

## Otros problemas

---

Otros problemas

---

## Sistemas de recomendación

# Sistemas de recomendación

## Introducción

### Definición

Los Sistemas de recomendación (SR) son algoritmos que ayudan a los usuarios a descubrir elementos relevantes (películas, textos, productos, etc.).

Los objetivos son múltiples:

- Aumentar las ventas.
- Facilitar las búsquedas.
- Mejorar la experiencia de usuario.
- Incrementar la duración de la interacción del usuario con la plataforma.

# Sistemas de recomendación

## Introducción

### Tipos

Según metodología empleada para realizar la recomendación, existen varios tipos.

- **Recomendaciones no personalizadas:** Iguales para todos los usuarios.
  - SR basados en popularidad.
- **Recomendaciones personalizadas:** Únicas para cada usuario.
  - SR colaborativos.
  - SR basados en contenido.

# Sistemas de recomendación

No personalizados

## Basados en popularidad

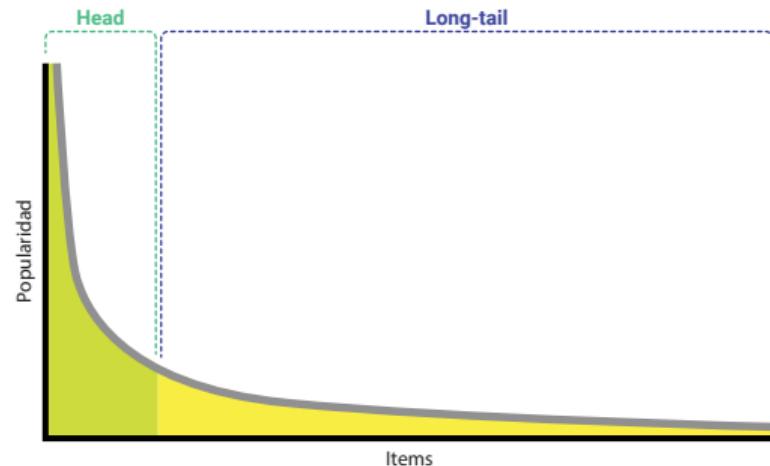
Recomiendan, para cualquier usuario, los elementos más consumidos/vistos/comprados entre todos los usuarios. Útiles cuando no se dispone de información específica del usuario.

Detalles:

- Fáciles de implementar.
- No recomienda items 'long-tail'.
- Buena solución para casos de 'cold-start'.

Ejemplos:

- Productos más vendidos.
- Artículos más leídos.



# Sistemas de recomendación

## Personalizados

### Filtros colaborativos

Hacen recomendaciones personalizadas basadas en usuarios o items **similares**.

Detalles:

- Utiliza las valoraciones **explícitas** o **implícitas**.
  - **Explícitas**: Un usuario valora el ítem con Like, estrellas, review, ...
  - **Implícitas**: Un usuario ha visto un video entero, pero no lo valoró.

Tipos:

- Usuario-usuario.
- Item-item.

# Sistemas de recomendación

## Personalizados

### Filtros colaborativos: Usuario-usuario

Replica el funcionamiento de las recomendaciones en la vida real. Nos dejamos asesorar por gente similar a nosotros.

Funcionamiento:

- 1 Para cada usuario, buscar su similitud con el actual.
- 2 Seleccionar los usuarios más parecidos.
- 3 Ponderar los votos de los usuarios seleccionados para el producto a recomendar.

Ejemplo

Un usuario compró  $p_1, p_2$  y  $p_3$ . Si otro usuario solo compró  $p_1$  y  $p_2$  le recomendaremos  $p_3$ .

# Sistemas de recomendación

Personalizados

## Filtros colaborativos: Usuario-usuario

Replica el funcionamiento de las recomendaciones en la vida real. Nos dejamos asesorar por gente similar a nosotros.

