哈希图半监督学习方法及其在图像分割中的应用

张晨光 [1, 2] 李玉鑑 [1]

摘 要 图半监督学习 (Graph based semi-supervised learning, GSL) 方法需要花费大量时间构造一个近邻图, 速度比较慢. 本文提出了一种哈希图半监督学习 (Hash graph based semi-supervised learning, HGSL) 方法, 该方法通过局部敏感的哈希函数进行近邻搜索, 可以有效降低图半监督学习方法所需的构图时间. 图像分割实验表明, 该方法一方面可以达到更好的分割效果, 使分割准确率提高 0.47 %左右; 另一方面可以大幅度减小分割时间, 以一幅大小为 300 像素 × 800 像素的图像为例, 分割时间可减少为图半监督学习所需时间的 28.5 %左右.

关键词 哈希图半监督学习, 图半监督学习, 局部敏感的哈希函数, 图像分割

DOI 10.3724/SP.J.1004.2010.01527

Hash Graph Based Semi-supervised Learning Method and Its Application in Image Segmentation

ZHANG Chen-Guang[1, 2] LI Yu-Jian[1]

Abstract Graph based semi-supervised learning (GSL) method runs slowly because of the need of much time to construct a neighbor graph. This paper presents a hash graph based semi-supervised learning (HGSL) method, which can search neighbors by locality sensitive hashing function and efficiently reduce the time for GSL to construct a neighbor graph. Image segmentation experiments show that HGSL has an improvement of 0.47 % in average segmenting accuracy, and can greatly reduce the segmenting time, e.g., it takes about 28.5 % of the time for GSL to segment an image with size of 300 × 800.

Key words Hash graph based semi-supervised learning (HGSL), graph based semi-supervised learning (GSL), locality sensitive hashing function, image segmentation

图半监督学习 (Graph based semi-supervised learning, GSL) 方法的核心思想是通过构造一个近邻图[1−3], 把已知标号信息逐步传递到未标号的数据点上, 并由此确定所有数据点的标号. 近邻图中, 顶点表示 (已标号和未标号) 数据点, 边表示数据点之间的相似程度.

图半监督学习过程, 可以看成是两个独立的步骤, 第一个步骤是构图的过程, 该步骤因为需要对数据集中每一个数据点寻找近邻, 具有比较高的时间复杂度 $O(n^2d)$, 第二个步骤是传递标号信息的过程, 时间复杂度为 $O(n^2kc)$, 这里 n, d 和 c 分别表示数据点的个数、维数和类别数, k 表示每个数据点的近邻数[1]. 图半监督学习技术因为受限于这两个过程中比较高的时间复杂度, 速度一般比较慢. 目前, 尽管有些方法可以通过精选子集的途径来降低图半监督学习的数据规模, 但是一方面通过这些方法得到的并不是 "精确" 解, 另一方面子集上的学习过程依然存在构图和标号传递时间复杂度过高的问题[1]. 所以, 如何降低图半监督学习的时间复杂度是一个非常重要的问题, 一种解决途径就是采用局部敏感的哈希函数.

局部敏感的哈希方法 (Locality sensitive hashing, LSH) 是一种有效的近似最近邻搜索算法, 它在语音识别、文本分类与数据挖掘中都有应用[4−6], 其基本原理是: 通过构造对局部敏感的哈希函数, 将距离相差较小的数据点以较大的概率映射到同一个哈希键值, 而距离相差较大的数据点以较大的概率散列到不同的哈希键值, 这样在检索近邻点时只需穷举分析具有相同哈希键值的数据点即可. 局部敏感的哈希方法检索近邻的时间复杂度是 $O(n^{\rho}\log_{1/p}n)$, 其中 $\rho << 1$ 且 $0 < p < 1/2$[7].

本文采用局部敏感的哈希函数改进图半监督学习的构图过程, 提出了哈希图半监督学习 (Hash graph based semi-supervised learning, HGSL) 方法. 该方法构图的时间复杂度仅为 $O(n^{\rho+1}\log_{1/p}n)$, 与原方法相比时间复杂度大为降低. 图像分割实验表明, 该方法不仅有更好的分割效果, 且分割时间大

收稿日期 2009-06-09 录用日期 2010-06-12
Manuscript received June 9, 2009; accepted June 12, 2010
国家自然科学基金 (60775010), 北京工业大学高层次人才建设项目资助
Supported by National Natural Science Foundation of China (60775010) and High-level Personnel Development Project of Beijing University of Technology

1. 北京工业大学计算机学院 北京 100124 2. 海南大学信息科学技术学院 海口 571737
1. College of Computer Science and Technology, Beijing University of Technology, Beijing 100124 2. College of Information Science and Technology, Hainan University, Haikou 571737

幅度减少. 比如, 当图像尺寸为 300 像素 × 800 像素时, 可减少 71.5 %左右.

1 图半监督学习算法框架

设 $\chi = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_l, \boldsymbol{x}_{l+1}, \cdots, \boldsymbol{x}_n\}$ 是一个向量点集, 其中前 l 个点有初始类别标号 $y_i \in \{1, 2, \cdots, c\}$ $(1 \le i \le l)$, 其余点没有标号. 半监督学习的目标就是利用 χ 中所有点 (包括有标号点和无标号点), 建立一个判别模型去标注 χ 中没有标号的点.

