Sparse Spectral Hashing

Group #10

Problem statement

Fast finding of similar data points to a given query from a large scale database using Sparse Spectral Hashing

Efficient encoding requirements

- The binary code has to be easily computed for new data points, outside the training dataset
- The code must be memory-efficient and use as less number of bits to be represented as possible
- The code must preserve similarity, irrespective of the data space

Hashing models proposed so far

- Locality Sensitive Hashing (LSH)
- Semantic Hashing
- Spectral Hashing
- Parameter Sensitive Hashing (PSH)
- Sparse Spectral Hashing (SSH)

Locality Sensitive Hashing

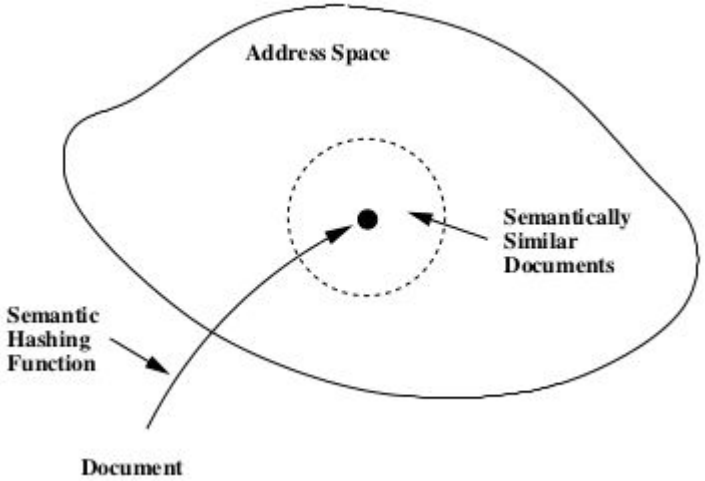
In LSH, a family of locality preserving hash functions are used to hash similar data-points in the high-dimensional space into the same bucket with high probability than non-similar datapoints. To conduct similarity search, LSH hashes a query datapoint into a bucket and take the data-points in the bucket as the retrieved candidates. A final step is then performed to rank the candidate data by calculating their distance to the query, or just return several candidate data-points randomly.

Drawbacks of LSH :

- The selection of number of hash functions(hyperplanes) is randomly decided.
- The Hamming codes are not sparse enough for small number of bits due to which frequent collisions occur.

Semantic Hashing

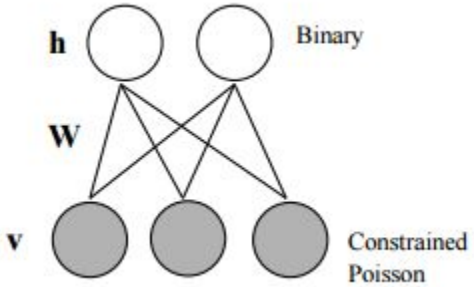
The main idea is to train the RBMs and Deep Encoders with data and convert visual word representation into binary codes which represent a address in an address space.

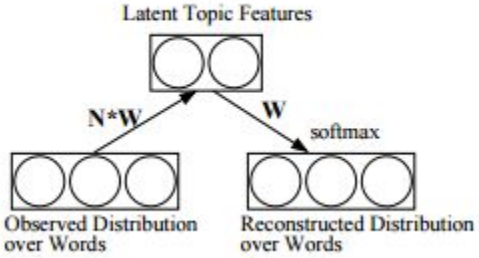


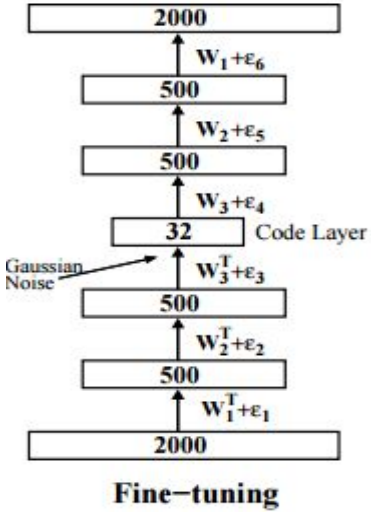
Documents are mapped to memory addresses in such a way that semantically similar documents are located at nearby addresses. Documents similar to a query document can then be found by simply accessing all the addresses that differ by only a few bits from the address of the query document.

The RBM model during Pre-training

The Poisson Constrained function is used and layer by layer of RBMs are trained recursively using softmax layer. Then they are finally enrolled into an auto-encoder.







Fine-Tuning the Data with Auto-encoder

The pretraining finds a point that lies in a good region of parameter space and the fine-tuning then performs a local gradient search that finds a nearby point that is considerably better. During the fine-tuning, we want backpropagation to find codes that are good at reconstructing the count data but are as close to binary as possible. To make the codes binary, we add Gaussian noise.