Funcionamiento:

	Item 1	Item 2	Item 3	Item 4	Item 5
Usuario 1	5	3	4	4	?
Usuario 2	3	-	2	-	5
Usuario 3	4	3	-	3	-
Usuario 4	3	1	1	5	4
Usuario 5	5	5	5	2	3

# Sistemas de recomendación

Personalizados

## Filtros colaborativos: Usuario-usuario

Replica el funcionamiento de las recomendaciones en la vida real. Nos dejamos asesorar por gente similar a nosotros.

Funcionamiento:

	Item 1	Item 2	Item 3	Item 4	Item 5
Usuario 1	5	3	4	4	?
Usuario 2	3	-	2	-	5
Usuario 3	4	3	-	3	-
Usuario 4	3	1	1	5	4
Usuario 5	5	5	5	2	3

# Sistemas de recomendación

## Personalizados

### Filtros colaborativos: Usuario-usuario

Replica el funcionamiento de las recomendaciones en la vida real. Nos dejamos asesorar por gente similar a nosotros.

Funcionamiento:

	Item 1	Item 2	Item 3	Item 4	Item 5
Usuario 1	5	3	4	4	?
Usuario 2	3	-	2	-	
Usuario 3	4	3	-	3	
Usuario 4	3	1	1	5	4
Usuario 5	5	5	5	2	3

# Sistemas de recomendación

## Personalizados

### Filtros colaborativos: Item-item

Buscar la valoración que tienen items similares al que se pretende recomendar.

Funcionamiento:

- 1 Determinar la similitud entre los items.
- 2 Seleccionar los más similares.
- 3 Ponderar las valoraciones dadas por el usuario a los items.

	Item 1	Item 2	Item 3	Item 4	Item 5
Usuario 1	5	3	4	4	?
Usuario 2	-	-	3	-	5
Usuario 3	4	3	-	3	-
Usuario 4	3	1	3	2	4
Usuario 5	5	2	5	2	3

# Sistemas de recomendación

## Personalizados

### Basados en contenido

Utilizan el contenido de los ítems o usuarios (no solo las interacciones) para representar cada uno de ellos y buscar similitudes.

Ejemplos de contenido en usuarios:

- ① Edad, género, dirección, ...
- ② Historial de consumo.
- ③ Imágenes que ha tomado.

Ejemplos de contenido en ítems:

- ① Descripción del producto.
- ② Géneros de una película.
- ③ Localización de un restaurante.
- ④ Reseñas (texto, imágenes, ...).

Otros problemas

---

## Sistemas de recomendación: Arquitecturas

# Sistemas de recomendación

## Arquitecturas

### Objetivo

El objetivo final es llenar todos los huecos de una matriz usuario item de forma coherente con los valores que ya existen.

En otras palabras, buscamos aprender  $f_{\theta}(\mathbf{u}, \mathbf{p})$ , siendo:

- $\mathbf{u}$  la representación de un usuario del conjunto de usuarios  $\mathbf{U}$ .
- $\mathbf{p}$  la representación de un ítem del conjunto de ítems  $\mathbf{P}$ .
- $\theta$  los parámetros de la función.

La salida de la función depende del problema a resolver, típicamente será:

- Numérica si intentamos predecir valoraciones (regresión o ranking).
- Categórica si intentamos predecir me gusta/no me gusta (clasificación).

# Sistemas de recomendación

## Arquitecturas

### Objetivo

El objetivo final es llenar todos los huecos de una matriz usuario item de forma coherente con los valores que ya existen.

Ejemplo de regresión:  $f_{\theta}(\mathbf{u}, \mathbf{p}) \rightarrow [0, 5]$

	Item 1	Item 2	Item 3	Item 4	Item 5
Usuario 1	5	-	-	4	-
Usuario 2	-	-	3	-	5
Usuario 3	4	0	-	3	-
Usuario 4	-	-	3	-	4
Usuario 5	5	-	-	2	-

# Sistemas de recomendación

## Arquitecturas

### Objetivo

El objetivo final es llenar todos los huecos de una matriz usuario item de forma coherente con los valores que ya existen.

