Empiric Introduction to Light Stochastic Binarization

Step towards fast and frugal approximate NN-search algorithm ?

Daniel Devatman Hromada[1,2]

[1] Slovak University of Technology
Faculty of Electronic Engeneering and Informatics
Department of Robotics and Cybernetics

[2] Université Paris 8
École Doctoralle Cognition, Langage, Interaction
Laboratoire Cognition Humaine et Artificielle

17th International Conference on Text, Speech and Dialogue,
Brno, Czech Republic, EU, 11.9.2014

Table of Contents

What SHALL & SHAN'T be presented

What shall NOT be presented

1. dozens incomprehensible formulas and analytical proofs
2. a sufficient overview of dimensionality-reduction and hashing techniques
3. the most performant, robust and fast text classification algorithm ever created

What SHALL be presented

1. Random Indexing and Reflected Random Indexing algorithms
2. simple intuitive method of hashing text into binary vectors
3. NN-search and/or text classification algorithm which could be potentially useful...
4. ...a red herring?

Table of Contents

Problem 1: Text classification

The task is to assign a document to one or more classes.

- supervised - involves labeled training data and (machine) learning phase
- unsupervised - document clustering without reference to external information

Don't forget

Both supervised and unsupervised document classification algorithms share one common aspect: they both start by characterisation of documents in terms of ordered sequences (i.e. vectors) of features they contain.

In current study, we characterise documents only in terms of simple word-occurence features. More advanced (morphophonologic, syntactic, pragmatic, stylistic, rhetoric etc.) features could be also exploitable.

Problem 2: Nearest-Neighbor search

Definition

Given a set S of points in a space M and a query point $q \in M$, find the closest point in S to q.

3 things to take into account

1. Indexing phase complexity
2. Query event complexity
3. Tradeoff between the two

Possible approaches

1. linear: complexity O(Nd)
2. space partitioning: k-d trees, R-trees (curse of dimensionality)
3. approximate NN: best bin first, Locally Sensitive Hashing

Locally Sensitive Hashing

LSH

Locality-sensitive hashing (LSH) is a method of performing probabilistic dimension reduction of high-dimensional data. The basic idea is to hash the input items so that similar items are mapped to the same buckets with high probability.

In Big Data scenarios, NN-search is a tough nut to crack. LSH-inspired solutions offer a way out.i Some studied hashing functions:

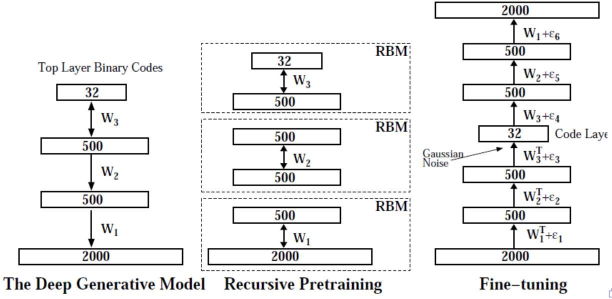
- bit sampling
- min-wise independent permutations
- stable distribution hashing
- Nilsimsa hashing

Note that the objective of hashing discussed here is to MAXIMIZE the probability of collision of similar items.

Semantic Hashing (Salakhutdinov & Hinton)

- deep-learning approach to text hashing problem
- uses multiple layers of Restricted Boltzmann Machines
- only restricted number of features (2000) enters the first RBM
- results in stack of encoders hashing texts into binary vectors
- hash distance in terms of Hamming metrics
- evaluated in text classification

Fig. reproduced from Salakhutdinov & Hinton, 2007



Random Projection (RP)

- group of Dimensionality Reduction techniques
- based on J-L lemma
- RP variant of LSH algorithm (Andoni&Indyk, 2008) chooses the random hyperplane to hash input vectors
- potentially the simplest implementations of RP approach are RI and RRI

J-L lemma

A small set of points in a high-dimensional space can be embedded into a space of much lower dimension in such a way that distances between the points are nearly preserved (Johnson & Lindenstrauss, 1984).

