w^t - \eta\left(\frac{\partial L}{\partial w} + \lambda w^t\right)$$

$$= \underline{(1-\eta\lambda)w^t} - \eta \frac{\partial L}{\partial w}$$

接近0

Weight Decay

Basic MF

filename	lab_reco_model_mf.ipynb
number of users	671
number of movies	9125
rating range	0.5 ~ 5分, ❑ 4分以上算正評 ❑ 未滿4分認為是負評或是不列入推薦名單
neural network	matrix factorization

Model All Metrics

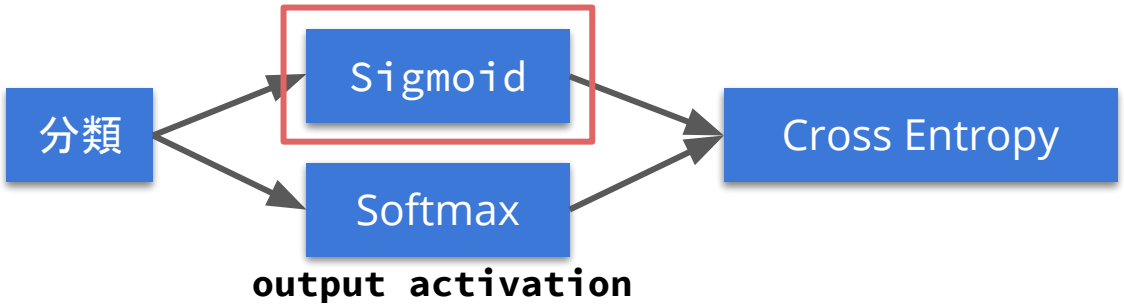
Model	Dummy Model(亂猜)	Basic MF
RMSE Loss	3.23	0.93
ROC AUC	0.50	0.74
NDCG	strict: 0.45 normal: 0.63	strict: 0.63 normal: 0.76
Precision at 10	strict: 0.37 normal: 0.54	strict: 0.50 normal: 0.63

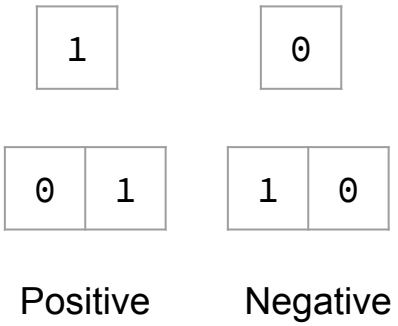
Recommendation Engine in Matrix Factorization

- Movie rating ⇒ explicit feedback
- **通常User不會這麼熱心的對你的商品評價**
  - implicit feedback
    - 商品點閱次數
    - Youtube影片觀看時間
    - ...

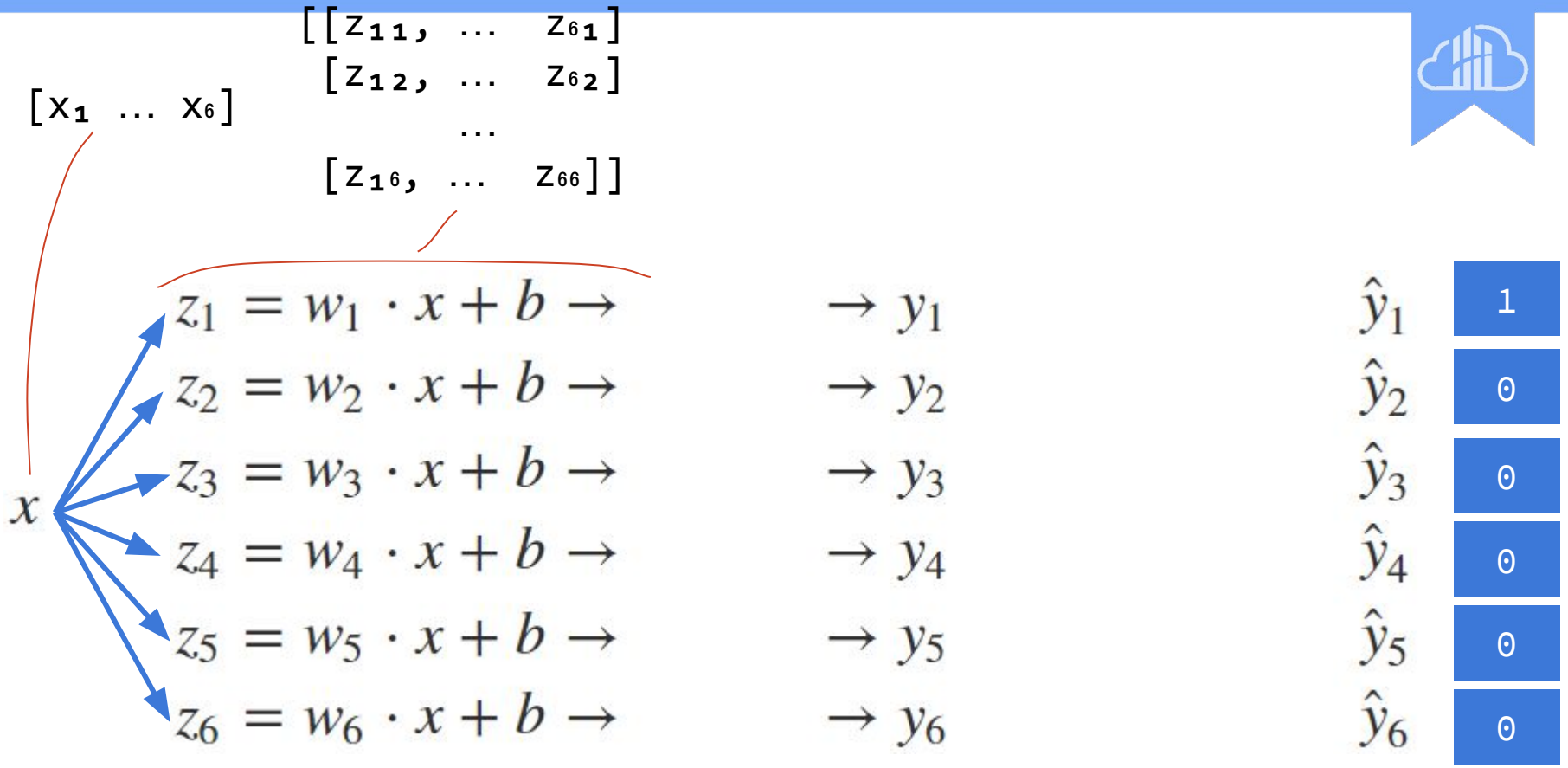
    ⇒ 通常只能得到1(Positive), 0(Negtive)

- **Label 0.5 ~ 5 ⇒ 1 or 0 (這裡4分以上 = 1 others 0)**
  - Regression ⇒ Binary classification
  - P(使用者A 喜歡 電影B) = ??





