cf2vec: Collaborative Filtering algorithm selection using graph distributed representations

Juan Valdivia

Advanced Analytics Prod

juan.valdivia@itau-unibanco.com.br

11/10/2021

Agenda

1. Introdução
   - Motivação
   - *No-free lunch theorem* e viés
2. *Sistemas de recomendação*
   - *Collaborative filtering* (*CF*)
3. *Metalearning*
   - Seleção de algoritmos
   - *Metafeatures* para *CF*
   - Recomendação de algoritmos usando *Metalearning*
   - *k-Nearest Neighbors Ranking Method*
4. *cf2vec*
5. *Resultados*

Objetivos

- Selecionar algoritmos para uma determinada tarefa pode ser um processo experimental árduo (abordagem de tentativa e erro), custoso, subjetivo, podendo levar ao *overfitting* e dificulta a reprodução de experimentos.
- Desafio: falta de orientação sobre qual algoritmo de *Collaborative Filtering* seria mais adequado para uma determinada tarefa (por exemplo: *Rating Prediction*).
- Solução: abordagem de *Metalearning* (*MtL*) que consiste basicamente em extrair metaconhecimento de tarefas resolvidas com sucesso por *ML* para usá-lo na solução de novas tarefas.

Porque *Metalearning*?

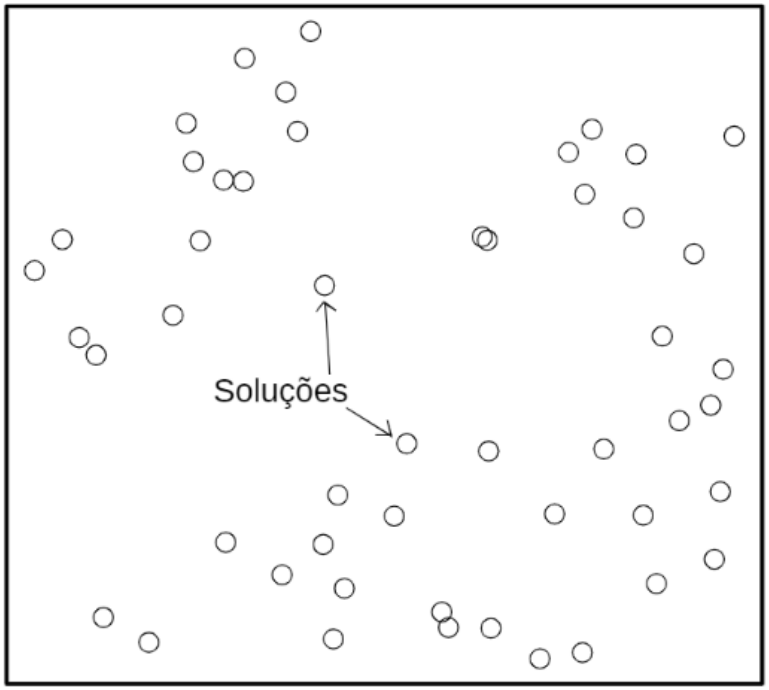
no-free lunch theorem

Existe um espaço de hipóteses H (conjunto infinito de possíveis funções h). O *no-free lunch theorem* diz que não existe um único classificador $h \subset H$ que seja adequado para todas as situações, pois cada algoritmo faz suposições sobre os dados.