Random indexing

Basic Idea

Given the set of N objects (e.g. documents) which can be described in terms of F features (e.g. occurence of the string in the document) one can obtain D-dimensional vectorial representation of any object by summing up the vectors associated to all features F_1, F_2 observable within X.

Generation of initial feature vectors

Initial feature vectors are generated in a way that out of D elements of vector, only S among them are randomly set to either -1 or 1 value. Other values contain zero. Since the "seed" parameter S is much smaller than the total number of elements in the vector (D), i.e. S <<D, initial feature vectors are very sparse, containing mostly zeroes, with occasional value of -1 or 1.

Reflective Random Indexing

Similiar to RI, RRI also involves the parameters:

- D: dimensionality
- S: "seed" - number of non-zero values in initial random vector

But also introduces 3rd parameter:

- I: number of "reflective" iterations

Reflective process

One can forget the initial randomly generated feature vectors of 0^{th} generation and obtain the feature vectors for feature F_X as a sum of vectors O_1,O_2 representing the objects within which one can observe the occurence of feature F_X. Subsequently, the object vectors can be once again calculated as a sum of feature vectors; feature vectors as a sum of object vectors etc.

RRI Algorithm

```
algorithm RRI ()
   #initial iteration is equivalent to plain Random Indexing
   foreach Feature
      Feature_Vectors[Feature] =  generate_Random_Vectors(Dimension, Seed)
      Feature_Vectors[Feature] *=  TFIDF_Weights[Feature] #optional
   foreach Object
      foreach Feature in Object2Feature[Object]
         Object_Vector[Object] += Feature_Vectors[Feature]
   normalize Feature_Vectors,Object_Vectors     #optional
   #reflective iterations
   repeat
      foreach Feature
         foreach Object in Feature2Object[Feature]
            Feature_Vector[Feature] += Object_Vectors[Object]
      foreach Object
         foreach Feature in Object
            Object_Vector[Object] += Feature_Vectors[Feature]
      Iteration = Iteration + 1
      normalize Feature_Vectors,Object_Vectors  #optional
   until Iteration == MaxIterations
   return Feature_Vectors, Object_Vectors
```

Table of Contents

LSB hashing phase

LSB=RRI+binarization

After all object vectors are calculated by RRI, we simply determine the median value (i.e. 50th percentile for every dimension (i.e. column) D of the resulting Nd matrix. In such a way we obtain a threshold value for every dimension and we assign into d^{th} element of final binary representation of object n the 0 value if its real-valued coordinate along d^{th} dimension is smaller than the determined threshold and 1 if it is above the threshold. Rare tie situations are broken randomly. Result is a set of binary hashes cut in two equally cardinal subsets by every dimension-denoting bit.

$$h_d(n) = \begin{cases} 0 & \text{if } n < \mathrm{median}(\mathrm{D_d}) \\ 1 & \text{if } n > \mathrm{median}(\mathrm{D_d}) \\ \mathrm{rand} & \text{if } n == \mathrm{median}(\mathrm{D_d}) \end{cases}$$

LSB queries

1. RRI attributes a real-valued vector to the text query.
2. The real-valued vector is binarized into LSB hash.
3. Collisions with LSB hashes stored in database are looked up.
4. Hamming sphere of radius X can be explored.

Note

Hamming distance(vec_1,vec_2) $=$ Hammingweight(XOR(vec_1,vec_2))

On modern CPUs which support instruction set $>$SSE4.2, Hamming weight of a binary vector can be calculated by a hardware-implemented POPCNT instruction.

Table of Contents

20-newsgroups corpus

- 20-newsgroups postings taken from the Usenet collection
- we have used solely 7531 postings from the corpus testing subset
- to each posting, one among 20 newsgroups (class) labels were associated
- newsgroup labels were used solely for evaluation (unsupervised scenario)

Processing

Words were extracted from postings by considering every non-word character as a word boundary - 93591 words were thus extracted, among which 41782 has occured in more than one posting. Occurence of ANY word was considered to yield a feature. Data were not processed in any other way.

