Matrix factorization

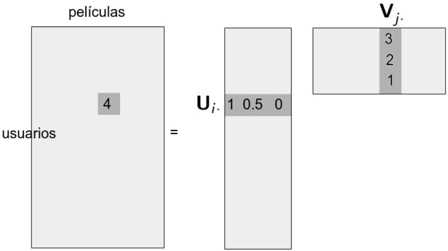
Yamila M. Barrera

24 de octubre de 2017

Matrix factorization

Contexto: Conocemos los puntajes que algunos usuarios dieron a algunas películas.
Problema: ¿Cómo predecir los puntajes desconocidos?

$$
\begin{array}{cc}
 & \text{películas} \\
\text{usuarios} & \\
R \;= & \begin{array}{|cccccccc|}
\hline
? & 2 & ? & ? & ? & ? & 3 & ? \\
5 & ? & 4 & ? & ? & ? & ? & 4 \\
5 & ? & 4 & ? & 1 & 2 & ? & ? \\
2 & ? & ? & 4 & 5 & 2 & 3 & ? \\
? & 1 & ? & 3 & ? & ? & ? & ? \\
4 & ? & ? & ? & 5 & ? & ? & 2 \\
? & ? & ? & ? & ? & ? & ? & ? \\
? & 3 & ? & ? & 2 & ? & ? & 1 \\
4 & ? & ? & ? & ? & ? & 5 & ? \\
? & 4 & ? & 3 & ? & 1 & ? & 1 \\
3 & 3 & ? & 5 & ? & ? & 2 & ? \\
\hline
\end{array}
\end{array}
$$



$$R = \mathbf{U}\mathbf{V}^t$$

$$r_{ij} = \mathbf{U}_{i\cdot}\mathbf{V}_{j\cdot}^t = \sum_{l=1}^{k} u_{il} v_{jl}$$

$$4 = 1 \cdot 3 + 0{,}5 \cdot 2 + 0 \cdot 1$$

$\mathbf{U}_{i\cdot}$: factores que describen al usuario i
$\mathbf{V}_{j\cdot}$: factores que describen a la película j

dim(U) = #usuarios x k
dim(V) = #películas x k

Como problema de optimización

$$\mathbf{U}, \mathbf{V} = argmin \frac{1}{2} \sum_{(i,j) \in S} (r_{ij} - \mathbf{U}_{i\cdot} \mathbf{V}_{j\cdot}^t)^2$$

donde

r_{ij}: rating del usuario i a la película j

$S = \{(i, j) : r_{ij} \text{ es conocido } \}$

Como problema de optimización

$$\mathbf{U}, \mathbf{V} = argmin \frac{1}{2} \sum_{(i,j)\in S} (r_{ij} - \mathbf{U}_{i\cdot}\mathbf{V}_{j\cdot}^t)^2$$

donde

r_{ij}: rating del usuario i a la película j

$S = \{(i,j) : r_{ij} \text{ es conocido } \}$

Como problema de optimización

$$\mathbf{U}, \mathbf{V} = argmin \frac{1}{2} \sum_{(i,j) \in S} (r_{ij} - \mathbf{U}_{i\cdot} \mathbf{V}_{j\cdot}^t)^2$$

donde

$$r_{ij}\text{: rating del usuario } i \text{ a la película } j$$
$$S = \{(i,j) : r_{ij} \text{ es conocido } \}$$

¿Cómo predecir ratings?

$$\hat{r}_{ij} = \mathbf{U}_{i\cdot} \mathbf{V}_{j\cdot}^t = \sum_{l=1}^{k} u_{il} v_{jl}$$

Alternating least squares

Si fijamos una de las incógnitas (**U**ó **V**) el problema de optimización es cuadrático y puede obtenerse un despeje analítico.

$$L = \frac{1}{2} \sum_{(i,j):r_{ij} \in S} (r_{ij} - \mathbf{U}_{i\cdot} \mathbf{V}^t_{j\cdot})^2$$

Fijada **V**, $\mathbf{U}_i$ es la solución de $\frac{\partial L(\mathbf{U},\mathbf{V})}{\partial U_i} = 0$
Fijada **U**, $\mathbf{V}_j$ es la solución de $\frac{\partial L(\mathbf{U},\mathbf{V})}{\partial V_j} = 0$

Gradient Descent

$$L = \frac{1}{2} \sum_{(i,j)\in S} \left(r_{ij} - \sum_{s=1}^{k} u_{is} v_{js} \right)^2$$

```
Algorithm GD (Ratings matrix R, learning rate α)
begin
  Randomly initialize matrices U and V
  S = {(i,j) : r_ij is observed}
  while not convergence do
  begin
    compute e_ij ∈ S as the observed entries of R − UV^t
    for each user-component pair (i,q) do u⁺_iq ← u_iq + α ∂L/∂u_iq
    for each item-component pair (j,q) do v⁺_jq ← v_jq + α ∂L/∂v_jq
    for each user-component pair (i,q) do u_iq ← u⁺_iq
    for each item-component pair (j,q) do v_jq ← v⁺_jq
    Check convergence condition
  end
end
```

