Factorización Matricial en Sistemas Recomendadores
Clase de Introducción

Denis Parra

30 de Septiembre de 2014

TOC

- Contexto
- Métodos de vecindario y de factores latentes
- El Netflix Prize
  - Yehuda Koren & Chris Volinski
  - La solución de Simon Funk para SVD

Hasta este punto …

Content-Based	Filtrado Colaborativo
Basado en le perfil del usuario, el cual puede tener diversas representaciones (vector de características, contenido de ítems consumidos, etc)	Basado en interacciones y valoraciones de los usuario
Principal Problema: Se requiere información externa	Principal Problema: Cold Start: New User, New Item
Ejemplo: Pandora – Music Genome Project	Ejemplo: Netflix (al menos el desafío presentado en el Netflix challenge)

Neighborhood methods

***Latent Factors Methods (MF)**

Basado en presentacion de Peng Xu: http://www.groupes.polymtl.ca/inf6304/Presentations/20133/matrix-fac.pdf

- Las siguientes slides son un resumen del Paper "Matrix Factorization Techniques for Recommender Systems" de Koren, Bell y Volinksy, ganadores del Netflix Prize

CF con modelos de Vecindario

- Modelos que hemos visto basados en calcular relaciones entre los usuarios o entre los ítems a recomendar:

- User Based CF (KNN, en la figura)
- Item Based CF

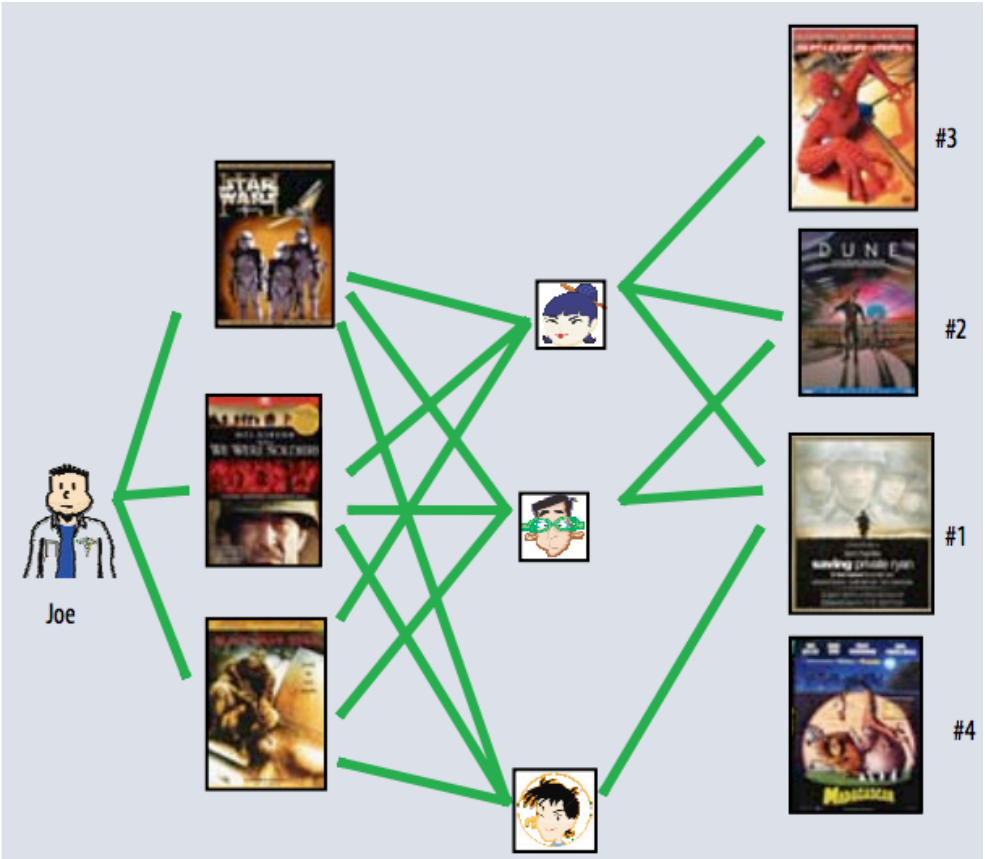


Figure 1. The user-oriented neighborhood method. Joe likes the three movies on the left. To make a prediction for him, the system finds similar users who also liked those movies, and then determines which other movies they liked. In this case, all three liked *Saving Private Ryan*, so that is the first recommendation. Two of them liked *Dune*, so that is next, and so on.