Ejemplo de clasificación:  $f_{\theta}(\mathbf{u}, \mathbf{p}) \rightarrow \{0, 1\}$

	Item 1	Item 2	Item 3	Item 4	Item 5
Usuario 1	1	-	-	0	-
Usuario 2	-	-	0	-	1
Usuario 3	1	1	-	0	-
Usuario 4	-	-	1	-	1
Usuario 5	0	-	-	1	-

# Sistemas de recomendación

## Arquitecturas

### Factorización de matrices

El método más utilizado para aprender la función  $f_{\theta}(\mathbf{u}, \mathbf{p})$  suele ser la factorización de matrices.

Este método:

- Proyecta los usuarios  $\mathbf{u} \in \mathbb{R}^{|U|}$  y los ítems  $\mathbf{p} \in \mathbb{R}^{|P|}$  en un espacio de dimensión  $\mathbb{R}^k$ .
- Aprende las matrices  $\mathbf{W}$  y  $\mathbf{V}$  que retornan las proyecciones  $\mathbf{Wu}$  y  $\mathbf{Vp}$ .
- La compatibilidad entre usuario y producto proviene del producto escalar de las proyecciones.

### Tipo de problema

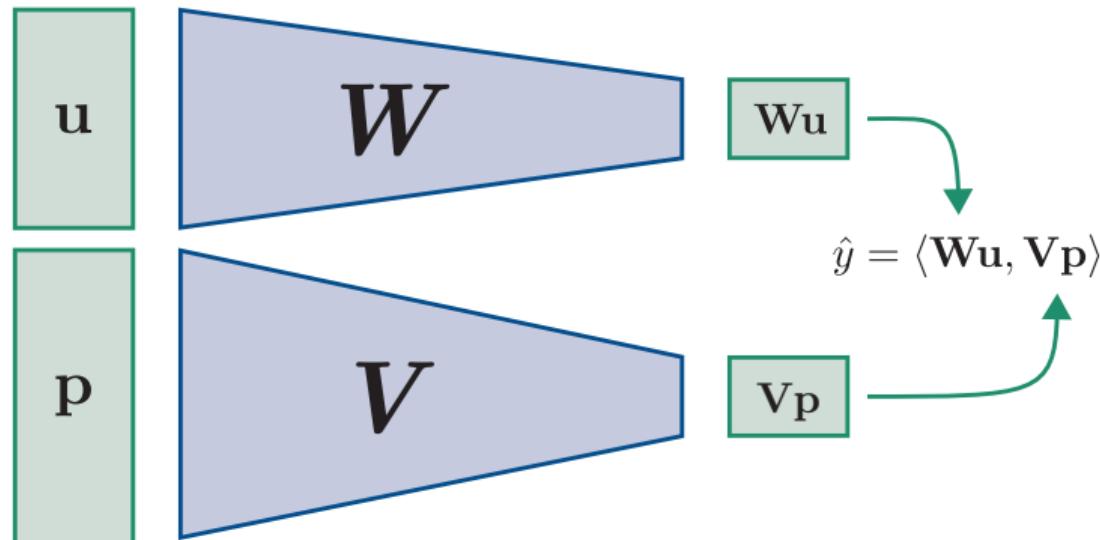
Podemos utilizar (basado en contenido) o no (filtro colaborativo) información sobre el contenido de los ítems y usuarios.

# Sistemas de recomendación

## Arquitecturas

### Factorización de matrices

El método más utilizado para aprender la función  $f_{\theta}(\mathbf{u}, \mathbf{p})$  suele ser la factorización de matrices.



# Sistemas de recomendación

## Arquitecturas

### Factorización de matrices

El método más utilizado para aprender la función  $f_{\theta}(\mathbf{u}, \mathbf{p})$  suele ser la factorización de matrices.

