

分类号 TP181 密级 公开

UDC _____

学 位 论 文

离散哈希学习

(题名和副题名)

蒋庆远

(作者姓名)

指导教师姓名、职务、职称、学位、单位名称及地址 李武军 副教授

南京大学计算机科学与技术系 南京市栖霞区仙林大道 163 号 210023

申请学位级别 博士 专业名称 计算机科学与技术

论文提交日期 2020 年 9 月 3 日 论文答辩日期 2020 年 8 月 20 日

学位授予单位和日期 _____

答辩委员会主席: 陈松灿 教授

评阅人: 盲审



南京大學

研究生畢業論文 (申請博士學位)

論 文 題 目 _____ 离散哈希学习 _____

作 者 姓 名 _____ 蒋庆远 _____

学 科、专 业 方 向 _____ 计算机科学与技术 _____

研 究 方 向 _____ 机器学习 _____

指 导 教 师 _____ 李武军 副教授 _____

2020 年 9 月 3 日

学 号：**DG1633008**

论文答辩日期：**2020年8月20日**

指 导 教 师： (签字)

Discrete Hashing Learning

by

Qing-Yuan Jiang

Supervised by

Associate Professor Wu-Jun Li

A dissertation submitted to
the graduate school of Nanjing University
in partial fulfilment of the requirements for the degree of
DOCTOR OF PHILOSOPHY
in
Computer Science and Technology



Department of Computer Science and Technology
Nanjing University

September 3, 2020

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目： 离散哈希学习

计算机科学与技术 专业 2016 级博士生姓名： 蒋庆远
指导教师（姓名、职称）： 李武军 副教授

摘 要

哈希学习已经被广泛应用到大规模检索任务中。哈希学习的目标是通过学习哈希函数将数据从原始特征表示映射到尽量保相似性的二值哈希编码表示。由于二值哈希编码定义在二值空间，哈希学习模型的优化变得十分困难。因此，部分哈希学习方法在训练过程中采用直接丢弃二值约束的松弛策略。然而，直接丢弃二值约束将使得哈希学习模型偏离原始的目标。这类方法的检索精度通常因此受损。离散哈希学习方法在学习过程中不丢弃二值约束。因此，离散哈希学习方法通常能达到比采用直接丢弃二值约束策略的哈希学习方法更高的检索精度。本文从非深度单模态、深度单模态、非深度多模态和深度多模态四个应用场景对离散哈希学习进行系统地研究。本文取得的创新成果如下：

- 在非深度单模态场景中，图哈希学习方法是一类重要的非深度单模态哈希学习方法。然而，现有的离散图哈希学习方法无法利用全部图相似度信息进行训练，他们的检索精度也因此受损。同时，现有的离散图哈希学习方法的训练过程低效。为了利用全部图相似度信息，本文提出一种基于特征变换的可扩展图哈希学习方法（**Scalable Graph Hashing with Feature Transformation**, 简称 **SGH**）。**SGH** 设计了一种隐式计算全部图相似度信息的方法，可以在训练过程中利用全部图相似度信息。通过隐式计算图相似度信息，**SGH** 能达到线性复杂度。**SGH** 还设计了一种逐比特离散优化算法学习二值哈希编码。实验结果表明，与现有的离散图哈希学习方法相比，**SGH** 方法可以达到更高的检索精度，同时，**SGH** 方法的训练也更高效。
- 在深度单模态场景中，现有的深度单模态哈希学习方法存在两个问题。一方面，现有的深度单模态哈希学习方法无法使用监督信息同时直接监督二值哈希编码学习和深度特征学习。另一方面，现有的深度单模态哈希学习方法都基于对称哈希框架，训练过程低效。本文首先提出一种深度离散监

督哈希学习方法 (Deep Discrete Supervised Hashing, 简称 DDSH)。DDSH 方法是第一个使用标签对信息同时直接监督二值哈希编码学习和深度特征学习的深度单模态哈希学习方法。同时, DDSH 能够实现两个过程的相互反馈。为了解决深度单模态哈希学习方法训练过程低效的问题, 本文还提出一种非对称深度监督哈希学习方法 (Asymmetric Deep Supervised Hashing, 简称 ADSH)。ADSH 使用非对称哈希建模, 同时设计了一种高效的离散优化算法。实验表明, 与现有的深度单模态哈希学习方法相比, DDSH 方法能达到更高的检索精度。同时, 与除 DDSH 方法外的对称深度单模态哈希学习方法相比, ADSH 方法能在最短时间内达到更高的检索精度。与 DDSH 方法相比, ADSH 方法的训练更加高效。

- 在非深度多模态场景中, 现有基于标签对信息的非深度跨模态离散哈希学习方法的复杂度为训练集大小的平方。计算资源有限时, 这些方法只能使用采样训练集进行训练。同时, 由于复杂度太高, 这些方法的训练过程低效。本文提出一种基于离散隐因子模型的跨模态哈希学习方法 (Discrete Latent Factor Model based Cross-Modal Hashing, 简称 DLFH)。DLFH 设计了一种可以直接学习二值哈希编码的离散优化算法, 并给出了该算法的收敛性证明。DLFH 还设计了一种基于随机采样策略的离散优化算法以提高训练效率。实验表明, 与现有的非深度跨模态哈希学习方法相比, DLFH 方法能达到更高的检索精度。同时, 与现有的跨模态离散哈希学习方法相比, DLFH 方法的训练更加高效。
- 在深度多模态场景中, 本文首次将深度特征学习技术引入到跨模态哈希学习中, 提出一种深度跨模态哈希学习方法 (Deep Cross-Modal Hashing, 简称 DCMH)。DCMH 方法是第一个将二值哈希编码学习和深度特征学习融合到同一个端到端的框架中的跨模态哈希学习方法。本文还提出一种基于深度离散隐因子模型的跨模态哈希学习方法 (Deep Discrete Latent Factor Model for Cross-Modal Hashing, 简称 DDLFH)。DDLFH 通过结合 DLFH 的二值哈希编码学习能力和深度学习的深度特征学习能力, 可以在同一框架中同时完成二值哈希编码学习和深度特征学习。实验表明, 与非深度跨模态哈希学习方法相比, DCMH 方法能达到更高的检索精度。同时, 与现有的非深度跨模态哈希学习方法和深度跨模态哈希学习方法相比, DDLFH 方法能达到更高的检索精度。

关键词: 哈希学习; 大规模数据检索; 离散优化; 离散哈希

南京大学研究生毕业论文英文摘要首页用纸

THESIS: Discrete Hashing Learning

SPECIALIZATION: Computer Science and Technology

POSTGRADUATE: Qing-Yuan Jiang

MENTOR: Associate Professor Wu-Jun Li

Abstract

Hashing has been widely used in many large-scale retrieval applications. The goal of hashing is to learn a hash function to map the data from original features as binary hash codes which can preserve the similarity as much as possible. The optimization of hashing model is difficult because the binary hash codes are defined over discrete space. Some hashing methods adopt the strategy which discards the binary constraint directly during training procedure. However, directly discarding the binary constraint will make the hashing model deviate the original goal of hashing. Thus, the retrieval accuracy of these methods is deteriorated. Discrete hashing learning can learn binary hash code during the training procedure. Hence, compared with the methods which discard the binary constraint directly, discrete hashing learning methods can achieve higher retrieval accuracy. This paper studies the discrete hashing learning from the following four application scenarios: non-deep single modal hashing, deep single modal hashing, non-deep multi-modal hashing and deep multi-modal hashing. The contributions of this thesis are outlined as follows:

- In non-deep single modal scenarios, graph hashing is one of the most important non-deep single modal hashing methods. However, existing discrete graph hashing cannot use the whole graph similarity for training. Hence, the retrieval accuracy of these methods is deteriorated. Furthermore, the training of existing discrete graph hashing is inefficient. To fully use the graph similarity, this paper proposes a graph hashing method, called scalable graph hashing with feature transformation (SGH). SGH designs an approach to compute the whole graph similarity implicitly. Hence, SGH can use the whole graph similarity during training procedure. By computing graph similarity implicitly, SGH can achieve linear complexity. Moreover,

SGH proposes a bitwise discrete optimization algorithm to learn binary hash codes. Experiments demonstrate that SGH can outperform existing discrete graph hashing methods. And the training of SGH is more efficient compared with existing discrete graph hashing methods.

- In deep single modal scenarios, existing deep single modal hashing methods have two issues. On one hand, existing deep single modal hashing methods cannot use supervised information to guide the binary hash codes learning and deep feature learning simultaneously and directly. On the other hand, existing deep single modal hashing methods are symmetric hashing methods and the training of these methods is inefficient. This paper proposes a deep hashing method, called deep discrete supervised hashing (DDSH). DDSH is the first deep single modal hashing method which can use the pairwise supervised information to directly guide the binary hash codes learning and deep feature learning. Thus these two learning procedures can give feedback to each other during training. To solve the inefficiency problem for training deep single modal hashing, this paper also proposes another deep hashing method, called asymmetric deep supervised hashing (ADSH). ADSH uses asymmetric hashing to model hashing problem and designs an efficient learning algorithm. Experiments demonstrate that DDSH can achieve higher retrieval accuracy compared with existing deep single modal hashing methods. Furthermore, ADSH can achieve higher retrieval accuracy within the shortest time compared with symmetric deep single modal hashing methods except for DDSH. Compared with DDSH, the training of ADSH is more efficient.
- In non-deep multi-modal scenarios, the computation complexity of existing pairwise non-deep cross-modal discrete hashing is the square of the training set size. These methods can only use sampled set for training when the computational resource is limited. Moreover, the training of these methods is inefficient due to the high complexity. This paper proposes a cross-modal hashing method, called discrete latent factor model based cross-modal hashing (DLFH). DLFH designs a discrete learning algorithm which can be proved to be convergent to learn binary hash codes. And DLFH designs a stochastic sampling strategy to improve the training efficiency. Experiments demonstrate that DLFH can achieve higher retrieval accuracy compared with existing non-deep cross-modal hashing methods. More-

over, the training of DLFH is more efficient compared with discrete cross-modal hashing methods.

- In deep multi-modal scenarios, this paper introduces the deep feature learning technique into cross-modal hashing for the first time and proposes a cross-modal hashing method, called deep cross-modal hashing (DCMH). DCMH is the first cross-modal hashing method which can integrate the binary hash codes learning and deep feature learning into an end-to-end learning framework. This paper also proposes a cross-modal hashing method, called deep discrete latent factor model for cross-modal hashing (DDLFH). DDLFH seamlessly integrates the discrete learning ability of DLFH and the feature learning ability of deep learning into the same learning framework. Experiments demonstrate that DCMH can achieve higher accuracy compared with non-deep cross-modal hashing methods, and DDLFH can achieve higher retrieval accuracy compared with existing non-deep cross-modal hashing methods and deep cross-modal hashing methods.

keywords: Hashing Learning; Large-scale Data Retrieval; Discrete Optimization; Discrete Hashing

目 次

目 次	vii
插图清单	xi
附表清单	xiii
1 绪论	1
1.1 引言	1
1.2 有待研究的问题	3
1.3 本文工作	4
1.4 论文组织	6
2 研究背景	9
2.1 哈希学习简介	9
2.1.1 哈希学习研究进展	10
2.1.2 在大规模检索中的应用	14
2.2 相关评价标准和常用数据集	17
2.2.1 相关评价标准	17
2.2.2 数据集	20
3 非深度单模态离散哈希	25
3.1 引言	25
3.2 问题定义	26
3.3 基于特征变换的可扩展图哈希学习方法 SGH	27
3.3.1 模型	27
3.3.2 学习算法	30
3.3.3 样本外扩展	32
3.3.4 复杂度分析	32
3.4 实验验证	33