如果令 Γ 表示 $n \times c$ 的非零实数矩阵的一个集合, 向量 $F_i = [F_{i1}, F_{i2}, \cdots, F_{ic}]$, 那么 Γ 中任意一个矩阵 $F = [F_1^{\mathrm{T}}, F_2^{\mathrm{T}}, \cdots, F_n^{\mathrm{T}}]^{\mathrm{T}}$ 对应于向量点集 χ 上的一个分类方法, 其中 $\boldsymbol{x}_i$ 的标号定义为

$$y_i = \arg \max_{0 \le j \le c} F_{ij}$$

图半监督学习算法通常包含的步骤如下[8]:

步骤 1. 构造初始状态矩阵 $Y_{n \times c}$, 假设 $\boldsymbol{x}_i$ $(i = 1, 2, \cdots, n)$ 的初始标号为 y_k, 则

$$Y_{ij} = \begin{cases} 1, & j = y_k \\ 0, & j \ne y_k \end{cases} \tag{1}$$

步骤 2. 对 χ 上每个数据点 $\boldsymbol{x}_i$ 寻找它的 K 近邻, 记为 $N(\boldsymbol{x}_i)$, $i = 1, 2, \cdots, n$. 构造 χ 上的相似距离矩阵 $W_{n \times n}$:

$$w_{ij} = \begin{cases} \exp(-\dfrac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}), & i \ne j \text{ 且 } j \in N(\boldsymbol{x}_i) \\ 0, & \text{否则} \end{cases} \tag{2}$$

w_{ij} 称为 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 的相似距离, 其中 σ 是一个常数, $\|\cdot\|$ 表示向量取范数.

步骤 3. 令 $S = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, 其中对角矩阵 $D = (D_{ij})_{n \times n}$, $D_{ii} = \sum_{j=1}^{n} w_{ij}$, $D_{ij} = 0\,(i \ne j)$.

步骤 4. 令 $F(0) = Y$, 迭代计算

$$F(t+1) = \alpha S F(t) + (1 - \alpha) Y \tag{3}$$

直到收敛, 这里 $F(t+1)$ 与 $F(t)$ 都属于 Γ, 且 α 是介于 0 到 1 之间的常数.

步骤 5. 假设 F^* 为迭代的结果, 则数据 $\boldsymbol{x}_i$ 的判别结果为 $y_i = \arg\max_{0 \le j \le c} F_{ij}^*$.

如果把 χ 中的向量看作图的顶点, 把 W 看作顶点与顶点之间的距离矩阵, 图半监督学习算法的本质相当于最小化图上的能量函数[8]:

$$Q(F) = \frac{1}{2}\Big(\sum_{i,j=1}^{n} w_{ij} \left\| \frac{1}{\sqrt{D_{ii}}} F_i - \frac{1}{\sqrt{D_{jj}}} F_j \right\|^2 + \mu \sum_{i=1}^{n} \|F_i - Y_i\|^2 \Big)$$

其中, $\mu > 0$, $Q(F)$ 的第一项表示相近点标号的相差程度, 而第二项表示最后的结果和初始状态的相差程度.

图半监督学习算法的计算时间主要集中在步骤 2 构图的过程和步骤 4 的标号传递的过程, 时间复杂度分别为 $\mathrm{O}(n^2 d)$ 和 $\mathrm{O}(n^2 kc)$, n, d 和 c 分别表示数据点的个数、维数和类别数, k 表示每个数据点的近邻数, 本文的主要目标就是降低步骤 2 的构图过程中的时间复杂度.

2 哈希图半监督学习方法

由于图半监督学习方法需要花费大量的时间在每个数据点的近邻搜索上, 因此提高近邻搜索速度是减少构图时间的关键. 目前, 近邻搜索算法包括近似排除搜索算法 (Approximating and eliminating search algorithm, AESA)、线性近似排除搜索算法 (Linear approximating and eliminating search algorithm, LAESA)、KD-tree、R-tree 和 LSH 等[7−11]. AESA 和 LAESA 需要大量的时间进行预处理, 并不适用于图半监督学习方法的近邻搜索. 基于 KD-tree 和 R-tree 等 tree 结构的近邻搜索算法需要先对搜索空间进行划分, 同样会造成时间损失, 而且在数据维数比较高的时候, 这些方法的性能会下降到与线性搜索一样[12], 因此也不适合图半监督学习方法. 相反, LSH 在进行近邻搜索时不仅具有非常快的检索速度 (高维下比 KD-tree 快 40 倍[7]), 而且可以避免因为数据维数增加导致的搜索性能下降. 如果用 LSH 构造近邻图, 就有可能大幅度提高图半监督学习的速度. 下面将对这一问题详细论述.

2.1 局部敏感的哈希方法

局部敏感的哈希方法 (LSH) 的基本思想是通过构造一组局部敏感的哈希函数完成近邻检索.

设 (S, d_x) 是一个度量空间, $B(\boldsymbol{q}, r) = \{\boldsymbol{p} \in S | d_x(\boldsymbol{p}, \boldsymbol{q}) \le r\}$, 其中 $\boldsymbol{q} \in S$, r 表示半径. 一组映射 $H = \{h : S \to U\}$ 称之为 (r_1, r_2, p_1, p_2) 敏感, 如果它们满足下面两个条件:

1) 对任意 $\boldsymbol{v}, \boldsymbol{q} \in S$, 如果 $\boldsymbol{v} \in B(\boldsymbol{q}, r_1)$, 则 $\Pr[h(\boldsymbol{q}) = h(\boldsymbol{v})] \ge p_1$;

2) 对任意 $\boldsymbol{v}, \boldsymbol{q} \in S$, 如果 $\boldsymbol{v} \notin B(\boldsymbol{q}, r_2)$, 则 $\Pr[h(\boldsymbol{q}) = h(\boldsymbol{v})] \le p_2$.