Machine Learning Framework



Probability:

- $1 > y_i > 0 \quad y_i = P(C_i \mid x)$
- $\sum_i y_i = 1$ → Sigmoid不滿足這個需求!

Recommendation Engine in Matrix Factorization

Activation function: Softmax

$$softmax(z) = \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}}$$

$$z_1 = w_1 \cdot x + b \quad 5$$
$$z_2 = w_2 \cdot x + b \quad 1$$
$$z_3 = w_3 \cdot x + b \quad 3$$
$$z_4 = w_4 \cdot x + b \quad -3$$
$$z_5 = w_5 \cdot x + b \quad -3$$
$$z_6 = w_6 \cdot x + b \quad 0$$

x

0.8	y_1
0.01	y_2
0.11	y_3
0	y_4
0	y_5
0	y_6

Cross Entropy

$$-\sum_{i=1} \hat{y}_i ln y_i$$

類別1

$\hat{y}_1$	1
$\hat{y}_2$	0
$\hat{y}_3$	0
$\hat{y}_4$	0
$\hat{y}_5$	0
$\hat{y}_6$	0

Probability:

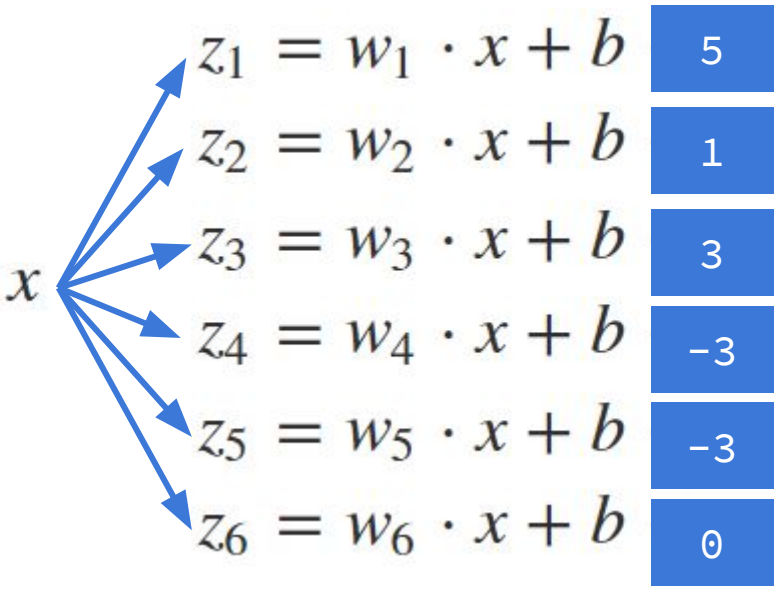
- $1 > y_i > 0 \quad y_i = P(C_i \mid x)$
- $\sum_i y_i = 1$

計算argmax, 最大的數字都在 index 0 ⇒ 命中！

Recommendation Engine in Matrix Factorization

Activation function: Sigmoid

$$\sigma(z) = \frac{1}{1 + exp(-z)}$$



算argmax是沒有意義的! 要考慮的是每個class的分數

Softmax與Sigmoid的差別

$$softmax(z) = \frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}}$$

- ❏ 所有類別視為一個母體
- ❏ Single-label for one record

一筆record一個label

$$\sigma(z) = \frac{1}{1 + exp(-z)}$$

- ❏ 不考慮類別間的關係
- ❏ Multi-label for one record

一筆record可有多個label

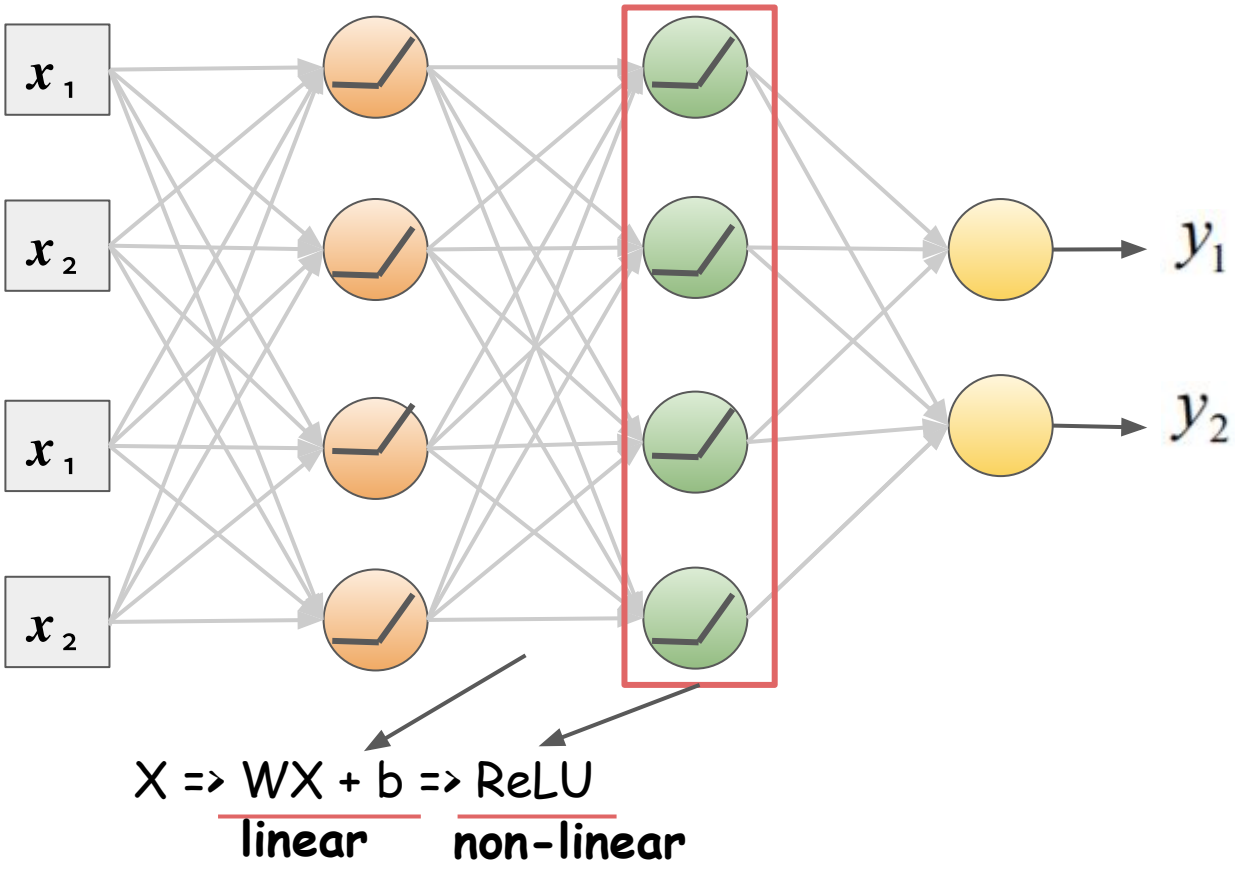
Computes sigmoid cross entropy given `logits`.