3.4.1	实验设置	33
3.4.2	性能对比	34
3.4.3	超参数实验	37
3.5	本章小结	38
4	深度单模态离散哈希	41
4.1	引言	41
4.2	问题定义	42
4.3	深度离散监督哈希学习方法 DDSH	43
4.3.1	模型	43
4.3.2	学习算法	45
4.3.3	样本外扩展	47
4.4	非对称深度监督哈希学习方法 ADSH	47
4.4.1	模型	48
4.4.2	学习算法	49
4.4.3	样本外扩展	51
4.4.4	复杂度分析	51
4.5	DDSH 方法的实验验证	52
4.5.1	实验设置	52
4.5.2	性能对比	54
4.5.3	有效性验证实验	56
4.6	ADSH 方法的实验验证	57
4.6.1	实验设置	57
4.6.2	性能对比	58
4.6.3	超参数实验	61
4.7	本章小结	62
5	非深度多模态离散哈希	63
5.1	引言	63
5.2	问题定义	64
5.3	基于离散隐因子模型的跨模态哈希学习方法 DLFH	65
5.3.1	模型	65
5.3.2	学习算法	66

目 次	ix
5.3.3 随机学习策略	70
5.3.4 样本外扩展	70
5.3.5 复杂度分析	71
5.4 实验验证	72
5.4.1 实验设置	72
5.4.2 收敛性分析	73
5.4.3 性能对比	74
5.4.4 超参数实验	78
5.5 本章小结	80
6 深度多模态离散哈希	81
6.1 引言	81
6.2 问题定义	82
6.3 深度跨模态哈希学习方法 DCMH	83
6.3.1 模型	83
6.3.2 学习算法	85
6.3.3 样本外扩展	87
6.4 基于深度离散隐因子模型的跨模态哈希学习方法 DDLFH	88
6.4.1 模型	88
6.4.2 学习算法	89
6.4.3 随机学习策略	94
6.4.4 样本外扩展	95
6.5 DCMH 方法的实验验证	95
6.5.1 实验设置	95
6.5.2 性能对比	97
6.5.3 有效性验证实验	99
6.5.4 超参数实验	99
6.6 DDLFH 方法的实验验证	100
6.6.1 实验设置	100
6.6.2 性能对比	101
6.6.3 有效性验证实验	105
6.6.4 超参数实验	106
6.7 本章小结	106

7 总结与展望	109
7.1 本文总结	109
7.2 未来工作展望	110
参考文献	113
A 符号及简称说明	127
A.1 本文使用的通用符号	127
A.2 本文使用的通用简称	129
致 谢	133
简历与科研成果	135

插图清单

1-1	论文组织	6
2-1	哈希方法示意图	10
3-1	SGH 方法中采用的近似	29
3-2	SGH 方法和基准方法在两个数据集上的查准率	36
3-3	SGH 方法对超参数 ρ 的敏感性实验	38
3-4	SGH 方法对超参数 n_k 的敏感性实验	38
4-1	ADSH 方法和基准方法在 MSCOCO 数据集上的训练时间	60
4-2	ADSH 方法对超参数 γ 和 $ \Psi $ 的敏感性实验	62
5-1	DLFH 方法损失函数值随训练变化趋势	74
5-2	DLFH 方法平均查准率均值随训练变化趋势	74
5-3	DLFH 方法和基准方法在三个数据集上的查准-查全率曲线	78
5-4	DLFH 方法对超参数 λ 的敏感性实验	79
5-5	DLFH 方法对超参数 m 的敏感性实验	80
6-1	DCMH 方法和基准方法的查准-查全率曲线（采样训练）	98
6-2	DCMH 方法有效性验证实验	99
6-3	DCMH 方法对超参数 γ 和 ν 的敏感性实验	100
6-4	DDLHFH 方法和基准方法的查准-查全率曲线（采样训练）	103
6-5	DDLHFH 方法和基准方法的查准-查全率曲线（全集训练）	104
6-6	DDLHFH 方法有效性验证实验	106
6-7	DDLHFH 方法对超参数 η_n 的敏感性实验	106

附表清单

2-1	哈希学习中常用的图片数据集	22
2-2	哈希学习中常用的图片-文本数据集	23
3-1	SGH 方法和基准方法在 TINY1M 数据集上的查准率	35
3-2	SGH 方法和基准方法在 FLICKR1M 数据集上的查准率	35
3-3	SGH 方法和基准方法在 TINY1M 数据集上的训练时间	37
3-4	SGH 方法和基准方法在 FLICKR1M 数据集上的训练时间	37
4-1	DDSH 方法中使用的深度神经网络配置细节	54
4-2	DDSH 方法和基准方法在 CIFAR10 数据集上的平均查准率均值	55
4-3	DDSH 方法和基准方法在 NUSWIDE 数据集上的平均查准率均值	55
4-4	DDSH 方法有效性验证实验	57
4-5	ADSH 方法和基准方法在 CIFAR10 数据集上的平均查准率均值	59
4-6	ADSH 方法和基准方法在 MSCOCO 数据集上的平均查准率均值	59
4-7	ADSH 方法和 DDSH 方法在 CIFAR10 数据集上的对比	61
5-1	DLFH 方法和基准方法在 IAPRTC12 数据集上的平均查准率均值	75
5-2	DLFH 方法和基准方法在 FLICKR25K 数据集上的平均查准率均值	75
5-3	DLFH 方法和基准方法在 NUSWIDE 数据集上的平均查准率均值	76
5-4	DLFH 方法和深度跨模态基准方法的对比	77
5-5	DLFH 方法和基准方法在 NUSWIDE 数据集上的训练时间	79
6-1	DCMH 方法中用于文本数据深度特征学习的深度神经网络配置 细节	96
6-2	DCMH 方法和基准方法在 IAPRTC12 数据集上的平均查准率均值	97
6-3	DCMH 方法和基准方法在 FLICKR25K 数据集上的平均查准率均值	98
6-4	DDLHFH 方法和基准方法在 IAPRTC12 数据集上的平均查准率均 值（采样训练）	101

6-5 DDLFH 方法和基准方法在 FLICKR25K 数据集上的平均查准率 均值 (采样训练)	102
6-6 DDLFH 方法和基准方法在 IAPRTC12 数据集平均查准率均值 (全集训练)	102
6-7 DDLFH 方法和基准方法在 FLICKR25K 数据集平均查准率均值 (全集训练)	103
6-8 DLFH、DCMH 和 DDLFH 在 FLICKR25K 数据集上的对比	105
A-1 本文使用的通用符号	127
A-2 本文使用的通用简称	129

第一章 绪论

1.1 引言

随着近年来信息技术的高速发展，互联网上每天产生的数据呈几何式增长，人类已经进入大数据时代^[1-3]。例如，在 2015 年，图片分享平台 Instagram^①的服务器储存的图片达到了 400 万亿张；视频分享网站 Youtube^②2017 年的数据显示，人们平均每天要观看 10 亿视频。大数据在科学研究、医疗、金融、气象等领域具有很高的研究价值。作为计算机科学中的重要研究课题，机器学习^[4,5]为挖掘、利用大数据中的有效信息提供了一种可靠的技术。最近邻检索^[6-9] (Nearest Neighbor Search, 简称 NNS) 是机器学习中的一个基础性的研究课题，其目标是从数据库中检索出和查询样本最近的数据样本。最近邻检索算法被广泛应用在信息检索^[10]、计算机视觉^[4]、数据挖掘^[5,11]等领域。然而，在大数据的应用中，从数据库中检索出查询样本的最近邻样本的时间开销很大。同时，很多实际应用也不要求检索系统检索出离查询样本最近的数据库样本。因此，近似最近邻检索^[12-16] (Approximate Nearest Neighbor Search, 简称 ANNS) 近年来受到越来越多的关注。

较为流行的近似最近邻检索算法包括 K 维树^[13,14] (K -D Tree)、乘积量化^[15,17] (Product Quantization, 简称 PQ)、哈希^[2,16,18-22] (Hashing) 等。在这些方法中，哈希方法具有高效存储海量数据和加速查询过程的优点。因此，越来越多的学者对哈希方法进行了研究并提出了很多的方法。哈希方法的目标是通过使用哈希函数将数据从原始特征表示映射到尽量保相似性的二值哈希编码表示。这里，保相似性的含义为，如果两个数据点在原始空间相似，这两个数据点在二值空间中也要尽量相似，反之，如果两个数据点在原始空间中不相似，这两个数据点在二值空间中也要尽量不相似。通常，使用海明距离来定义二值哈希编码表示的相似性，这里海明距离定义为两个二值哈希编码对应位上不同编码的位数。一方面，基于二值哈希编码表示，数据能高效地储存在

^①<https://en.wikipedia.org/wiki/Instagram>

^②<https://en.wikipedia.org/wiki/YouTube>

计算机系统中。例如，假设使用 128 维的双精度浮点数尺度不变特征变换^[23] (Scale-Invariant Feature Transform, 简称 SIFT) 表示一张图片，10 亿张图片大约需要 1TB 的存储空间。然而，如果使用 32 维的二值哈希编码表示一张图片，保存同样数量的图片仅需要 4GB 的存储空间。另一方面，基于二值哈希编码表示，可以使用海明排序 (Hamming Ranking) 或者哈希表查询 (Hash Table Lookup) 加速检索过程。

根据是否使用数据指导哈希函数或二值哈希编码的学习，哈希可以分为数据独立哈希 (Data-Independent Hashing) 和数据依赖哈希 (Data-Dependent Hashing)。数据独立哈希方法的哈希函数通常是人工设计或者随机生成。代表性的数据独立哈希方法包括局部敏感哈希^[16,24,25] (Locality Sensitive Hashing, 简称 LSH)、核局部敏感哈希^[26] (Kernelized Locality Sensitive Hashing, 简称 KLSH) 和分数位哈希^[27] (Minwise Hashing)。数据依赖哈希又称为哈希学习^[1,28,29] (Hashing Learning) 方法。与数据独立哈希方法不同，哈希学习方法使用数据来指导哈希函数学习。哈希学习方法通常可以使用更短的二值哈希编码达到和数据独立哈希方法差不多的检索精度。

按照哈希学习方法的应用场景，哈希学习方法可以分为单模态哈希学习方法^[21,30-46] (Single Modal Hashing) 和多模态哈希学习方法^[47-59] (Multi-Modal Hashing)。单模态哈希学习方法应用在查询样本和数据库样本均只有一个模态的场景中，例如文本检索文本、图片检索图片等应用场景。早期的单模态哈希学习方法使用提取好的单模态数据的特征进行学习，无法进行深度特征学习。这类方法称为非深度单模态哈希学习方法^[21,34,44,60]。随着深度学习^[61] 技术的兴起，深度特征学习技术被引入到单模态哈希学习方法中。可以使用深度神经网络进行深度单模态特征学习的单模态哈希学习方法称为深度单模态哈希学习方法^[22,62-65]。多模态哈希学习方法应用在数据模态的数目大于等于两个时的场景中。多模态哈希可分为多源哈希学习方法^[66] (Multiple Source Hashing) 和跨模态哈希学习方法^[48,51,56,57,67] (Cross-Modal Hashing)。多源哈希学习方法要求所有样本对应的所有模态的数据都已观测到。跨模态哈希学习方法假设查询样本和数据库样本来自不同的数据模态。跨模态哈希学习方法可应用在以文搜图、以图搜文等场景中。与多源哈希学习方法相比，跨模态哈希学习方法不要求数据所有模态的信息都被观测到，因此跨模态哈希的应用场景更加灵活。早期的跨模态哈希学习方法使用提取好的多模态数据的特征进行学习，无法进行深度特征学习。这类方法称为非深度跨模态哈希学习方法^[47,48,51-53,56,57,68,69]。可以使