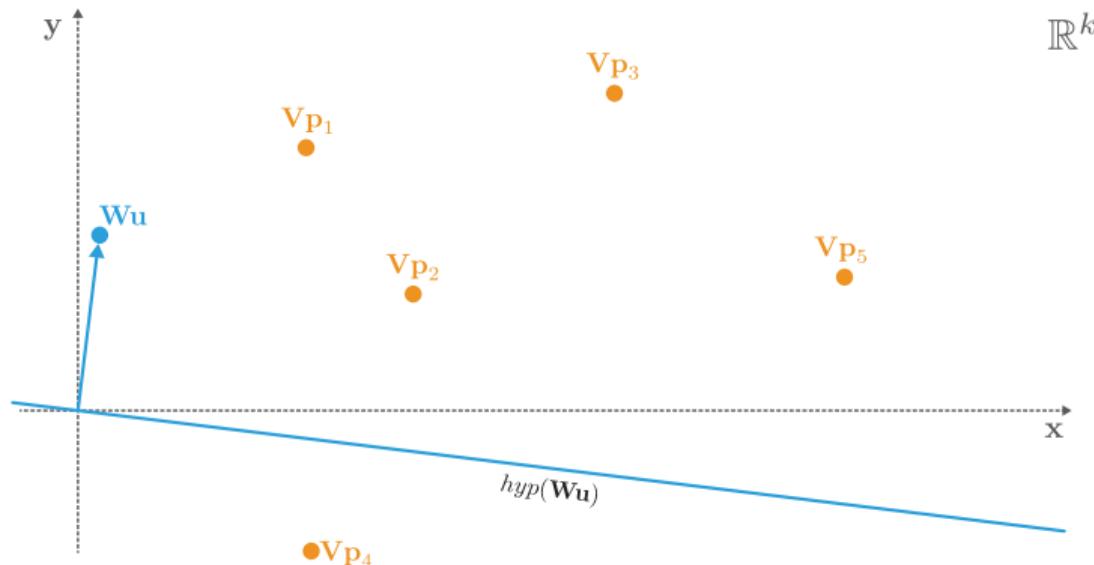


# Sistemas de recomendación

## Arquitecturas

### Factorización de matrices

El método más utilizado para aprender la función  $f_{\theta}(\mathbf{u}, \mathbf{p})$  suele ser la factorización de matrices.