Measures the probability error in discrete classification tasks in which each class is independent and not mutually exclusive. For instance, one could perform multilabel classification where a picture can contain both an elephant and a dog at the same time.

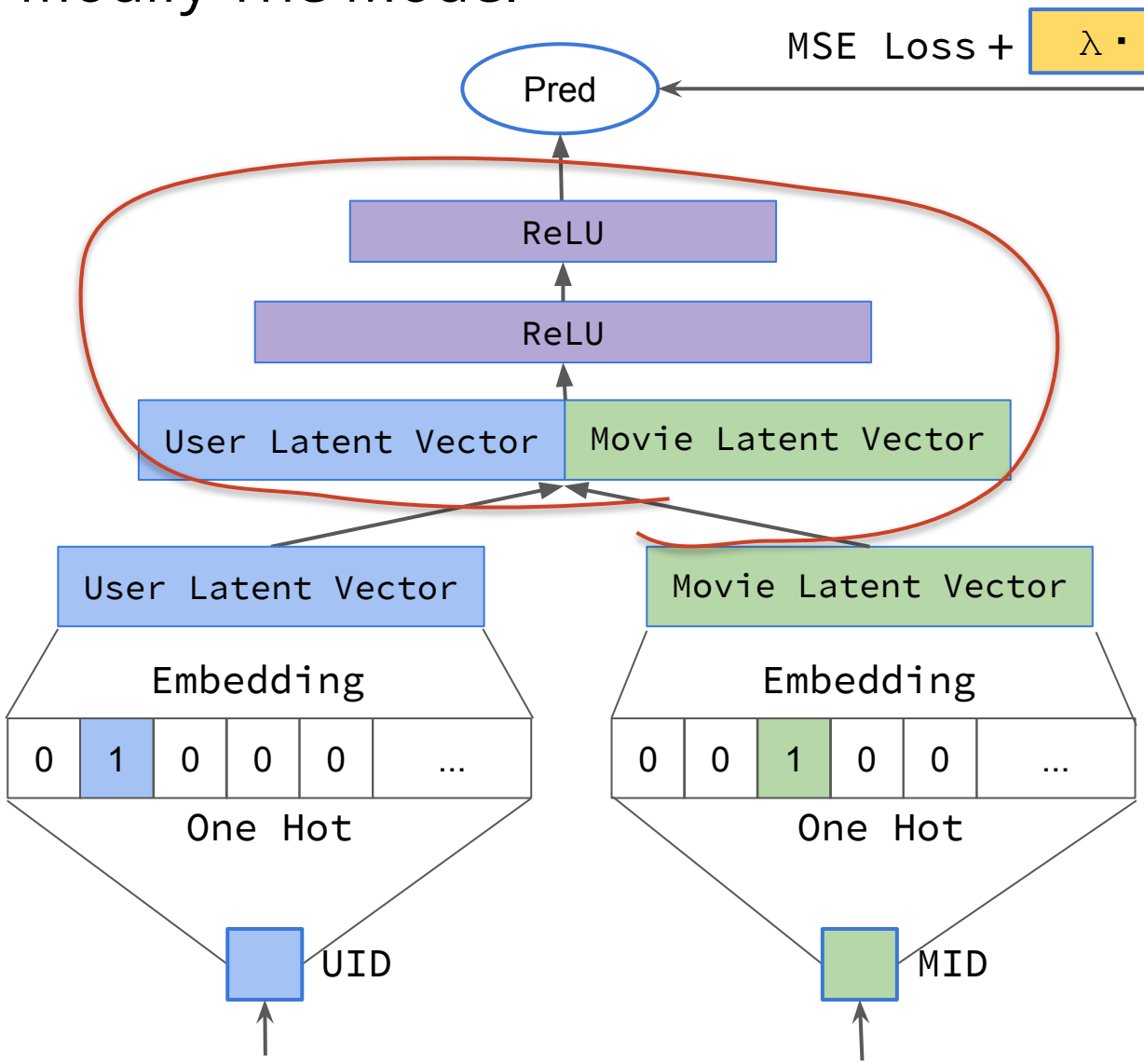
整個Model似乎是線性的...?

$$f_{u,m}(x) = u_i \cdot m_j + b_u + b_m + b_{global}$$

Dense ReLU Layer



Modify The Model



The **non-linear model** maybe got better result ?