在条件 1) 和 2) 中, $1 \ge p_1 > p_2 \ge 0$ 且 $0 < r_1 < r_2$.

通过这组映射可以构造局部敏感的哈希函数如下:

$$g(\boldsymbol{x}) = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \cdots, h_k(\boldsymbol{x})]^{\mathrm{T}}$$

其中, 哈希映射 $h_i \in H$, $i = 1, 2, \cdots, k$, $\boldsymbol{x} \in S$.

在预处理阶段, 每个数据点都通过哈希函数 g_1, $g_2,\cdots,g_l$ 计算出 l 个哈希键值, 然后将该数据点依次装入哈希表 T 中该哈希键值对应的位置, 其中 g_1, $g_2,\cdots,g_l$ 都是由 H 中随机选取 k 个映射构成. 在检索阶段, 对于待检索点 $\boldsymbol{q}$, 只需搜索哈希表 T 中 $g_1(\boldsymbol{q})$, $g_2(\boldsymbol{q})$, $\cdots$, $g_l(\boldsymbol{q})$ 对应的位置即可.

k 和 l 的取值都会影响 LSH 进行检索的速度和准确度. 一般地, l 增大, 则检索的准确度增加, 但相应的候选数据点也会增加, 从而降低检索速度; 相反, k 增大, 则检索速度增加, 但检索的准确度会下降. 在实际应用中, k 和 l 的取值可以从数据集中选出一些数据点进行仿真实验估算得到[13]. 关于 k 和 l 的取值, 以及 LSH 的算法时间复杂度有下面的结论[7, 13]:

假设 H 是度量空间 (S,d_x) 上的一组 (r_1,r_2,p_1,p_2) 敏感映射, $\boldsymbol{q}$ 为待检索的点, 则存在 k 和 l 使得下面两个事件以大概率成立:

1) 如果存在 $\boldsymbol{p}^*\in B(\boldsymbol{q},r_1)$, 则存在某些 $j=1,2,\cdots,l$, 使得 $g_j(\boldsymbol{p}^*)=g_j(\boldsymbol{q})$;

2) 与 $\boldsymbol{q}$ 距离大于 r_2 且与 $\boldsymbol{q}$ 具有相同哈希键值的数据点的个数小于 $3l$, 即:

$$\sum_{j=1}^{l}\left|(S-B(\boldsymbol{q},r_2))\cap g_j^{-1}(g_j(\boldsymbol{q}))\right|<3l$$

且 LSH 完成检索的空间复杂度为 $\mathrm{O}(dn+n^{1+\rho})$, 所需距离的平均计算次数为 $\mathrm{O}(n^{\rho})$, 哈希映射的平均计算次数为 $\mathrm{O}(n^{\rho}\log_{1/p_2}n)$, 其中 n 为数据点个数, d 为数据点的维数, $\rho=\frac{\log 1/p_1}{\log 1/p_2}$, $|\cdot|$ 表示集合中的点数.

从上面的结论可以看出, 对于给定的度量空间, 只要能找到该度量下的一组 (r_1,r_2,p_1,p_2) 敏感映射, 那么就可以通过这组映射构造出一组恰当的哈希函数, 从而保证以大概率检索出与待检索数据点之间距离小于 r_1 的所有近邻点, 且 LSH 对单个数据点完成检索的总时间复杂度为 $\mathrm{O}(n^{\rho}\log_{1/p_2}n+n^{\rho})$, 是数据规模的次线性函数. 因此, 如果数据集比较大, 且维数比较高, LSH 就能够显著加快检索速度.

2.2　$\boldsymbol{p}$-stable 分布及敏感哈希函数的构造

在给定度量空间中找到一组 (r_1,r_2,p_1,p_2) 敏感映射是进行 LSH 检索的关键. 下面将介绍 l_p 距离下的局部敏感哈希函数的构造方法.

如果数据点 $\boldsymbol{x}=[x_1,x_2,\cdots,x_d]^{\mathrm{T}}$ 和 $\boldsymbol{y}=[y_1,y_2,\cdots,y_d]^{\mathrm{T}}$, 则 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 之间的 l_p 距离定义为

$$\|\boldsymbol{x}-\boldsymbol{y}\|_p=\left(\sum_{i=1}^{d}(x_i-y_i)^p\right)^{\frac{1}{p}}$$

给定实数集上的分布 D 和常数 $p,\xi_1,\xi_2,\cdots,\xi_p$ 是服从 D 的独立同分布的随机变量, 如果对任意给定的 p 个实数 $v_1,v_2,\cdots,v_p$, $\sum_{i=1}^{p}v_i\xi_i$ 和 $(\sum_{i=1}^{p}|v_i|^p)^{1/p}\xi$ 具有相同的分布, 则称 D 是 p-stable 分布, 其中 ξ 是服从分布 D 的随机变量.

对给定数据点 $\boldsymbol{x}$, $\boldsymbol{y}$ 和 p-stable 分布 D, 如果 ξ 是服从分布 D 的随机变量, $\boldsymbol{a}$ 是 d 维向量且 $\boldsymbol{a}$ 的每一个分量都是服从分布 D 且相互独立的随机变量, 那么 $\boldsymbol{x}\cdot\boldsymbol{a}-\boldsymbol{y}\cdot\boldsymbol{a}$ 和 $\|\boldsymbol{x}-\boldsymbol{y}\|_p\xi$ 具有相同的分布, 这意味着 $\boldsymbol{x}$ 与 $\boldsymbol{y}$ 之间的 l_p 距离可以通过 $\boldsymbol{x}-\boldsymbol{y}$ 在 $\boldsymbol{a}$ 上的投影进行估计. 因此, 可以构造 (r_1,r_2,p_1,p_2) 敏感映射[14] 如下:

$$h_{\boldsymbol{a},b}(\boldsymbol{x})=\left\lfloor\frac{\boldsymbol{a}\cdot\boldsymbol{x}+b}{w}\right\rfloor \tag{4}$$

其中, $w>0$ 是一个常数, b 是取自均匀分布 $[0,w]$ 的一个实数.