Modelos de Factores Latentes

- Intenta explicar ratings de usuarios caracterizando a items y usuario sobre factores inferidos a partir de patrones de ratings

- Los factores nonecesariamente deben conocerse de antemano

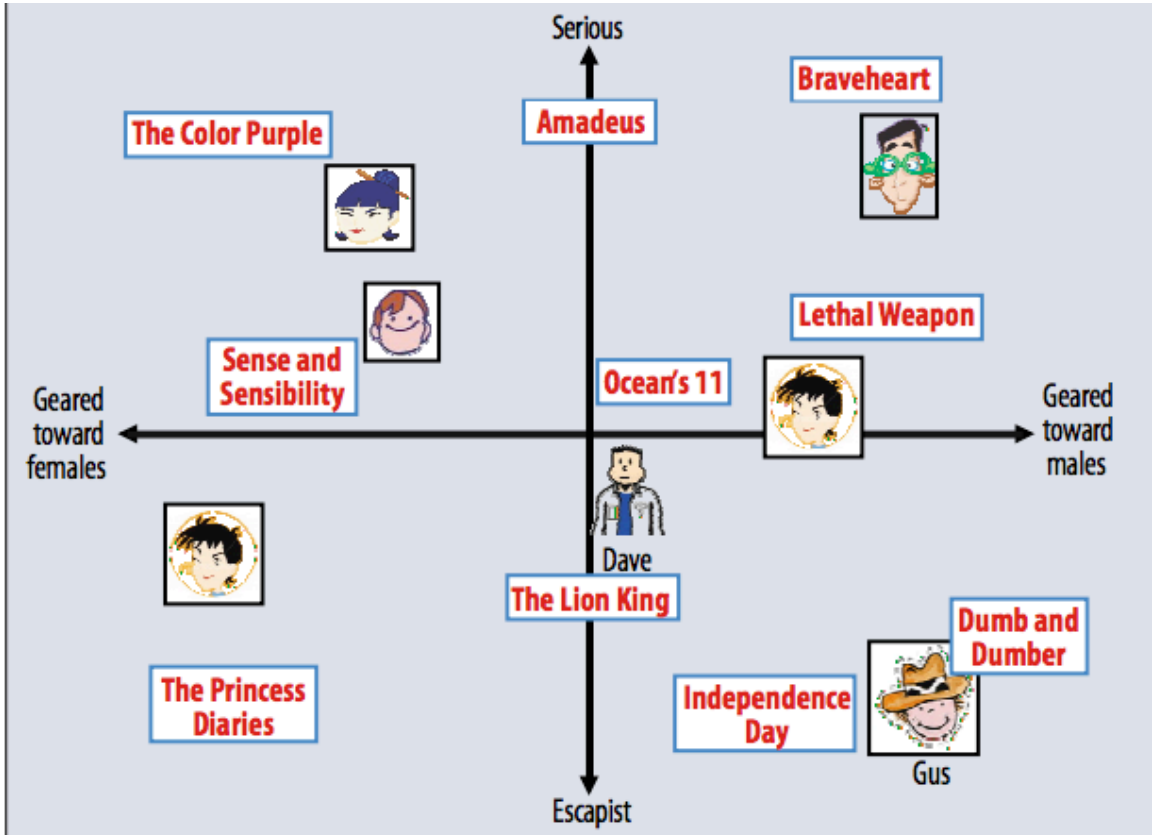


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Por qué Factorización Matricial?