Drawbacks of Semantic Hashing

- RBM has an extremely complex model which means the parameters of RBM is very difficult to determine.
- This Machine Learning approach suffers from the inaccuracy of the data points outside the training set.
- For very large document set, documents with similar addresses have similar content but the converse is not necessarily true. Given the way we learn the deep belief net, it is quite possible for the semantic address space to contain widely separated regions that are semantically very similar.

Parameter Sensitive Hashing (Introduction)

Example based methods are effective for parameter estimation problems when the underlying system is simple or the dimensionality is low. For complex and high-dimensional problems such as Pose Estimation, the number of required examples and the computational complexity rapidly become prohibitively high.

Many problems in computer vision can be naturally formulated as parameter estimation problems :

- Given an image or a video sequence x , we estimate the parameters of a model describing the scene or the object of interest
- Classic methods for example-based learning, such as the k-nearest neighbor rule (k-NN) and locally-weighted regression (LWR), are appealing due to their simplicity and the asymptotic optimality of the resulting estimators.
- Complexity of nearest neighbor search makes this approach difficult to apply to the very large numbers of examples needed for general articulated pose estimation with image-based distance metrics.

We overcome the problem of computational complexity by using LSH in our hashing approach.

While LSH provides a technique for quickly finding close neighbors in the input space, these are not necessarily close neighbors in the parameter space.

Parameter Sensitive Hashing

- An extension of LSH.
- PSH uses hash functions sensitive to the similarity in the parameter space, and retrieves approximate nearest neighbors in that space in sublinear time.
- The key construction is a new feature space that is learned from examples in order to more accurately reflect the proximity in parameter space

Spectral Hashing

Let x_i be d-dimensional vectors in the Euclidean space, with i = 1 to N. Let y_i be the corresponding Hamming codes of length k.

Assuming a distance metric of $\in$, the similarity between x_i and x_j is given by an affinity matrix W, where $W(i,j) = \exp(-\|x_i - x_j\|^2/\epsilon^2)$

Thus, average Hamming distance between similar neighbours is given by :

$$\sum_{ij} W_{ij}\|y_i - y_j\|^2$$

with $y_i \in \{-1,1\}^k$, $\sum_i y_i = 0$ and $\frac{1}{n}\sum_i y_i y_i^T = I$

We aim to minimize $\sum_{ij} W_{ij}\|y_i - y_j\|^2$

Spectral Relaxation

Let $Y_{n \times k}$ be a matrix whose j^{th} row is y_j^T and $D_{n \times n}$ be a diagonal matrix such that D(i,i) = $\sum_j W(i,j)$

The optimization problem can now be re-written as minimizing trace($Y^T(D - W)Y$), with $Y^T 1 = 0$ and $Y^T Y = I$. The solutions to the trace equation are the k eigenvectors of D - W with minimal eigenvalue.

However, in order to account for data points from outside the sample set, we assume the data points x_i are coming from a probability distribution. Then, the solutions are eigenfunctions of weighted Laplacian operators L_p.

L_p: f $\longrightarrow$ g, with g(x) = D(x)f(x)p^2(x) - p(x) $\int_s W(s,x) f(s) p(s) ds$ with D(x) = $\int_s W(x,s)$
Here, p(x) = $\prod_i u_i(x_i)$, where u_i is a uniform distribution.

Drawbacks of Spectral Hashing

- It assumes a multi-dimensional uniform distribution to have generated the data points. This is not suitable for many real-life datasets.
- Data points are over-complete with redundant features being involved in linear combinations.

Sparse Spectral Hashing

We have a collection of N - *d* dimensional data-points in Euclidean Space

$$\{(\boldsymbol{x}_i) \in \mathbb{R}^d : i = 1, 2, \ldots, N\},$$

where X_i represents the feature vector for the ***i*** th data-point, d represents the dimension of the features from the training data.

$$\Theta : \boldsymbol{x}_i \in \mathbb{R}^d \rightarrow \boldsymbol{y}_i \in \{-1, 1\}^m$$

H is an efficient indexing function to map each x i from d-dimensional Euclidean space to m-dimensional Hamming space y_i , we define H as follows.

Θ is a semantic hashing function

Sparse Spectral Hashing

Step 1:

We initially have X= n x d dimensional space. We transform it into n x m space (m << d). We do this using sparse PCA.

Given the covariance matrix $\boldsymbol{\Sigma}$ of $\boldsymbol{X}$ ($\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$), a d-dimensional sparse PC $\boldsymbol{p}$ can be obtained according to convex relaxation mentioned before. Then the covariance matrix $\boldsymbol{\Sigma}$ is updated as following:

$$\boldsymbol{\Sigma} := \boldsymbol{\Sigma} - (\boldsymbol{p}^T \boldsymbol{\Sigma} \boldsymbol{p}) \boldsymbol{p} \boldsymbol{p}^T \quad (5)$$

The update of $\boldsymbol{\Sigma}$ in formula (5) is repeated by m times and convex relaxation is implemented to each updated $\boldsymbol{\Sigma}$. By this way, we can get m sparse PCs and construct a matrix $\boldsymbol{M} \in \mathbb{R}^{d \times m}$ whose columns are the sparse PCs. The matrix $\boldsymbol{B}$ can be defined by matrix multiplication as $\boldsymbol{B} = \boldsymbol{X}\boldsymbol{M}$.