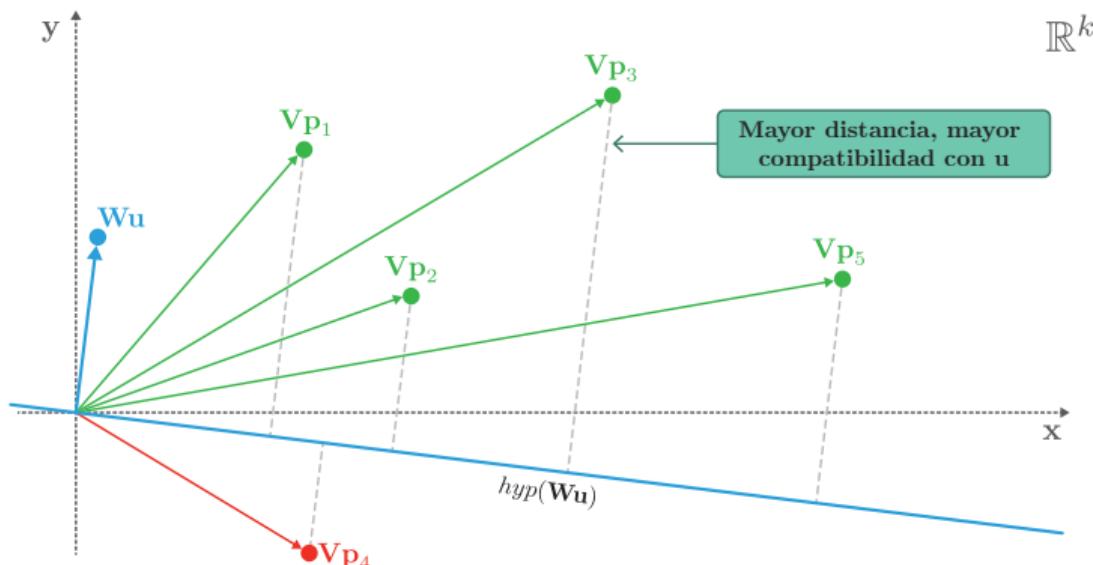


# Sistemas de recomendación

## Arquitecturas

### Factorización de matrices

El método más utilizado para aprender la función  $f_{\theta}(\mathbf{u}, \mathbf{p})$  suele ser la factorización de matrices.

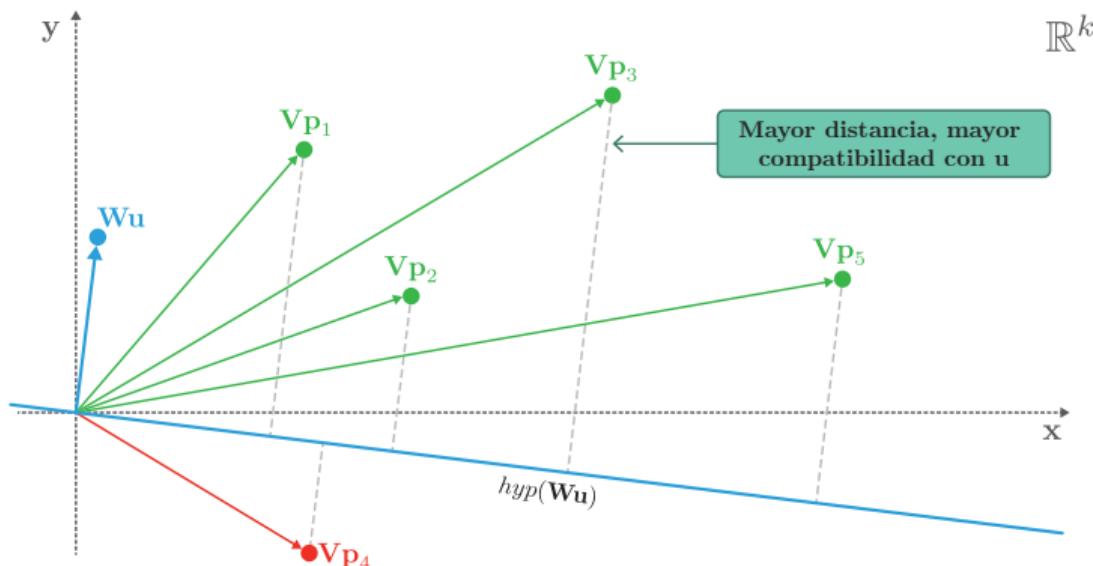


# Sistemas de recomendación

## Arquitecturas

### Factorización de matrices

El método más utilizado para aprender la función  $f_{\theta}(\mathbf{u}, \mathbf{p})$  suele ser la factorización de matrices.



# Sistemas de recomendación

## Arquitecturas

### Clasificación multi-etiqueta

Si no existe un alto número de ítems, se puede plantear el problema de recomendación como una clasificación multi-etiqueta.

Detalles:

- Existirán tantas clases como ítems.
- Suele ser un problema complejo de aprender (muchos ceros).
- La lista final de ítems recomendados proviene de ordenar las predicciones de mayor a menor.

# Sistemas de recomendación

## Arquitecturas

Si planteamos el problema como un *filtro colaborativo*, cualquiera de las propuestas anteriores se puede resolver mediante redes poco profundas.

### Deep Learning

En el momento que utilicemos el *contenido* para codificar (usuarios, items o ambos), necesitaremos complicar las arquitecturas vistas manteniendo la esencia básica.

