CLUSTERING THE SKETCH: DYNAMIC COMPRESSION FOR EMBEDDING TABLES

Meta

Henry Tsang
Meta

NORMAL

Thomas Ahle
Normal Computing

thomasahle.github.io/cce

BETTER COMPRESSION BY COMBINING HASHING AND CLUSTERING

Embedding Tables

We address the challenges associated with large categorical feature tables in Recommendation Systems by co-learning a hash projection matrix and a set of embeddings. The hash projections are learned by occasionally decompressing and performing k-means clustering on the embeddings, and the embeddings are learned by SGD.

The methods can be used in any Neural Network that uses embedding tables, and is today using hashing or quantization.

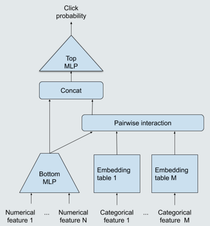
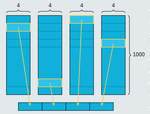


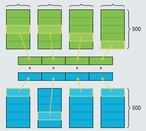
Table Compression

The Hashing Trick: Each ID is hashed to one location in a table (here with 1000 rows) and it is assigned the embedding vector stored at the location. Many IDs are likely to share the same vector.



Compositional Embeddings, CE: Each ID is hashed to a location in each of, say, 4 different tables. The 4 vectors stored there are concatenated into the final embedding vector. Given the large number of possible combinations (here 1000^4), it is unlikely that two IDs get assigned the exact same embedding vector, even if they may share each part with some other IDs.

Sum Hash Embeddings: Each ID is hashed to two rows, one per table, and its embedding vector is assigned to be the sum of those two vectors. Sometimes a separate table is used for weights.



Clustered Compositional Embeddings, CCE: Combining Compositional Embeddings with Sum Hashing allows us to introduce a clustering step, shown in the big central picture on the poster.
Each ID gets assigned a vector that is the concatenation of smaller sums. In each of the four blocks, we apply clustering every epoch, setting the results in the green tables, and replacing the hash functions in the blue tables with new random values.

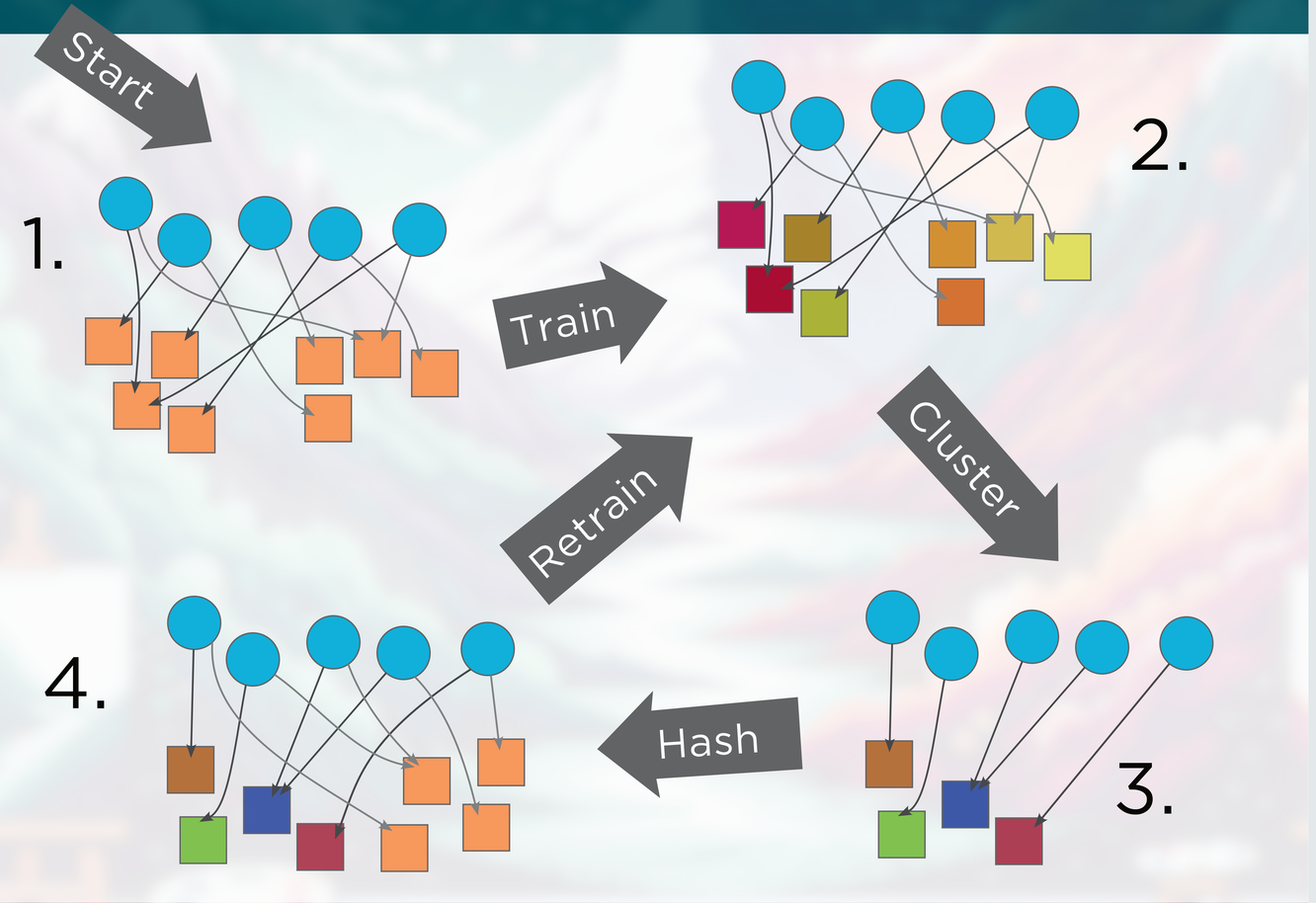
Our repository github.com/thomasahle/cce implements more than 11 different table compression algorithms.