用深度神经网络进行深度多模态特征学习的跨模态哈希学习方法称为深度跨模态哈希学习方法^[56,59]。

由于二值哈希编码定义在二值空间中，哈希学习通常被建模成为一个离散优化问题。因此，哈希学习模型的优化通常是一个 NP 难的问题。为了避免由于二值哈希编码的离散性带来的问题，部分哈希学习方法采用了直接丢弃二值约束的松弛策略来设计优化算法。采用这种松弛策略本质上改变了原始哈希学习模型的目标，因此通常会导致检索精度受损。为了提高哈希学习的检索精度，研究者们提出了一些可以直接学习二值哈希编码的离散哈希学习方法^[21,22,38,44,60]。早期的离散哈希学习方法大部分应用在非深度单模态场景中。在非深度单模态哈希学习中，代表性的离散哈希学习方法包括迭代量化^[21] (Iterative Quantization, 简称 ITQ)、离散图哈希^[38] (Discrete Graph Hashing, 简称 DGH)、快速哈希^[60] (Fast Hashing, 简称 FastH) 和基于列采样的离散监督哈希^[44] (Column Sampling based Discrete Supervised Hashing, 简称 COSDISH) 等。随着深度哈希学习方法的兴起，研究者们提出了一些应用于深度单模态场景的离散哈希学习方法。代表性的方法包括基于标签对的深度监督哈希^[22] (Deep Pairwise Supervised Hashing, 简称 DPSH)、深度监督哈希^[70] (Deep Supervised Hashing, 简称 DSH) 等。在多模态哈希的场景中，基于典型相关分析的矢量迭代^[21] (Canonical Correlation Analysis based Iterative Quantization, 简称 CCAITQ)、语义相关最大化^[51] (Semantic Correlation Maximization, 简称 SCM) 和语义保持哈希^[53] (Semantic-Preserving Hashing, 简称 SePH) 等方法都是非深度跨模态离散哈希的代表方法。

1.2 有待研究的问题

虽然近年来研究者们提出了一些离散哈希学习算法，但离散哈希学习中仍存在问题尚未解决。这里从非深度单模态、深度单模态、非深度多模态和深度多模态四个场景对离散哈希学习存在的问题进行总结。

在非深度单模态哈希中，图哈希学习方法是一类重要的非深度单模态哈希学习方法。部分图哈希学习方法在训练过程中采用了直接丢弃二值约束的松弛策略。例如，SH^[30]、AGH^[32] 等方法都采用了这种策略。DGH^[38] 方法是代表性的离散图哈希学习方法。DGH 方法无法使用全部图相似度信息进行训练。DGH 的检索精度也因此受损。同时，DGH 方法需要使用锚图^[32,71] 进行建模，

但锚图的构造过程低效，导致 DGH 的训练过程低效。

在深度单模态哈希中，现有的深度单模态离散哈希学习方法存在两个方面的问题。一方面，现有的深度单模态离散哈希学习方法无法使用监督信息同时直接监督二值哈希编码学习和深度特征学习。具体来说，代表性的深度单模态离散哈希学习方法包括：DPSH^[22]、DSH^[70] 等方法。DPSH、DSH 等方法设计了离散优化算法来学习二值哈希编码。这些方法在学习过程引入一个实值变量，并使用深度神经网络学习数据的实值表示，同时尽量降低二值哈希编码和实值表示之间的误差。这意味着，这些方法无法使用监督信息来直接监督二值哈希编码的学习。另一方面，现有的深度单模态离散哈希学习方法都是对称哈希学习方法。当监督信息为标签对信息或者三元组信息时，基于对称哈希的深度哈希学习方法至少需要拟合训练集大小的平方的监督信息，因此这些方法的训练过程低效。

在非深度多模态哈希中，现有的基于标签对信息的非深度跨模态离散哈希学习方法存在复杂度高的问题。具体来说，代表性的非深度跨模态离散哈希学习方法包括：CCAITQ^[21]、交替协同量化^[69] (Alternating Co-Quantization, 简称 ACQ)、相关性量化哈希^[52] (Quantized Correlation Hashing, 简称 QCH)、SCM^[51] 和 SePH^[53] 等。其中，CCAITQ、ACQ 和 QCH 是无监督方法，与监督方法相比检索精度较差。SCM 需要使用类别标签来完成训练，无法应用在给定标签对信息的场景中。为了拟合标签对监督信息，SePH 的复杂度达到了平方级别。当计算资源有限时，SePH 只能在采样训练集上进行训练，其检索精度也因此受限。同时，尽管使用采样训练集，SePH 方法的训练过程仍旧低效。

在深度多模态哈希中，离散哈希学习的必要性还有待验证。具体来说，深度跨模态哈希场景中，离散哈希学习面临着与其他场景不同的问题。例如，深度跨模态离散哈希需要学习多个数据模态的二值哈希编码等。因此，在深度跨模态场景中是否需要离散哈希学习需要进一步验证。同时，深度跨模态哈希中，使用何种策略监督二值哈希编码学习和深度多模态特征学习也需要进一步探讨。

1.3 本文工作

本文从非深度单模态、深度单模态、非深度多模态和深度多模态四个场景对离散哈希学习进行系统地研究，并提出针对不同场景中不同问题的解决方

案。这里根据应用场景的不同将本文的工作划分为四个部分，分别对应论文的第三章到第六章。

第一部分针对非深度单模态哈希的应用场景，提出一种基于特征变换的可扩展图哈希学习方法（Scalable Graph Hashing with Feature Transformation, 简称 SGH）。SGH 方法通过使用特征变换隐式地计算全部图相似度信息，可以避免平方级别的计算和储存复杂度。此外，SGH 方法还设计了一种逐比特离散优化算法来学习二值哈希编码。与现有的离散图哈希学习方法相比，SGH 能达到更高的检索精度。同时，SGH 的训练也更高效。本部分的主要研究成果已发表在 CCF-A 类国际会议 IJCAI 2015。

第二部分针对深度单模态哈希的应用场景，首先提出一种深度离散监督哈希学习方法（Deep Discrete Supervised Hashing, 简称 DDSH）。DDSH 方法可以使用监督信息同时直接监督二值哈希编码和深度特征的学习。本部分还提出一种非对称深度监督哈希学习方法（Asymmetric Deep Supervised Hashing, 简称 ADSH）。ADSH 方法使用非对称哈希建模深度哈希学习模型，将针对查询样本的深度特征学习过程和针对数据库样本的二值哈希编码学习过程区别对待。与现有的深度单模态哈希学习方法相比，DDSH 方法能达到更高的检索精度。与除 DDSH 方法外的对称深度单模态哈希学习方法相比，ADSH 方法能在最短时间内达到更高的检索精度。与 DDSH 方法相比，ADSH 方法的训练更加高效。本部分的主要研究成果已分别发表在 CCF-A 类国际期刊 TIP 2018 和 CCF-A 类国际会议 AAAI 2018。

第三部分针对非深度多模态哈希的应用场景，提出一种基于离散隐因子模型的跨模态哈希学习方法（Discrete Latent Factor Modal based Cross-Modal Hashing, 简称 DLFH）。DLFH 方法使用离散隐因子模型对跨模态哈希学习方法建模，并设计了一种高效的二值哈希编码学习算法。同时本部分给出了二值哈希编码学习算法的收敛性证明。与现有的非深度跨模态哈希学习方法相比，DLFH 方法能达到更高的检索精度，同时其训练也更高效。本部分的主要研究成果已发表在 CCF-A 类国际期刊 TIP 2019。

第四部分针对深度多模态哈希的应用场景，首次将深度多模态特征学习引入到跨模态哈希学习中，提出一种深度跨模态哈希学习方法（Deep Cross-Modal Hashing, 简称 DCMH）。DCMH 方法使用两个深度神经网络对数据进行深度特征学习。同时，DCMH 方法可以直接学习数据的二值哈希编码表示。本部分还提出一种基于深度离散隐因子模型的跨模态哈希学习方法（Deep

Discrete Latent Factor Modal for Cross-Modal Hashing, 简称 DDLFH)。DDL FH 方法将 DL FH 方法的二值哈希编码学习能力和深度学习的深度特征学习能力整合到一个统一的学习框架, 能够实现两种能力的相互反馈。本部分的主要研究成果已发表在 CCF-A 类国际会议 CVPR 2017 及投稿到计算机学报^①。

1.4 论文组织

本文一共分为七个章节, 论文的主要方法包含在第三章到第六章, 它们之间的逻辑结构如图 1-1 所示。

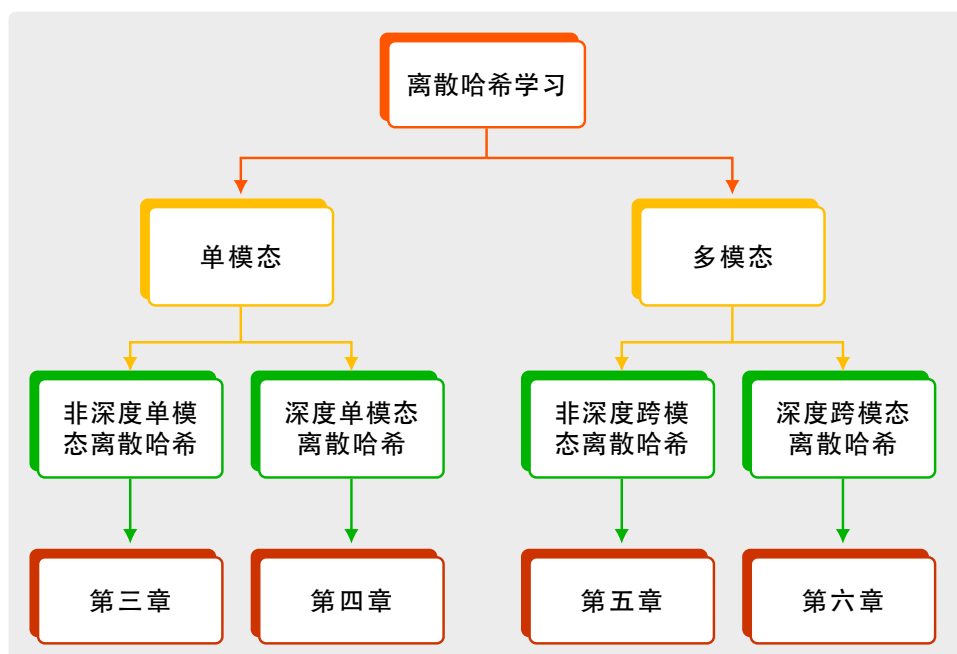


图 1-1: 论文组织

第一章为绪论部分, 主要对哈希学习进行简介, 并介绍离散哈希学习的发展现状和存在的问题。针对这些问题, 本章概括性地介绍本文的工作。

第二章介绍本文的研究背景, 具体来说, 第二章首先介绍哈希学习, 然后介绍相关评价标准和常用数据集。

第三章介绍已有非深度单模态哈希中图哈希学习方法面临的问题和本文提出的基于特征变换的可扩展图哈希学习方法 SGH。

第四章介绍已有深度单模态哈希中离散哈希学习面临的问题, 并介绍本文提出的深度离散监督哈希学习方法 DDSH 和非对称深度监督哈希学习方法

^①<http://cjc.ict.ac.cn/>

ADSH。

第五章介绍已有非深度多模态哈希中离散哈希学习面临的问题。同时第五章介绍本文提出的基于离散隐因子模型的跨模态哈希学习方法 **DLFH**。

第六章介绍已有深度多模态哈希中离散哈希学习面临的问题。同时第六章介绍本文提出的深度跨模态哈希学习方法 **DCMH** 和基于深度离散隐因子模型的跨模态哈希学习方法 **DDL FH**。

最后，第七章回顾并总结本文的研究内容和贡献，并展望离散哈希学习的未来研究方向。

此外，附录 A 介绍本文使用的一些通用符号的定义和通用简称。

第二章 研究背景

本章主要对哈希学习相关的理论和技术做概括性地阐述和总结。其中，第2.1节介绍哈希学习，包括哈希学习的研究进展及其在大规模检索中的应用。第2.2节介绍用于评价哈希学习方法的相关评价标准和在哈希学习方法中常用的数据集。

2.1 哈希学习简介

哈希学习旨在学习哈希函数将数据从原始特征表示映射到尽量保相似性的二值哈希编码表示。这里，保相似性是指当两个数据样本在原空间中相似，通过哈希函数将数据表示为二值哈希编码后，希望这两个数据样本在二值空间中也相似；反之，如果两个数据样本在原空间中不相似，通过哈希学习将数据表示为二值哈希编码后，希望这两个数据样本在二值空间中也不相似。图2-1给出了哈希学习的示意图。图2-1使用了三张图片作为例子，其中，第一张图片和后两张图片不相似，后两张图片相似。通过哈希函数 $h(\cdot)$ 的作用，后两张图片将被表示为相似的二值哈希编码，即后两张图片的二值哈希编码间的海明距离较小；而第一张图片和第二张（或者第三张）图片不相似，因此第一张图片和第二张（或者第三张）图片被映射为不相似的二值哈希编码，即第一张图片和第二张（或者第三张）图片的二值哈希编码之间的海明距离较大。一方面，通过二值哈希编码表示，存储空间将会减小。另一方面，二值哈希编码间的海明距离计算通常可以使用位操作完成，而通过将二值哈希编码组织成倒排表等结构，检索过程的时间开销将会减小。因此，哈希学习可达到降低存储开销、加速计算和检索的目的。

