Tree of Nets

1 Preliminares

This section is devoted to the description of the cosidered problem, namely the predecessor Search and of some preliminary definitions about Neural Networks, the model on which is based the solution to the problem proposed in this study.

L'idea principale di questo studio è di trovare un modello efficente in tempo e spazio che possa accedere ad un elemento di una lista ordinata in tempo e con una struttura dati che occupa spazio O(1). Solitamente per questo tipo di problema vengono usati i B-Tree che invece occupano spazio O(n) e effettuano la ricerca in O(log n)

1.1 Precedessor Search

Let X be a subset of part of a universe U, sorted according to the $\leq$ order relation definex on U. We assume that each element of U can be rappresented by d bits. We also assume that lexicographic ordering of the binary representation of each element in U preserve the order relation $\leq$. That is, a comparison between two elements of U can be done using esing either the binary representation as a string or a primitive comparison operation. The *predecessor search problem* consist in finding the position of the largest key inX not greater than a given input key x, that is denoted by pos(x) the position of x $\in$ X in the sorted sequence, the problem is determining pred(x) = pos(z), z=$\max_{y \in X} y \leq x$

Example: Let X be

X = {2,3,4,5,12,15,18}

Let us assume that 7 is the key to search. The predecessor search should return the position pred(7) = 4. Analogously, pred(5) = 4.

Given $|X| = n$, let F_X be the *Empirical Cumulative distribution* of elements of U with respect to (sample) X, for each $x \in$ U, $F(x)_X = \frac{|y \in X | y \leq x|}{n}$. To simplify the notation we denote F_X by F. In the example above, it holds $F(2) = \frac{1}{7}$, $F(5) = \frac{4}{7}$, $F(6) = \frac{4}{7}$, $F(18) = 1$. The knowledge of F provides a solution to our problem since, given an element x of U, only one evaluation of F provides pred(x) = $\lceil F(z) * n \rceil$. Our task is find a good approximation $\tilde{F}$ of F.

1.2 Neural Network

2 Methods

We concentrate on *feed forward network* to determinate $\tilde{F}$. We want to do it by considering limited resources both in time and space.

We compute the space of model with respect to the dataset in Kb, $spaceOVH(Model, dataset) = \frac{ModelSpace*1024}{NumberOfExample}$

We consider 3 model with some common settings:

- The loss function is the most common choise, mean square error $E(\vec{A}, \vec{B}) = \frac{1}{m}\sum_{i=1}^{m}(a_i - b_i)^2$.
- Hiddens and outputs neurons have Leaky-ReLU activation function ($\alpha = 0.05$)
- learning rate = 0.1
- Batch-size = 64
- Optimizer = SGD with Momentum ($\mu = 0.9$)
- Initializer for weights connection and bias is Random Normal with $\mu = 0$ and $\sigma = 0.05$
- Stopping Criteria: the loss don't improve with patience 10 or number of epochs is 20000

NN1K model have 64 inputs neurons and 1 outputs neurons.
NN2K model have 64 inputs neurons, 256 hiddens neurons and 1 outputs neurons.
NN3K model have 64 inputs neurons, 2 layers of 256 hiddens neurons and 1 outputs neurons.

Tree with NN0 leaf The idea is to split the dataset in s parts. So there are s Neural nets like previus paragraph with 64 input neurons and 1 output neurons.
NN0 model.

- The loss function is the most common choise, mean square error $E(\vec{A}, \vec{B}) = \frac{1}{m}\sum_{i=1}^{m}(a_i - b_i)^2$.
- The activation function of output neuros is Leaky-ReLU ($\alpha = 0.05$)
- learning rate = 0.1
- Batch size = 32
- Optimizer = SGD with Momentum ($\mu = 0.9$)

- Initializer for weights connection and bias is Random Normal with $\mu = 0$ and $\sigma = 0.05$
- Stopping Criteria: the mean absolute error don't improve with patience 10 or number of epochs is 20000

3 Results

In our test we used 3 differents dataset, all with the uniform distribution:

- file3 2^9 elements
- file7 2^{13} elements
- file10 2^{20} elements

Tables of NN1K, NN2K, NN3K model

- ϵ = Max error
- SpaceOVH = space of the model in KB with respect to dataset

NNxK file 3

Model	ϵ	SpaceOVH
NN1K	9	8.93×10^{-2}
NN2K	4	1.29×10^{1}
NN3K	4	6.31×10^{1}

NNxK file 7

Model	ϵ	SpaceOVH
NN1K	53	3.09×10^{-3}
NN2K	33	8.06×10^{-1}
NN3K	40	3.94

NNxK file 10

Model	ϵ	SpaceOVH
NN1K	905	2.42×10^{-5}
NN2K	1031	6.29×10^{-3}
NN3K	1270	3.08×10^{-2}

Tables of Tree-NN0 model

- Split = Number of splits on dataset (and number of NN0 used)
- ϵ = Max of error max of each NN0
- μ = Mean of max error of each NN0
- SpaceOVH = space of the model in KB with respect to dataset