Algorithm 2 Sparse CCE for Least Squares

Require: $X \in \mathbb{R}^{n\times d_1}, Y \in \mathbb{R}^{n\times d_2}, k \in \mathbb{N}$
$H_0 = 0 \in \mathbb{R}^{d_1\times 2k}$ ▷ Initialize assignments
$M_0 = 0 \in \mathbb{R}^{2k\times d_2}$ ▷ Initialize codebook
for $i = 0, 1, \dots$ **do**
  $T_i = H_i M_i \in \mathbb{R}^{d_1\times d_2}$
  $A = \text{kmeans}(T_i) \in \{0,1\}^{d_1\times k}$
  $C \sim \text{countsketch}() \in \{-1, 0, 1\}^{d_1\times k}$
  $H_{i+1} = [A \mid C] \in \mathbb{R}^{d_1\times 2k}$
  $M_{i+1} = \arg\min_M \|XH_{i+1}M - Y\|_F^2$
end for

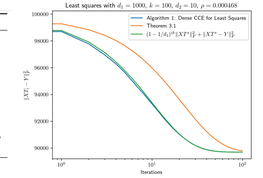
Single iteration of CCE

1. Starting from a random embedding table, each ID is hashed to a vector in each of 2 small tables.
2. During training, the embedding of an ID is taken to be the mean of the two referenced code words.
3. After training for an epoch, the vectors for all (or a sample of) the IDs are computed and clustered. This leaves a new small table in which similar IDs are represented by the same vector.
4. We can choose to combine the cluster centers with a new random table (and new hash function), after which the process can be repeated for an increasingly better understanding of which ID should be combined.



Algorithm 1 Dense CCE for Least Squares

Require: $X \in \mathbb{R}^{n\times d_1}, Y \in \mathbb{R}^{n\times d_2}, k \in \mathbb{N}$
$H_0 = 0 \in \mathbb{R}^{d_1\times 2k}$
$M_0 = 0 \in \mathbb{R}^{2k\times d_2}$
for $i = 0, 1, \dots$ **do**
  $T_i = H_i M_i$
  $N \sim N(0,1)^{d_1\times k}$
  $H_{i+1} = [T_i \mid N]$
  $M_{i+1} = \arg\min_M \|XH_{i+1}M - Y\|_F^2$
end for



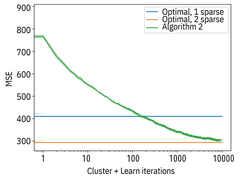
Theorem 3.1. *Assume $d_1 > k > d_2$ and let $T^* \in \mathbb{R}^{d_1\times d_2}$ be the matrix that minimizes $\|XT^* - Y\|_F^2$, then $T_i = H_i M_i$ from Algorithm 1 exponentially approaches the optimal loss in the sense*

$$E[\|XT_i - Y\|_F^2] \le (1-\rho)^{ik}\|XT^*\|_F^2 + \|XT^* - Y\|_F^2,$$

where $\rho = \|X\|_2^2/\|X\|_F^2 \approx 1/d_1$ is the smallest singular value of X squared divided by the sum of singular values squared.

Algorithm 2 Sparse CCE for Least Squares

Require: $X \in \mathbb{R}^{n\times d_1}, Y \in \mathbb{R}^{n\times d_2}, k \in \mathbb{N}$
$H_0 = 0 \in \mathbb{R}^{d_1\times 2k}$ ▷ Initialize assignments
$M_0 = 0 \in \mathbb{R}^{2k\times d_2}$ ▷ Initialize codebook
for $i = 0, 1, \dots$ **do**
  $T_i = H_i M_i \in \mathbb{R}^{d_1\times d_2}$
  $A = \text{kmeans}(T_i) \in \{0,1\}^{d_1\times k}$
  $C \sim \text{countsketch}() \in \{-1, 0, 1\}^{d_1\times k}$
  $H_{i+1} = [A \mid C] \in \mathbb{R}^{d_1\times 2k}$
  $M_{i+1} = \arg\min_M \|XH_{i+1}M - Y\|_F^2$
end for



Algorithm 3 Clustered Compositional Embeddings with c columns and $2k$ rows

class CCE:
  method Initialize:
    for $i = 1$ to c **do**:
      $h_i, h_i' \leftarrow$ i.i.d. $\sim$ random functions from $[d_1]$ to $[k]$ ▷ H in Algorithm 2
      $M_i, M_i' \leftarrow$ i.i.d. $\sim N(0,1)^{k\times d_2/c}$
  method GetEmbedding(id):
    return CONCAT($M_i[h_i(id)] + M_i'[h_i'(id)]$ for $i = 1$ to c)
  method Cluster($items$):
    for $i = 1$ to c **do**:
      $T \leftarrow [M_i[h_i(id)] + M_i'[h_i'(id)]$ for $id \in [d_1]]$ ▷ See discussion below
      centroids, assignments $\leftarrow$ K-MEANS(T) ▷ Find k clusters and assign T to them
      $h_i \leftarrow$ assignments
      $M_i \leftarrow$ centroids
      $h_i' \leftarrow$ random function from $[d_1]$ to $[k]$
      $M_i' \leftarrow 0^{k\times d_2/c}$