Stochastic Gradient Descent

$$\hat{r}_{ij} = \sum_{q=1}^{k} u_{iq} v_{jq}$$

Idea: Mirar de a una por vez las entradas conocidas de R y sólo actualizar las correspondientes $2k$ entradas en las matrices U y V

$$u_{iq} \leftarrow u_{iq} - \alpha \frac{\partial L}{\partial u_{iq}} \quad \forall q \in 1, \ldots, k$$

$$v_{jq} \leftarrow v_{jq} - \alpha \frac{\partial L}{\partial v_{jq}} \quad \forall q \in 1, \ldots, k$$

Stochastic Gradient Descent

$$\hat{r}_{ij} = \sum_{q=1}^{k} u_{iq} v_{jq}$$

Idea: Mirar de a una por vez las entradas conocidas de R y sólo actualizar las correspondientes $2k$ entradas en las matrices U y V

$$u_{iq} \leftarrow u_{iq} - \alpha e_{ij} v_{jq} \quad \forall q \in 1, \dots, k$$

$$v_{jq} \leftarrow v_{jq} - \alpha e_{ij} u_{iq} \quad \forall q \in 1, \dots, k$$

Stochastic Gradient Descent (SGD)

Algorithm SGD (Ratings matrix R, learning rate α)
begin
  Randomly initialize matrices U and V
  S = $\{(i,j) : r_{ij}$ is observed$\}$
  while not convergence do
  begin
    Randomly shuffle obserede entries in S
    for each $(i,j) \in S$ in shuffle order do
    begin
      $e_{ij} \leftarrow r_{ij} - \sum_{s=1}^{k} u_{is} v_{js}$
      for each $q \in \{1,\ldots,k\}$ do $u_{iq}^{+} \leftarrow u_{iq} + \alpha \frac{\partial L}{\partial u_{iq}}$
      for each $q \in \{1,\ldots,k\}$ do $u_{iq}^{+} \leftarrow u_{iq} + \alpha \frac{\partial L}{\partial v_{jq}}$
      for each $q \in \{1,\ldots,k\}$ do $u_{iq} \leftarrow u_{iq}^{+}$
      for each $q \in \{1,\ldots,k\}$ do $v_{jq} \leftarrow v_{jq}^{+}$
    end
    Check convergence condition
  end
end

Regularización

Agregamos un término de regularización para evitar overfitting:

$$\mathbf{U}, \mathbf{V} = argmin \frac{1}{2} \sum_{(i,j):r_{ij} \in S} (r_{ij} - \mathbf{U}_{i\cdot} \mathbf{V}_{j\cdot}^t)^2 + \lambda_u ||\mathbf{U}||_F^2 + \lambda_v ||\mathbf{V}||_F^2$$

Modelo con bias

$$\hat{r}_{ij} = \mu + b_i + c_j + \mathbf{U}_{i\cdot} \mathbf{V}_{j\cdot}{}^t$$

μ: bias general
b_i: bias asociado a la película i
c_j: bias asociado al usuario j

$$L = \frac{1}{2} \sum_{(i,j):r_{ij} \text{is known}} (r_{ij} - \hat{r}_{ij})^2 + \lambda_u ||\mathbf{U}||_F^2 + \lambda_v ||\mathbf{V}||_F^2 + \lambda_b ||\mathbf{b}||^2 + \lambda_c ||\mathbf{c}||^2$$

$$\mathbf{U}, \mathbf{V}, \mathbf{b}, \mathbf{c} = argmin\ L$$

Implementación en Python

Surprise

A Python scikit for recommender systems.

Implementación usando numpy y pandas:
`http://blog.ethanrosenthal.com/2016/01/09/explicit-matrix-factorization-sgd-als/`

- **Parámetros del modelo:**
  - k: cantidad de factores
  - λ_*: regularización
- **Parámetros del problema de optimización**
  - α: learning rate
  - *n_epochs*: cantidad de iteraciones sobre los datos
  - inicialización de U y V

Modelo con confianza

Supongamos ahora que r_{ij} es la cantidad de veces que el usuario i miró el programa de televisión j.

$$p_{ij} = \begin{cases} 1 \text{ si } r_{ij} > 0 \\ 0 \text{ si } r_{ij} = 0 \end{cases}$$

Definimos la confianza que tenemos en que el usuario i guste del programa j como

$$c_{ij} = 1 + \alpha r_{ij}$$

Y planteamos el problema de optimización

$$min \sum_{(i,j)} c_{ij}(p_{ij} - \mathbf{U}_{i\cdot}\mathbf{v}_{j\cdot}^t)^2 + \lambda_u ||\mathbf{U}||_F^2 + \lambda_v ||\mathbf{V}||_F^2$$

Problema: ahora la suma es sobre todos los (i, j)

Bibliografía

- Matrix factorization techniques for recommender systems, *Yehuda Koren,Robert Bell, Chris Volinsky*
- Collaborative filtering for implicit Feedback datasets, *Yifan Hu,Yehuda Koren, Chris Volinsky*
- Ethan Rosenthal. Intro to recommender systems: collaborative filtering. `http://blog.ethanrosenthal.com/2015/11/02/intro-to-collaborative-filtering/`
- Explicit matrix factorization: ALS, SGD, and all that jazz. `http://blog.ethanrosenthal.com/2016/01/09/explicit-matrix-factorization-sgd-als/`