Posibles variaciones:

- *Utilizar imágenes*: Añadir redes CNN para proyectar en un espacio común.
- *Utilizar texto*: Añadir redes recurrentes o mecanismos de atención.

Otros problemas

---

## Arquitecturas diseñadas para el procesamiento de conjuntos

# Arquitecturas diseñadas para conjuntos

Existen arquitecturas de red especialmente diseñadas para tratar y modelar **conjuntos**. La clave en este tipo de problemas es que el orden de los elementos del conjunto no debería de influenciar la salida de la red. Algunas aplicaciones podrían ser:

- Detección de anomalías en un conjunto.
- Predicción de estadísticos de un conjunto.
- Cuantificación



# Arquitecturas diseñadas para conjuntos

## Definición del problema:

Dado un conjunto de entrenamiento  $D = (\{x_i\}_{i=1}^n, y)$ , es decir, un conjunto de muestras etiquetadas, tratamos de aprender un modelo que sea capaz de predecir una nueva muestra de test  $T = \{x_i\}_{i=1}^m$ .

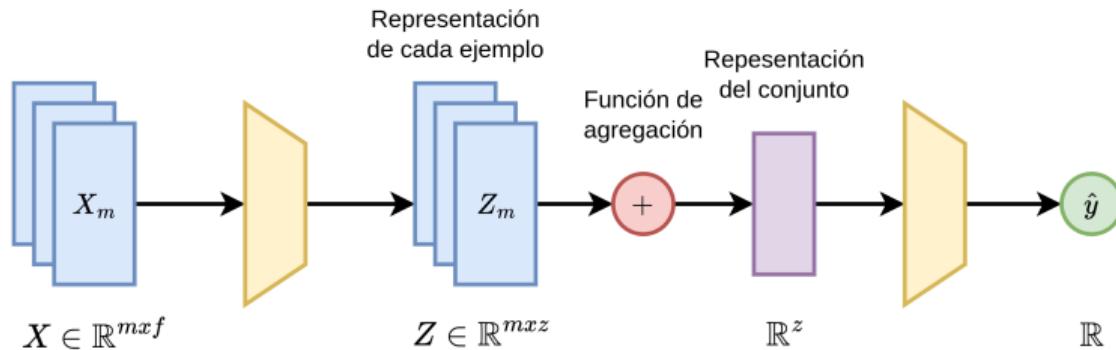
Para resolver este problema necesitamos arquitecturas que sean capaces de procesar conjuntos de datos y representarlos de manera **invariante al orden**. Es decir, la evaluación de la red para una muestra  $T$  debería de cumplir:

$$f(\{x_i\}_{i=1}^m) = f(\phi(\{x_i\}_{i=1}^m)),$$

siendo  $\phi$  una permutación cualquier de los ejemplos de la muestra.

# Arquitecturas diseñadas para conjuntos: DeepSets

DeepSets es una arquitectura de red que propone una capa de representación basada en **funciones de agregación simples**:



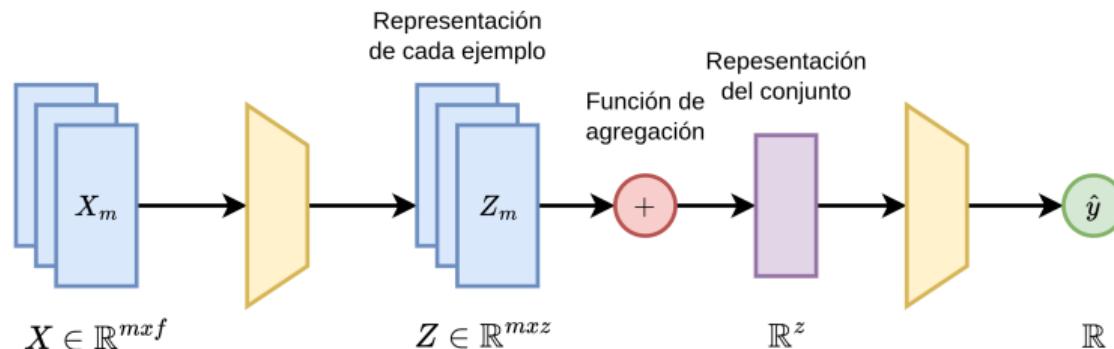
donde  $m$  sería el tamaño del conjunto de entrada y  $f$  la dimensión de cada ejemplo.

