Sistemas Recomendadores
IIC-3633

Recomendación basada en feedback implícito

Esta clase

1. Repaso de factores latentes
2. Feedback implícito (sólo interacciones)
3. Formas de abordar este problema.
4. Recomendación basada en factores latentes (BPR)

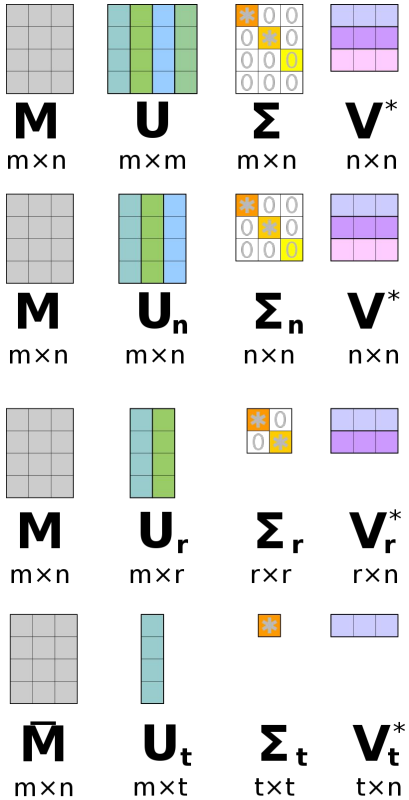
Repaso

¿Por qué no usamos SVD para aprender los vectores de usuarios e ítems?

Las dimensiones de la matriz son demasiado grandes y es muy costoso computacionalmente

El algoritmo para hacer SVD original está hecho para una matriz que tiene todos los valores.

Necesitamos agregarle regularización para prevenir overfitting (FunkSVD).



¿Por qué reducimos dimensionalidad?

Para disminuir la cantidad de computo

Para reducir la maldición de la dimensionalidad, ahora tendré vectores más pequeños y más densos.

En alta dimensión los vectores pueden quedar a distancias muy similares.

Solución parcial: FunkSVD

- FunkSVD a diferencia de SVD agrega un factor de regularización, a diferencia de SVD encuentra directamente vectores de usuarios e ítems.

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

donde

r_{ui}: rating que asignó el usuario u al ítem i

q_i: vector latente del ítem i

p_u: vector latente del usuario u

λ: constante de regularización

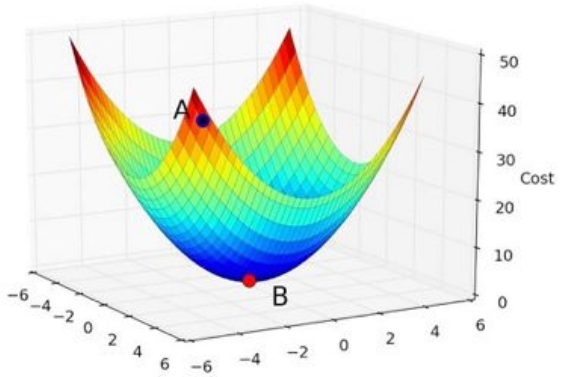
Factorización Matricial - Descenso de Gradiente

- Descenso de gradiente es un procedimiento para optimizar funciones.
- La intuición es mover los parámetros del modelo en dirección contraria del gradiente
- En el caso de *FunkSVD* la función de aprendizaje se aplica para cada par (u, i)

Función objetivo $\min_{\theta} \frac{1}{2} \sum_{i} (y_i - f(x_i, \theta))^2$

Función de error $E_i = \frac{1}{2}(y_i - f(x_i, \theta))^2$

Actualización de parámetros $\Delta w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}$

