We propose an unsupervised hashing method called Anchor-based Probability Hashing (i.e. APHash) to preserve the similarities by exploiting the distribution of data points:

- Distances are transformed into probabilities in both original and hash spaces.
- Instead of constructing $n \times n$ probability matrices within the whole training set as in SePH[1], we first randomly select a small set of $m$ anchors then construct asymmetric probability matrices of size $m \times n$ to avoid high complexity issue.

### Method

**Step 1**

In the original space, we construct probability matrix $P$ between the small set of $m$ anchors $C$ and the whole training set $X$ of $n$ data items. Define $p_{ij}$ as the probability of assigning $x_j$ to anchor $c_i$. $P$ is normalized row by row.

$$d(c_i, x_j) = \text{distance}(c_i, x_j)$$

$\theta$ indicates the threshold indicating the average distance between $c_i$ and its $k$ nearest neighbors computed as follows:

$$\theta = \frac{\sum_{j \in N_k(c_i)} d(c_i, x_j)}{k}$$

**Step 2**

In hash space, we define $Q$ as the probability distribution with Hamming distance. Inspired by t-SNE[2], we utilize t-distribution with one degree freedom to transform Hamming distance into probabilities.

$$q_{ij} = \frac{(1 + g(h_i, b_j))^{-1}}{\sum_{t \in [1,m]} (1 + g(h_t, b_j))^{-1}}$$

$h_i$ and $b_j$ denote hash codes of anchor point and training set item respectively. Hamming distance can be transformed to Euclidean distance with $g(h_i, b_j) = \frac{1}{4} ||h_i - b_j||_2^2$.

During optimization process, they are relaxed to real-value vectors $\bar{h}$ and $\bar{b}$ to make the problem tractable.

**Step 3**

The overall objective function of APHash containing two parts: KL-divergence loss and Quantization loss.

$$J = J_0 + \lambda J_1$$

$\lambda$ is a hyper parameter to balance two parts.

$J_0$: KL-divergence loss measures the difference between $P$ and $Q$ to make them as consistent as possible.

$$J_0 = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

$J_1$: Quantization loss forces the relaxed entries of matrices $\bar{H}$ and $\bar{B}$ to be closed to $\pm 1$ during optimization.

$$J_1 = 1/Z_H ||\bar{H} - 1||_2^2 + 1/Z_B ||\bar{B} - 1||_2^2$$

We apply alternating stochastic gradient descent method to optimize the model.

- We compute the derivative w.r.t. $\bar{h}$ and $\bar{b}$ as $\frac{\delta J}{\delta \bar{h}}$ and $\frac{\delta J}{\delta \bar{b}}$.
- The overall objective is optimized w.r.t one parameter while fixing another until model converges.
- We use $\text{sign}()$ function to obtain final hash code $H$ and $B$.

**Step 4**

For out-of-sample extension, linear model is applied to learn hash function with the learned binary codes of anchor set $H$. The objective function is

$$L = \min_W ||H - W^T C||_2^2 + \alpha ||W||_2^2$$

The learned binary code $B$ is fixed and treated as index of database.

### Experimental Results

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR-10@8-bit</th>
<th>YouTube Faces@8-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0.6100</td>
<td>0.6975</td>
</tr>
<tr>
<td>LSH</td>
<td>0.1170</td>
<td>0.1222</td>
</tr>
<tr>
<td>SH</td>
<td>0.1295</td>
<td>0.1303</td>
</tr>
<tr>
<td>ACH</td>
<td>0.1507</td>
<td>0.1575</td>
</tr>
<tr>
<td>DSH</td>
<td>0.1470</td>
<td>0.1580</td>
</tr>
<tr>
<td>Sph</td>
<td>0.1465</td>
<td>0.1487</td>
</tr>
<tr>
<td>OEH</td>
<td>0.1373</td>
<td>0.1331</td>
</tr>
<tr>
<td>ITQ</td>
<td>0.1545</td>
<td>0.1650</td>
</tr>
<tr>
<td>APHash</td>
<td>0.1650</td>
<td>0.1698</td>
</tr>
</tbody>
</table>

### References
