LECTURE 10

Yue Gong 2020.02.18

Review

Hash-based data structure
- Hash Tables / Maps / Sets
  - Exact equality lookup
- Hash-based similarity lookup
  - Minhash
- Approximate Membership Query (is q in a set)
  - Bloom Filter
- Approximate Counting Query (how many times has a q shown up)
  - Count-Min Sketch

- Approximate Distinct Count
- Relations to Machine Learning

} This Lecture

Distinct Count Problem

Estimate the number of distinct elements in a list with a single pass

Exact Solution:

O(# Distinct Elems) Memory

- What if we only have $O(m)$ space $m \ll D$?

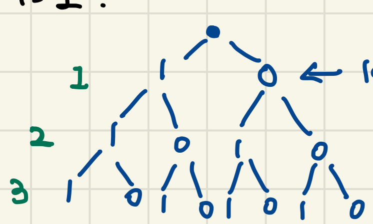
hash function: $h: S \to \{0, \dots, 2^L - 1\}$

What is the probability a hash code is odd ?

$\frac{1}{2}$

$h: S \to \{0,1\}^L$

What is the probability that the leftmost bit is 1 ?



x = the position of the least-significant 1

$P(X=1) = \frac{1}{2}$

$P(X=2) = \frac{1}{4}$

$P(X=3) = \frac{1}{8}$

$\boxed{P(X=L) = \frac{1}{2^L}}$, where x is the position of the least-significant 1.

BMP[0, 1, ..., L-1]

for s in S:

  p = first index with a 1 from the left in the hash(s)

  BMP[p] = 1

BMP ⊔ ⊔ ⊔ ⊔ ... ⊔

$n/2$ $n/4$ .... $n/2^{L-1}$

Estimation: Calculate R → right most bit that is 1

Then, estimate 2^R as the # distinct values

This is called "Flajolet-Martin Sketch"

Relations to Machine Learning

→ high-dimensional sparse features

one-hot embedding

A, B, C

		A/C	B
1. A, B, C	→ [1, 1, 1] →	[1	1]
2. A, C	→ [1, 0, 1] →	[1	0]
3. B	→ [0, 1, 0] →	[0	1]
4. A, C	→ [1, 0, 1] →	[1,	0]
5. B	→ [0, 1, 0] →	[0,	1]

Are they compressible ?

Yes ! Because A always appears with C.

Feature Hashing

$[v_1, v_2, v_3, \dots, v_j]$

⇓

$[v_1', v_2', \dots, v_m']$

Reduce Feature dimension

Feature Hashing (Yahoo Paper)

v = original feature vector $0, 1, \ldots, i-1$

x = compressed feature vector $0, 1, \ldots, m-1$

$h_1 = \{0, \ldots, i-1\} \rightarrow \{0, \ldots, m-1\}$

maps and randomly groups together features

$h_2 = \{0, \ldots, i-1\} \rightarrow \{-1, 1\}$

Assign a negative one or positive one, to each one of the features

$$\rightarrow x[j] = \sum_{l=0}^{i-1} \delta(h_1(l) = j) \cdot v[l] \cdot \underbrace{h_2(l)}_{\text{randomly flips the sign}}$$

One Interesting Property

$$\langle v, v' \rangle = \sum_{l=0}^{i-1} v[l] \cdot v'[l]$$

$E(\langle x, x' \rangle) = \langle v, v' \rangle$, feature hashing preserves the inner product between examples.

why inner product can be preserved?

$$\langle x, x' \rangle = \sum_{l=0}^{i-1} \sum_{m=0}^{i-1} \delta(h_1(l) = h_1(m)) \cdot h_2(l) h_2(m) \cdot v(l) v(m)$$

add together

$h_2(l), h_2(m)$ have the same sign $\rightarrow h_2(l) h_2(m) = 1$

$h_2(l), h_2(m)$ have different signs $\rightarrow h_2(l) h_2(m) = -1$

cancellation

Why is preserving inner product so good?

$$\min_{\theta} \| X^T\theta - y \|_2^2$$

if X is full rank, $\theta = \underbrace{(X^TX)^{-1}}_{\text{Matrix of Inner Product}} X^T y$

Advantages over PCA

If using PCA. needs to materialize a dictionary of words

→ if you use Feature Hashing, you can skip materialization

e.g. { the quick brown .... }

hash↓ hash↓ hash↓