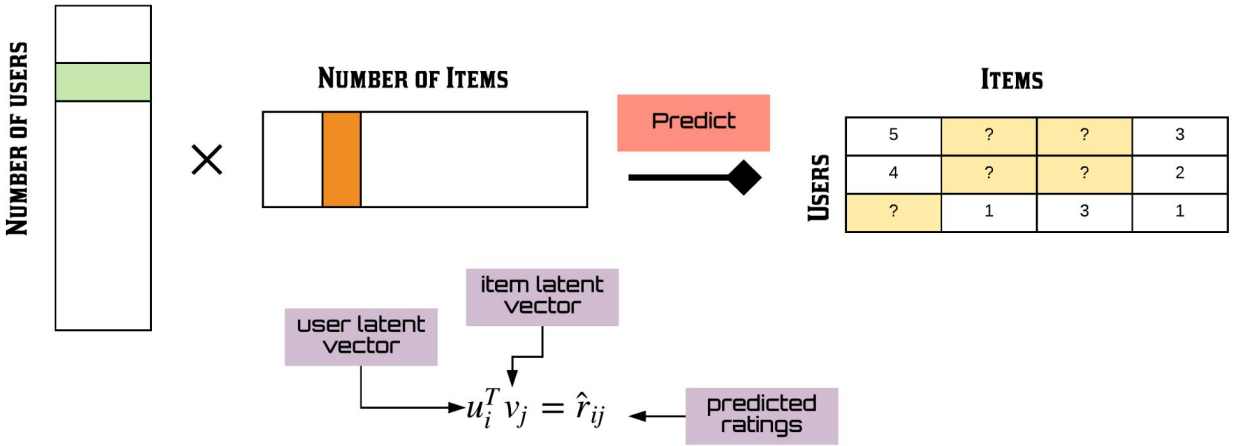


¿Cómo aprendemos qi y pu?

1. Inicializar aleatoriamente ambos vectores.
2. Usando las reglas de actualización voy iterativamente actualizando los valores de qi y pu
3. Repetir hasta que converja el error.

¿Cómo generamos la recomendación?

Una vez aprendidos los vectores qi y pu



Feedback explícito vs feedback implícito

Valerie

★★★★★ **Muy feliz**

Calificado en Estados Unidos el 15 de abril de 2023

Tamaño: L2 | **Compra verificada**

Nunca he jugado mucho al tenis, pero he tenido una pareja a la que le gusta jugar y una nueva oportunidad de jugar. Tenía unas cuantas raquetas más baratas aquí, Head and Wilson, ya que esta raqueta es una mejora definitiva para mí. Yo lo recomendaría.