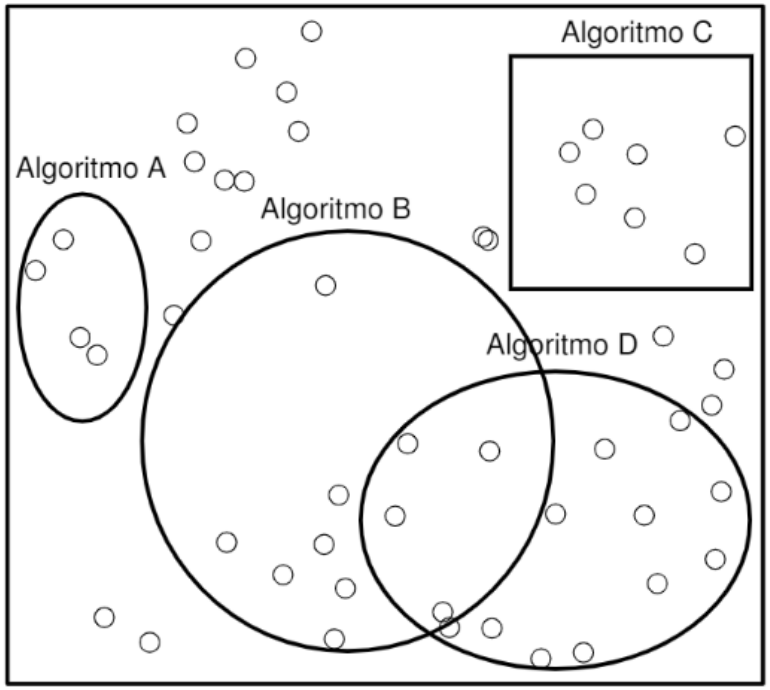
Viés

Preferência na escolha da função (hipótese) que deve ser usada. Essa preferência é muito importante para garantir o aprendizado, pois ela garante que se tenha um conjunto finito de funções a serem testadas.

Espaço de hipóteses e viés



(a) Espaço de busca.



(b) Exemplos de possíveis vieses de alguns algoritmos.

Figure: Espaço de hipóteses e viés

Por que sistemas de recomendação?

- Necessidade de diminuir a sobrecarga de informação em serviços online para oferecer experiências cada vez mais personalizadas aos usuários.
- Utéis para aumentar a satisfação dos clientes, agregação de valor no oferecimento de produtos, conhecer melhor o comportamento dos usuários e ajudam a aumentar as receitas das empresas.
- Mas como filtrar esses grandes fluxos de dados para permitir que os usuários encontrem os itens com mais facilidade?

Collaborative filtering

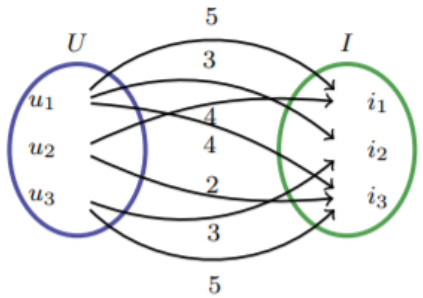
- Abordagem de sistemas de recomendação que consiste em filtrar informações ou padrões usando técnicas que envolvem colaboração entre vários usuários, agentes e fontes de dados.
- *CF* usa a interação dos usuários com os itens por meio de um *feedback*.
- *Feedback* explícito: usuários tomam a iniciativa de indicar seus interesses.
- Problema: usuários podem não ter interesse em avaliar os itens
- *Feedback* ímplicito: presente no comportamento dos usuários nos serviços online (histórico de compras, histórico de navegação e cliques)
- Problema: comportamento do usuário pode não refletir necessariamente de maneira direta a sua opinião sobre o produto.

Como são os conjuntos de dados?

- Para o caso do *Feedback* explícito, podemos representar o conjunto de dados como um matriz de *ratings* ou como um grafo bipartido.

	i_1	i_2	i_3
u_1	**5**	**3**	**4**
u_2	**4**	...	**2**
u_3	...	**3**	**5**

(a) Rating Matrix. Rows represent users U, while columns represent items I. Some cells have the rating assigned by an user to an item.



(b) Bipartite Graph. The graph has two node subsets, representing users U and items I. Ratings are weighted edges between nodes of both subsets.

Figure: Representações de um conjunto de dados de filtragem colaborativa

Categorias de técnicas de *CF*

- baseadas em memória: computam funções de similaridade entre usuários e itens para fazer as recomendações baseadas nesses valores de similaridade (algoritmos baseados em *nearest neighbors*).
- baseadas em modelo: por exemplo *Matrix factorization* (*MF*), que é um modelo de fator latente que lida bem com dados esparsos e de grande escala. Técnicas baseadas em modelo podem ser adaptadas com flexibilidade para usar redes neurais.
- Técnicas de *CF* hibridas: combinam técnicas de *CF* baseadas em memória e modelo.

Tarefas de *CF*

Podem ser diferenciadas com base nos tipos de *feedback* e dados de entrada:

- *Rating Prediction*: previsão das classificações explícitas (*rating*) dadas pelos usuários aos itens. As métricas *NMAE* e *RMSE* podem ser usadas.
- *Item Recommendation*: recomendação de uma lista de itens que é mais adequada para um determinado usuário. As métricas *NDCG* e *AUC* podem ser usadas.