Step 2:

The mapping of B in Euclidean space to Y in Hamming space.

Generally, we do it using a function: $$z(i,j) = \Theta\left(\delta_j^{min}, B(i,t)\right) = \sin\left(\frac{\pi}{2} + \frac{k\pi}{B_{(:,t)}^{max} - B_{(:,t)}^{min}} B(i,t)\right)$$

We calculate the threshold t by using weak learners(Adaboost) while using known binary codes in the training set. An appropriate t value is selected and plucked into the below equation to get the predicted binary codes z(i,j).

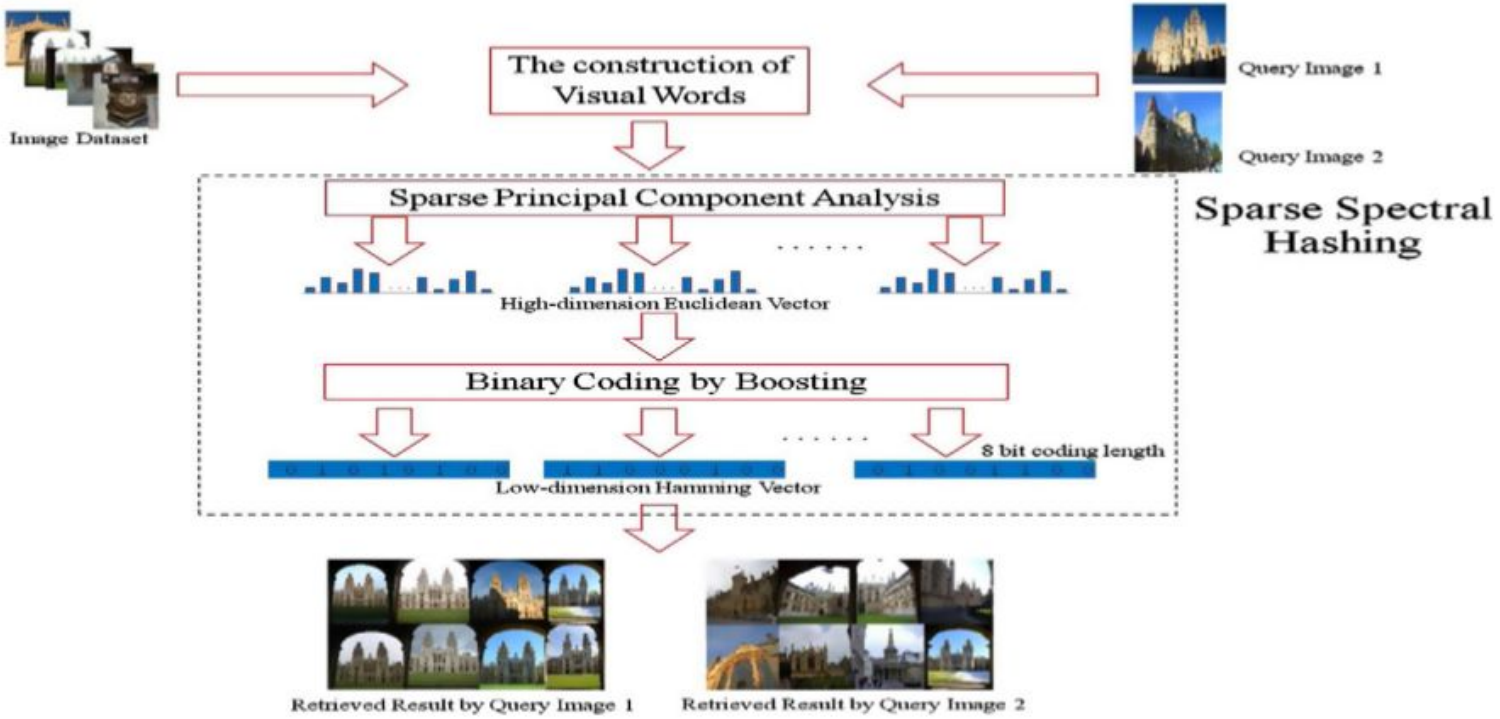
$$y(i,j) = \begin{cases} +1, \text{if } z(i,j) \leqslant T_j \\ -1, \text{otherwise} \end{cases}$$

Final Retrieval

We now have binary codes for the query using the semantic hash function.

We use the Hamming distance to get the similar documents from the Hamming space.

Hamming distance (d) = count_different_bits(binary code1, binary code2).



Flowchart of working of SSH

In our next presentation, we'll be explaining the hash function (SSC Bosting) in detail and the implementation of Sparse Spectral Hashing.

THANK YOU