Betty

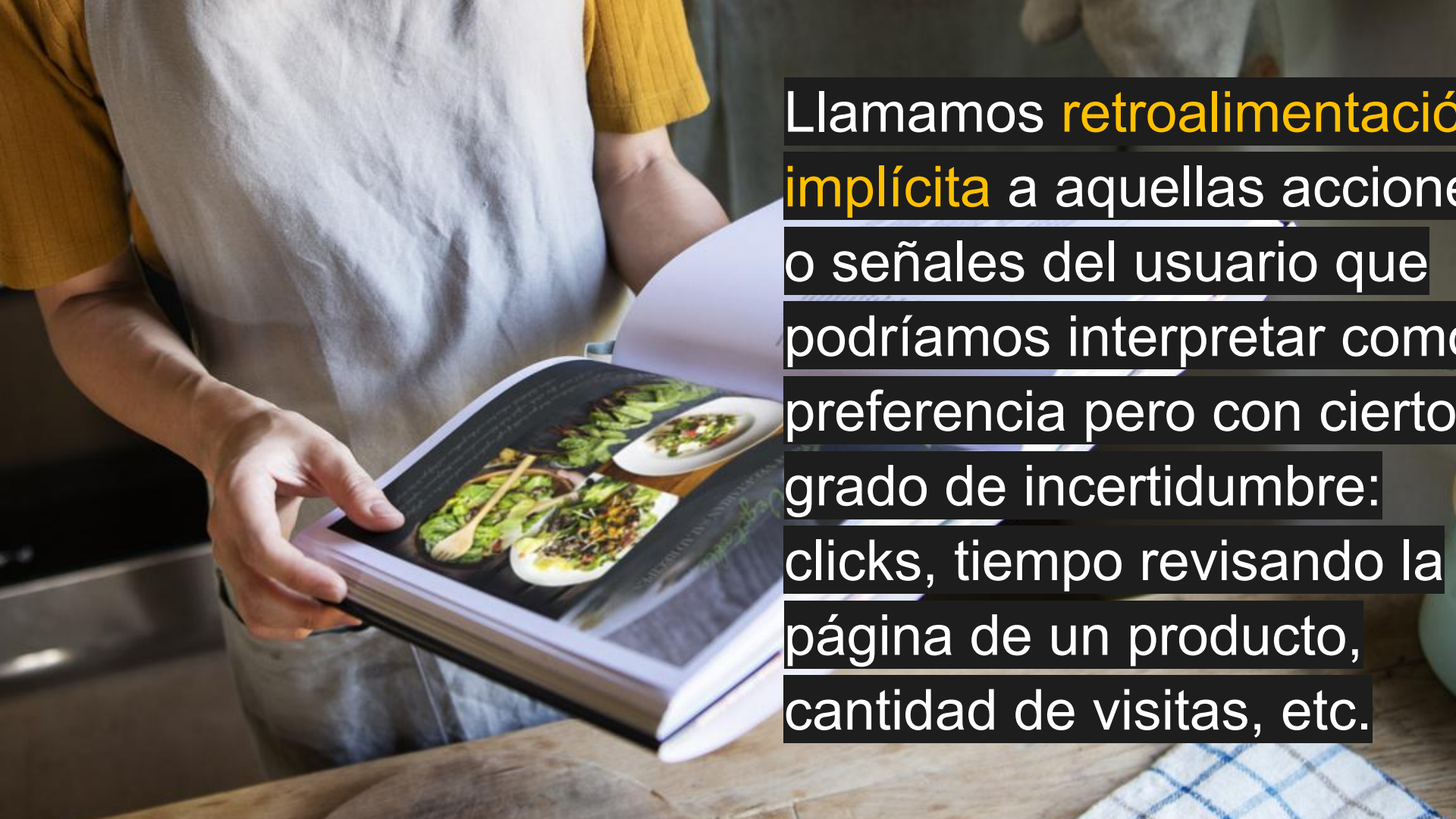
★★★☆☆ **Principiante**

Calificado en Estados Unidos el 7 de marzo de 2023

Tamaño: L2 | **Compra verificada**

La raqueta tiene colores reservados, pero es para personas muy principiantes, es buena opción para tenerla en el bolso y prestársela a algún amigo que no juegue tenis diariamente

Llamamos retroalimentación explícita a aquellas acciones donde el usuario nos indica directamente sus preferencias, por ejemplo las calificaciones o ratings, y las acciones ”pulgar arriba”



Llamamos retroalimentació
implícita a aquellas accione
o señales del usuario que
podríamos interpretar com
preferencia pero con cierto
grado de incertidumbre:
clicks, tiempo revisando la
página de un producto,
cantidad de visitas, etc.

Características de la retroalimentación implícita

1. Es más fácil de obtener que la retroalimentación explícita. No siempre la gente quiere dar su opinión a través de calificaciones o ratings.

2. La RI no contiene esencialmente información de preferencias negativas, a diferencia de calificaciones o ratings explícitos.

3. La RI contiene ruido y es difícil cuantificar preferencias de los usuarios y confianza sobre esas preferencias.

4. No podemos usar métricas como RMSE o MAE para medir el rendimiento, debemos usar métricas de ranking

Retroalimentación Implícita o Implicit Feedback

- Hu, Y., Koren, Y., & Volinsky, C. (2008) modificaron el modelo de SVD para incluir *feedback* implícito.

Binarización de los datos

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

- interacción si se pasa umbral de preferencia (ej. veces que consume el producto).

Función de confianza

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

- definido por un alpha (hiper-parámetro).
- valor mínimo de confianza = 1

matrices de feedback implícito

P_{ui}

	item 1	item 2	item 3	….
usuario 1	1	0	0	
usuario 2	0	1	1	
….				

C_{ui}

	item 1	item 2	item 3	….
usuario 1	1.1	0	0	
usuario 2	0	1.5	1.02	
….				

$$\min_{x^*,y^*} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda(||x_u||^2 + ||y_i||^2)$$

Predicción de Factorización Matricial

Regularización L2

Factor de confianza del usuario u en el item i

Para feedback implícito incorporamos un factor de confianza Cui

Optimización de recomendación basada en factorización matricial con feedback implícito

1. Alternate Least Squares (ALS / WRMF)

2. Bayesian Personalized Ranking (BPR)

Filtrado Colaborativo con Retroalimentación Implícita

El artículo “Collaborative Filtering for Implicit Feedback Datasets” de Hu, Koren y Volinsky introduce el método de factorización matricial ALS (Alternating Least Squares).