Recommendation Engine with DNN

Recommedation Engine in Matrix Factorization

Variant of Matrix Factorization

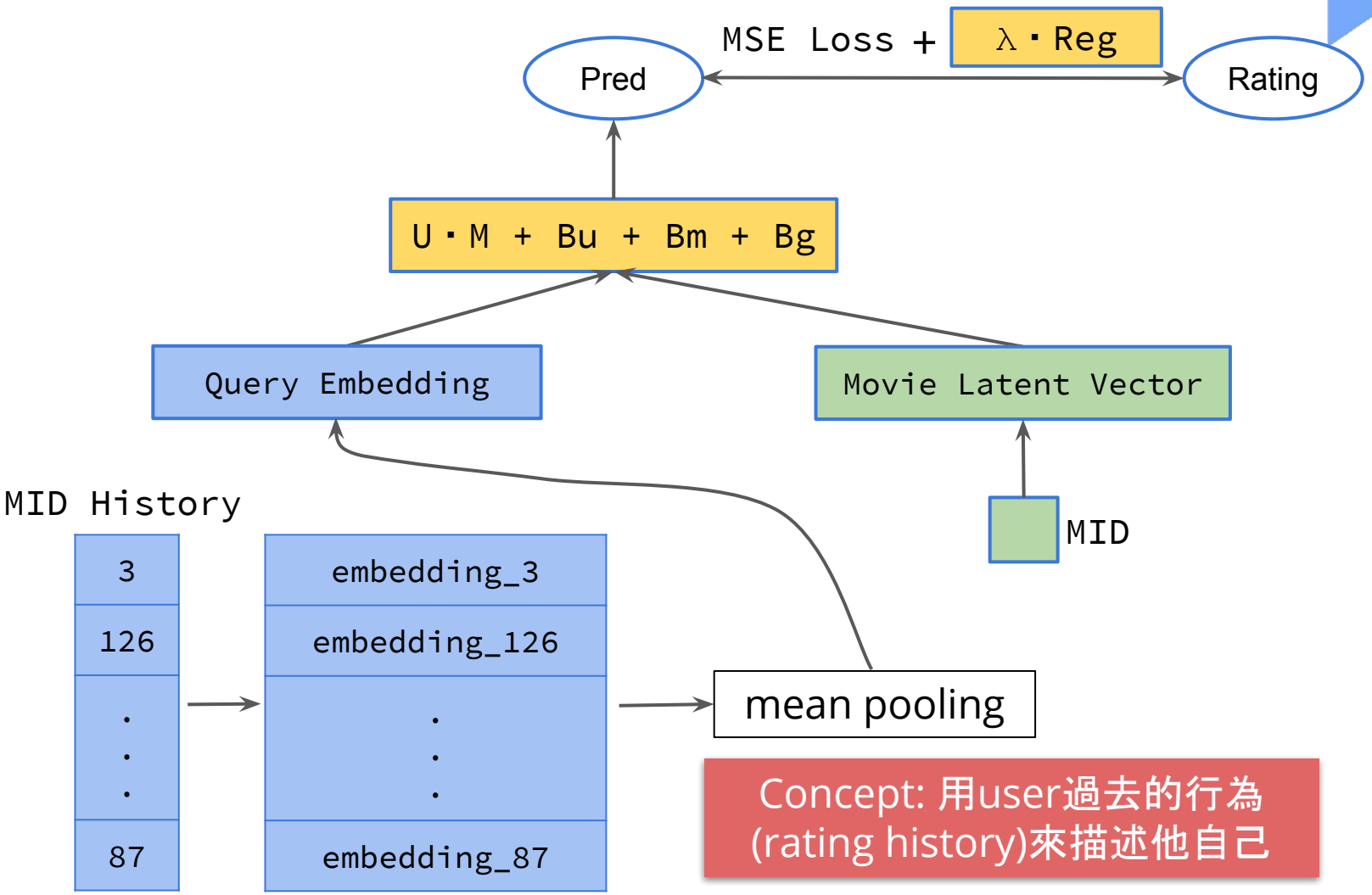
Model - MF with History

- 如果新的User進來...
  - 推薦最受歡迎的電影 (完全沒User rating)
  - 就算累積了Rating history，Model無法使用(不認識新ID)

在不重train model的狀況下，希望能夠對新來的user(有rating history)做推薦

Variant of Matrix Factorization

Model - MF with History

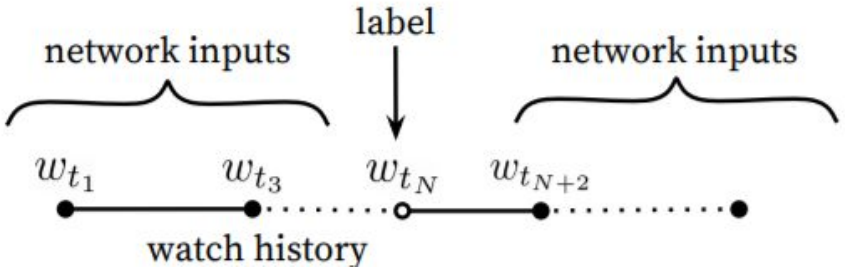


Model - MF with History

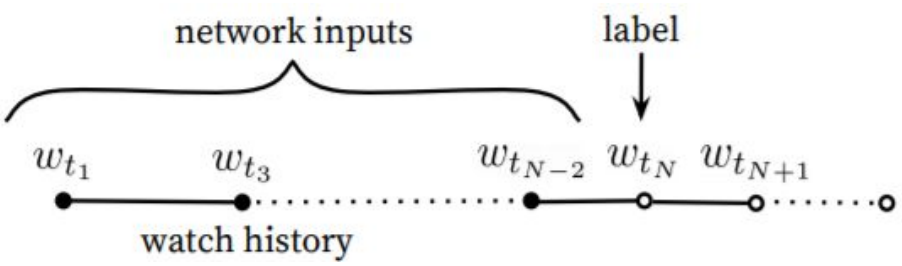
	user_id	query_movie_ids	query_movie_ids_len	candidate_movie_id	rating
0	47	[263, 2860, 3856, 6034, 2409, 2374, 7128, 5339...	357	1393	4.5
1	175	[966, 5026, 4395, 6042, 3871, 6521, 5020, 953,...	177	4002	3.0
2	261	[45, 1104, 154, 4514, 4171, 3801, 3727, 3644, ...	471	3089	2.0

- 觀察47號user
  - query_movie_ids: train data中47號的有rating的movie id，排除1393號
    ⇒ leave one out (這裡的設計也可以只用47號喜歡的電影)

leave one out for predict



leave one out(考慮時間)