## Funciones de agregación

Las funciones de agregación a utilizar pueden ser: suma, máximo, mínimo, media, mediana, etc.

# Arquitecturas diseñadas para conjuntos: DeepSets

Las funciones de agregación utilizadas por DeepSets tienen como objetivo obtener una **representación única de la muestra en un espacio latente**.



A partir de esta representación la segunda parte de la red puede aprender a asociar estos vectores con su etiqueta final.

# Arquitecturas diseñadas para conjuntos: DeepSets

Algunos puntos importantes sobre DeepSets son:

- La arquitectura está pensada funcionar como un **aproximador universal** de cualquier función que trabaje con conjuntos.
- La función de agregación es una manera **demasiado simple** de representar los ejemplos del conjunto, perdiéndose mucha información en el proceso.
- La función de agregación no captura **iteraciones entre los elementos del conjunto**.
- Algunas funciones de agregación como la suma pueden no sirven para **conjuntos de tamaño variable**.

# Arquitecturas diseñadas para conjuntos: SetTransformers

SetTransformers es otra arquitectura diseñada para trabajar con conjuntos que explota el concepto de **atención** de los **transformers**.

- **Modelan relaciones** entre los diferentes ejemplos del conjunto.
- Aprovechan la atención para capturar **relaciones entre elementos**.
- Flexibilidad para manejar **conjuntos de tamaños variables**.
- Mejoran el rendimiento de DeepSets.

## Arquitecturas diseñadas para conjuntos: SetTransformers

La idea básica de los SetTransformers es calcular la atención entre los elementos del conjunto. El principal problema de esta arquitectura es que su complejidad es  $O(n \times n)$ , siendo  $n$  el número de ejemplos del conjunto.

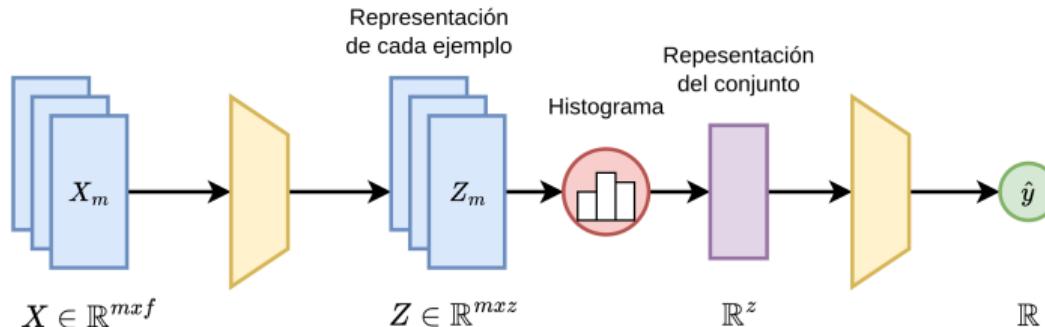
Para resolver este problema los autores de SetTransformers calculan la atención de cada ejemplo a una serie de puntos llamados **inducing points**, reduciendo por tanto la complejidad a  $O(n \times i)$ , siendo  $i$  el número de inducing points.

### Nota importante

Ten en cuenta que **los transformers por defecto son invariantes al orden**. Es por ello que en problemas de NLP se les añade la posición de cada palabra en la frase para poder capturar esa información que es importante a la hora de procesar texto.

# Arquitecturas diseñadas para conjuntos: HistNetQ

HistNetQ es un arquitectura de red pensada específicamente para **cuantificación**. En esta arquitectura se utiliza como capa de representación de la muestra **histogramas**. La idea es que un histograma es una herramienta de representación adecuada para este problema.