Collaborative Filtering for Implicit Feedback Datasets

Yifan Hu
AT&T Labs – Research
Florham Park, NJ 07932

Yehuda Koren*
Yahoo! Research
Haifa 31905, Israel

Chris Volinsky
AT&T Labs – Research
Florham Park, NJ 07932

Filtrado Colaborativo con Retroalimentación Implícita

El modelo ALS permite:

1. **Aprender representaciones latentes de usuarios e ítems a partir de feedback implícito como clics o cantidad de reproducciones de una canción o serie de TV.**
2. **Escalar la cantidad de usuarios e ítems en el modelo al paralelizar el entrenamiento con Mínimos Cuadrados Alternados (ALS en inglés)**
3. **Evaluar el modelo propuesto en un caso real, como un proveedor de TV digital.**

Modelamiento de los clicks de usuarios

Al solo contar con “cantidad de veces que se ve un producto” y no con calificaciones explícitas, Hu et al. modelaron el entrenamiento de manera distinta a FunkSVD

1

Introducen una variable de preferencia p_{ui} que solo toma valores 1 o 0

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases}$$

donde r_{ui} indica cuántas veces el usuario *u* consumió el producto *i*

Modelamiento de confianza sobre las preferencias

Otro concepto que introduce el método de Hu et al. es el de confianza en la retroalimentación implícita. Como no es una señal explícita como las calificaciones en forma de 1 a 5 estrellas, se incorpora el concepto de confianza.

2

Introducen la variable c_{ui} que mide la confianza en observar la preferencia p_{ui} y se plantea de la siguiente forma:

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

donde r_{ui} indica cuántas veces el usuario u consumió el producto i, y α es un hiper-parámetro que se puede optimizar posteriormente, que indica la tasa a la que crece la confianza que se tiene en la preferencia a medida que aumenta el consumo de i

Función de pérdida de ALS

Finalmente, la nueva función que espera encontrar los vectores latentes x_* e y_* que minimizan la pérdida es:

$$\min_{x_\star, y_\star} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

p_{ui}: Preferencia del usuario u por el ítem i (1 / 0)
c_{ui}: confianza en la preferencia del usuario

x_u: vector de factores latentes del usuario
y_i : vector de factores latentes del ítem
λ : coeficiente de regularización

Paralelización del cálculo

El vector *Xu* para cada usuario u se calcula a continuación, donde la matriz Y es la matriz de factores de items, la matriz C^u es la matriz de "confianzas" de los ítems consumidos por el usuario u y p(u) son las preferencias del usuario u.

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$$

El vector *Yi* para cada ítem *i* se calcula a continuación, donde la matriz X es la matriz de factores de los usuarios, la matriz C^i es la matriz de "confianzas" de los items y p(i) son todas las preferencias por los items i.

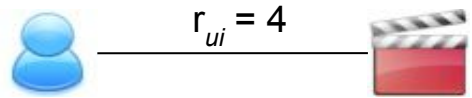
$$y_i = (X^T C^i X + \lambda I)^{-1} X^T C^i p(i)$$

Optimización de recomendación basada en factorización matricial con feedback implícito

1. Alternate Least Squares (ALS)

2. Bayesian Personalized Ranking (BPR)

Tipos de aprendizaje

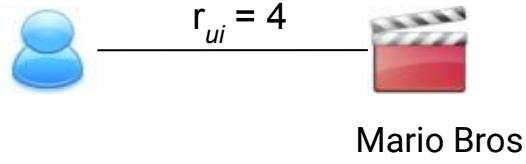


Mario Bros

Aprendizaje de preferencia directa

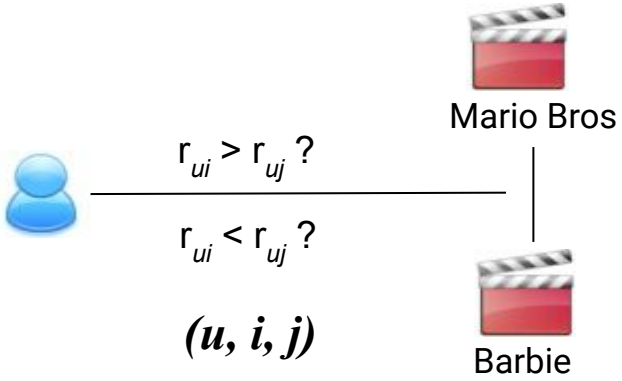
(point-wise learning)