Model - MF with History

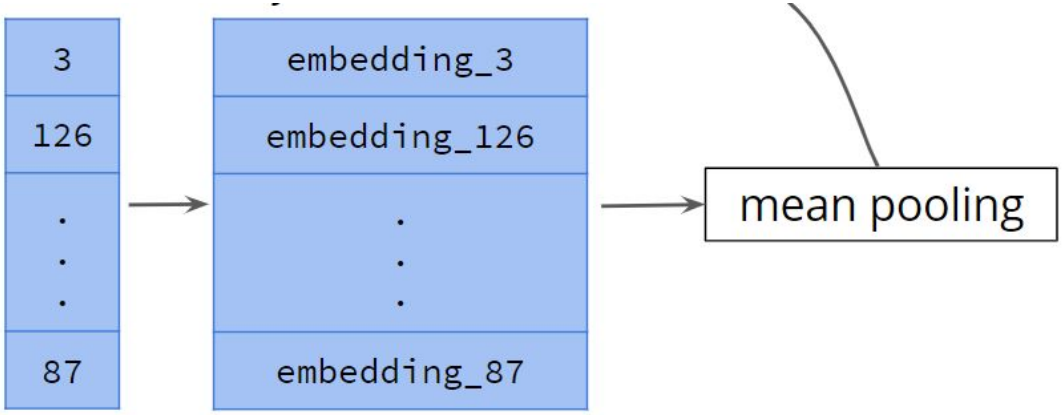
	user_id	query_movie_ids	query_movie_ids_len	candidate_movie_id	rating
0	47	[263, 2860, 3856, 6034, 2409, 2374, 7128, 5339...	357	1393	4.5
1	175	[966, 5026, 4395, 6042, 3871, 6521, 5020, 953,...	177	4002	3.0
2	261	[45, 1104, 154, 4514, 4171, 3801, 3727, 3644, ...	471	3089	2.0

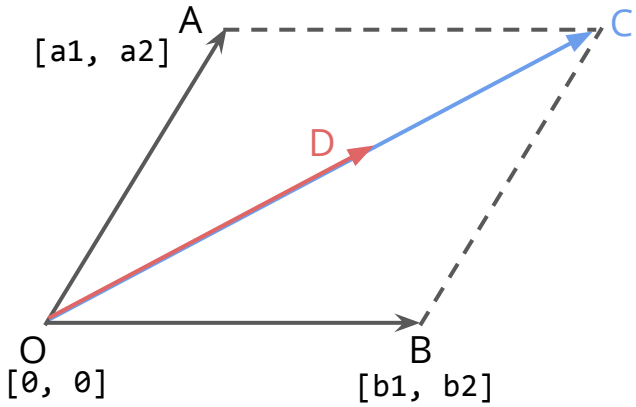
- 觀察47號user
  - query_movie_ids: train data中47號的有rating的movie id，排除1393號
    ⇒ leave one out (這裡的設計也可以只用47號喜歡的電影)
  - query_movie_ids_len: 描述query_movie_ids movie id的個數(tensorflow限制，mini batch裡面的變數長度必須要一致)

所以47號User在train data裡只有一筆資料了?

Variant of Matrix Factorization

Vector Computation



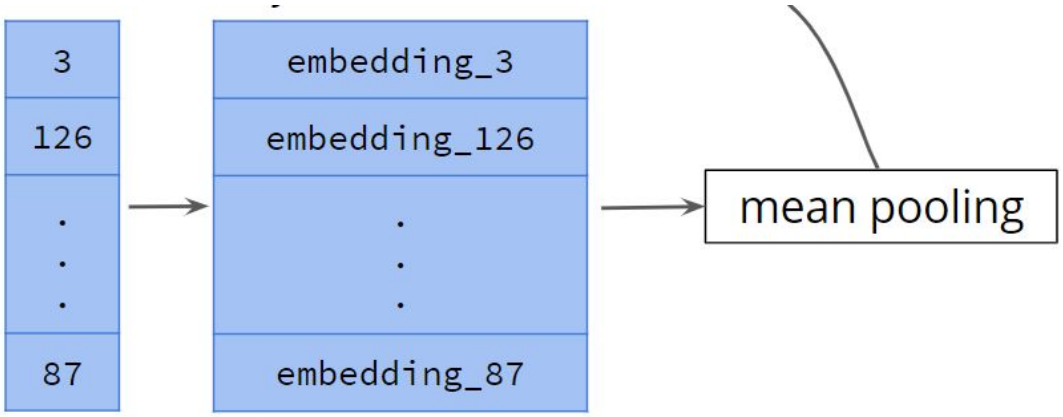


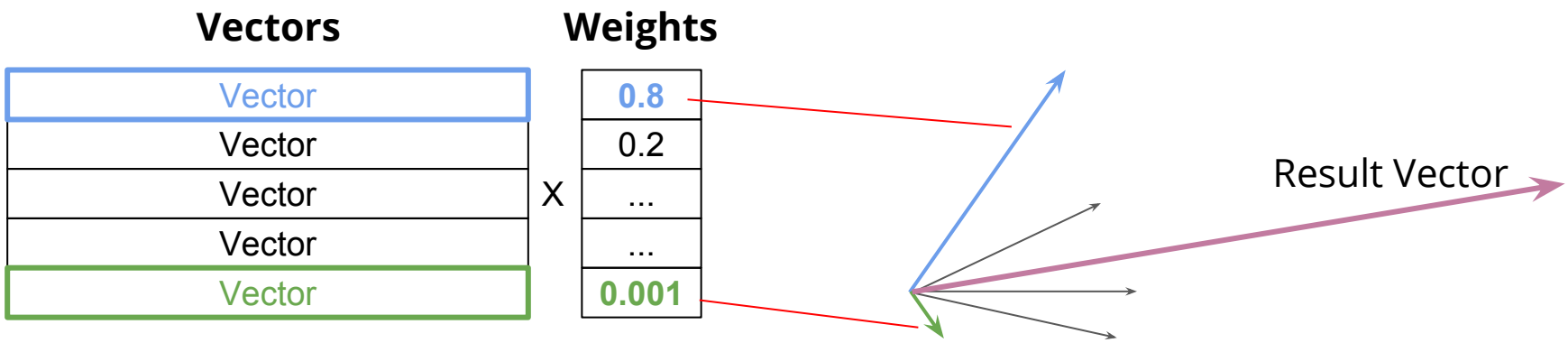
OC = OA + OB
OA = [a1, a2]
OB = [b1, b2]

SUM	OC = [a1 + b1, a2 + b2]
MEAN	OD = [(a1 + b1) / 2, (a2 + b2) / 2]