NN0 file 3

Split	ϵ	μ	SpaceOVH
1	8	8.0	4.96×10^{-2}
2	8	6.5	9.92×10^{-2}
3	7	5.0	1.49×10^{-1}
4	9	6.0	1.98×10^{-1}
5	6	4.4	2.48×10^{-1}
6	5	3.5	2.98×10^{-1}
7	5	3.14	3.47×10^{-1}
8	5	3.25	3.97×10^{-1}
9	3	2.33	4.46×10^{-1}
10	4	2.8	4.96×10^{-1}
11	4	2.45	5.45×10^{-1}
12	3	1.92	5.95×10^{-1}
13	3	1.85	6.45×10^{-1}
14	2	1.57	6.94×10^{-1}
15	4	1.67	7.44×10^{-1}
16	2	1.19	7.93×10^{-1}
17	2	1.12	8.43×10^{-1}
18	2	1.17	8.93×10^{-1}
19	4	1.16	9.42×10^{-1}
20	1	1.0	9.92×10^{-1}
21	2	1.1	1.04
22	2	1.05	1.09
26	1	1.0	1.29
30	2	1.07	1.49

NN0 file 7

Split	ϵ	μ	SpaceOVH
1	41	41.0	3.1×10^{-3}
2	36	29.0	6.2×10^{-3}
3	33	27.67	9.3×10^{-3}
4	43	30.5	1.24×10^{-2}
5	27	20.2	1.55×10^{-2}
6	31	22.17	1.86×10^{-2}
7	32	19.86	2.17×10^{-2}
8	29	21.5	2.48×10^{-2}
9	33	19.0	2.79×10^{-2}
10	24	15.6	3.1×10^{-2}
11	25	15.09	3.41×10^{-2}
12	28	14.92	3.72×10^{-2}
13	24	14.46	4.03×10^{-2}
14	26	13.64	4.34×10^{-2}
15	24	12.4	4.65×10^{-2}
16	21	13.25	4.96×10^{-2}
17	23	12.65	5.27×10^{-2}
18	21	12.44	5.58×10^{-2}
19	19	10.79	5.89×10^{-2}
20	27	11.55	6.2×10^{-2}
21	20	11.05	6.51×10^{-2}
22	20	10.77	6.82×10^{-2}
26	13	9.35	8.06×10^{-2}
30	17	9.3	9.3×10^{-2}
34	16	8.53	1.05×10^{-1}
38	13	7.39	1.18×10^{-1}
42	11	7.14	1.3×10^{-1}
46	13	7.13	1.43×10^{-1}
50	12	6.8	1.55×10^{-1}
54	13	6.57	1.67×10^{-1}
58	12	6.24	1.8×10^{-1}
62	11	6.02	1.92×10^{-1}
64	21	6.0	1.98×10^{-1}
72	9	5.24	2.23×10^{-1}
80	9	4.92	2.48×10^{-1}
88	10	4.65	2.73×10^{-1}
96	9	4.38	2.98×10^{-1}
104	8	4.13	3.22×10^{-1}
112	7	3.83	3.47×10^{-1}
120	10	3.75	3.72×10^{-1}
128	6	3.52	3.97×10^{-1}

NN0 file 10

Split	ϵ	μ	SpaceOVH
1	714	714.0	2.42×10^{-5}
2	593	557.5	4.84×10^{-5}
3	494	420.33	7.26×10^{-5}
4	515	428.75	9.69×10^{-5}
5	545	360.0	1.21×10^{-4}
6	561	319.17	1.45×10^{-4}
7	538	292.57	1.69×10^{-4}
8	465	298.5	1.94×10^{-4}
9	453	240.33	2.18×10^{-4}
10	425	228.6	2.42×10^{-4}
11	338	222.64	2.66×10^{-4}
12	311	200.08	2.91×10^{-4}
13	207	183.85	3.15×10^{-4}
14	238	175.07	3.39×10^{-4}
15	254	176.53	3.63×10^{-4}
16	305	196.75	3.87×10^{-4}
17	266	169.53	4.12×10^{-4}
18	246	154.06	4.36×10^{-4}
19	193	147.79	4.6×10^{-4}
20	233	144.3	4.84×10^{-4}
21	277	150.67	5.08×10^{-4}
22	357	143.0	5.33×10^{-4}
26	209	131.85	6.3×10^{-4}
30	233	116.7	7.26×10^{-4}
34	196	106.15	8.23×10^{-4}
38	177	104.55	9.2×10^{-4}
42	214	101.38	1.02×10^{-3}
46	153	88.57	1.11×10^{-3}
50	143	84.88	1.21×10^{-3}
54	145	85.07	1.31×10^{-3}
58	133	82.4	1.4×10^{-3}
62	143	80.34	1.5×10^{-3}
64	142	79.77	1.55×10^{-3}
72	121	69.71	1.74×10^{-3}
80	134	66.75	1.94×10^{-3}
88	153	65.88	2.13×10^{-3}
96	109	60.6	2.32×10^{-3}
104	107	59.13	2.52×10^{-3}
112	99	56.11	2.71×10^{-3}
120	118	54.53	2.91×10^{-3}
128	302	56.22	3.1×10^{-3}