O que é necessário para selecionar algoritmos usando *Metalearning*

- Conjunto de diversos *datasets* p_s que pertencem a um problema P_s.
- *Metafeatures* (F): extraem propriedades dos *datasets* p_s na forma $f(p_s)$. Essas propriedades devem prover evidência sobre o desempenho futuro das técnicas investigadas, devem distinguir problemas com diferentes níveis de dificuldade e preferencialmente devem ser calculadas com um baixo custo computacional.
- Conjunto de algoritmos A: cada algoritmo $a \subset A$ é aplicado a cada conjunto de dados p_s, ou seja tem-se, $a(p_s)$.
- Métrica de avaliação: para cada métrica $m \subset M$ é feita a avaliação dos modelos obtidos após a aplicação de cada um dos algoritmos a nos conjuntos de dados p_s, ou seja, $m(a(p_s))$.

Arcabouço de seleção de algoritmos

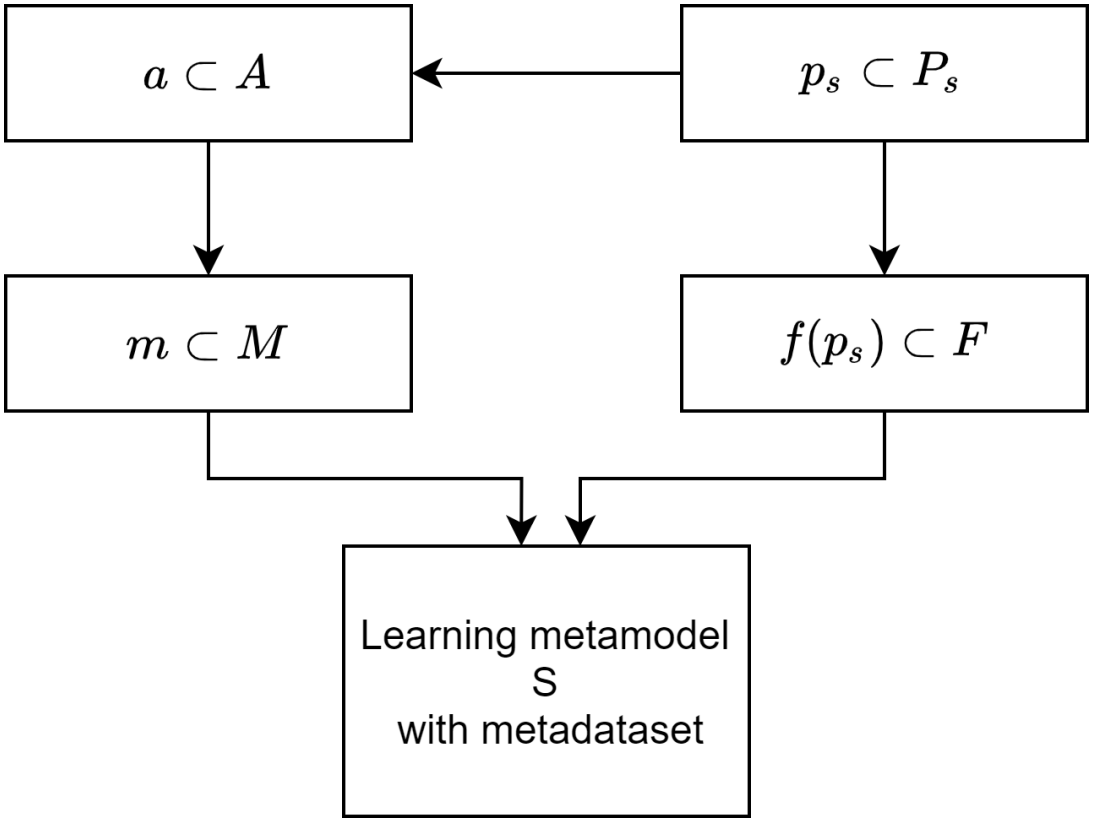


Figure: Arcabouço de seleção de algoritmos

Principais *metafeatures* para *datasets* de *CF*

- *Rating Matrix metafeatures*: estatísticas (min, max, moda, curtose, entropia) sobre as linhas/colunas, número de usuários, estatísticas sobre os *ratings*.
- *Subsampling landmarkers*: extraídas usando o desempenho estimado para amostras aleatórias dos *datasets* originais.
- *Graph metafeatures*: os *datasets* de *CF* são modelados como um grafo e as características extraídas são medidas usadas em redes complexas.
- *Comprehensive metafeatures*: todas as abordagens anteriores para extrair *metafeatures* são agregadas. Para obter as *metafeatures* mais significativas é usada seleção de *features* por correlação.