Tipos de aprendizaje



Aprendizaje de preferencia directa

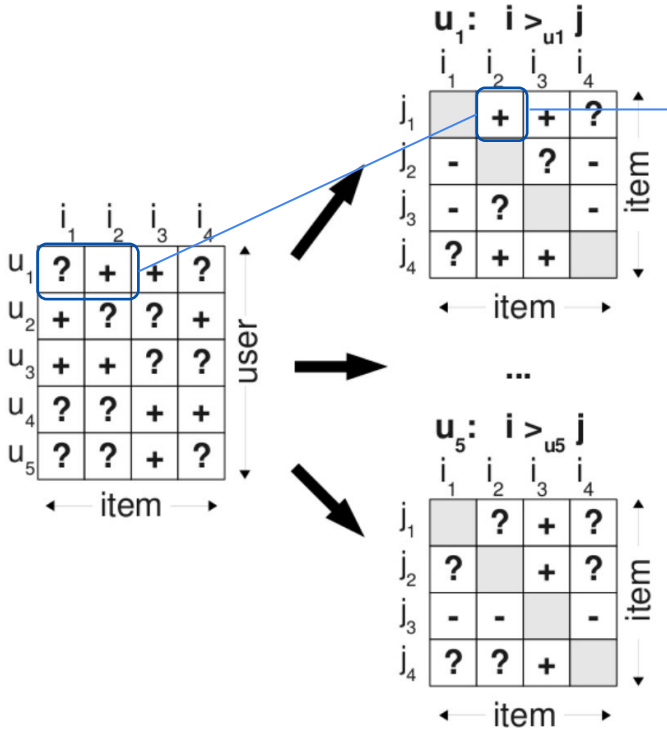
(point-wise learning)



Aprendizaje de preferencia por comparación de pares

(pairwise-wise learning)

Transformando retroalimentación en positiva y negativa



$$(u_1, i_2, j_1)$$

Cada usuario (ui) se representa como una matriz de preferencias.

$$\mathcal{D}_p = \{(u, i, j) \in \mathcal{D} | i \in \mathcal{I}_u \wedge j \in \mathcal{I} \setminus \mathcal{I}_u\}$$

BPR

BPR significa “Bayesian Personalized Ranking” :

1. Su objetivo es encontrar una función de ranking personalizado.

2. Uno de los métodos de “Aprender a rankear” más relevantes para el usuario objetivo.

3. BPR por sí mismo no es un algoritmo: más bien una función de pérdida y un framework para llevar a cabo la optimización.

4. Algoritmo = modelo + función de pérdida + aprendizaje

Ejemplo del algoritmo BPR-MF

BPR-MF es un algoritmo que usa el framework de BPR con factorización matricial

1. Modelo: MF (factorización matricial)

2. Función de pérdida: BPR-OPT

3. Aprendizaje: BPR-Learn (basado en SGD)

¿Cómo funciona BPR?

Los datos de entrenamiento son tuplas de la forma:

(u , i , j)

Que indica que el **usuario u** prefiere el **item i** por sobre el **item j**

BPR intenta clasificar el ítem positivo más alto que el ítem negativo.

Este proceso proporciona un ranking personalizado para cada usuario.

SUPUESTO FUERTE: items que aún no han sido consumido son menos preferidos.

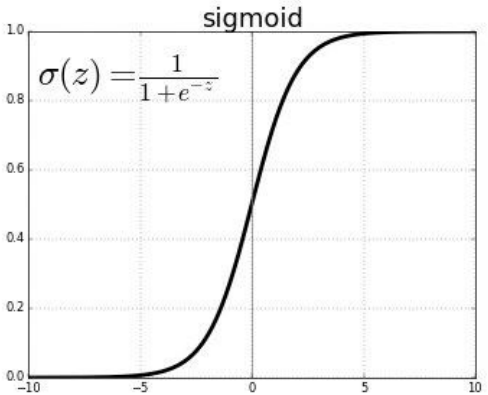
Optimización con BPR (BPR-opt)

BPR utiliza una pairwise loss function y modela la probabilidad de que el usuario u prefiera el ítem i sobre el ítem j

Factorización matricial de vectores de items preferidos (**Vi**) y no preferidos (**Vj**) por el usuario U. U , Vi y Vj comienzan aleatorios.

Xui = U X ViT (preferidos)

Xuj = U X VjT (no preferidos)



Diferencia entre resta de los resultados anteriores:

X_uij = Xui - Xuj

Función de pérdida:

BPR-opt = - log_prob + regularización

Donde:

log_prob = sum (log(sigmoid(Xuij)))

regularización = lambda * (sum(||U||^2) + sum(||Vi||^2) + sum(||Vj||^2))

Función de pérdida BPR

Recordemos la función de pérdida, donde $\hat{\Phi}_{ui}$ corresponde a la predicción de preferencia del usuario u por el ítem i:

$$\arg\max_{\Theta} \underbrace{\sum_{(u,i,j)\in\mathcal{D}_p} ln\,\sigma(\hat{\phi}_{ui} - \hat{\phi}_{uj}) - \lambda\|\Theta\|^2}_{BPR-OPT}$$

De esta ecuación nos interesa a continuación aprender los parámetros Θ que maximizan la expresión BPR-OPT, que no son otra cosa que los vectores latentes de usuarios y de ítems.