当实际应用中存在存储开销过大或者计算检索速度慢的问题时，可以使用哈希学习解决这些问题。例如，在大规模数据检索^[1,3,24]、大规模推荐系统^[49]、神经网络压缩^[72,73]等应用场景中都可以使用哈希学习来进行压缩存储或加速计算和检索过程。本文主要关注哈希学习在大规模检索场景中的应用。本节剩余部分主要介绍哈希学习的研究进展及其在大规模检索中的应用。

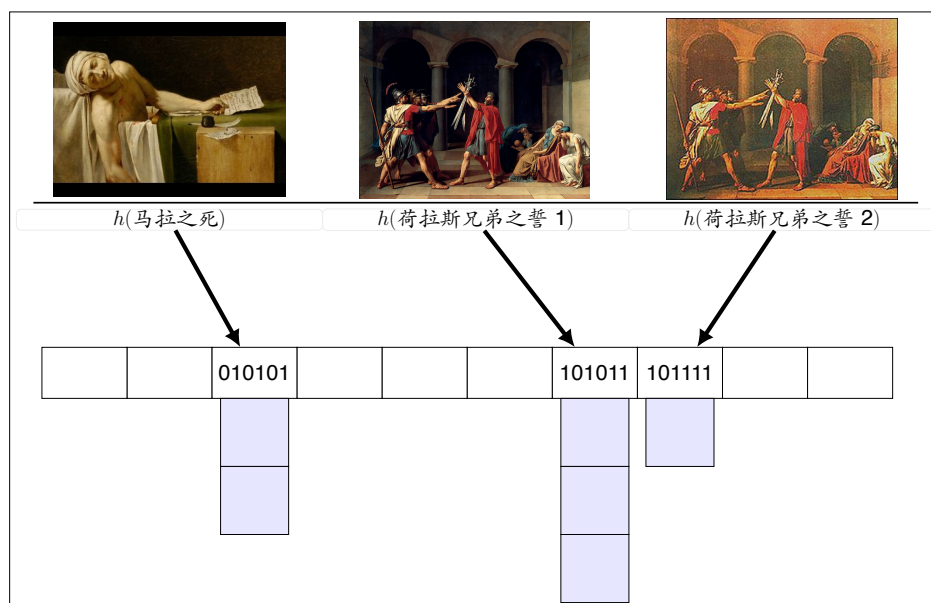


图 2-1: 哈希方法示意图

2.1.1 哈希学习研究进展

哈希学习方法使用数据来指导哈希函数的学习。Salakhutdinov 和 Hinton 在 2007 年^[74]首次提出语义哈希学习方法 (Semantic Hashing)，将哈希方法推广到机器学习领域。随后，研究者们提出了很多哈希学习方法。本节剩余部分首先根据不同的应用场景分类介绍哈希学习，然后介绍离散哈希学习的研究进展。

2.1.1.1 哈希学习分类

按照哈希学习的应用场景不同，哈希学习可以划分为单模态哈希学习方法^[21,34,38,44,60,75]和多模态哈希学习方法^[47,48,51-54,69]。单模态哈希学习方法应用在检索样本与数据库样本的数据来自同一个模态的场景中。单模态哈希学习方法可以用在图片检索图片^[21,62]、文本搜索文本^[31,76-78]、音频检索^[79]、视频检索^[80]等应用场景中。多模态哈希学习方法应用在数据模态数目大于等于两个的场景中。这类方法可应用在图片搜索文本、文本搜索图片^[47,51,56]等应用场景中。

早期的单模态哈希学习方法使用提取好的单模态数据的特征进行学习，无法进行深度特征学习。这类方法称为非深度单模态哈希学习方法。在非深度单模态哈希学习方法中，部分方法不使用人工标记的标签，仅使用训练样本

本身进行训练。代表性的方法包括：谱哈希^[30] (Spectral Hashing, 简称 SH)、锚图哈希^[32] (Anchor Graph Hashing, 简称 AGH)、DGH^[38]、局部线性哈希^[81] (Locality Linear Hashing, 简称 LLH)、主成分分析哈希^[21] (Principal Component Analysis Hashing, 简称 PCAH)、ITQ、等方差哈希^[34] (Isotropic Hashing, 简称 IsoHash)、二值自动编码器^[82] (Binary Autoencoder, 简称 BA)、K 聚类哈希^[37] (K-Means Hashing, 简称 KH) 等。SH、AGH、DGH 和 LLH 属于图哈希学习方法。其中, SH、AGH 和 DGH 使用拉普拉斯降维方法将数据投影到低维实值空间, 再使用不同的策略来学习二值哈希编码。LLH 使用局部线性嵌入来学习数据的低维实值表示, 同时通过降低二值哈希编码和实值表示之间的量化误差来学习二值哈希编码。PCAH、ITQ 和 IsoHash 方法首先使用主成分分析方法将数据样本从原始特征空间投影到低维实值空间中。然后, PCAH 直接将数据样本的实值表示量化为二值哈希编码表示, ITQ 通过降低二值哈希编码和实值表示之间的量化误差来学习二值哈希编码, IsoHash 则通过学习一个正交矩阵来保证各维实值表示的方差尽量相同。BA 方法使用自动编码器模型来学习数据的二值表示。KH 采用 K-Means 算法来建模哈希学习问题。

另外一些非深度单模态哈希学习方法通过使用人工标记的语义标签来指导哈希学习。代表性的方法包括: 序列投影学习哈希^[83] (Sequential Projection Learning Hashing, 简称 SPLH)、自学习哈希^[31] (Self-Taught Hashing, 简称 STH)、两步走哈希^[84] (Two-Step Hashing, 简称 TSH)、离散监督哈希^[85] (Supervised Discrete Hashing, 简称 SDH)、基于核的监督哈希^[86] (Kernelized Supervised Hashing, 简称 KSH)、隐因子模型哈希^[75] (Latent Factor Hashing, 简称 LFH)、FastH^[60]、COSDISH^[44]、海明距离度量学习^[87] (Hamming Distance Metric Learning, 简称 HDML)、列生成哈希^[88] (Column Generation Hashing, 简称 CGH)、基于排序的监督哈希^[89] (Ranking based Supervised Hashing, 简称 RSH) 和保序哈希^[90] (Ranking Preserving Hashing, 简称 RPH) 等方法。SDH 方法使用类别标签指导二值哈希编码的学习。SPLH、STH、TSH、KSH、LFH、FastH 和 COSDISH 使用标签对监督信息来指导二值哈希编码的学习。CGH、HDML 使用三元组标签信息来指导二值哈希编码的学习。RSH、RPH 方法使用序列标签来指导哈希模型学习。

随着深度学习^[89] 技术的兴起, 深度特征学习技术被引入到单模态哈希中。这类方法称为深度单模态哈希学习方法。Xia 等人提出卷积神经网络哈希^[62] (Convolutional Neural Network Hashing, 简称 CNNH), 首次使用深

度神经网络来完成深度特征学习。CNNH 采用类别标签来指导哈希模型学习。DPSH^[22]、深度哈希网络^[65] (Deep Hashing Network, 简称 DHN)、DSH^[70] 等方法采用标签对信息来指导哈希模型的学习。内建网络哈希^[63] (Network in Network Hashing, 简称 NINH)、深度正则相似度比较哈希^[91] (Deep Regularized Similarity Comparison Hashing, 简称 DRSCH)、深度语义排序哈希^[92] (Deep Semantic Ranking Hashing, 简称 DSRH) 和深度三元组监督哈希^[93] (Deep Triplet Supervised Hashing, 简称 DTSH) 使用三元组监督信息来指导哈希模型学习。这些方法都是使用监督信息进行学习的代表性方法。深度哈希^[64] (Deep Hashing, 简称 DH)、深度比特哈希^[94] (Deep Bit, 简称 DeepBit)、二值生成对抗神经网络^[95] (Binary Generative Adversarial Network, 简称 binGAN) 和生成对抗神经网络哈希^[96] (Hashing for Generative Adversarial Network, 简称 hashGAN) 等方法则不使用监督信息指导哈希模型学习。

多模态哈希学习方法又可以分为多源哈希^[66,97-99] 和跨模态哈希^[48-51,56,57]。多源哈希使用多个模态的数据来学习哈希函数和二值哈希编码。具体来说, 多源哈希要求查询样本与数据库样本的数据均具有所有模态的信息。跨模态哈希假设查询样本与数据库样本来自不同的数据模态。由于跨模态哈希对数据模态的要求没有多源哈希对数据模态的要求高, 因此跨模态哈希通常具有更广泛的应用场景。代表性的跨模态哈希学习方法包括: 二值哈希编码重构嵌入^[100] (Binary Reconstruction Embedding, 简称 BRE)、多模态隐二值嵌入^[68] (Multimodal Latent Binary Embedding, 简称 MLBE)、协同量化哈希^[48] (Co-Regularized Hashing, 简称 CRH)、CCAITQ^[21]、协同矩阵哈希^[50] (Collective Matrix Factorization Hashing, 简称 CMFH)、SCM^[51]、QCH、ACQ、SePH^[53]、语义主题多模态哈希^[101] (Semantic Topic Multimodal Hashing, 简称 STMH)、监督矩阵分解哈希^[102] (Supervised Matrix Factorization Hashing, 简称 SMFH) 和广义语义保持哈希^[54] (Generalized Semantic Preserving Hashing, 简称 GSPH)。其中, BRE、CCAITQ、CRH、CMFH、QCH 和 ACQ 不使用跨模态相似度进行训练, 属于无监督哈希学习方法。MLBE、SCM、SePH、STMH、SMFH 和 GSPH 使用跨模态相似度进行训练, 属于监督哈希学习方法。

2.1.1.2 离散哈希学习研究进展

哈希学习问题的本质是一个离散优化问题。为了避免二值约束带来的问题, 很多哈希学习方法采用丢弃二值哈希编码的松弛策略。例如, SH^[30] 和

AGH^[32] 等方法都在训练过程中直接丢弃二值约束。这类方法在模型学习阶段直接丢弃二值约束，首先学习数据的实值表示。训练过程结束后，使用取符号函数将实值表示量化为二值哈希编码表示。采用这种松弛策略本质上改变了哈希学习模型的原始目标。松弛后的模型的解很可能已经偏离满足原始模型的最优解。这类方法的检索精度因此也会受损。

为了进一步提高哈希学习模型的检索精度，研究者们提出了一些离散哈希学习方法^[21,38,44,60,85]。早期的离散哈希学习大部分应用在非深度单模态场景中。在非深度单模态哈希学习场景中，代表性的离散哈希学习方法包括 ITQ^[21]、DGH^[38]、FastH^[60]、SDH^[85] 和 COSDISH^[44] 等方法。ITQ^[21] 方法首先使用 PCA 将数据降维到低维空间，然后设计了一种迭代量化算法来学习二值哈希编码。具体来说，迭代量化算法迭代学习一个单位正交投影矩阵和二值哈希编码。DGH^[38] 方法使用锚图哈希进行建模。在 DGH 模型中同时存在二值哈希编码平衡约束和比特独立约束^[30]。DGH 引入一个实值变量，将这两个约束施加到实值变量上，并同时优化该实值变量和二值变量的量化损失。然后，DGH 设计了一种具有收敛保证的取符号梯度上升方法^[38]（Signed Gradient Ascent Method）来学习二值哈希编码。ITQ 和 DGH 都是无监督哈希学习方法。ITQ 是非图哈希学习方法，DGH 是图哈希学习方法。DGH 证明了图哈希学习方法能达到比非图哈希学习方法更好的检索精度。FastH^[60] 采用图割算法进行二值哈希编码学习。SDH^[85] 设计了一种离散循环坐标下降方法（Discrete Cyclic Coordinate Descent）来学习二值哈希编码。COSDISH^[44] 采用分而治之的思想，将二值哈希编码学习分为两个子问题，并使用离散二次规划^[103]（Binary Quadratic Programming, 简称 BQP）来完成较复杂部分的二值哈希编码学习，使用 1 范数近似的方法来完成较简单部分的二值哈希编码学习。FastH、SDH 和 COSDISH 都是监督哈希学习方法。