不难证明, $h_{\boldsymbol{a},b}(\boldsymbol{x})$ 是 (r_1,r_2,p_1,p_2) 敏感映射. 事实上, 令 $f_p(t)$ 表示 p-stable 分布绝对值的概率密度函数, $r=\|\boldsymbol{x}-\boldsymbol{y}\|_p$, 则 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 具有相同哈希值的概率为[14]:

$$p(r)=\Pr[h_{\boldsymbol{a},b}(\boldsymbol{x})=h_{\boldsymbol{a},b}(\boldsymbol{y})]=\int_0^w\frac{1}{r}f_p(\frac{t}{r})(1-\frac{t}{w})\mathrm{d}t \tag{5}$$

$p(r)$ 是关于 r 的单调下降函数, 因此距离越小的点映射到相同哈希值的概率越大, 距离越大的点映射到相同哈希值的概率越小, 即 $h_{\boldsymbol{a},b}(\boldsymbol{x})$ 的确是 l_p 距离下的一组 $(r_1,r_2,p(r_1),p(r_2))$ 敏感映射.

2.3　相似距离矩阵的快速构造

下面将利用局部敏感哈希函数给出一种快速构造相似距离矩阵的方法.

根据式 (2), 计算 χ 的相似距离矩阵 $W_{n\times n}$, 需要对任意 $\boldsymbol{x}_i\in\chi$ 都预先得到 $\boldsymbol{x}_i$ 的 K 近邻数据点集 $N(\boldsymbol{x}_i)$. 所以, 要快速构造 $W_{n\times n}$, 关键是如何快速计算 $N(\boldsymbol{x}_i)$.

由文献 [15] 可知, 在 l_2 距离下一个重要的 2-stable 分布就是高斯分布:

$$\Psi(x)=\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{x^2}{2}}$$

因此, 通过高斯分布就可以在 l_2 距离下构造一组 (r_1,r_2,p_1,p_2) 敏感映射 $H=\{h_{\boldsymbol{a},b}(\boldsymbol{x})\}$, 其中向量 $\boldsymbol{a}$ 的每个分量都是服从高斯分布的实数, b 是在 $[0,w]$ 上服从均匀分布的一个实数, $w>0$ 是一个常数. 对任意的 $\boldsymbol{x}_i\in\chi$, 通过哈希函数 $g_1,g_2,\cdots,g_l$ 计算出 l 个哈希键值, 并将 $\boldsymbol{x}_i$ 依次存入哈希表中分别与这

l 个哈希键值对应的 l 个哈希桶 $B_1, B_2, \cdots, B_l$ 中, 令 $N_h(\boldsymbol{x}_i) = \cup_k B_k$.

由 LSH 的性质可知, $N_h(\boldsymbol{x}_i)$ 的数据点数以大概率小于 $3l$, 但理论上应包含 $N(\boldsymbol{x}_i)$ 中的绝大数点, 因此 $N_h(\boldsymbol{x}_i)$ 可以看作是近邻数据点集 $N(\boldsymbol{x}_i)$ 的一种近似, 也就是说, 可以用 $N_h(\boldsymbol{x}_i)$ 代替 $N(\boldsymbol{x}_i)$ 构造相似距离矩阵. 所以, χ 上的相似距离矩阵 $W_{n\times n}$ 可以用下面的快速方法近似构造:

$$w_{ij} = \begin{cases} \exp\left(-\dfrac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right), & i \neq j \text{ 且 } j \in N_h(\boldsymbol{x}_i) \\ 0, & \text{否则} \end{cases}$$

其中, σ 是一个常数, $\|\cdot\|$ 表示向量取范数.

由于计算 $N_h(\boldsymbol{x}_i)$ 的时间实际上就是用 LSH 散列 $\boldsymbol{x}_i$ 所需的时间, 因此计算 $W_{n\times n}$ 的时间复杂度为 $\mathrm{O}(n^{\rho+1}\log_{1/p_2} n + n\cdot 2l) \approx \mathrm{O}(n^{\rho+1}\log_{1/p_2} n)$. 考虑到图半监督学习的构图过程本质上就是计算相似距离矩阵 $W_{n\times n}$, 所以哈希图半监督学习方法的构图时间复杂度可以近似为 $\mathrm{O}(n^{\rho+1}\log_{1/p_2} n)$.

此外, 图半监督学习中近邻图的结构对半监督学习的准确性有一定的影响[1]. 采用 $N(\boldsymbol{x}_i)$ 构造近邻图, 有可能因为近邻过于紧密造成标号信息的损失, 而 $N_h(\boldsymbol{x}_i)$ 则有可能避免类似的情况, 从而提高半监督学习的准确性, 下面以极端情况下的一个例子说明这个问题: 令 $\chi = \{\boldsymbol{x}_i | 1 \le i \le 9\}$, 将它分成三组 $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$, $\{\boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6\}$ 和 $\{\boldsymbol{x}_7, \boldsymbol{x}_8, \boldsymbol{x}_9\}$, 假设 $N(\boldsymbol{x}_i)$ 只包含 $\boldsymbol{x}_i$ 所在组的另两个点, 并且只有 $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_9$ 分别具有标号 $+1$ 和 -1. 显然, 如果采用 $N(\boldsymbol{x}_i)$ 构造近邻图, 则 $\boldsymbol{x}_4$, $\boldsymbol{x}_5$ 和 $\boldsymbol{x}_6$ 不能从其余点中得到任何的标号信息, 而如果采用 $N_h(\boldsymbol{x}_i)$ 构造近邻图, 则因为 $\boldsymbol{x}_4$, $\boldsymbol{x}_5$ 和 $\boldsymbol{x}_6$ 有可能成为其余点的近似近邻, 从而获得标号信息.