- Podemos encontrar factores latentes que nos permiten hacer predicciones, pero además obtender información de relaciones entre usuarios e ítems
- Más específicamente:
  - Tiene buena escalabilidad
  - Predicciones más precisas
  - Flexibilidad

Un modelo básico de FM

$q_i \in \mathbb{R}^f$ representa al item i

$p_u \in \mathbb{R}^f$ represents al usuario u

y el producto interno $q_i^T p_u$

captura la interaccion entre u e i

Luego, el rating se predice como:

$$\hat{r}_{ui} = q_i^T p_u$$

Desafío: Obtener vectores latentes

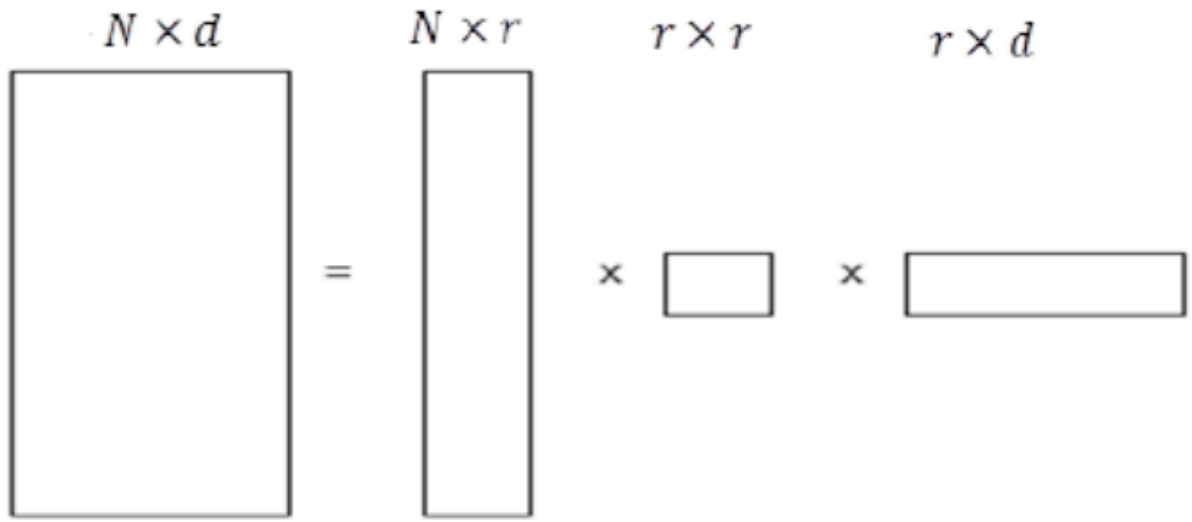
- Técnica factible: SVD (la misma que vimos la clase pasada para LSI)
- SVD tradicional no está definida para una matriz muy incompleta
- Primeros sistemas llenaban la matriz para hacerla más densa: Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). “Application of dimensionality reduction in recommender system-a case study “

Llenado de matriz

- Se llenaban vacíos usando el rating promedio del usuario o del item
- Procedimiento
  - factor R_{norm} using SVD to obtain U, S and V.
  - reduce the matrix S to dimension k
  - compute the square-root of the reduced matrix S_k, to obtain $S_k^{1/2}$
  - compute two resultant matrices: $U_k S_k^{1/2}$ and $S_k^{1/2} V_k'$

SVD de la clase pasada

$$\mathbf{X} = \mathbf{U} \times \mathbf{\Sigma} \times \mathbf{V^T}$$



$$(\mathbf{t}_i^T) \rightarrow \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} = (\hat{\mathbf{t}}_i^T) \rightarrow \begin{bmatrix} \begin{bmatrix} \\ \mathbf{u}_1 \\ \\ \end{bmatrix} \dots \begin{bmatrix} \\ \mathbf{u}_l \\ \\ \end{bmatrix} \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} \cdot \begin{bmatrix} [& \mathbf{v}_1 &] \\ & \vdots & \\ [& \mathbf{v}_l &] \end{bmatrix}$$

X $(\mathbf{d}_j)$ ↓ — U — Σ — V^T $(\hat{\mathbf{d}}_j)$ ↓

Consideraciones Prácticas

- Componente off-line (entrenar el modelo, elegir número de dimensiones) y componente online (realizar las predicciones)
- Decomposición de matriz usuario-item se hace offline. Es $O((m+n)^3)$, mientras que la recomendación basada en correlación es $O(m^2n)$. Sin embargo, para la predicción, SVD es eficiente pues debe almacenar sólo del orden $O(m+n)$.
- Número de dimensiones latentes debe determinarse por cross-validation.