Como recomendar algoritmos usando *Metalearning*?

- Objetivo: construir um sistema de recomendação de algoritmos que seja capaz de reduzir o tempo de experimentos ao ter um conjunto de algoritmos que quando comparados com os melhores algoritmos possíveis tenham uma mínima perda na qualidade dos resultados.
- Necessário: ter um *metadaset* que contenha *metafeatures* (dados que descrevam as características dos problemas) e *metatargets* (dados que descrevam o desempenho dos algoritmos).
- Assim, os *metatargets* podem ser usados para fazer *ranking* de algoritmos.

Como seria esse sistema?

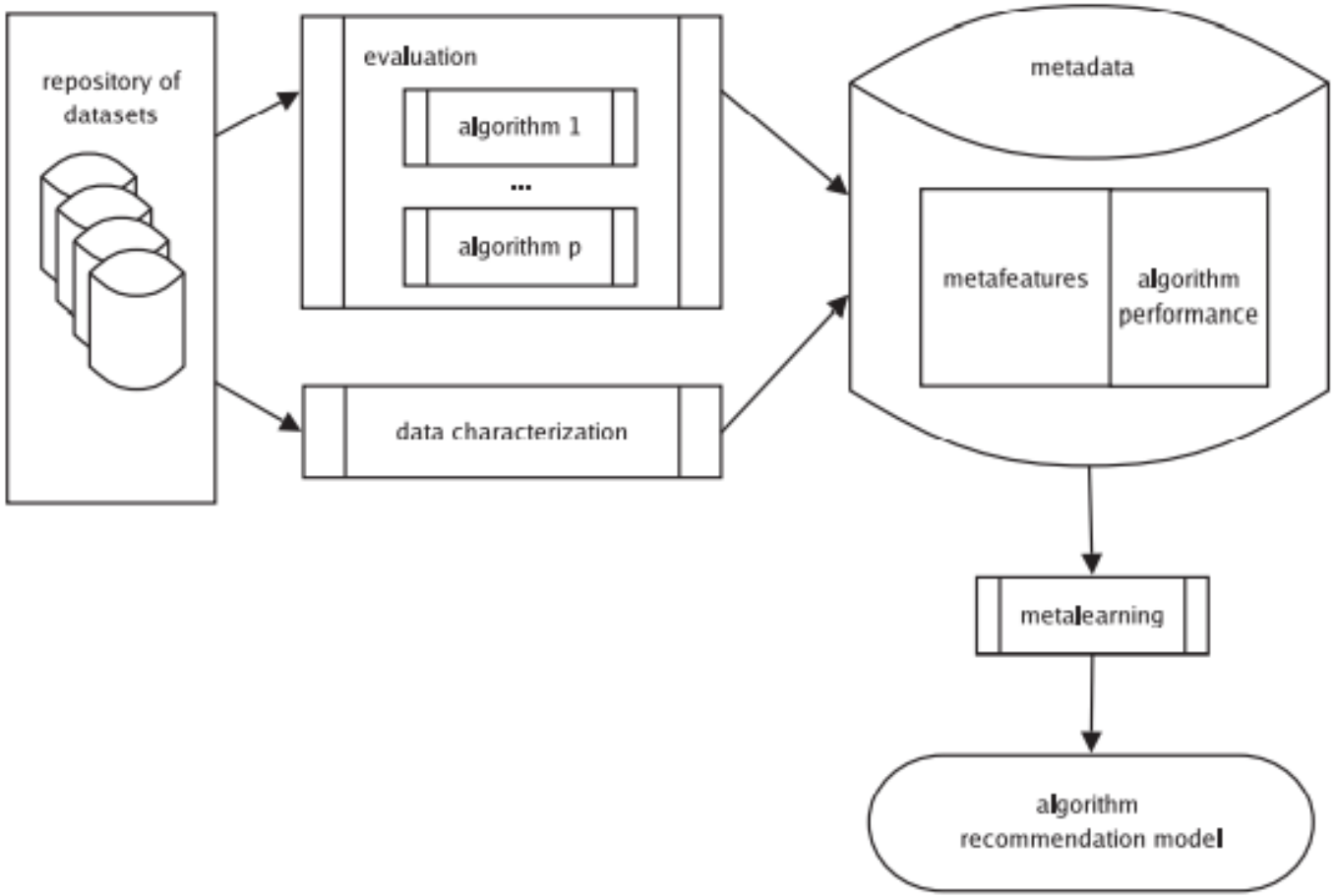


Figure: Sistema de recomendação de algoritmos usando *Metalearning*

k-Nearest Neighbors Ranking Method

Para treinar *metadatasets* é necessário um *meta-learner*. O *k-Nearest Neighbors Ranking Method* é uma adaptação do *k-Nearest Neighbors* para a tarefa de fazer um *ranking* de algoritmos.