3 图像分割实验

为了说明哈希图半监督学习方法的实际应用效果, 本文将其应用于图像分割, 具体步骤包括: 1) 图像手工标注; 2) 参数 k 和 l 的估算及哈希函数的构造; 3) 图像分割过程; 4) 实验结果分析.

3.1 图像手工标注

在如图 1 (a) 所示的原始图像上通过画线的形式随意标注一些背景区域中的点和一些待分割区域中的点, 得到如图 1 (b) 所示的标注结果, 其中用手工标注了几条线, 外圈黑线表示背景区域点, 中间白线表示待分割区域点.

3.2 参数 k 和 l 的估算及哈希函数的构造

将待分割图像划分成大小相等的小像素块 $s\times s$, s 的值一般可根据速度和精度要求选择为 $1\sim 10$ 之间的整数. 所有像素块按照其中包含的待分割区域点与背景区域点的多少给予相应的标号: 待分割区域点多则该像素块标为 1, 反之标为 0; 但如果不包含待分割区域点和背景区域点, 则不对该像素块标号. 每个像素块均用 5 维向量 (r, g, b, x, y) 表示, 其中 r, g, b 表示像素块内各点的 RGB 颜色分量均值, x 和 y 表示像素块中心相对于左上角的坐标值. χ 表示所有这些 5 维向量组成的集合, Y 是按照式 (1) 构造的初始状态矩阵.

(a) 原始图像 (b) 标注区域

(a) Original image (b) Marking regions

图 1 原始图像及标注示意图

Fig. 1 An example of image marking

为了最小化向量散列到哈希表 T 中的时间, 可以用以下方法对参数 k 和 l 进行估算:

1) 从 χ 中任意选出两组固定数目的向量, 构成新集合 χ_t 和 χ_q; 选定 k 为常数 (比如 $k = 16$), 取 $l = \lceil \log\delta / \log(1 - p_1^k) \rceil$ ($\lceil\cdot\rceil$ 表示上取整)[7], 以保证成功检索近邻点的概率大于等于 δ (这里取为 0.7), 其中 $p_1 = p(1)$ 由式 (5) 计算.

2) 生成 l 个 k 维复合向量 $\boldsymbol{a}_i = (\boldsymbol{a}_{i1}, \boldsymbol{a}_{i2}, \cdots, \boldsymbol{a}_{ik})$ $(1 \le i \le l)$ 和 l 个实数 b_i $(1 \le i \le l)$, 其中 $\boldsymbol{a}_{ij} = (\boldsymbol{a}_{ij1}, \boldsymbol{a}_{ij2}, \boldsymbol{a}_{ij3}, \boldsymbol{a}_{ij4}, \boldsymbol{a}_{ij5})$ $(1 \le i \le l, 1 \le j \le k)$ 且 $\boldsymbol{a}_{ijz}$ $(1 \le i \le l, 1 \le j \le k, 1 \le z \le 5)$ 均为取自标准正态分布的实数, b_i $(1 \le i \le l)$ 为取自均匀分布 $\mathrm{U}(0, w)$ 的实数.

3) 将 χ_t 中的每一个向量 $\boldsymbol{x}_t$ 用式 (1) 散列到哈希表 T 中, 通过对 χ_q 中的每一个向量 $\boldsymbol{x}_q$ 进行查询, 计算 u, v 和 g 的值, 其中, u 表示计算一次 $h_{\boldsymbol{a},b}(\boldsymbol{x}_i)$ 的平均时间, v 表示检索一次哈希桶的平均时间, g 表示计算一次 w_{ij} 的平均时间.

4) 利用 u, v 和 g 的值, 重新估算新的 k 和 l. 由于 $\boldsymbol{x}_q \in \chi_q$ 和 $\boldsymbol{x}_t \in \chi_t$ 具有相同哈希键值的概率为

$$p_{\boldsymbol{x}_t} = \int_0^w \left(\frac{1}{r}\right)\left(\frac{\sqrt{2}}{\pi}\right) \mathrm{e}^{\frac{-x^2}{2r^2}} \left(1 - \frac{x}{w}\right) \mathrm{d}x$$

其中, r 为 $\boldsymbol{x}_t$ 与 $\boldsymbol{x}_q$ 之间的 l_2 距离, 因此 χ_t 中与 $\boldsymbol{x}_q$ 具有相同键值的向量个数可估计为

$$c_t = \sum_{\boldsymbol{x}_t \in \chi_t} p_{\boldsymbol{x}_t}$$

所以, 对于给定的 k 和 l 值, 对所有 $\boldsymbol{x}_q \in \chi_q$ 计算 $N_h(\boldsymbol{x}_q)$ 的总时间为

$$T_{k,l} = \sum_{\boldsymbol{x}_q \in \chi_q} (u \times k \times l + v \times l + g \times c_t)$$

其中, $l = \left\lceil \log \delta / \log(1 - p_1^k) \right\rceil$. 把 k 限定在一定范围 (比如 $20 \sim 40$) 最小化 $T_{k,l}$ 即可得到较好的参数 k 和 l[13].