…Volviendo al Netflix prize

- Llenar la matriz (imputación) puede ser computacionalmente costoso e incrementa el numero de datos, además de ser sensible a sesgos introducidos por imputación.
- El método puede inducir a overfitting
- **Solución**: optimización considerando regularización

Minimizar el error con Regularización

- La idea es “aprender” los vectores de factores q_i y p_u usando:

$$\min_{q^*,p^*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(|| q_i ||^2 + || p_u ||^2)$$

- K : número de pares (u,i) donde r_{ui} es conocido
- El efecto de la regularización: penalizar valores muy altos en q_i y p_u

Algoritmos de Aprendizaje

- Dos métodos son mencionados en el paper de Koren et al.
- Stochastic Gradient Descent
  - Fácil Implementación
  - Tiempo de ejecución relativamente rápido
- Alternating Least Squares
  - Se puede paralelizar

SGD: Procedimiento iterativo

- Se inicializan pi y qu de forma aleatorio
- Se calcula un error

$$e_{ui} \overset{def}{=} r_{ui} - q_i^T p_u.$$

Y se actualizan q_i y p_u en la dirección del gradiente

$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$$
$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$$

SGD de Simon Funk para el Netflix Prize:

http://sifter.org/~simon/journal/20061211.html

El problema de recomendación como FM -

- Matriz Original: 17.000 peliculas x 500.000 usuarios: 8.5*10^9
- Supongamos que elegimos 40 factores latentes: nuestro modelo necesitaría almacenar: 40*(17K+500K) parámetros

ratingsMatrix[user][movie] = sum (userFeature[f][user] * movieFeature[f][movie]) for f from 1 to 40

Cómo actualizar el modelo?

```
userValue[user] += lrate * err * movieValue[movie];
movieValue[movie] += lrate * err * userValue[user];
/**…….**/
/* Where:
 *  real *userValue = userFeature[featureBeingTrained];
 *  real *movieValue = movieFeature[featureBeingTrained];
 *  real  lrate = 0.001;
 */
static inline
void train(int user, int movie, real rating)
{
      real err = lrate * (rating - predictRating(movie, user));

      userValue[user] += err * movieValue[movie];
      movieValue[movie] += err * userValue[user];
}
```

ALS

- Como q_i y p_u son desconocidos, la función es no convexa, pero se puede rotar entre fijar los qi's para optimizar los pu's, y luego fijar los pu's para optmizar qi's
- SGD es más fácil de implementar y más rápido, pero ALS se puede paralelizar y funciona de forma más óptima para casos de implificit feedback (matriz más densa)

Agregando los biases

$$b_{ui} = \mu + b_i + b_u$$

- Predicción es luego

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

- Y la función de error es

$$\min_{p^*,q^*,b^*} \sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda$$

$$(\| p_u \|^2 + \| q_i \|^2 + b_u^2 + b_i^2)$$

Feedback Implícito y otras variables

- *N(u)* denota el conjunto de ítems para los cuáles el usuario ha expresado preferencia implícita, y el item *i* está asociado con x_i
- Un usuario que mostró preferencia por items en N(u) puede ser caracterizado como:

$$\sum_{i \in N(u)} x_i$$

- Y normalizando, esto se puede escribir como

$$|N(u)|^{-0.5} \sum_{i \in N(u)} x_i$$

Otras variables

- Atributos asociados al usuario A(u), pueden ayudar a describir al usuario como

$$\sum_{a \in A(u)} y_a$$

- Luego, el modelo de factorización extendido será:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \left[p_u + |N(u)|^{-0.5} \sum_{i \in N(u)} x_i + \sum_{a \in A(u)} y_a \right]$$