随着深度哈希学习算法的兴起，研究者们提出了一些应用在深度单模态场景中的离散哈希学习方法。代表性的方法包括 DPSH^[22]、DSH^[70] 等方法。DPSH^[22] 是第一个端到端的深度单模态离散哈希学习方法。DPSH 方法使用深度神经网络将数据投影到低维实值空间，并使用标签对监督信息来指导实值特征的学习。为了学习二值哈希编码，DPSH 引入一个二值变量，并优化二值变量和实值编码之间的量化损失。DPSH 证明了采用深度特征学习策略和进行二值哈希编码学习能取得更好的检索精度。DSH^[70] 方法使用对比损失^[104]（Contrastive Loss）来学习实值的低维表示。同时 DSH 设计了一个实值输出和

二值哈希编码之间的 1 范数损失项来使得实值表示尽量地近似二值哈希编码。并优化由对比损失和 1 范数损失项构成的目标函数来完成深度特征学习和二值哈希编码学习。

在多模态场景中, CCAITQ^[21]、ACQ^[69]、QCH^[52]、SCM^[51] 和 SePH^[53] 等方法都是代表性的非深度跨模态离散哈希学习方法。CCAITQ^[21] 首先使用 CCA 将多模态数据投影到子空间中, 然后再使用迭代量化算法学习单位正交投影矩阵与二值哈希编码表示。ACQ^[69] 和 QCH^[52] 则将降维和二值哈希编码学习过程融合到同一过程中。CCAITQ、ACQ、QCH 都是无监督方法。SCM^[51] 利用类别标签来指导模型学习, 并设计了一种逐比特优化的二值哈希编码学习算法。SePH^[53] 在同时学习实值表示的同时, 通过优化实值表示和 ± 1 之间的量化损失来学习二值哈希编码。SCM 和 SePH 使用监督信息来指导跨模态哈希学习, 属于监督哈希学习方法。

2.1.2 在大规模检索中的应用

在大规模检索的实际应用场景中, 给定数据库样本集合和查询样本集合, 基于哈希学习的大规模检索通常包含三个过程, 即: 二值哈希编码生成、初排过程、重排过程。二值哈希编码的生成过程主要使用哈希函数将数据表示为二值哈希编码。对于哈希学习方法, 哈希函数需使用哈希学习方法学习得到。哈希学习方法可以根据不同的应用场景来选择。初排过程最常用的两个策略是海明排序和哈希表查询。重排过程通常使用基于实值特征的排序算法。本节主要介绍初排策略和重排策略。

2.1.2.1 初排策略

通过哈希学习方法, 可以得到可将数据表示为二值哈希编码的哈希函数。查询样本和数据库样本被表示为二值哈希编码后, 使用初排过程从数据库中选取前 K 个与查询样本最近的样本集合。初排过程最常用的两个策略是海明排序和哈希表查询。海明排序通过使用基于二值哈希编码的排序方法对整个数据库的样本进行排序, 从而返回所需的前 K 个样本。具体来说, 假设查询样本 $\mathbf{x}^{(q)}$ 的二值哈希编码为 $\mathbf{b}^{(q)}$, 并且, 给定包含 n 个数据样本的数据库二值哈希编码 $\mathbf{B}^{(d)} = \{\mathbf{b}_i^{(d)}\}_{i=1}^n$ 。海明排序首先计算查询样本的二值哈希编码 $\mathbf{b}^{(q)}$ 与数据库样本的二值哈希编码之间的海明距离, 然后再根据海明距离进行排序。最后根据排序的结果返回前 K 个与查询样本最近的样本集合。该过程总结在算

法 2.1 中。

从算法 2.1 中可以看到，海明查询最重要的两个过程包含计算查询样本的二值哈希编码与数据库样本的二值哈希编码之间的海明距离和排序两个过程。首先，由于数据被表示成为二值哈希编码，海明距离的计算可以使用位操作来实现加速。其次，假设比特长度为 c ，则海明距离的值属于集合 $\{0, 1, \dots, c\}$ 。因此仅需使用 $c + 1$ 个集合来标记各个海明距离就可以完成排序过程。可得基于海明距离的排序复杂度为 $\mathcal{O}(n)$ 。这意味着通过海明排序可以达到 $\mathcal{O}(n)$ 的查询速度。

算法 2.1 海明排序算法

输入：

编码长度 c ，查询样本二值哈希编码 $\mathbf{b}^{(q)}$ ，数据库样本二值哈希编码 $\mathbf{B}^{(d)}$ 。

输出：

包含 K 个备选样本的集合 \mathbf{C} 。

- 1: 初始化备选样本集合 $\mathbf{C} = \emptyset$;
 - 2: 计算 $\mathbf{b}^{(q)}$ 与数据库编码的 $\mathbf{B}^{(d)} = \{\mathbf{b}_i^{(d)}\}_{i=1}^n$ 之间的海明距离;
 - 3: 根据海明距离对数据编码进行升序排序;
 - 4: 选择前 K 个样本加入备选集合 \mathbf{C} 。
-

哈希表查询是另一种重要的基于哈希的初排方法。基于数据的二值哈希编码表示，数据可以被组织成倒排索引^[6,105]结构。当有查询样本到来时，仅需使用查询样本的二值哈希编码表示进行哈希桶查询，直到找到前 K 个满足条件的数据库样本即可。具体来说，假设给定 n 个数据库样本的二值哈希编码 $\mathbf{B}^{(d)} = \{\mathbf{b}_i^{(d)}\}_{i=1}^n$ ，根据 $\mathbf{B}^{(d)}$ 构造倒排索引表。给定查询样本 $\mathbf{x}^{(q)}$ 的二值哈希编码 $\mathbf{b}^{(q)}$ ，哈希表查询使用 $\mathbf{b}^{(q)}$ 作为索引检索倒排表。首先返回搜索半径 $R = 0$ 的海明桶 $\mathbf{b}^{(q)}$ 中的样本，如果以 $\mathbf{b}^{(q)}$ 为索引的哈希桶中没有足够的备选样本，则依次扩大搜索半径继续查找，直至找到足够的样本。该过程总结在算法 2.2 中。

理想情况下，查询所需的样本可以在以 $\mathbf{b}^{(q)}$ 为索引的哈希桶中找到，此时查询的复杂度为 $\mathcal{O}(1)$ 。当以 $\mathbf{b}^{(q)}$ 为索引的哈希桶无法查询到足够数量的样本时，通常需要扩大搜索半径，此时查询复杂度将以几何的速度增长。最差情况，哈希表查询复杂度将达到 $\mathcal{O}(2^c)$ 。为了缓解这一问题，一种可行的方案是使用二值哈希编码构造多桶哈希表，同时在多个哈希表里进行查询。

算法 2.2 哈希表查询算法**输入:**编码长度 c , 查询样本二值哈希编码 $\mathbf{b}^{(q)}$, 数据库样本二值哈希编码 $\mathbf{B}^{(d)}$ 。**输出:**包含 K 个备选样本的集合 \mathbf{C} 。

- 1: 将二值哈希编码组织成倒排表 T ; 初始化备选样本集合 $\mathbf{C} = \emptyset$;
- 2: **for** $r = 0 \rightarrow c$ **do**
- 3: **if** $|\mathbf{C}| > K$ **then**
- 4: 满足条件, 返回。
- 5: **end if**
- 6: **if** $r = 0$ **then**
- 7: 查询以 $\mathbf{b}^{(q)}$ 为索引的哈希桶;
- 8: 将哈希桶的样本加入 \mathbf{C} ;
- 9: **else**
- 10: 翻转查询样本编码 $\mathbf{b}^{(q)}$ 的 r 个比特, 依次查询以翻转后的编码为索引的哈希桶;
- 11: 将对应哈希桶中的样本加入 \mathbf{C} ;
- 12: **end if**
- 13: **end for**

2.1.2.2 重排策略

通过初排策略可以得到与查询样本最相似的前 K 个样本的集合 \mathbf{C} 。然后使用重排策略从集合 \mathbf{C} 中选取与查询样本最近的前 k 个样本, 其中, $k < K$ 。最常用的重排策略使用查询样本的实值特征和备选样本的实值特征之间的欧氏距离进行排序以得到前 k 个样本。该过程总结在算法 2.3 中。重排过程中使用的实值特征通常为手工提取的特征或者深度神经网络提取的深度特征。在实际系统中, 由于参数 k 和 K 的值小于原始数据库大小 n , 因此重排过程的计算量通常也在可接受的范围之内。

算法 2.3 重排算法**输入:**查询样本的实值特征 $\mathbf{x}^{(q)}$, 备选样本的集合 \mathbf{C} 的实值特征。**输出:**包含 k 个备选样本的集合 \mathbf{D} 。

- 1: 初始化备选样本集合 $\mathbf{D} = \emptyset$;
- 2: 计算查询样本 $\mathbf{x}^{(q)}$ 与 \mathbf{C} 中备选样本之间的欧氏距离;
- 3: 根据欧氏距离对数据编码进行升序排序;
- 4: 选择前 k 个样本加入备选集合 \mathbf{D} 。

2.2 相关评价标准和常用数据集

本节介绍评价哈希学习方法的常用相关评价标准和数据集。

2.2.1 相关评价标准

对于哈希学习方法，通常从检索精度和检索速度两个方面进行评价。本节剩余部分主要从检索精度相关标准和检索速度相关标准两个方面进行介绍。

2.2.1.1 检索精度相关标准

为了评价哈希学习方法，需要首先将查询样本和数据库样本表示为二值哈希编码，并使用二值哈希编码来模拟检索过程。然后，使用评价检索过程的评价标准来评价哈希学习方法。最常用的评价标准包括查准率（Precision）、查全率（Recall）和平均查准率均值（Mean Average Precision）。

给定查询样本的二值哈希编码 $\mathbf{b}_j^{(q)}$ 和数据库二值哈希编码 $\{\mathbf{b}_i^{(d)}\}_{i=1}^n$ ，假设返回序列的索引集合为： $\{\pi_1, \dots, \pi_n\}$ ，返回序列可以表示为： $\{\mathbf{b}_{\pi_1}^{(d)}, \dots, \mathbf{b}_{\pi_n}^{(d)}\}$ 。在返回序列的截断位置 K 的查准率（Precision at Cutoff K ）为：

$$\text{cPre}(K, \mathbf{b}_j^{(q)}) = \frac{N_{j,K}^+}{K},$$

其中， $N_{j,K}^+$ 表示在返回的序列中，前 K 个样本中与查询样本 $\mathbf{b}_j^{(q)}$ 是相似样本的数目。对于查询样本集 $\mathbf{B}^{(q)} = \{\mathbf{b}_j^{(q)}\}_{j=1}^m$ ，在返回序列的截断位置 K 的平均查准率可按如下的公式计算：

$$\text{cPre}(K, \mathbf{B}^{(q)}) = \frac{1}{m} \sum_{j=1}^m \text{cPre}(K, \mathbf{b}_j^{(q)}).$$

无特殊说明的情况下， $\text{cPre}(K, \mathbf{B}^{(q)})$ 简记为 $\text{cPre}(K)$ 。根据 $\text{cPre}(K)$ 的定义可知， $\text{cPre}(K) \in [0, 1]$ 。学习得到的二值哈希编码表示越好，查询样本的二值哈希编码与其在数据库中相似样本二值哈希编码的海明距离越小，在返回序列中与查询样本相似的样本越靠前， $\text{cPre}(K)$ 值越高。

由于二值哈希编码和海明距定义在二值空间，哈希学习方法的精度还可以根据海明球（Hamming Ball）来评价。具体来说，给定查询样本的二值哈希编码 $\mathbf{b}_j^{(q)}$ 和数据库二值哈希编码 $\{\mathbf{b}_i^{(d)}\}_{i=1}^n$ 。根据 $\mathbf{b}_j^{(q)}$ 和 $\mathbf{B}^{(d)} = \{\mathbf{b}_i^{(d)}\}_{i=1}^n$ 之间的海