在获得较好的参数 k 和 l 后, 重新生成 l 个 k 维复合向量 $\boldsymbol{a}_i = (\boldsymbol{a}_{i1}, \boldsymbol{a}_{i2}, \cdots, \boldsymbol{a}_{ik})$ $(1 \le i \le l)$ 和 l 个实数 b_i $(1 \le i \le l)$, 从而得到性能较好的局部敏感哈希函数:

$$g_i(\boldsymbol{x}) = [h_{i1}(\boldsymbol{x}), h_{i2}(\boldsymbol{x}), \cdots, h_{ik}(\boldsymbol{x})]^{\mathrm{T}},\ i = 1, 2, \cdots, l$$

其中, $h_{ij}(\boldsymbol{x}) = \left\lfloor \frac{\boldsymbol{a}_{ij} \cdot \boldsymbol{x} + b_i}{w} \right\rfloor$ (w 为常数).

3.3　图像分割过程

按照第 2.3 节的方法, 令 $\sigma = 100$ 计算相似距离矩阵 W, 并把 W 归一化为

$$S = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

其中, D 为斜对角矩阵, 且 $D_{ii} = \sum_j w_{ij}$.

然后, 令 $F(0) = Y$, 迭代计算式 (3) 直到收敛 (F 中各个数的平均变化程度 < 0.001), 假设 F^* 为迭代的结果, 无标号向量 $\boldsymbol{x}_i$ 的最终标号取决于 F_i^* 中最大的分量, 即: 若 $F_{i1}^* > F_{i2}^*$, 则该向量对应的像素块属于待分割区域; 否则, 该向量对应的像素块属于背景区域.

3.4　实验结果分析

本文在 DELL OPTIPLEX GX 620 (3.20 GHz CPU, 1.00 GB 内存) 型号的机器上用 VC 2005 构建了一个基于哈希图半监督学习的图像分割系统, 并在数据集 BSD (Berkeley segmentation dataset)[16] 和 SED (Segmentation evaluation database)[17] 上对 HGSL, GSL 和 Lazy snapping[18] 进行了比较实验, 其中 Lazy snapping 是一种改进的图切 (Graph cut) 方法[19]. 部分实验结果列举在图 2 中, 其中每 4 张图片为一组, 第 1, 2 组的图片取自 SED, 其余组则包含了 BSD 测试集第一部分中的所有图片. 每组中的第 1, 2, 3 张图片分别是 Lazy snapping, GSL 和 HGSL 的分割结果.

从图 2 中不难看出, HGSL 的分割效果能够达到与 Lazy snapping 相当的水平, 而且略好于 GSL. 为了更准确地对比 HGSL 和 GSL 的分割效果, 本文

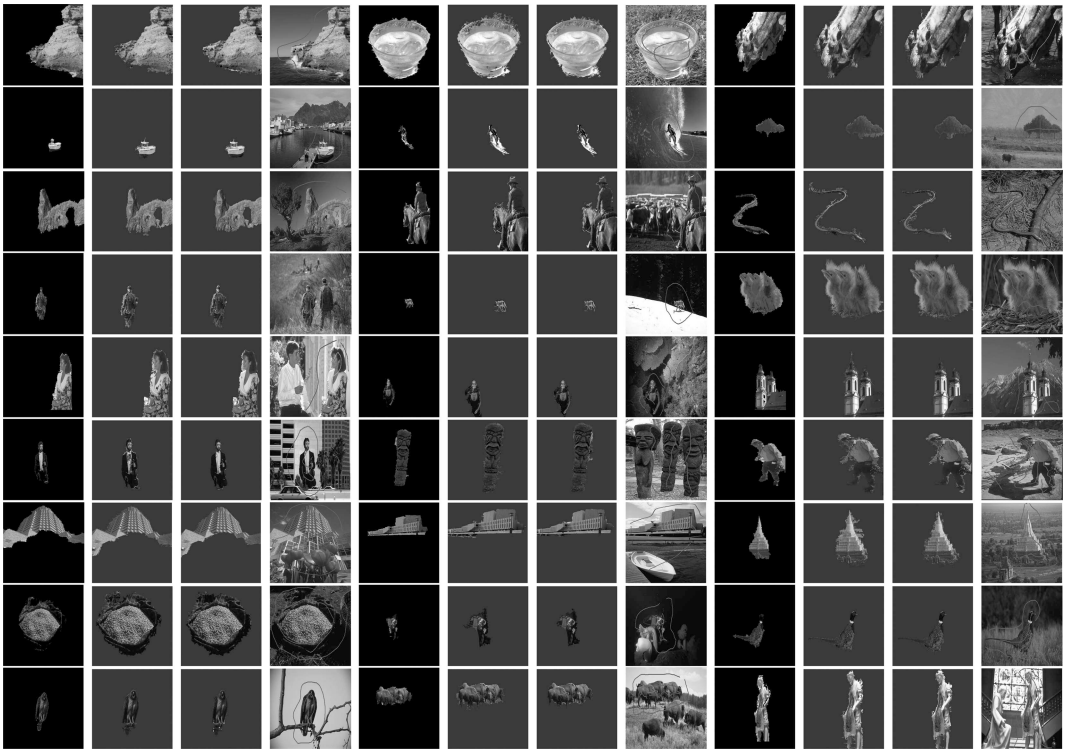
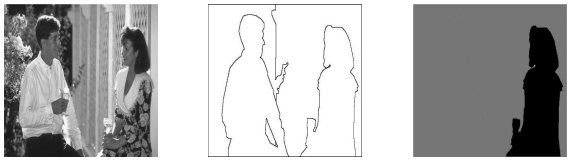