FM accuracy

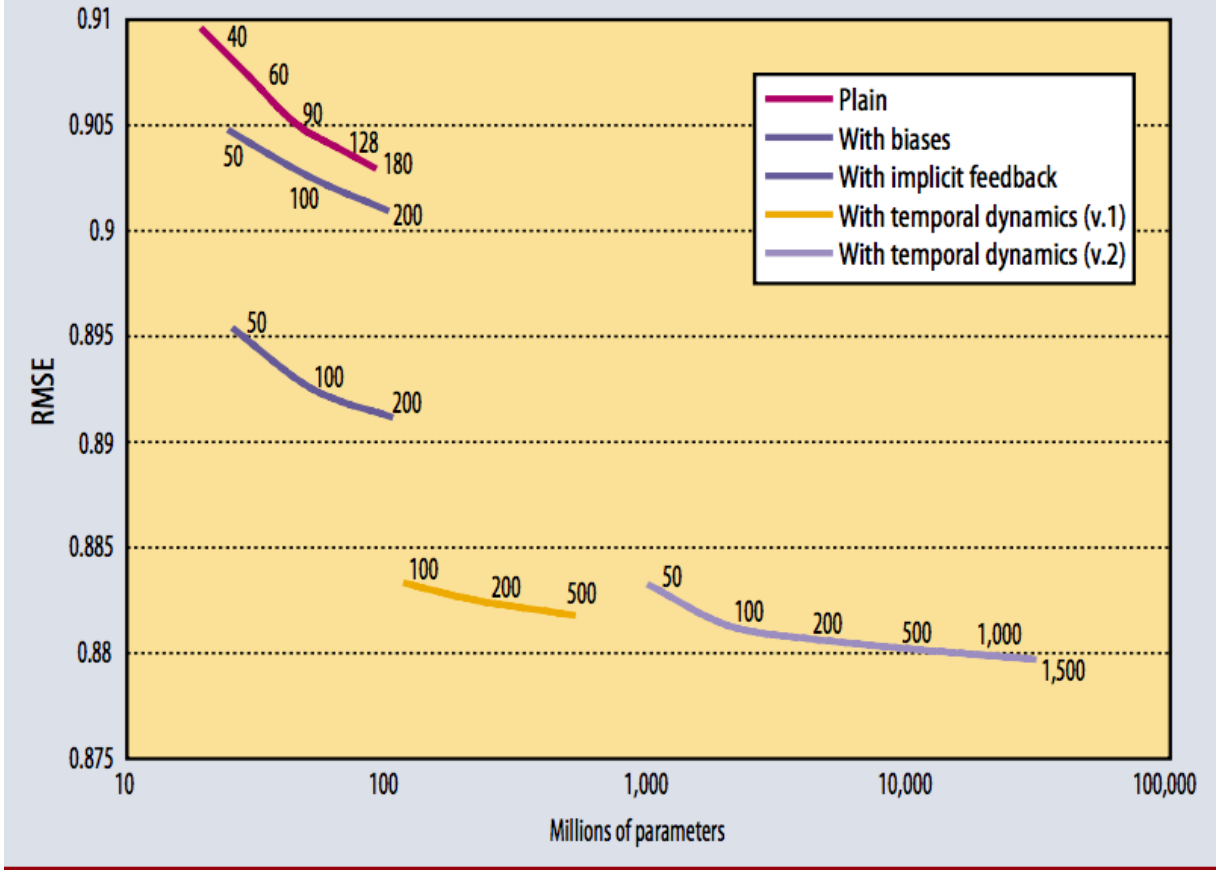


Figure 4. Matrix factorization models' accuracy. The plots show the root-mean-square error of each of four individual factor models (lower is better). Accuracy improves when the factor model's dimensionality (denoted by numbers on the charts) increases. In addition, the more refined factor models, whose descriptions involve more distinct sets of parameters, are more accurate. For comparison, the Netflix system achieves RMSE = 0.9514 on the same dataset, while the grand prize's required accuracy is RMSE = 0.8563.