明距离，可以将数据库集合划分为 $c + 1$ 个集合，其中 c 表示二值哈希编码长度。在半径为 R 的海明球处的查准率（Precision at R -Hamming Ball）使用如下公式进行计算：

$$\text{rPre}(R, \mathbf{b}_j^{(q)}) = \frac{M_{j,R}^+}{M_{j,R}},$$

其中， $M_{j,R}^+$ 表示在海明半径小于等于 R 的所有海明球中，与查询 $\mathbf{b}_j^{(q)}$ 是相似样本的数目， $M_{j,R}$ 表示在海明半径小于等于 R 的所有海明球中所有样本的数目。对于查询样本集 $\mathbf{B}^{(q)} = \{\mathbf{b}_j^{(q)}\}_{j=1}^m$ ，在半径为 R 的海明球处的平均查准率可按如下的公式计算：

$$\text{rPre}(R, \mathbf{B}^{(q)}) = \frac{1}{m} \sum_{j=1}^m \text{rPre}(R, \mathbf{b}_j^{(q)}).$$

无特殊说明的情况下， $\text{rPre}(R, \mathbf{B}^{(q)})$ 简记为 $\text{rPre}(R)$ 。根据 $\text{rPre}(R)$ 的定义可知， $\text{rPre}(R) \in [0, 1]$ 。学习得到的二值哈希编码表示越好，查询样本的二值哈希编码与其在数据库中相似样本二值哈希编码的海明距离越小，相似样本落在海明半径小的海明桶中的可能性越高， $\text{rPre}(R)$ 值越高。

类似地，给定查询样本的二值哈希编码 $\mathbf{b}_j^{(q)}$ ，在返回序列的截断位置 K 的查全率（Recall at Cutoff K ）为：

$$\text{cRec}(K, \mathbf{b}_j^{(q)}) = \frac{N_{j,K}^+}{N_j^+},$$

其中， N_j^+ 表示数据库中与查询样本 $\mathbf{b}_j^{(q)}$ 相似的样本的数目。对于查询样本集 $\mathbf{B}^{(q)} = \{\mathbf{b}_j^{(q)}\}_{j=1}^m$ ，在返回序列的截断位置 K 的平均查全率可按如下的公式计算：

$$\text{cRec}(K, \mathbf{B}^{(q)}) = \frac{1}{m} \sum_{j=1}^m \text{cRec}(K, \mathbf{b}_j^{(q)}).$$

无特殊说明的情况下， $\text{cRec}(K, \mathbf{B}^{(q)})$ 简记为 $\text{cRec}(K)$ 。根据 $\text{cRec}(K)$ 的定义可知， $\text{cRec}(K) \in [0, 1]$ 。学习得到的二值哈希编码表示越好，查询样本的二值哈希编码与其在数据库中相似样本二值哈希编码的海明距离越小，在返回序列中与查询样本相似的样本越靠前， $\text{cRec}(K)$ 值越高。

对于查询样本集 $\mathbf{b}_j^{(q)}$ ，在半径为 R 的海明球处的查全率（Recall at R -

Hamming Ball) 为:

$$\text{rRec}(R, \mathbf{b}_j^{(q)}) = \frac{M_{j,R}^+}{N_j^+}.$$

对于查询样本集 $\mathbf{B}^{(q)} = \{\mathbf{b}_j^{(q)}\}_{j=1}^m$, 在半径为 R 的海明球处的平均查全率可按如下的公式计算:

$$\text{rRec}(R, \mathbf{B}^{(q)}) = \frac{1}{m} \sum_{j=1}^m \text{rRec}(R, \mathbf{b}_j^{(q)}).$$

无特殊说明的情况下, $\text{rRec}(R, \mathbf{B}^{(q)})$ 简记为 $\text{rRec}(R)$ 。根据 $\text{rRec}(R)$ 的定义可知, $\text{rRec}(R) \in [0, 1]$ 。学习得到的二值哈希编码表示越好, 查询样本的二值哈希编码与其在数据库中相似样本二值哈希编码的海明距离越小, 相似样本落在海明半径小的海明桶中的可能性越高, $\text{rRec}(R)$ 值越高。

除了查准率与查全率, 常用的指标还有在返回序列截断位置 K 的平均查准率均值 (Mean Average Precision at Cutoff K)。给定查询 $\mathbf{b}_j^{(q)}$, 在序列返回位置 K 的查准率均值可以使用如下公式计算:

$$\text{cAP}(K, \mathbf{b}_j^{(q)}) = \frac{1}{\text{rel}(K, \mathbf{b}_j^{(q)})} \sum_{k=1}^K \text{cPre}(k, \mathbf{b}_j^{(q)}) \cdot \mathbf{1}(\mathbf{b}_j^{(q)} \text{ 与 } \mathbf{b}_l^d \text{ 是相似样本}),$$

其中, $\text{rel}(K, \mathbf{b}_j^{(q)})$ 表示在返回序列的前 K 个位置中, 与查询样本 $\mathbf{b}_j^{(q)}$ 相似的样本数目, $\mathbf{1}(\cdot)$ 表示指示函数, 其定义为:

$$\mathbf{1}(\text{condition}) = \begin{cases} 1 & \text{if condition is true,} \\ 0 & \text{otherwise.} \end{cases}$$

对于查询样本集 $\mathbf{B}^{(q)} = \{\mathbf{b}_j^{(q)}\}_{j=1}^m$, 在返回序列截断位置 K 的平均查准率均值可以使用如下的公式计算:

$$\text{cMAP}(K, \mathbf{B}^{(q)}) = \frac{1}{m} \sum_{j=1}^m \text{cAP}(K, \mathbf{b}_j^{(q)}).$$

无特殊说明的情况下, $\text{cMAP}(K, \mathbf{B}^{(q)})$ 简记为 $\text{cMAP}(K)$ 。根据 $\text{cMAP}(K)$ 的定义可知, $\text{cMAP}(K) \in [0, 1]$ 。学习得到的二值哈希编码越好, 查询样本的二值哈希编码与其在数据库中相似样本二值哈希编码的海明距离越小, 在返回序

列中相似样本越靠前， $cMAP(K)$ 值越高。

2.2.1.2 检索速度相关标准

最直观的评价检索速度的标准为实际检索时间，为了得到实际检索时间，需要使用二值哈希编码构造基于哈希学习检索的系统。基于构造的系统，可通过给定的查询样本集执行检索过程得到实际检索时间。

除了检索时间之外，常用的用于评价检索速度的标准还包括哈希表查询成功率^[38,106]等能间接反映查询速度的评价标准。给定查询样本的二值哈希编码 $b^{(q)}$ 和查询半径 R ，哈希表查询成功表示在海明半径小于等于 R 的海明桶中，返回的样本数目大于零。当返回的样本数目为零时，表示查询失败。据此可以计算哈希表查询成功率。

2.2.2 数据集

研究者们公开了很多可用于哈希学习研究的数据集，包括图片数据集^[107,108]、文本数据集^①、图片-文本数据集^[109,110]、音频数据集^[79,111,112]、视频数据集^[80]等。本文主要介绍用于图片检索和图片-文本检索的常用数据集。

哈希学习中常用的图片数据集有：TINY1M^②、FLICKR1M^[113]、MNIST^[107]、CIFAR10^[108]、MSCOCO^[114]等。常用的图片-文本数据集有：FLICKR25K^[110]、WIKI^[115]、IAPRTC12^[116]、NUSWIDE^[38]等。本节剩余部分分别简单介绍这两类数据集。

2.2.2.1 图片数据集

TINY1M 数据集包含一百万张图片，这些图片是从八千万张图片的集合中采样得到的。每张图片的原始尺寸为 32×32 ，被表示成 384 维度的 GIST 特征。

FLICKR1M^[113] 数据集包含了一百万张从 Flickr 网站^③下载的图片。每张图片被表示成 512 维度的特征。

MNIST^[107] 数据集包含七万张手工书写的图片。这七万张图片被划分为十个类别，这十个类别分别表示手写数字 0 到 9。MNIST 是一个单类别数据集，

^①<http://qwone.com/~jason/20Newsgroups/>

^②<http://horatio.cs.nyu.edu/mit/tiny/data/index.html>

^③<https://www.flickr.com/>

每个类别包含了七千张图片。每张图片的原始尺寸为 28×28 。

CIFAR10^[108] 数据集包含六万张图片。这六万张图片被划分为十个类别，这十个类别为：飞机、汽车、鸟、卡车、猫、鹿、狗、青蛙、马、船。CIFAR10 是一个单类别数据集，每个类别包含了六千张图片。每张图片的原始尺寸为 32×32 。

MSCOCO^[114] 数据集的图片分为训练集和验证集，训练集包含 82783 张图片，验证集包含 40504 张图片。MSCOCO 是一个多类别数据集，每张图片可能同时属于多个类别。MSCOCO 数据集的类别总数为 91。除了原始的图片，本文还使用在 ImageNet 数据集^[61] 上预训练的神经网络模型 CNNF 提取了 4096 维的深度特征。

表 2-1 中给出了 TINY1M、FLICKR1M、MNIST、CIFAR10 和 MSCOCO 数据集中的一些统计信息，包括数据集大小和随机选择的一个样本示例。

2.2.2.2 图片-文本数据集

FLICKR25K^[110] 数据集包含 25000 个数据点，每个样本对应着一张图片及其对应的文本标签。FLICKR25K 中的图片从 Flickr 网站下载得到。这个数据集是一个多类数据集，类别数目为 24。通常将文本标签数量小于 20 的样本移除，得到包含 20015 个样本的子集来进行实验。除了原始图片和文本，本文中还为图片数据提供了 512 维的 GIST 特征和使用在 ImageNet 数据集^[61] 上预训练的神经网络模型 CNNF 提取的 4096 维的深度特征，为文本数据提供了 1386 维的词袋特征 (Bag-of-Words, 简称 BOW)。

WIKI^[115] 数据集包含 2866 个图片文本对，这些图片文本对从维基百科网站^①上搜集而来。所有的样本被划分为 10 个类别，这十个类别分别是：艺术与建筑、生物学、地理、历史、文学与戏剧、媒体、音乐、皇室与贵族、运动、战争。除了原始图片和文本数据，作者提供了图片和文本数据的手工特征。具体来说，每张图片被表示为 128 维的 SIFT 特征，每个文本样本被表示为 10 维的隐狄利克雷分布特征^[117] (Latent Dirichlet Allocation, 简称 LDA)。

IAPRTC12^[116] 数据集包含 20000 个数据样本点，每个样本包含一张图片及图片对应的文本标签。IAPRTC12 数据集是一个多类数据集，每个样本被标记了多个类别。类别总数为 255。除了原始图片和文本数据，本文中还为图片数据提供了 512 维的 GIST 特征和使用在 ImageNet 数据集^[61] 上预训练的神经网络

^①<https://www.wikipedia.org/>

表 2-1: 哈希学习中常用的图片数据集

数据集	大小	图片示例
TINY1M	1000000	
FLICKR1M	1000000	
MNIST	70000	
CIFAR10	60000	
MSCOCO	123287	





网络模型 CNNF 提取的 4096 维的深度特征，为文本数据提供了 2912 维 BOW 特征。

NUSWIDE^[109] 数据集包含 269648 个样本。每个样本有其对应的图片和文本模态数据。NUSWIDE 的图片从 Flickr 网站收集而来。NUSWIDE 是一个多类数据集，每张图片可能被标记了多个类别标签。类别总数为 81。除了原始的图片 and 文本数据，作者还为图片数据提供了 500 维的视觉词袋特征 (Bag-of-Visual Words, 简称 BOVW)，为文本数据提供了 1000 维的 BOW 特征。本文中还为图片数据提供了使用在 ImageNet 数据集^[61] 上预训练的深度神经网络模型 CNNF 提取的 4096 维的深度特征。

表 2-2 中给出了 FLICKR25K、WIKI、IAPRTC12 和 NUSWIDE 数据集的一些统计信息，包括了数据集大小和随机选择的文本、图片示例。在表 2-2 中，

WIKI 数据集展示的数据样本取自维基百科词条“Ima Hogg”^①，由于文本太长，这里只展示了部分文本，其余部分可从该词条主页找到。

表 2-2: 哈希学习中常用的图片-文本数据集

数据集	大小	文本示例	图片示例
FLICKR25K	25000	explore, beach, people, sea, summer, boat, jump, playa, mar, coast, gente, shore.	
WIKI	2866	Music was always present at the Hogg household, and Ima began learning to play the piano at age three. Although her younger brothers attended public school, Ima was enrolled at a private school and received private music lessons.....	
IAPRTC12	20000	cascading, waterfall, middle, jungle, pool, dirty, water, foreground.	
NUSWIDE	269648	sunset, night, architecture, city, building, lake, lights, colorful, buildings, tower, america, evening, downtown, work, chicago, office, great, cityscape, streets, illinois, business.	