Variant of Matrix Factorization

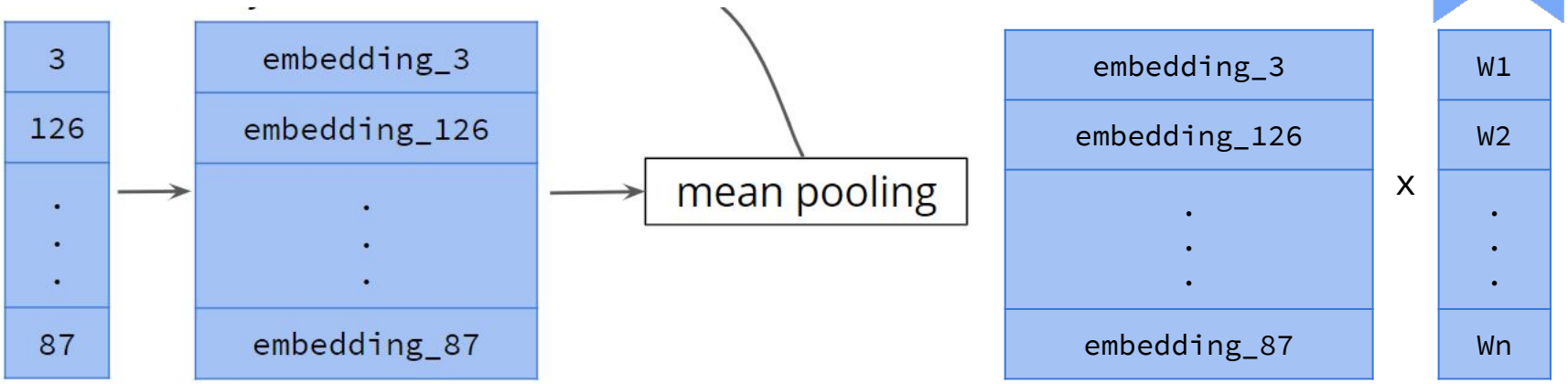
向量的加權和 (Weighted Sum)





Variant of Matrix Factorization

Movie ID Aggregation(Pooling)?

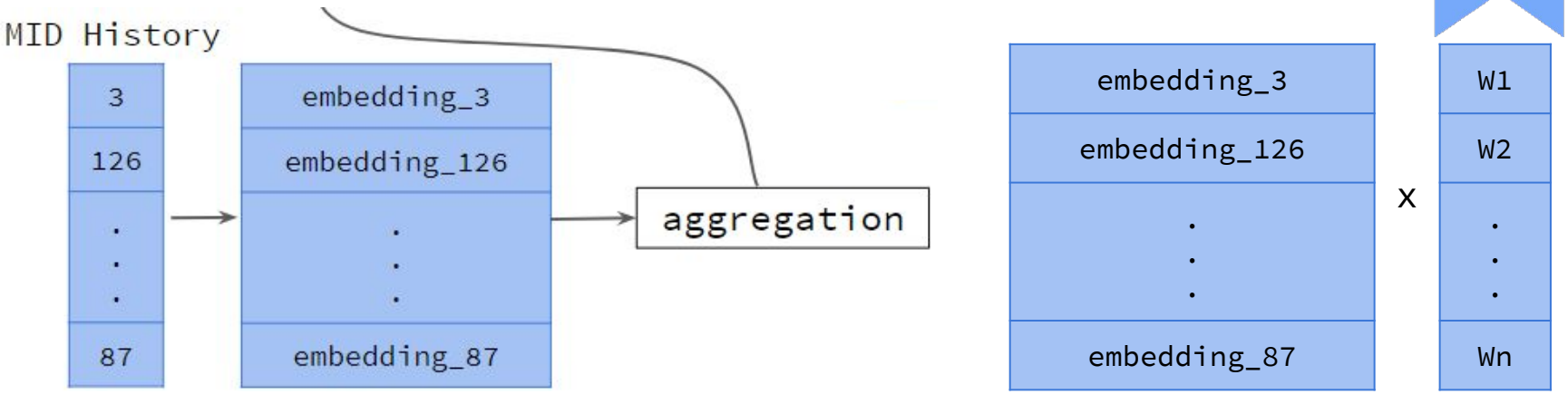


Tensorflow documentation上找到處理multivalent embedding的方式	
Sum	embedding向量相加
Mean	embedding向量相加後平均
SqrtN	將weights normalize之後再對embeddings做weighted sum

通常效果較好

Variant of Matrix Factorization

Movie ID Aggregation(Pooling)?



$X = [X_1, X_2, \ldots, Xn]$, $W = [W_1, W_2, \ldots, Wn]$

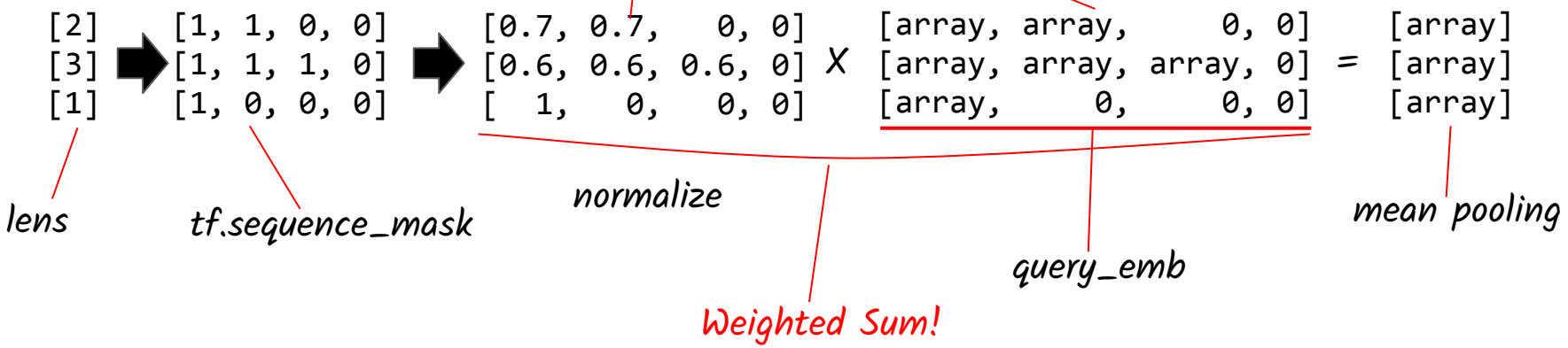
$$sqrtn = \frac{x_1 \cdot w_1 + x_2 \cdot w_2 + \ldots + x_n \cdot w_n}{\sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}} = [x_1, x_2, \ldots, x_n] \cdot \frac{[w_1, w_2, \ldots, w_n]}{\sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}}$$

$$= X \cdot l2\_nomalize(W)$$

Variant of Matrix Factorization

Function **sqrtn**

```
def sqrtn(x):
    qry, lens = x
    lens = tf.reshape(lens, [-1])
    weights = tf.nn.l2_normalize(tf.sequence_mask(lens, dtype=tf.float32), 1)
    weights = tf.expand_dims(weights, -1)
    return tf.reduce_sum(qry * weights, 1)
emb_query = Lambda(sqrtn, name='emb_query')([emb_query, inp_query_len])
```