Dimensiones latentes

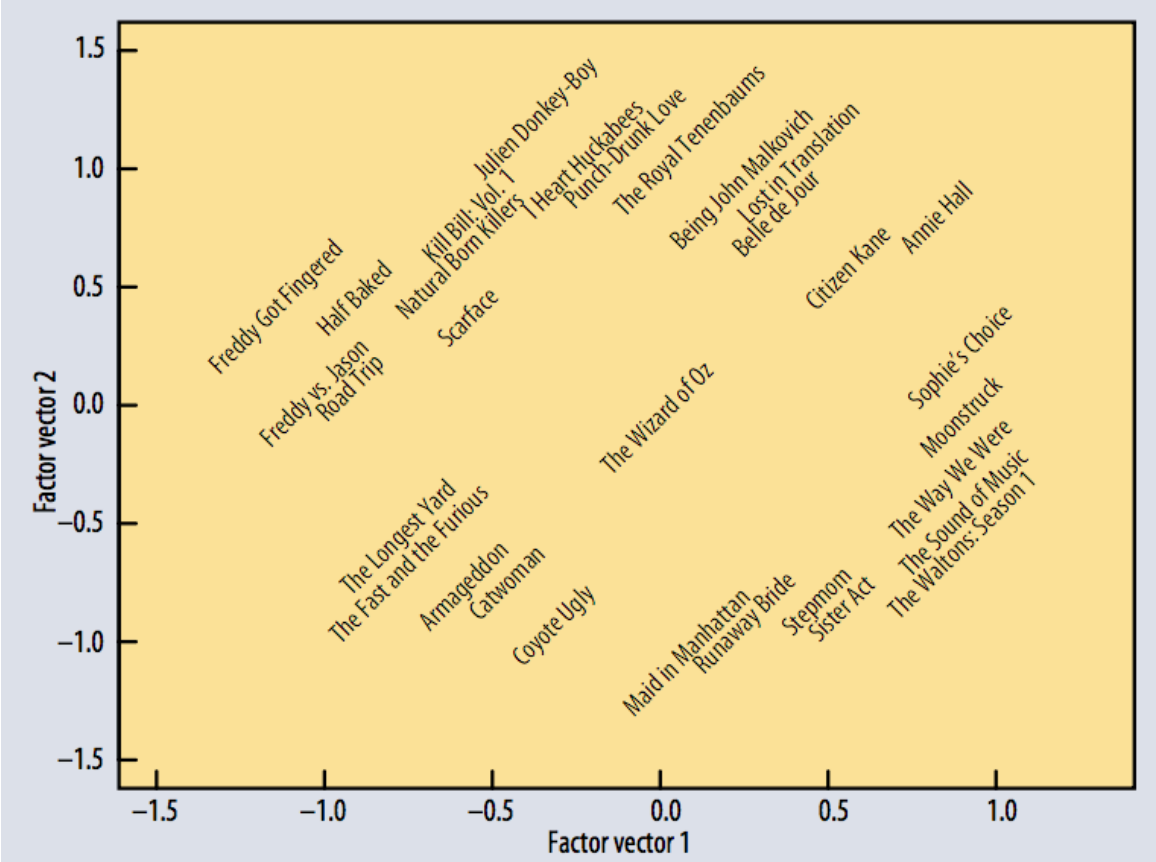


Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.

Próxima clase:

- Agregar información temporal
- Factorización de tensores
- Varios versiones de Factorización:
  - PMF
  - NMF
  - WALS
  - Factorization Machines

References

- Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.
- Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Application of dimensionality reduction in recommender system-a case study (No. TR-00-043). Minnesota Univ Minneapolis Dept of Computer Science.
- Hofmann, T. (2003, July). Collaborative filtering via gaussian probabilistic latent semantic analysis. In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval (pp. 259-266). ACM.