Case Study 2: Document Retrieval

Clustering Documents

Announcements:

• HW2 posted
• Project Milestones

• Shameless plug for my talk
  – Talk: Accelerating Stochastic Gradient Descent
  – Next Tue at 1:30 in CSE 303
  – It’s a very promising directions....

• Today:
  – Review: locality sensitive hashing
  – Today: clustering and map-reduce
Case Study 2: Document Retrieval

Locality-Sensitive Hashing
Random Projections for NN Search

Machine Learning for Big Data
CSE547/STAT548, University of Washington
Sham Kakade
April 18, 2017

Intuition (?): NN in 1D and Sorting

- How do we do 1-NN searches in 1 dim?
  - Pre-processing time: \( O(N\log N) \)
  - Query time: \( O(1) \)

©Sham Kakade 2017
Using Hashing to Find Neighbors

- KD-trees are cool, but...
  - Non-trivial to implement efficiently
  - Problems with high-dimensional data
- Approximate neighbor finding...
  - Don’t find exact neighbor, but that’s OK for many apps, especially with Big Data
- What if we could use hash functions:
  - Hash elements into buckets:
    - Look for neighbors that fall in same bucket as x:
- But, by design...

What to hash?

- Before: we were hashing ‘words’/strings
- Remember, we can think of hash functions abstractly:
  \[ h : X \rightarrow \{ \ell_1, \ldots, m \} \]
- Idea of LSH: try to has similar items into same buckets and different items into different buckets
Locality Sensitive Hashing (LSH)

- Suppose we have a set of functions $H$ and a distribution over these functions.
- A LSH family $H$ satisfies (for example), for some similarity function $d$, for $r>0$, $\alpha>1$, $1>P_1,P_2>0$:
  - $d(x,x') \leq r$, then $\Pr_{h \sim H}(h(x)=h(x'))$ is high, with prob $>P_1$
  - $d(x,x') > \alpha r$, then $\Pr_{h \sim H}(h(x)=h(x'))$ is low, with prob $<P_2$
  - (in between, not sure about probability)

---

LSH: basic paradigm

- Step 0: pick a ‘simple’ way to construct LSH functions
- Step 1: (amplification) make another hash function by repeating this construction
  $\phi(x) = (h_1(x), \ldots, h_k(x))$
- Step 2: the output of this function $\phi$ specifies the index to a bucket.
- Step 3: use multiple hash tables, for recall, search for similar items in the same buckets.
Example: hashing binary strings

- Suppose $x$ and $x'$ are binary strings
- Hamming distance metric $|x-x'|$
- What is a simple family of hash function?
  \[ h^{(i)}(x) = x_i \]
- Suppose $|x-x'|$ are $R$ close, what is $P_1$?
  \[ P_1 = 1 - \frac{R}{\alpha} \]
- Suppose $|x-x'| > cR$, what is $P_2$?
  \[ P_2 = 1 - \frac{cR}{\alpha} \]

Amplification

- Improving $P_1$ and $P_2$
- Now the hash function is:
  \[ \phi = (h_1^{(i)}(x), h_2^{(i)}(x), \ldots) \]
  \[ \phi_d = (h_1^{(i)}(x), \ldots, h_k^{(i)}(x)) \]
- The choice $m$ is a parameter.
Review: Random Projection Illustration

- Pick a random vector $v$:
  - Independent Gaussian coordinates
    $y(x) = v \cdot x$

- Preserves separability for most vectors
  - Gets better with more random vectors

Multiple Random Projections: Approximating Dot Products

- Pick $m$ random vectors $v(i)$:
  - Independent Gaussian coordinates

- Approximate dot products:
  - Cheaper, e.g., learn in smaller $m$ dimensional space

- Only need logarithmic number of dimensions!
  - $N$ data points, approximate dot product within $\varepsilon > 0$:

$$m = O\left(\frac{\log N}{\varepsilon^2}\right) \quad \Rightarrow \quad |x \cdot x'| \approx |\phi(x) - \phi(x')| \pm \varepsilon$$

- But all sparsity is lost
LSH Example function: Sparser Random Projection for Dot Products

- Pick random vector $v$
- Simple 0/1 projection: $h(x) = s_{\frac{x \cdot v}{\|v\|^2}}(v \cdot x)$

- Now, each vector is approximated by a single bit
- This is an LSH function, though with poor $\alpha$ and $P_2$

---

LSH Example continued: Amplification with multiple projections

- Pick random vectors $v^{(i)}$
- Simple 0/1 projection: $\phi_i(x) =$

- Now, each vector is approximated by a bit-vector
- Dot-product approximation:
LSH for Approximate Neighbor Finding

• Very similar elements fall in exactly same bin:

\[ \phi(y) = (\phi_1(x), \ldots, \phi_k(x)) \]

• And, nearby bins are also nearby:

• Simple neighbor finding with LSH:
  – For bins \( b \) of increasing hamming distance to \( \phi(x) \):
    • Look for neighbors of \( x \) in bin \( b \)
  – Stop when run out of time

• Pick \( m \) such that \( N/2^m \) is “smallish” + use multiple tables

LSH: using multiple tables

\[ \phi_1^{(i)}(x) \]

\[ \phi_2 \]

\[ \phi_3 \]

©Sham Kakade 2017
Hash Kernels: Even Sparser LSH for Learning

- Two big problems with random projections:
  - Data is sparse, but random projection can be a lot less sparse
  - You have to sample many huge random projection vectors
    - And, we still have the problem with new dimensions, e.g., new words
- **Hash Kernels**: Very simple, but powerful idea: combine sketching for learning with random projections
- Pick 2 hash functions:
  - \( h \): Just like in Count-Min hashing
  - \( \xi \): Sign hash function
    - Removes the bias found in Count-Min hashing (see homework)
- Define a “kernel”, a projection \( \phi \) for \( x \):

### NN complexities

<table>
<thead>
<tr>
<th></th>
<th>Query time</th>
<th>Space used</th>
<th>Preprocessing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vornoi</td>
<td>( O(2^d \log n) )</td>
<td>( O(n^{d/2}) )</td>
<td>( O(n^{d/2}) )</td>
</tr>
<tr>
<td>Kd-tree</td>
<td>( O(2^d \log n) )</td>
<td>( O(n) )</td>
<td>( O(n \log n) )</td>
</tr>
<tr>
<td>LSH</td>
<td>( O(n^\rho \log n) )</td>
<td>( O(n^{1+\rho}) )</td>
<td>( O(n^{1+\rho} \log n) )</td>
</tr>
</tbody>
</table>
Hash Kernels, Random Projections and Sparsity

\[ \phi_i(x) = \sum_{j: h(j) = i} \xi(j) x_j \]

- Hash Kernel as a random projection:
  - What is the random projection vector for coordinate \( i \) of \( \phi \):
  - Implicitly define projection by \( h \) and \( \xi \), so no need to compute apriori and automatically deals with new dimensions
  - Sparsity of \( \phi \), if \( x \) has \( s \) non-zero coordinates:

What you need to know

- **Locality-Sensitive Hashing (LSH):** nearby points hash to the same or nearby bins
- LSH uses random projections
  - Only \( O(\log N/\varepsilon^2) \) vectors needed
  - But vectors and results are not sparse
- Use LSH for nearest neighbors by mapping elements into bins
  - Bin index is defined by bit vector from LSH
  - Find nearest neighbors by going through bins
- Hash kernels:
  - Sparse representation for feature vectors
  - Very simple, use two hash functions
    - Can even use one hash function, and take least significant bit to define \( \xi \)
  - Quickly generate projection \( \phi(x) \)
  - Learn in projected space