1. Selecionar o conjunto de k exemplos mais similares para um novo exemplo (no caso um *dataset* em *MtL*) com relação às *metafeatures*. Para isso deve ser usada uma métrica de distância para medir a similaridade.
2. Combinar os valores dos *targets* dos k exemplos selecionados para gerar uma previsão para o novo exemplo (*dataset*).

Como o sistema de recomendação de algoritmos do *cf2vec* funciona?

- Converte a matriz de filtragem colaborativa em um grafo bipartido.
- Amostra o grafo bipartido para reduzir a complexidade do problema.
- Aprende *embeddings* (representações distribuídas para cada um dos grafos) usando *graph2vec*.
- Utiliza os *embeddings* aprendidos para cada um dos grafos como *metafeatures* do *metadataset* que será construído.

framework do *cf2vec*

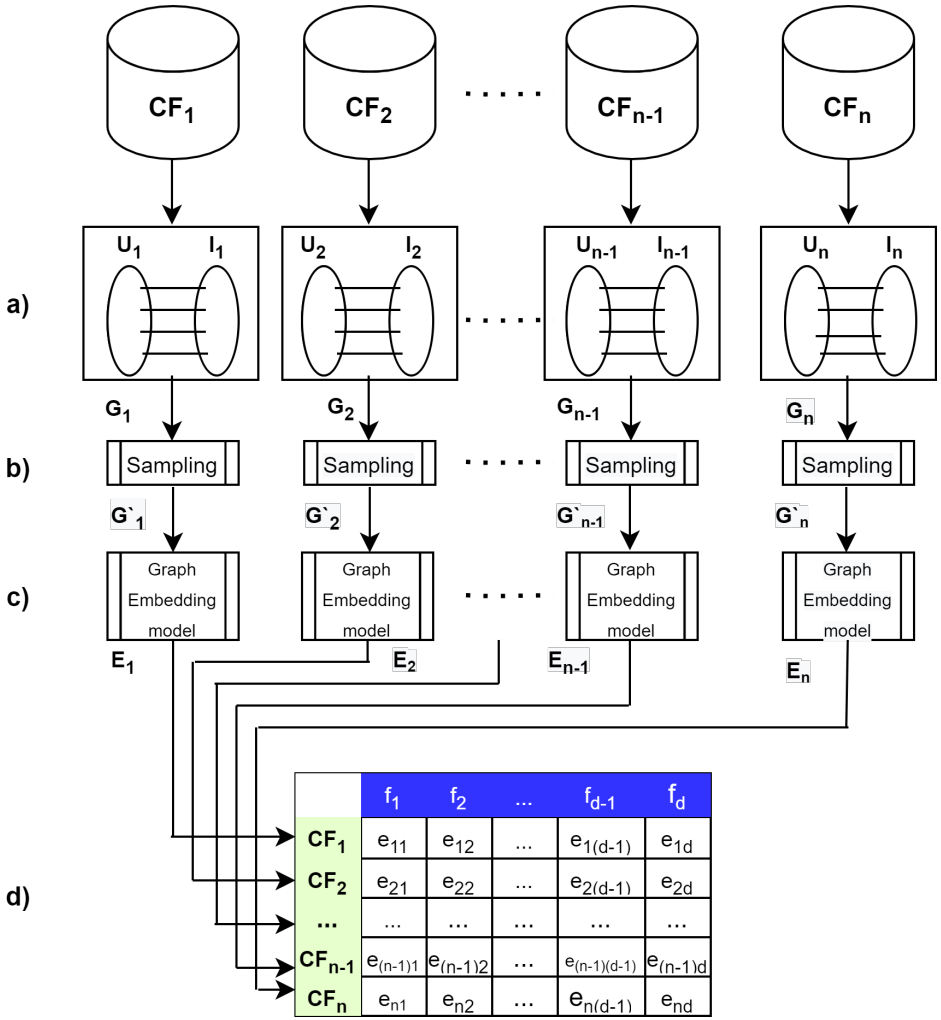
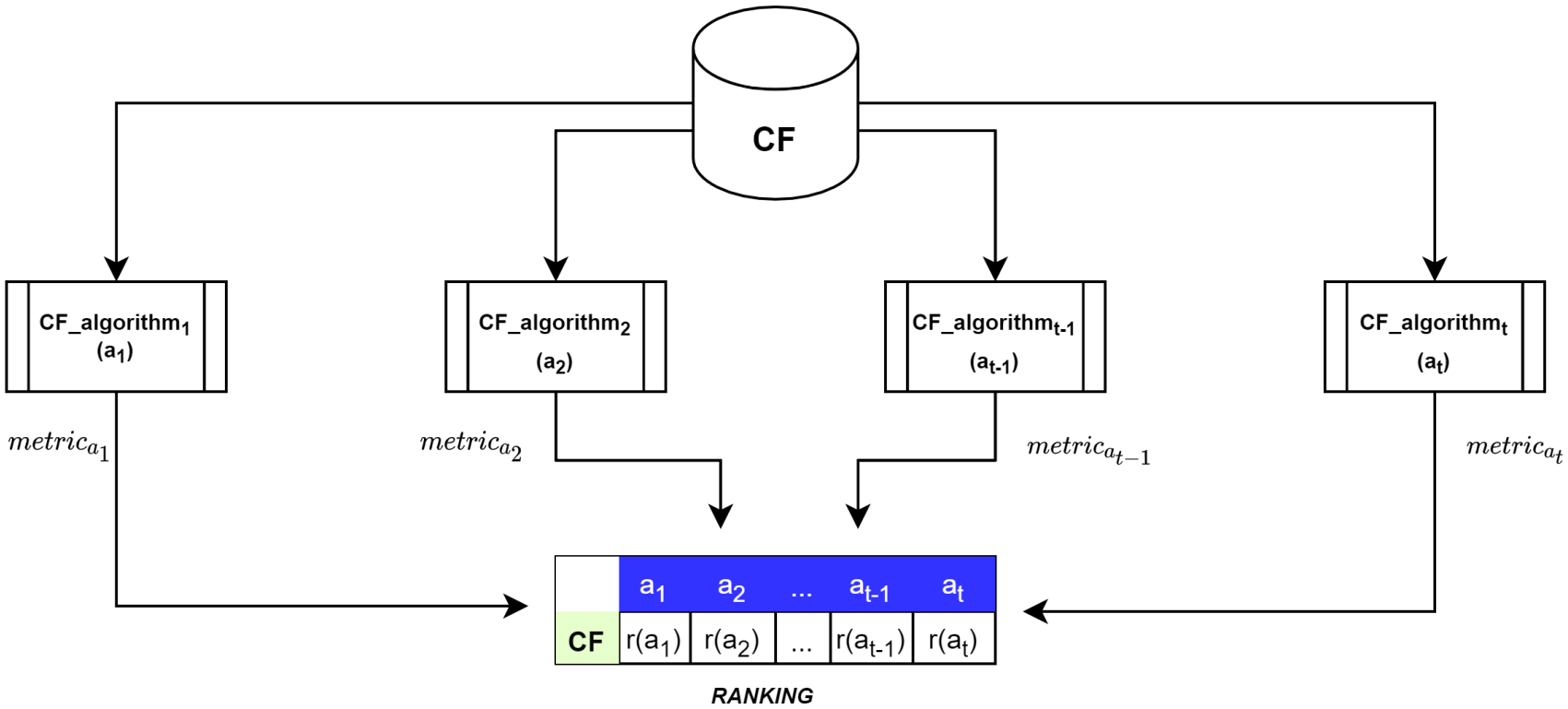


Figure: Extração de *metafeatures* usando o *framework cf2vec*

Criação dos *metatargets*

Para cada conjunto de dados aplica-se t algoritmos de *CF* e avalia-se usando uma métrica específica para cada tarefa que foi realizada. Dessa forma, a partir da métrica é possível gerar um *ranking* para poder recomendar os algoritmos.