Feature Engineering - User Side

	userId	user_rating_freq	user_rating_mean
0	0	-0.558471	-2.351626
1	1	-0.316104	-0.362525
2	2	-0.424303	-0.188878
3	3	0.237878	1.465967
4	4	-0.212232	0.535923

```
data.groupby(“userId”).rating.agg([‘size’, ‘mean’])
```

- user_rating_freq: User過去rating過多少電影
- user_rating_mean: User過去rating的平均

Variant of Matrix Factorization

Feature Engineering - Movie Side

genres	genres_len	avg_rating	freq_rating	year	candidate_movie_id	rating
[8]	1	1.102099	1.102099	-0.153507	931	4.0
[1, 6, 17]	3	0.830070	0.830070	-1.083561	1515	4.0
[9, 11, 15, 17]	4	0.006851	0.006851	0.001502	1083	3.5
[3, 4, 8, 13]	4	0.466777	0.466777	-2.633649	833	3.0
[17]	1	0.288261	0.288261	0.208180	859	3.0

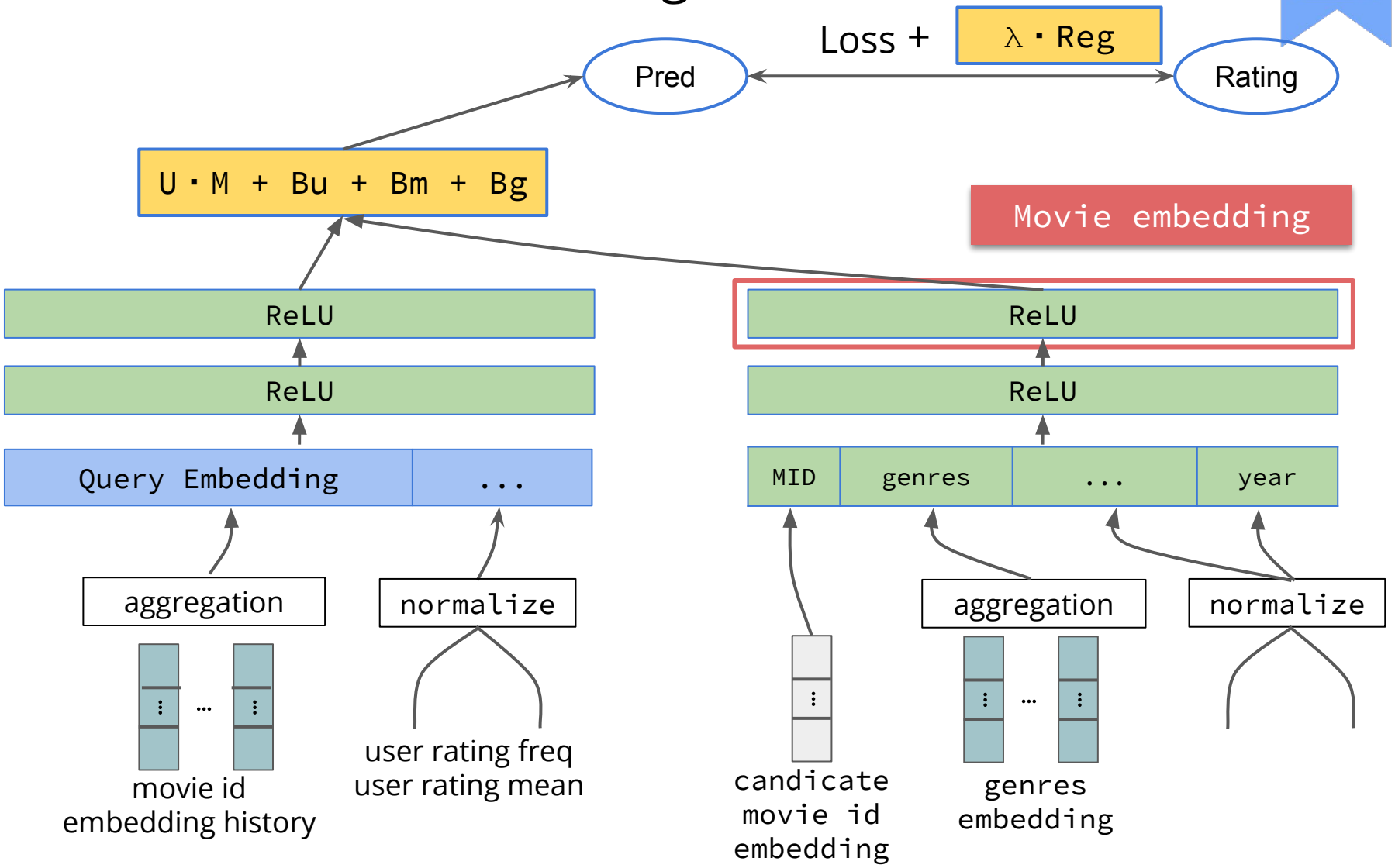
```
data.groupby("movidId").rating.agg(['size', 'mean'])
```

```
movies.title.str.findall("\(\s*(\d+)\s*\)")\
      .map(lambda lst: int(lst[-1]) if len(lst) else None)
```

- genres: multivalent variable ⇒ embedding ⇒ average
- genres_len: 描述genres長度
- avg_rating: 電影的平均評分(已normalize)
- freq_rating: 電影的平均評分(已normalize)
- year: 從title extract出來(已normalize)

Variant of Matrix Factorization

Model - MF with DNN (Regression)