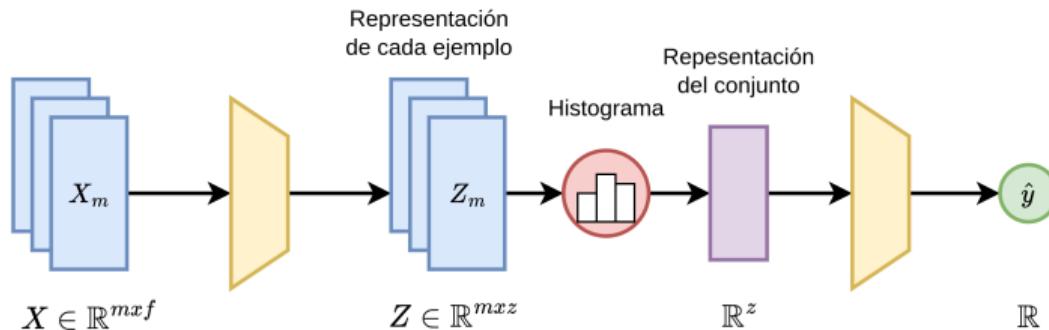


## Cuantificación

Recuerda que la cuantificación consiste en estimar la **prevalencia de las clases** en un conjunto de test.

# Arquitecturas diseñadas para conjuntos: HistNetQ

La arquitectura es similar a la de **DeepSets** pero utilizando histogramas para representar las muestras. La red calcula un histograma por cada una de las características de la representación de ejemplos.

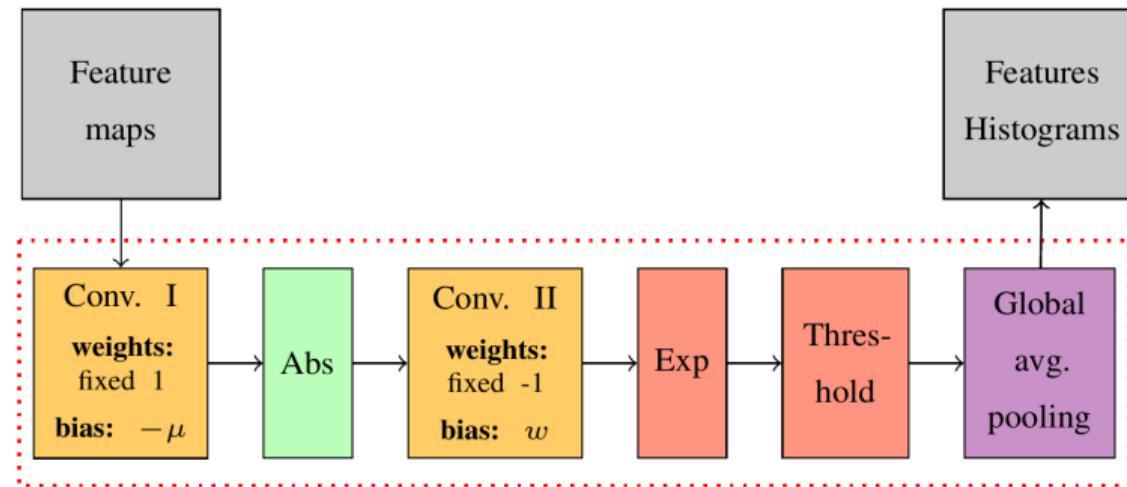


## Como usar histogramas en una red

El principal problema de usar histogramas es que **el cálculo de un histograma no es una operación diferenciable**.

# Arquitecturas diseñadas para conjuntos: HistNetQ

La solución es utilizar **histogramas diferenciables**.



## Histogramas diferenciables

Los histogramas diferenciables utilizan operaciones comunes (como las convoluciones), para calcular una aproximación a un histograma.