^①https://en.wikipedia.org/wiki/Ima_Hogg

第三章 非深度单模态离散哈希

3.1 引言

图哈希学习方法是一类重要的非深度单模态哈希学习方法。Weiss 等人^[30]在 2008 年提出了 SH 方法，首次在哈希学习中引入图的概念。具体来说，给定两个数据样本 $(\mathbf{x}_i, \mathbf{x}_j)$ ，这两个数据样本在欧氏空间的相似度定义如下：

$$S_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\rho}}.$$

这里， $\rho > 0$ 表示一个超参数， $\|\cdot\|_2$ 表示向量的 2 范数， $S_{ij} \in (0, 1]$ 。当给定 n 个样本时，样本间的相似度组成的矩阵 $\mathbf{S} = \{S_{ij}\}_{i,j=1}^n$ 表示相似度图。图哈希的目标为学习样本的二值表示，使得在二值空间中的二值表示仍能尽量保持原空间的相似度。如果两个样本在原空间相似，即 S_{ij} 趋向于 1，那么，希望这两个样本在二值空间也尽量相似，即希望他们之间的海明距离尽量小，反之，如果两个样本在原空间中不相似，即 S_{ij} 趋向于 0，那么，希望这两个样本在二值空间中也尽量不相似，即希望他们的海明距离尽量大。根据相似度定义可知，相似度图的计算复杂度为 $\mathcal{O}(n^2)$ 。

SH^[30]、AGH^[32]、DGH^[38] 等方法是代表性的图哈希学习方法。SH 使用拉普拉斯降维方法建模图哈希问题。为了避免二值约束带来的困难，SH 在训练过程中采用了直接丢弃二值约束的方式，首先学习样本的实值表示，训练过程结束后，再将实值表示投影到二值空间中得到样本的二值哈希编码。SH 假设数据服从均匀分布，并使用一维拉普拉斯谱图特征分解来避免高复杂度。AGH 方法引入了锚点的概念，使用锚图来近似相似度图。具体来说，AGH 通过锚点和数据样本之间的距离作为桥梁，重新定义两个样本之间的相似度。根据新定义的相似度图，作者设计了一种 $\mathcal{O}(n)$ 复杂度的学习算法。优化过程中，AGH 也采用了丢弃二值约束的策略来避免二值约束存在带来的问题。DGH 同样使用了锚点来重新定义相似度图并建模。与 AGH 不同的是，DGH 设计了一种直接学习二值哈希编码的离散优化方法。具体来说，DGH 首先将二值哈希编码平衡约束与二值哈希编码独立约束松弛到实值空间中，然后设计了一种取符号梯

度上升方法来学习二值哈希编码。作者证明了所提出的取符号梯度上升方法的收敛性。实验中 DGH 与 AGH 的比较证明了离散优化方法能达到更好的检索精度。然而，DGH 方法无法使用全部相似度信息进行训练，其检索精度也因此受损。同时，由于锚图的构造过程效率不高，导致 DGH 的训练过程低效。

为了提高离散图哈希学习方法的检索精度，更好的利用整个相似度图的监督信息，本章提出一种基于特征变换的可扩展图哈希学习方法（Scalable Graph Hashing with Feature Transformation, 简称 SGH）。本章工作的主要贡献包括以下三个方面：

- 本章提出的 SGH 方法设计了一种可以隐式地计算整个相似度图的特征变换方法。在学习过程中，SGH 方法可以避免平方级别的复杂度，并提高离散图哈希算法的训练效率。
- 本章提出的 SGH 方法设计了一种高效的逐比特离散优化的二值哈希编码学习算法，该学习算法通过优化残差损失来学习二值哈希编码。
- 实验证明与现有的离散图哈希学习方法相比，本章提出的 SGH 方法能达到最好的检索精度，同时其训练过程也更高效。

本章剩余部分组织如下：第 3.2 节中介绍问题定义；第 3.3 节介绍本章提出的 SGH 方法；第 3.4 节给出本文提出的 SGH 方法和基准方法在两个数据集上的实验；第 3.5 节总结本章提出的方法。

3.2 问题定义

假设训练样本集为 $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$ ，其中 n 表示训练集大小， d 表示特征维度。不失一般性，假设训练数据集已经经过中心化处理，即： $\sum_{i=1}^n \mathbf{x}_i = \mathbf{0}_d$ ，其中 $\mathbf{0}_d$ 表示元素全部为 0 的 d 维向量。哈希学习目标是將样本表示为二值哈希编码^①，即： $\mathbf{b}_i \in \{-1, +1\}^c$ ，其中 c 表示二值哈希编码长度。使用 c 个哈希函数 $\{h_k(\cdot) \mid k = 1, \dots, c\}$ 来將 \mathbf{x}_i 投影到二值空间中，即： $\mathbf{b}_i = [h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \dots, h_c(\mathbf{x}_i)]^\top$ 。

对于图哈希学习方法，本章使用 SH^[30] 定义的相似度来衡量两个数据样本之间的相似度。图哈希学习方法的目標是学习具有保相似度性质的二值哈希编码，具体来说，如果两个样本在原空间中是相似的，那么希望他们的二值哈希

^①这里，二值哈希编码被表示成 -1 与 $+1$ 。学习结束后， -1 被替换成 0，得到用 0 和 1 表示的二值哈希编码。

编码之间的海明距离应该尽量小；如果两个数据点在原空间中是不相似的，那么希望他们的二值哈希编码之间的海明距离尽量大。

3.3 基于特征变换的可扩展图哈希学习方法 SGH

本节详细地介绍本章提出的基于特征变换的可扩展图哈希学习方法 SGH，包括模型、学习算法、样本外扩展和复杂度分析四个部分。

3.3.1 模型

3.3.1.1 目标函数

给定训练集 $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top$ ，可得到整个训练集的相似度图矩阵为： $\mathbf{S} = \{S_{ij}\}_{i,j=1}^n$ 。图哈希学习目标函数是学习二值哈希编码来逼近相似度矩阵 \mathbf{S} ，因此，图哈希学习方法尝试优化如下的目标函数：

$$\begin{aligned} \min \mathcal{J}(\mathbf{B}) &= \sum_{i=1}^n \sum_{j=1}^n (c\tilde{S}_{ij} - \mathbf{b}_i^\top \mathbf{b}_j)^2, \\ \text{subject to: } \mathbf{B} &= [\mathbf{b}_1, \dots, \mathbf{b}_n]^\top \in \{-1, +1\}^{n \times c}. \end{aligned} \quad (3-1)$$

这里， $\tilde{S}_{ij} \triangleq 2S_{ij} - 1$ 。根据 S_{ij} 的定义可得： $\tilde{S}_{ij} \in (-1, 1]$ 。给定二值哈希编码对 $(\mathbf{b}_i, \mathbf{b}_j)$ ，他们之间的海明距离 $\text{dist}_H(\mathbf{b}_i, \mathbf{b}_j)$ 可以通过如下公式计算：

$$\text{dist}_H(\mathbf{b}_i, \mathbf{b}_j) = \frac{1}{2}(c - \mathbf{b}_i^\top \mathbf{b}_j). \quad (3-2)$$

根据 (3-2) 可得， $\text{dist}_H(\mathbf{b}_i, \mathbf{b}_j) \in \{0, 1, \dots, c\}$ 。同时可以通过计算 \mathbf{b}_i 与 \mathbf{b}_j 之间的内积来得到海明距离。具体来说， $\text{dist}_H(\mathbf{b}_i, \mathbf{b}_j)$ 越小， \mathbf{b}_i 与 \mathbf{b}_j 之间的内积越大，对应的两个数据样本 \mathbf{x}_i 与 \mathbf{x}_j 应该趋向于相似，即： $\tilde{S}_{ij} = 1$ ；反之， $\text{dist}_H(\mathbf{b}_i, \mathbf{b}_j)$ 越大， \mathbf{b}_i 与 \mathbf{b}_j 之间的内积越小，对应的两个数据样本 \mathbf{x}_i 与 \mathbf{x}_j 应该趋向于不相似，即相似度 \tilde{S}_{ij} 趋向于 -1 。因此，可以看到通过优化问题 (3-1) 得到的二值哈希编码能尽可能的保持由相似度 $\tilde{\mathbf{S}}$ 定义的相似度。

给定数据样本特征 \mathbf{X} ，可以使用线性投影函数将 \mathbf{X} 投影到 \mathbb{R}^c 空间。然而，在线性不可分的情况下，采用线性投影无法达到令人满意的检索精度。受到 KLSH^[26]、KSH^[86] 等方法的启发，本章采用基于核的方法将数据表示为核

化特征。具体来说，定义第 k 个哈希函数 $h_k(\cdot)$ 如下：

$$h_k(\mathbf{x}_i) = \mathbf{sign}\left(\sum_{j=1}^{n_k} W_{kj}\phi(\mathbf{x}_i, \mathbf{x}_j) + z_k\right).$$

这里 $\mathbf{W} \in \mathbb{R}^{c \times n_k}$ 表示权重矩阵， $\phi(\mathbf{x}_i, \mathbf{x}_j)$ 表示核函数， n_k 表示核基的数目， $\mathbf{sign}(\cdot)$ 表示逐元素取符号函数，其定义为：

$$\mathbf{sign}(x) = \begin{cases} +1, & \text{if } x \geq 0, \\ -1, & \text{otherwise.} \end{cases}$$

z_k 表示偏置项，定义为： $z_k = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^{n_k} W_{kj}\phi(\mathbf{x}_i, \mathbf{x}_j)$ ，偏置项的作用是使得均值归零。

因此，哈希函数可以定义为： $h_k(\mathbf{x}) = \mathbf{sign}(k(\mathbf{x})\mathbf{w}_i)$ ，其中， $\mathbf{w}_i = \mathbf{W}_{k*}^\top$ ，函数 $k(\mathbf{x})$ 的定义为：

$$k(\mathbf{x}) = \mathbf{concat}\left(\phi(\mathbf{x}, \mathbf{x}_1) - \frac{1}{n} \sum_{i=1}^n \phi(\mathbf{x}_i, \mathbf{x}_1), \dots, \phi(\mathbf{x}, \mathbf{x}_{n_k}) - \frac{1}{n} \sum_{i=1}^n \phi(\mathbf{x}_i, \mathbf{x}_{n_k})\right),$$

其中， $\mathbf{concat}(\cdot, \dots)$ 表示向量拼接函数。给定任意两个列向量 $\mathbf{c} \in \mathbb{R}^{n_c}$ ， $\mathbf{d} \in \mathbb{R}^{n_d}$ ，向量拼接函数定义如下：

$$\mathbf{concat}(\mathbf{c}, \mathbf{d}) = [\mathbf{c}; \mathbf{d}] = [c_1, \dots, c_{n_c}, d_1, \dots, d_{n_d}]^\top \in \mathbb{R}^{n_c+n_d}.$$

这里使用两个向量作为输入。本文中假设 $\mathbf{concat}(\cdot, \dots)$ 函数可接受任意多个向量作为输入。因此，目标函数 (3-1) 可重写如下：

$$\min_{\mathbf{W}} \mathcal{J}(\mathbf{W}) = \|\mathbf{c}\tilde{\mathbf{S}} - \mathbf{sign}(\mathbf{K}\mathbf{W}^\top)\mathbf{sign}(\mathbf{K}\mathbf{W}^\top)^\top\|_F^2, \quad (3-3)$$