Aprendizaje de parámetros con BPR

BPR-Learn: Maximizar BPR-OPT

Para encontrar los valores de los parámetros Θ, derivamos BPR-OPT respecto a los parámetros

$$\frac{\partial \text{BPR-OPT}}{\partial \Theta} = \sum_{(u,i,j) \in D_S} \frac{\partial}{\partial \Theta} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \frac{\partial}{\partial \Theta} ||\Theta||^2$$

$$\propto \sum_{(u,i,j) \in D_S} \frac{-e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} - \lambda_\Theta \Theta$$

y con esto podemos obtener la regla de actualización para aplicar SGD (Descenso de Gradiente Estocástico)

Reglas de Actualización

La siguiente ecuación nos indica la regla de actualización de parámetros

$$\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \Theta \right)$$

- **Si recordamos la derivación de otros modelos como FunkSVD, veremos que se inicializan los valores de Θ de forma aleatoria, se fijan los valores de los hiperparámetros alfa y lambda, y se va actualizando de forma iterativa los valores de Θ.**

- **La forma final de la derivada de $\hat{x}_{uij}$ dependerá de cómo definamos el score de preferencia que el usuario u tiene por el ítem i al compararlo con el ítem j**

Código en pytorch de BPR

W[u,:] , W[i,:] , W[j, :]→ busca indices de usuarios (u), ítems preferidos (i) e ítems no preferidos (j) en matriz W

```
class BPR(nn.Module):
    def __init__(self, user_size, item_size, dim, weight_decay):
        super().__init__()
        self.W = nn.Parameter(torch.empty(user_size, dim))
        self.H = nn.Parameter(torch.empty(item_size, dim))
        nn.init.xavier_normal_(self.W.data)
        nn.init.xavier_normal_(self.H.data)
        self.weight_decay = weight_decay

    def forward(self, u, i, j):
        u = self.W[u, :]
        i = self.H[i, :]
        j = self.H[j, :]
        x_ui = torch.mul(u, i).sum(dim=1)
        x_uj = torch.mul(u, j).sum(dim=1)
        x_uij = x_ui - x_uj
        log_prob = F.logsigmoid(x_uij).sum()
        regularization = self.weight_decay * (u.norm(dim=1).pow(2).sum() + i.norm(dim=1).pow(2).sum() + j.norm(dim=1).pow(2).sum())
        return -log_prob + regularization
```

Matriz H se actualiza de manera paralela con ítems preferidos (i) y no preferidos (j).

Recomendación

```python
class BPR(nn.Module):
    def __init__(self, user_size, item_size, dim, weight_decay):
        super().__init__()
        self.W = nn.Parameter(torch.empty(user_size, dim))
        self.H = nn.Parameter(torch.empty(item_size, dim))
        nn.init.xavier_normal_(self.W.data)
        nn.init.xavier_normal_(self.H.data)
        self.weight_decay = weight_decay

    def forward(self, u, i, j):
        u = self.W[u, :]
        i = self.H[i, :]
        j = self.H[j, :]
        x_ui = torch.mul(u, i).sum(dim=1)
        x_uj = torch.mul(u, j).sum(dim=1)
        x_uij = x_ui - x_uj
        log_prob = F.logsigmoid(x_uij).sum()
        regularization = self.weight_decay * (u.norm(dim=1).pow(2).sum() + i.norm(dim=1).pow(2).sum() + j.norm(dim=1).pow(2).sum())
        return -log_prob + regularization

    def recommend(self, u):
        u = self.W[u, :]
        x_ui = torch.mm(u, self.H.t())
        pred = torch.argsort(x_ui, dim=1)
        return pred
```

Recomendación:
Producto punto entre vectores aprendidos de usuario (W) y matriz de ítems (H) transpuesta (T).

Predicción final ordena items por similaridad.

Trucos de BPR

- Como son muchas combinaciones de (u,i,j) los autores hacen sampling aleatorio con repetición. (bootstrap sampling)

- La actualización de los factores latentes de usuario e item se actualizan con **descenso de gradiente** , **buscando maximizar la distancia entre los preferidos sobre los no preferidos.**