PRIN17 - Multicriteria Data Structures and Algorithms: from compressed to learned indexes, and beyond
Meeting - 12.03.21

Dipartimento Informatica

Componenti Unità
Marco Frasca RTD B (coordinator)
Giorgio Valentini PO
Dario Malchiodi PA
Marco Mesiti PA
Paolo Perlasca RU
Giosuè Marinò Collaboratore
Alessandro Petrini Assegnista
Jessica Gliozzo Dottoranda
**Attempt.** Simulating the PGM-index through a (Deep) Neural Network (NN) to evaluate potential extensions

Preliminary experiments ongoing

- **PROBLEM INPUT:** sorted sequence of integers $X$ [extendable to other type of elements]
- **PROBLEM OUTPUT:** a learned indexed to perform the $\text{rank}[x] := |y \in X | y \leq x|
- **STEP 1.** Learning the PGM index over $X$, and taking the segments $s_j = (l_j, c_j, i_j)$ at the lowest level
- **STEP 2.** Construct the network
- **STEP 3.** NN training
- **STEP 4 [optional].** NN compression
- **STEP 5.** Index validation
[T1] Compare classic Data Structures vs Purely Learned Indexes

**Step 1.** Learning the PGM index over $X$, and taking the segments $s_j = (l_j, c_j, i_j)$ at the lowest level

**Step 2.** NN structure for PGM.

![Diagram of PGM network structure](image)

Key $x$

Input layer

$W_1 = (l_1, l_2, ..., l_m)$

$W_2 = (c_1, c_2, ..., c_m), (i_1, i_2, ..., i_m)$

RBF/Step layer

PGM layer: $f_k(x) = (x-l_k)c_k + i_k$

Multiply

Dense
Initializing $W_2$ with **PGM segments** (right part). **Clamping** $W_1$ (left part)

**Size of dataset**: $10^6$

$L_{MAE}(x_1, \ldots, x_n) = \frac{1}{n} \sum_{i=1}^{n} |x_i|$, mean absolute error

$L_{MAX}(x_1, \ldots, x_n) = \log(\sum_{i=1}^{n} e^{|x_i|})$, loss approximating the maximum function

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ini mean err</th>
<th>ini max err</th>
<th>mean error</th>
<th>max error</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Loss MAE</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wiki</td>
<td>29.35</td>
<td>64</td>
<td>24.23</td>
<td>96</td>
</tr>
<tr>
<td>books</td>
<td>21.3</td>
<td>64</td>
<td>20.57</td>
<td>100</td>
</tr>
<tr>
<td>longitude</td>
<td>28.9</td>
<td>64</td>
<td>27.66</td>
<td>133</td>
</tr>
<tr>
<td><strong>Loss MAX</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wiki</td>
<td>29.35</td>
<td>64</td>
<td>29.1</td>
<td>69</td>
</tr>
<tr>
<td>books</td>
<td>21.3</td>
<td>64</td>
<td>21.1</td>
<td>67</td>
</tr>
<tr>
<td>longitude</td>
<td>28.9</td>
<td>64</td>
<td>28.1</td>
<td>67</td>
</tr>
</tbody>
</table>
Compare classic Data Structures vs Purely Learned Indexes

**Random** initializing $W_2$ (right part).  
**Clamping** $W_1$ (left part)

Size of dataset: $10^6$

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ini mean err</th>
<th>ini max err</th>
<th>mean error</th>
<th>max error</th>
</tr>
</thead>
<tbody>
<tr>
<td>wiki</td>
<td>29.35</td>
<td>64</td>
<td>1864.6</td>
<td>$\simeq 10^5$</td>
</tr>
<tr>
<td>books</td>
<td>21.3</td>
<td>64</td>
<td>2455.21</td>
<td>$\simeq 10^5$</td>
</tr>
<tr>
<td>longitude</td>
<td>28.9</td>
<td>64</td>
<td>2990.5</td>
<td>$\simeq 10^5$</td>
</tr>
</tbody>
</table>

**Learning Issues**

- Too many local attractors
- Structure constrained by PGM segments: optimal for the given segments
Ongoing activities and short term objectives:

– Experiments with alternative NN structure
– Learning the segment partition
– Verifying whether compression strategies may help
[T1] Compare classic Data Structures vs Purely Learned Indexes

Second activity in T1: study the impact of ML models in Learned Bloom Filters (LBFs)

- Preliminary research
- Recent works have shown that ‘classical’ BF might be improved in terms of space occupancy by using ML models

Bloom Filter:

- space efficient probabilistic data structure for representing a set \( X \subset S \)
- support membership queries: given \( y \in S \), does \( y \) belong to \( X \)?
- It uses \( k \) hash functions and a \( m \)-bit array

- When \( x \notin S \), the filter always answers correctly
- When \( x \in S \), the filter answers correctly with probability \( 1 - \epsilon \)
- \( 0 < \epsilon < 1 \) false positive rate (FPR)
- To reduce \( \epsilon \), more bits are required:
  \[ m = |X| \frac{\ln(1/\epsilon)}{(\ln(2))^2} \]
Learned Bloom Filters [3]. **Idea:** substitute the BF with a binary classifier $f: S \rightarrow [0, 1]$

- Negative samples $U \subset S \setminus X$ are needed to train $f$
- **Problem:** the classifier might have false negative predictions (FNs)
- **Solution:** use a *backup* BF $B$ for the classifier FNs

Given $x \in S$, the LBF $(f, \tau, B)$ answers:

- $x \in X$ if $f(x) > \tau$, or if $f(x) \leq \tau$ and $B$ says $x \in X$;
- $x \notin X$ otherwise

- $\tau \in [0, 1]$ threshold tuned so as to obtain the desired FPR of $f$
- The overall space required is considerably reduced (36% for FPR 0.01, 15% for FPR 0.001)
- **Limitations:** reducing the FPR leads to increase of the size of $B$ (more FNs)
- Query time increased: even more than $100x$ [4] (in [3] $f$ is a GRU)
- Further extension: Sandwiched LBF, add a BF at beginning to filter out negatives [5]


https://github.com/JasonMa2016/CS222

Research aim on LBF: on a proposal of UNIPA, to verify whether adopting classifiers $f$ different than RNN might reduce the limitation described above.

E.g. using classifiers faster than RNN to be queried.

So far we just reproduced the experiments in [4].
The aim of our unit in this task is to evaluate the capability of different compression techniques for Neural Network (NN) models. Reduce the model space requirements while possibly preserving (or even improving) its accuracy:

- Connection Pruning
- Weight quantization
- Sparsified NN training
- Knowledge distillation
- ...
- Combined approaches

Results might change with the input problem.

NN models for different categories of problems: multi-class/binary classification, regression, dimensionality reduction and so on.
Compress ML models

- Applied pruning and different techniques of weight quantization and evaluated their performance
- Introduced a compact format for compressed fully connected layers, able to take advantage from both sparse and repeated weights
- Validated our approach on two publicly available CNN for multi-class classification and for regression, on 4 different datasets
- Compression ratio up to 165 while preserving accuracy


[T2] Compressed ML models

UNIMI short- mid-term objectives:

- Extending the set of NN compression techniques
- Considering also the compression of convolutional layers
- Build-up an automatic optimizer across the range of possible compression techniques so as to respect a given constraint of the maximum space occupied by the compressed model
  - multiple criteria: (accuracy loss w.r.t. uncompressed model, space, time?)
- Hopefully to be used in tasks T3 and T4
- Further collaborations are welcome