Variant of Matrix Factorization

➔ **Functions of Regression**

Model prediction function

$$f_{u,m}(x) = u_i \cdot m_j + b_u + b_m + b_{global}$$

Loss function: mean squared error

$$L = \sum (f_{u,m}(x) - r_{ij})^2 + \lambda \cdot \frac{1}{2}(\sum u^2 + \sum m^2 + \sum b_u^2 + \sum b_m^2)$$

➔ **Functions of Classification**

Model prediction function

$$f_{u,m}(x) = \sigma(u_i \cdot m_j + b_u + b_m + b_{global})$$

Loss function: cross entropy

$$L = -\sum (\hat{y} \cdot lnf_{u,m}(x) + (1 - \hat{y})(1 - lnf_{u,m}(x))) + \lambda \cdot \frac{1}{2}(\sum u^2 + \sum m^2 + \sum b_u^2 + \sum b_m^2)$$

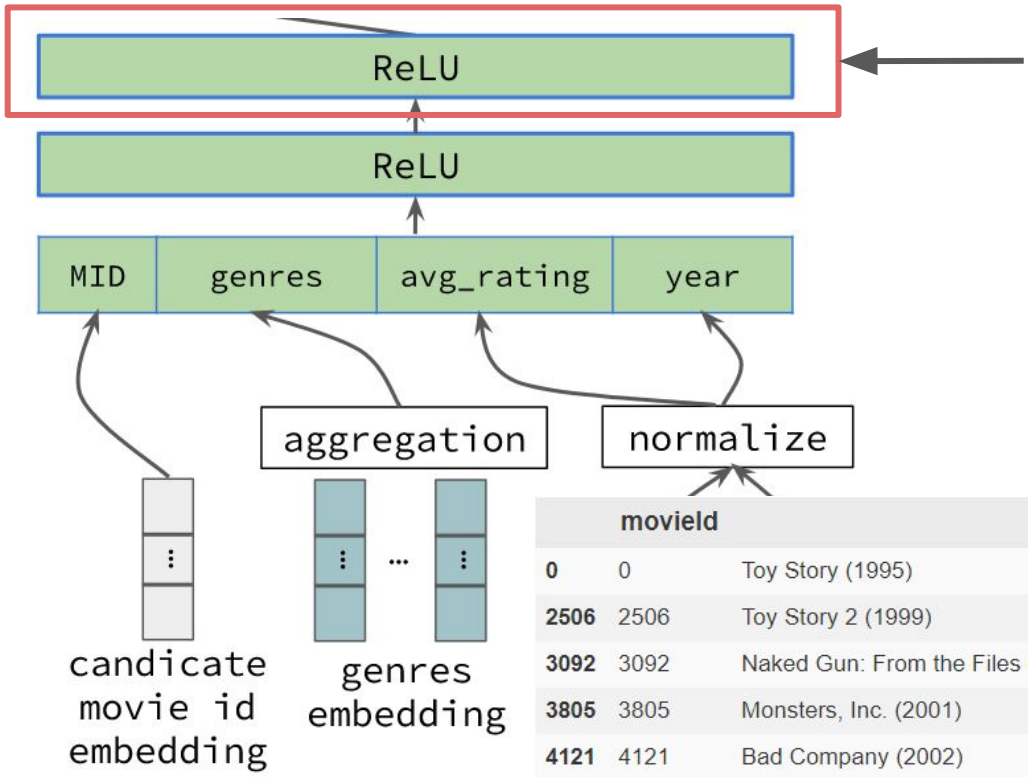
Variant of Matrix Factorization

Model - MF with DNN

filename	lab_reco_model_mf_dnn.ipynb
number of users	671
number of movies	9125
rating range	0.5 ~ 5分, ❑ 4分以上算正評 ❑ 未滿4分認為是負評或是不列入推薦名單
neural network	matrix factorization with dnn

Variant of Matrix Factorization

MF with DNN - 活用Laten Factor(Embedding)!



找出跟Toy Story相似的電影

- ❏ Toy Story2
- ❏ Monsters, Inc
- ❏ ...

	movieId	title	genres
0	0	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2506	2506	Toy Story 2 (1999)	Adventure|Animation|Children|Comedy|Fantasy
3092	3092	Naked Gun: From the Files of Police Squad!, Th...	Action|Comedy|Crime|Romance
3805	3805	Monsters, Inc. (2001)	Adventure|Animation|Children|Comedy|Fantasy
4121	4121	Bad Company (2002)	Action|Comedy|Crime
607	607	Wallace & Gromit: The Best of Aardman Animatio...	Adventure|Animation|Comedy
7850	7850	Mike's New Car (2002)	Animation|Comedy
4139	4139	Minority Report (2002)	Action|Crime|Mystery|Sci-Fi|Thriller
6423	6423	Protector, The (a.k.a. Warrior King) (Tom yum ...	Action|Comedy|Crime|Thriller
6248	6248	Kiss Kiss Bang Bang (2005)	Comedy|Crime|Mystery|Thriller
4558	4558	Cowboy Bebop: The Movie (Cowboy Bebop: Tengoku...	Action|Animation|Sci-Fi|Thriller

Model - 實際上使用Embedding的例子

使用User embedding找出該user偏好的衣服類型

甚至可以找不同類型的搭配(馬克思禮服找到一堆寬腿褲)

https://making.dia.com/embedding-everything-for-anything2anything-recommendations-fca7f58f53ff

MF with DNN - Metrics and Comparison

Model	MF	MF with DNN(MSE)	MF with DNN (Cross Entropy)
RMSE Loss	0.92	**0.82**	1.44 **cross entropy loss: 0.53(無法與RMSE相比)**
ROC AUC	0.74	0.80	**0.81**
NDCG	strict: 0.63 normal: 0.76	**strict: 0.73** **normal: 0.83**	strict: 0.72 normal: 0.82
Precision at 10	strict: 0.50 normal: 0.63	**strict: 0.59** **normal: 0.68**	**strict: 0.59** **normal: 0.68**

Model MF with DNN 彈性的應用

- 對於新的user
  - 無使用紀錄
    - 推薦最受歡迎商品
  - 一點點使用紀錄
    - Content base recommendation (movied embedding cosine similarity)
  - 大量使用紀錄
    - Model recommendation

- 對於新進來的電影
  - 將metadata帶入model取得embedding就可以使用
    - ex: genres、avg_rating、year、director、actors(actresses)...
  - 不使用指向性的Feature ⇒ Ex: Movie ID