图 2　图像分割效果举例, 其中每 4 张图为一组, 每组的第 1, 2, 3 张分别是 Lazy snapping, GSL 和 HGSL 的分割结果

Fig. 2　Examples of image segmentation results, where four images are in each group the first, second, and third images of the group are generated from lazy snapping, GSL, and HGSL, respectively

在图像库 SED (100 幅图) 和 BSD 测试集 (100 幅图) 上对 HGSL 和 GSL 的分割效果进行了量化: 首先, 按照图像库附带的人工分割边界图, 把待分割物体的颜色填充为黑色, 如图 3 所示. SED 自带有人工填充的结果, 所以此步可以省略. 然后, 将图像中待分割区域中的像素点作为一类, 背景中的像素点作为另一类, 把图像分割看成是两类分类问题, 对比填充图与 HGSL 和 GSL 的分割结果, 以每幅图像上像素点的分类准确率作为衡量其分割效果的标准, 即

$$\text{分割准确率} = \frac{\text{判断正确的像素点数}}{\text{总像素点数}}$$



(a) 原始图像 (b) 边界图 (c) 填充结果

(a) Original image (b) Border image (c) Filling result

图 3 对人工分割边界图进行颜色填充的示意图

Fig. 3 An example of filling segmentation region with color on hand-labeled border images

HGSL 和 GSL 在两个图像库上的平均分割准确率见表 1 所示. 图 4 是把区间 $[0,1]$ 等分成 135 份, 统计两个图像库上分割准确率出现在各个小区间的图像数目得到的直方图.

表 1 HGSL 和 GSL 在 BSD 和 SED 上的平均分割准确率

Table 1 The average segmentation accuracies of HGSL and GSL on BSD and SED

Method	BSD (%)	SED (%)
HGSL	97.670	97.534
GSL	97.198	97.054

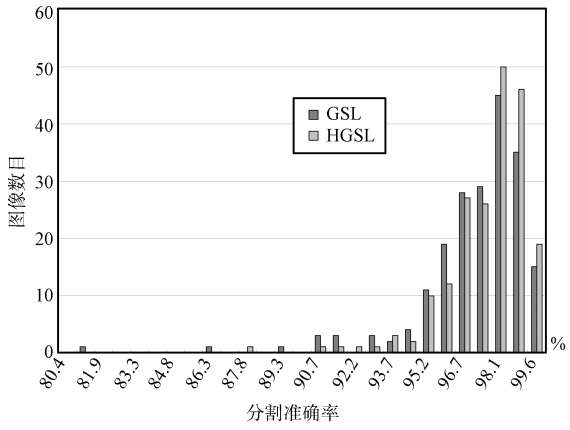


图 4 分割准确率的直方图

Fig. 4 The histograms of segmentation accuracy

从表 1 可以看到 HGSL 在两个数据库上的平均分割准确率都要比 GSL 高 0.47 %, 而从图 4 可以看到当分割准确率小于 98 % 时, 在大部分小区间上采用 HGSL 得到的图像数目都要比 GSL 小, 而在准确率大于 98 % 的各个区间上, HGSL 得到的图像数目都要远大于 GSL, 这充分说明了 HGSL 具有比 GSL 更好的分割效果.

此外, 为了说明 HGSL 在提高 GSL 速度方面的作用, 本文对从 SED 中通过选取、剪切或变换得到的 80 幅图像进行了分割时间的比对实验, 其中大小为 300 像素 × 190 像素的图像共 4 幅 (库中 1 幅, 3 幅由其他 300 像素 × 400 像素的图像剪切得到), 300 像素 × 200 像素的 8 幅, 300 像素 × 225 像素的 40 幅, 300 像素 × 400 像素的 24 幅, 300 像素 × 800 像素的 4 幅 (由 300 像素 × 400 像素的图像通过 Photoshop 改变尺寸得到). 在进行实验时, 分块的大小取为 5 像素 × 5 像素, 分割所需的平均时间见表 2 和图 5. 本文未在 BSD 上比较 HGSL 和 GSL 的速度, 这是因为这些图像的大小均为 481 像素 × 321 像素或者 321 像素 × 481 像素, HGSL 和 GSL 对其中每幅图像的分割时间分别是 5.2 秒和 14.4 秒左右.

表 2 HGSL 和 GSL 对不同大小图像的平均分割时间 (秒)

Table 2 The average segmentation times of HGSL and GSL for different image sizes (s)

Method	300 × 190	300 × 200	300 × 225	300 × 400	300 × 800
GSL	2.609	3.309	3.623	12.269	69.259
HGSL	1.312	1.742	1.998	4.925	19.7

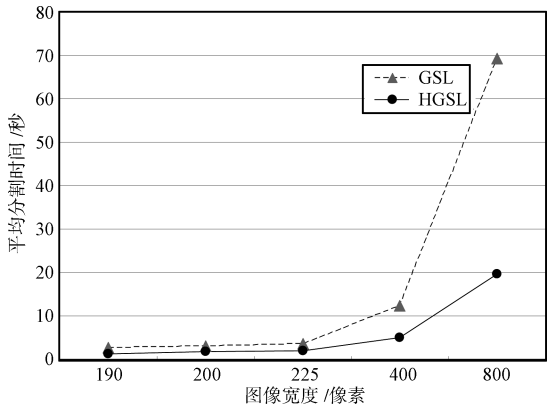


图 5 HGSL 和 GSL 的分割时间随图像宽度的变化

Fig. 5 The variations of segmentation times of HGSL and GSL with different image widths

从表 2 和图 5 易知, 哈希图半监督学习方法 HGSL 能够有效地减少图半监督学习方法 GSL 的图像分割时间, 当图像尺寸达到 300 像素 × 800 像

素时, 分割时间可减少 71.5 %, 而且随着图像的增大, 数据量的增加, HGSL 完成图像分割的时间要远小于 GSL.