Metamodelo

Metamodelos são avaliados usando a métrica *Kendall's Tau* (coeficiente que representa o grau de concordância entre duas colunas de *rankings*). Essa métrica é calculada usando a Equação 1 na qual C é o número de pares concordantes e D é o número de pares desconcordantes.

$$\tau = \frac{C - D}{C + D} \tag{1}$$

Metamodelo

- Avalia-se o metamodelo usando *leave one out cross-validation*.
- hiperparâmetros são ajustados usando *grid search*.

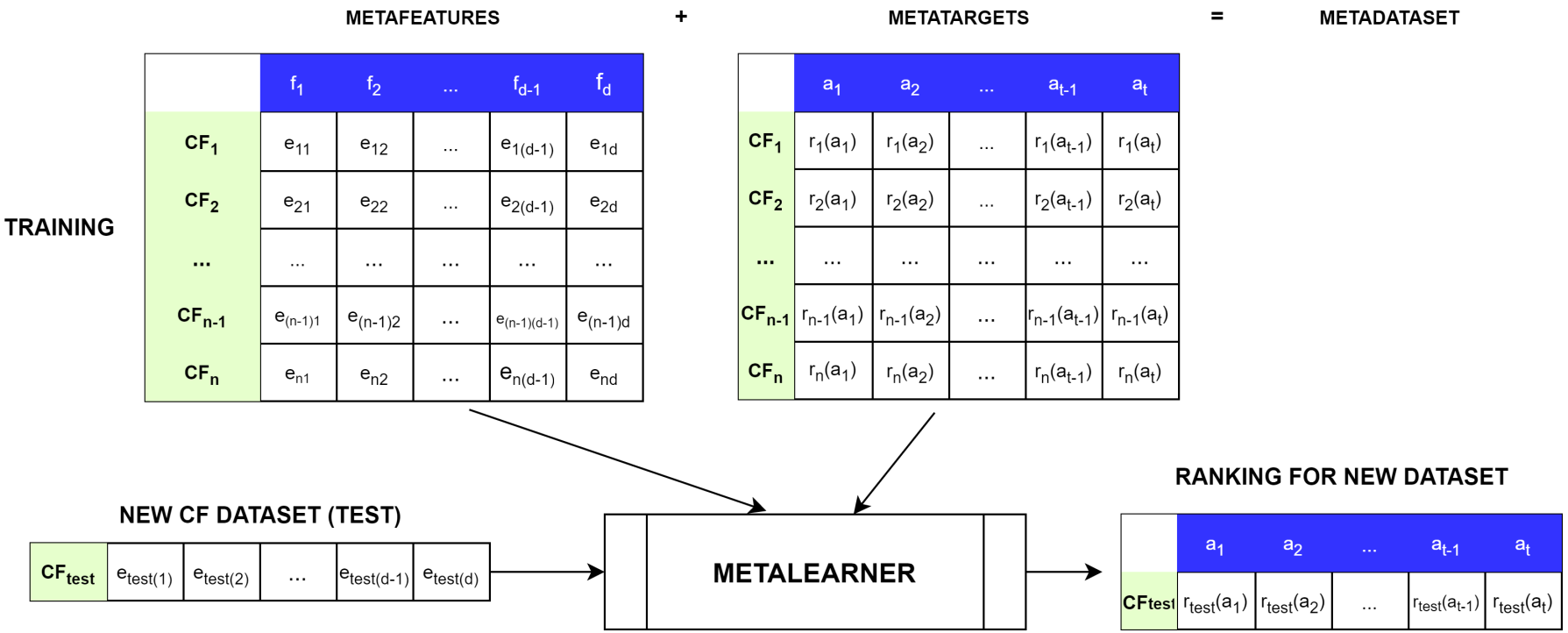


Figure: *Metamodel* do *cf2vec*

Como o sistema de recomendação de algoritmos do *cf2vec* funciona?

- Converte a matriz de filtragem colaborativa em um grafo bipartido.
- Amostra o grafo bipartido para reduzir a complexidade do problema.
- Aprende *embeddings* (representações distribuídas para cada um dos grafos) usando *graph2vec*.
- Utiliza os *embeddings* aprendidos para cada um dos grafos como *metafeatures* do *metadataset* que será construído.

Figure

Uncomment the code on this slide to include your own image from the same directory as the template .TeX file.

Citation

An example of the `\cite` command to cite within the presentation:

This statement requires citation [Smith, 2012].

References

John Smith (2012)

Title of the publication

Journal Name 12(3), 45 – 678.

The End