Inductive Hashing on Manifolds
Fumin Shen, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, Zhenmin Tang

Problem
Most hashing methods are designed to generate binary codes that preserve the Euclidean distance in the original space. Manifold learning techniques, in contrast, are better able to preserve the intrinsic geodesic distance. However, the following problems hinders the use of manifold learning for hashing:

1. Prohibitive computational cost
2. Out-of-sample extension problem – Most manifold learning methods are non-parametric.

Existing methods - All based on Laplacian eigenmaps
- Spectral Hashing: uniform data assumption
- Anchor Graph Hashing: Nystrom extension
- Self-Taught Hashing: out-of-sample extension by SVM

Contributions
We showed how to learn compact binary embeddings on their intrinsic manifolds. The proposed approach here is inspired by Delalleau et al.[2], where they have focused on semi-supervised classification. Our contributions include

1. Make semantic hashing on data manifolds practical by an inductive hashing framework
   - Efficient: Linear indexing time \(O(n)\) and Constant query time \(O(1)\)
   - Effective: Better than L2 scan with t-SNE et al.
2. Connect manifold learning and hashing
   - Any manifold learning methods can be applied in the hashing framework.
   - Evaluation of 9 manifold learning methods for hashing

Formulation
Denote the training data by \(X := \{x_1, x_2, \ldots, x_n\}\) and their manifold embedding by \(Y := \{y_1, y_2, \ldots, y_n\}\). Given a new data point \(x_r\) we aim to generate an embedding \(y_r\) which preserves the local neighborhood relationships:

\[
\min_{y} \sum_{i=1}^{n} w(x_r, x_i) |y_r - y_i|^2.
\]

where \(w(x_r, x_i)\) is the similarity. which is only non-zero for its \(k\) nearest neighbors. This results in

\[
y_r^* = \frac{\sum_{i=1}^{n} w(x_r, x_i) y_i}{\sum_{i=1}^{n} w(x_r, x_i)}.
\]

This provides a simple inductive formulation for the embedding of a new data point by a linear combination of the base embeddings.

We developed a prototype algorithm which was able to approximate \(y_r\) using only a small base set with a good bound: \(m\) clusters were used to cover \(Y\). Observing that the cluster centers have the largest overall weight w.r.t the points from their own cluster, i.e., \(\sum_{i\in j} w(e_i, x_i)\), we then approximately select all cluster centers to express \(y_r\) for efficiency.

We obtain our general inductive hash function by binarizing the low-dimensional embedding

\[
h(x) = \text{sgn}\left(\sum_{i=1}^{m} w(x, c_i) y_i \right),
\]

where \(Y_B := \{y_1, y_2, \ldots, y_m\}\) is the embedding for the base set \(B := \{c_1, c_2, \ldots, c_m\}\), which is the cluster centers obtained by K-means. With this, the embedding for the training data becomes

\[
Y = W_{XB} Y_B.
\]

where \(W_{XB}\) is defined such that \(W_{ij} = \frac{w(x_i, c_j)}{\sum_{j=1}^{m} w(x_i, c_j)}\) for \(x_i \in X, c_j \in B\). We term our hashing method Inductive Manifold-Hashing (IMH). For IMH, any manifold learning methods can be applied to generate the low dimensional embedding \(Y_B\) as a base.

Algorithm
Algorithm 1: Inductive Manifold-Hashing
Input: Training data \(X := \{x_1, x_2, \ldots, x_n\}\), code length \(r\), base set size \(m\), neighborhood size \(k\).
1. Generate the base set \(B\) by random sampling or clustering (e.g., K-means);
2. Embed \(B\) into the low dimensional space by any appropriate manifold learning method;
3. Obtain the low dimensional embedding \(Y\) for the whole dataset inductively by (4);
4. Threshold \(Y\) at zero;
Output: Binary codes \(Y := \{y_1, y_2, \ldots, y_n\} \in \mathbb{R}^{n \times r}\)

Evaluation
Evaluation of manifold learning methods

Computation times (seconds) on MNIST

Classification accuracy with linear SVM

Results
Retrieval results on CIFAR-10 (60K)

Retrieval results on MNIST (70K)

Retrieval results on SIFT and GIST1M

References