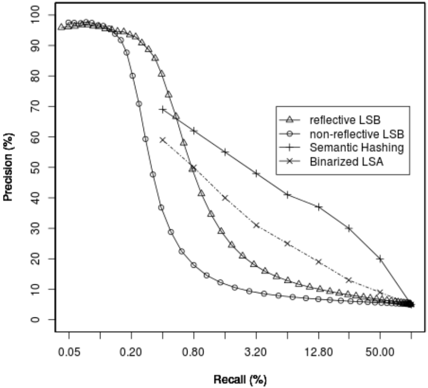
Evaluation

$$Recall = \frac{\text{Number of retrieved relevant documents}}{\text{Total number of all relevant documents}}$$

$$Precision = \frac{\text{Number of retrieved relevant documents}}{\text{Total number of retrieved documents}}$$

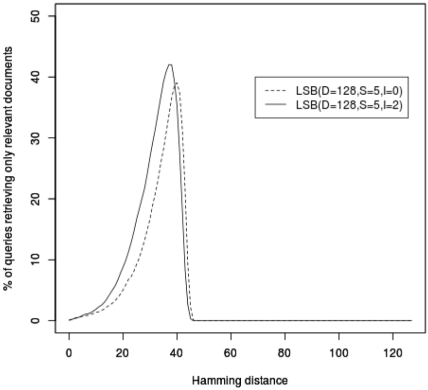
Analogicaly to study of (Salakhutdinov & Hinton, 2007) with which we compare our data, the retrieved document is considered to be relevant to the query document when they have the same class label.

Comparison of results



Comparison of reflective LSB(l=2) and unreflective LSB(l=0) LSB with Semantic Hashing and binarized Latent Semantic Analysis.

Other results



More than 40% of queries are accompanied, within the Hamming ball of radius 38, only by neighbors belonging to the same newsgroup category.

Example of hash collision

```
< New since version of 2 May 1993:
<   * Added info on ImageViewer for NeXT.
---
> New since version of 18 April 1993:
>   * New version of XV supports 24-bit viewing for X Windows.
>   * New versions of DVPEG & Image Alchemy for DOS.
>   * New versions of Image Archiver & PMView for OS/2.
>   * New listing: MGIF for monochrome-display Ataris.
461,463c464,466
<     PMView 0.85: JPEG/GIF/BMP/Targa/PCX viewer.  GIF viewing very fast,
<     JPEG viewing roughly the same speed as the above two programs.  Has
<     image manipulation & slideshow functions.  Shareware, $20.
---
>     PMView 0.85: JPEG/GIF/BMP viewer.  GIF viewing very fast, JPEG viewing
>     fast if you have huge amounts of RAM, otherwise about the same speed
>     as the above programs.  Strong 24-bit display support.  Shareware, $20.
632,641d634
< NeXT:
<
< ImageViewer is a PD utility that displays images and can do some format
< conversions.  The current version reads JPEG but does not write it.
< ImageViewer is available from the standard NeXT archives at
< sonata.cc.purdue.edu and cs.orst.edu, somewhere in /pub/next (both are
< currently being re-organized, so it's hard to point to specific
< sub-directories).  Note that there is an older version floating around that
< does not support JPEG.
```

Table of Contents

Repetitio est mater studiorum

- LSB = Random Indexing + Binarization
- one can use RI to reduce dimensionality to quite small number (D=128)
- every dimension is binarized by thresholding at 50th percentile
- LSB gives, even without machine learning, comparable results to those of Semantic Hashing (especially in high-precision, low recall region which is analogous to NN-search)
- all features can be taken into account, application of RI can potentially help "noisy features to cancel themselves out"

Future directions

1. focus less on text classification and more on NN-search problem
2. evaluate with different corpora
3. assess the importance of D, S, I parameters tuning
4. study whether multiple repetitions of RRI shall not increase the performance
5. use different (both more exhaustive and more restricted) feature sets
6. develop LSH's supervised variant (for example with evolutionary algorithms?)

Thank You for Your attention.

http://wizzion.com

(slava Ukrajine!)