4 结论

通过结合图半监督学习方法和局部敏感的哈希函数, 本文提出了一种哈希图半监督学习方法, 该方法不仅能够保证图半监督学习方法的分类精度 (这里相当于分割效果), 而且能够有效地减少图半监督学习方法的分类时间. 在 BSD 测试集 100 幅图像和 SED 的 100 幅图像上完成的分割实验表明, 哈希图半监督学习方法的分割准确率可提高 0.47 %左右, 且具有手工操作简单和分割速度更快的优点. 特别是, 当图像尺寸为 300 像素 × 800 像素时, 哈希图半监督学习方法所需分割时间约为图半监督学习方法的 1/3 左右, 这充分说明哈希图半监督学习方法对提高图半监督学习方法的速度具有重要作用. 在今后的工作中, 本文将借鉴种子图像选择 (Seed image selection)[20] 技术, 进一步改进哈希图半监督学习方法, 使其只需在图像库中标注部分图像, 就能够完成其他相关图像的无标注自动分割.

References

1 Chapelle O, Scholkopf B, Zien A. *Semi-Supervised Learning*. Cambridge: The MIT Press, 2006. 333−341

2 Wang F, Zhang C S, Shen H C, Wang J D. Semi-supervised classification using linear neighborhood propagation. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. New York, USA: IEEE, 2006. 160−167

3 Jia Y Q, Zhang C S. Learning distance metric for semisupervised image segmentation. In: Proceedings of the 15th International Conference on Image Processing. San Diego, USA: IEEE, 2008. 3204−3207

4 Shivakumar N. Detecting Digital Copyright Violations on the Internet [Ph. D. dissertation], Stanford University, USA, 2000

5 Havliwala T H, Gionis A, Indyk P. Scalable techniques for clustering the web. In: Proceedings of the 3rd International Workshop on the Web and Databases. Texas, USA: ACM, 2000. 129−134

6 Yang C. Macs: music audio characteristic sequence indexing for similarity retrieval. In: Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics. New York, USA: IEEE, 2001. 123−126

7 Gionis A, Indyk P, Motwani R. Similarity search in high dimensions via hashing. In: Proceedings of the 25th International Conference on Very Large Data Bases. San Francisco, USA: Morgan Kaufmann Publishers, 1999. 518−529

8 Zhou D Y, Bousquet O, Lal T N, Weston J, Scholkopf B. Learning with local and global consistency. In: Proceedings of the 18th Annual Conference on Neural Information Processing Systems. Cambridge, USA: The MIT Press, 2000. 321−328

9 Rico-Juan J R, Mico L. Comparison of AESA and LAESA search algorithms using string and tree-edit-distance. *Pattern Recognition Letters*, 2003, **24**(9-10): 1417−1426

10 Guttman A. A dynamic index structure for spatial searching. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. Massachusetts, USA: ACM, 1984. 47−57

11 Sproull R F. Refinements to nearest-neighbor search in k-dimensional trees. *Algorithmica*, 1991, **6**(4): 579−589

12 Weber R, Schek H J, Blott S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: Proceedings of the 24th International Conference on Very Large Data Bases. New York, USA: Morgan Kaufmann Publishers, 1998. 194−205

13 Andoni A, Indyk P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 2008, **51**(1): 117−122

14 Datar M, Immorlica N, Indyk P, Mirrokni V S. Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the 12th Annual Symposium on Computational Geometry. New York, USA: ACM, 2004. 253−262

15 Nolan J P. *Stable Distributions-Models for Heavy Tailed Data*. Boston: Birkhauser, 2010. 5−10

16 Martin D, Fowlkes C, Tal D, Malik J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proceedings of the 8th IEEE International Conference on Computer Vision. Vancouver, Canada: IEEE, 2001. 416−423

17 Alpert S, Galun M, Basri R, Brandt A. Image segmentation by probabilistic bottom-up aggregation and cue integration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Minnesota, USA: IEEE, 2007. 1−8

18 Li Y, Sun J, Tang C K, Shum H Y. Lazy snapping. *ACM Transactions on Graphics*, 2004, **23**(3): 303−308

19 Boykov Y Y, Jolly M P. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In: Proceedings of the 8th IEEE International Conference on Computer Vision. Vancouver, Canada: IEEE, 2001. 105−112

20 Batra D, Parikh D, Kowdle A, Chen T, Luo J. Seed image selection in interactive cosegmentation. In: Proceedings of the IEEE International Conference on Image Processing. Cairo, Egypt: IEEE, 2009. 2393−2396

张晨光 海南大学讲师. 2009 年获北京工业大学硕士学位. 主要研究方向为图像处理和模式识别. E-mail: zhangchenguang12@emails.bjut.edu.cn

(**ZHANG Chen-Guang** Lecturer at Hainan University. He received his master degree from Beijing University of Technology in 2009. His research interest covers pattern recognition and image processing.)

李玉鑑 北京工业大学计算机学院教授. 主要研究方向为模式识别和人工智能. 本文通信作者.
E-mail: liyujian@bjut.edu.cn

(**LI Yu-Jian** Professor at the College of Computer Science and Technology, Beijing University of Technology. His research interest covers pattern recognition and artificial intelligence. Corresponding author of this paper.)