其中， $\mathbf{K} = [k(\mathbf{x}_1), \dots, k(\mathbf{x}_n)]^\top \in \mathbb{R}^{n \times n_k}$ 表示全部训练集的基于核函数的特征矩阵， $\|\cdot\|_F$ 表示矩阵的 Frobenius 范数。为了保证二值哈希编码的独立性，在问题 (3-3) 中引入了施加在实值投影上的比特独立约束。因此可得：

$$\begin{aligned} \min_{\mathbf{W}} \mathcal{J}(\mathbf{W}) &= \|\mathbf{c}\tilde{\mathbf{S}} - \mathbf{sign}(\mathbf{K}\mathbf{W}^\top)\mathbf{sign}(\mathbf{K}\mathbf{W}^\top)^\top\|_F^2, & (3-4) \\ \text{subject to: } & \mathbf{W}\mathbf{K}^\top\mathbf{K}\mathbf{W}^\top = \mathbf{I}_c, \end{aligned}$$

其中， \mathbf{I}_c 表示维度为 $c \times c$ 的单位矩阵。

3.3.1.2 特征变换

考虑到直接计算全部相似度信息需要 $\mathcal{O}(n^2)$ 的计算与存储复杂度，本章提出一种基于特征变换方法来隐式地计算相似度 $\tilde{\mathbf{S}}$ 。首先定义两个函数 $f(\cdot)$ 和 $g(\cdot)$ 如下：

$$\begin{aligned} f(\mathbf{x}) &= \text{concat}\left(\sqrt{\frac{2(e^2-1)}{e\rho}}e^{-\frac{\|\mathbf{x}\|_2^2}{\rho}}\mathbf{x}, \sqrt{\frac{e^2+1}{e}}e^{-\frac{\|\mathbf{x}\|_2^2}{\rho}}, 1\right), \\ g(\mathbf{x}) &= \text{concat}\left(\sqrt{\frac{2(e^2-1)}{e\rho}}e^{-\frac{\|\mathbf{x}\|_2^2}{\rho}}\mathbf{x}, \sqrt{\frac{e^2+1}{e}}e^{-\frac{\|\mathbf{x}\|_2^2}{\rho}}, -1\right), \end{aligned} \quad (3-5)$$

由上式，可得： $f(\mathbf{x}), g(\mathbf{x}) \in \mathbb{R}^{(d+2)}$ 。

根据函数 $f(\cdot)$ 和 $g(\cdot)$ 的定义，对于任意的样本对 $(\mathbf{x}_i, \mathbf{x}_j)$ ，有：

$$\begin{aligned} f(\mathbf{x}_i)^\top g(\mathbf{x}_j) &= 2\left[\frac{e^2-1}{2e} \times \frac{2\mathbf{x}_i^\top \mathbf{x}_j}{\rho} + \frac{e^2+1}{2e}\right]e^{-\frac{\|\mathbf{x}_i\|_2^2 + \|\mathbf{x}_j\|_2^2}{\rho}} - 1 \\ &\approx 2e^{\frac{-\|\mathbf{x}_i\|_2^2 - \|\mathbf{x}_j\|_2^2 + 2\mathbf{x}_i^\top \mathbf{x}_j}{\rho}} - 1 = 2e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\rho}} - 1 \\ &= \tilde{S}_{ij}. \end{aligned} \quad (3-6)$$

这里使用近似 $\frac{e^2-1}{2e}x + \frac{e^2+1}{2e} \approx x, x \in [-1, 1]$ 。这个近似展示在图 3-1 中。为了保证近似的准确性，本章假设 $-1 \leq \frac{2}{\rho}\mathbf{x}_i^\top \mathbf{x}_j \leq 1$ 。并通过设置 $\rho = 2 \max\{\|\mathbf{x}_i\|_2^2\}_{i=1}^n$ 来保证 $-1 \leq \frac{2}{\rho}\mathbf{x}_i^\top \mathbf{x}_j \leq 1$ 。

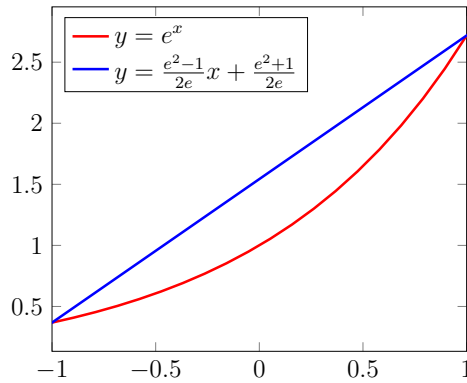


图 3-1: SGH 方法中采用的近似

由等式 (3-6)，容易得到： $\tilde{\mathbf{S}} \approx \hat{\mathbf{X}}\bar{\mathbf{X}}^\top$ ，其中 $\hat{\mathbf{X}} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)]^\top \in \mathbb{R}^{n \times (d+2)}$ ， $\bar{\mathbf{X}} = [g(\mathbf{x}_1), \dots, g(\mathbf{x}_n)]^\top \in \mathbb{R}^{n \times (d+2)}$ 。可以看到，直接计算相似度矩阵 $\tilde{\mathbf{S}}$ 的时间复杂度及存储复杂度仍然是 $\mathcal{O}(n^2)$ 。在下一节中，本文设计了一种

逐比特离散优化的二值哈希编码学习算法，并不需要显示地计算相似度矩阵 $\tilde{\mathbf{S}}$ 。因此，本章通过这种构造避免了 $\mathcal{O}(n^2)$ 的复杂度。

3.3.2 学习算法

直接优化问题 (3-4) 中的目标函数是一个 NP 难的问题。一种可能的方法为直接丢弃二值约束，将哈希函数松弛为实值的函数。SH^[30]、AGH^[32] 均使用了这种松弛的方法。但是，直接丢弃二值约束的松弛方法通常会导致最终的检索精度受损^[34]。本章设计了一种逐比特优化的序列学习算法来学习离散的二值哈希编码。

假设当前已学习得到 $t-1$ 个哈希函数，其参数为 $\{\mathbf{w}_i\}_{i=1}^{t-1}$ 。根据这 $t-1$ 个哈希函数可定义残差矩阵如下：

$$\mathbf{R}_t = c\tilde{\mathbf{S}} - \sum_{i=1}^{t-1} \text{sign}(\mathbf{K}\mathbf{w}_i)\text{sign}(\mathbf{K}\mathbf{w}_i)^\top.$$

学习第 t 个哈希函数时，目标函数可重新定义为：

$$\begin{aligned} \min_{\mathbf{w}_t} \mathcal{J}_r(\mathbf{w}_t) &= \|\mathbf{R}_t - \text{sign}(\mathbf{K}\mathbf{w}_t)\text{sign}(\mathbf{K}\mathbf{w}_t)^\top\|_F^2. \\ \text{subject to: } &\mathbf{w}_t^\top \mathbf{K}^\top \mathbf{K} \mathbf{w}_t = 1. \end{aligned} \quad (3-7)$$

优化 (3-7) 中的问题仍然是一个 NP 难的问题，因此采用松弛策略将问题 (3-7) 重写为：

$$\begin{aligned} \min_{\mathbf{w}_t} \tilde{\mathcal{J}}_r(\mathbf{w}_t) &= \|\mathbf{R}_t - \mathbf{K}\mathbf{w}_t\mathbf{w}_t^\top \mathbf{K}^\top\|_F^2 \\ \text{subject to: } &\mathbf{w}_t^\top \mathbf{K}^\top \mathbf{K} \mathbf{w}_t = 1. \end{aligned} \quad (3-8)$$

问题 (3-8) 中的目标函数可化简为：

$$\begin{aligned} \tilde{\mathcal{J}}_r(\mathbf{w}_t) &= \|\mathbf{R}_t - \mathbf{K}\mathbf{w}_t\mathbf{w}_t^\top \mathbf{K}^\top\|_F^2 \\ &= \text{tr}[(\mathbf{R}_t - \mathbf{K}\mathbf{w}_t\mathbf{w}_t^\top \mathbf{K}^\top)(\mathbf{R}_t - \mathbf{K}\mathbf{w}_t\mathbf{w}_t^\top \mathbf{K}^\top)^\top] \\ &= \text{tr}[\mathbf{K}\mathbf{w}_t\mathbf{w}_t^\top \mathbf{K}^\top \mathbf{K}\mathbf{w}_t\mathbf{w}_t^\top \mathbf{K}^\top] \\ &\quad - 2\text{tr}(\mathbf{w}_t^\top \mathbf{K}^\top \mathbf{R}_t \mathbf{K} \mathbf{w}_t) + \text{tr}(\mathbf{R}_t \mathbf{R}_t^\top) \\ &= -2\text{tr}(\mathbf{w}_t^\top \mathbf{K}^\top \mathbf{R}_t \mathbf{K} \mathbf{w}_t) + \text{const}. \end{aligned}$$

这里， $\text{tr}(\cdot)$ 表示矩阵的迹， const 表示与参数 \mathbf{w}_t 无关的常量。

优化问题 (3-8) 可重写为：

$$\begin{aligned} \min_{\mathbf{w}_t} \tilde{\mathcal{J}}_r(\mathbf{w}_t) &= -\text{tr}(\mathbf{w}_t^\top \mathbf{K}^\top \mathbf{R}_t \mathbf{K} \mathbf{w}_t) + \text{const} \\ \text{subject to: } &\mathbf{w}_t^\top \mathbf{K}^\top \mathbf{K} \mathbf{w}_t = 1. \end{aligned} \quad (3-9)$$

优化问题 (3-9) 的拉格朗日函数为：

$$\mathcal{L}(\mathbf{w}_t, \lambda) = -\text{tr}(\mathbf{w}_t^\top \mathbf{K}^\top \mathbf{R}_t \mathbf{K} \mathbf{w}_t) + \lambda(\mathbf{w}_t^\top \mathbf{K}^\top \mathbf{K} \mathbf{w}_t - 1).$$

根据拉格朗日函数 $\mathcal{L}(\mathbf{w}_t, \lambda)$ ，令：

$$\begin{cases} \frac{\partial \mathcal{L}(\mathbf{w}_t, \lambda)}{\partial \mathbf{w}_t} = -2\mathbf{K}^\top \mathbf{R}_t \mathbf{K} \mathbf{w}_t + 2\lambda \mathbf{K}^\top \mathbf{K} \mathbf{w}_t = \mathbf{0}_{n_k}, \\ \frac{\partial \mathcal{L}(\mathbf{w}_t, \lambda)}{\partial \lambda} = \mathbf{w}_t^\top \mathbf{K}^\top \mathbf{K} \mathbf{w}_t - 1 = 0. \end{cases} \quad (3-10)$$

这里， $\mathbf{0}_{n_k}$ 表示元素全部为 0 的 n_k 维向量。

根据等式 (3-10)，可得如下的广义特征值优化问题：

$$\mathbf{K}^\top \mathbf{R}_t \mathbf{K} \mathbf{w}_t = \lambda \mathbf{K}^\top \mathbf{K} \mathbf{w}_t.$$

定义 $\mathbf{A}_t = \mathbf{K}^\top \mathbf{R}_t \mathbf{K}$ 。根据 \mathbf{R}_t 的定义，当 $t = 1$ 时，可得：

$$\begin{aligned} \mathbf{A}_1 &= c\mathbf{K}^\top \tilde{\mathbf{S}} \mathbf{K} \\ &= c\mathbf{K}^\top \hat{\mathbf{X}} \bar{\mathbf{X}}^\top \mathbf{K} \\ &= c[\mathbf{K}^\top \hat{\mathbf{X}}][\bar{\mathbf{X}}^\top \mathbf{K}]. \end{aligned} \quad (3-11)$$

当 $t > 1$ 时，可得：

$$\begin{aligned} \mathbf{A}_t &= \mathbf{K}^\top \mathbf{R}_t \mathbf{K} \\ &= \mathbf{K}^\top \left[c\tilde{\mathbf{S}} - \sum_{i=1}^{t-1} \text{sign}(\mathbf{K} \mathbf{w}_i) \text{sign}(\mathbf{K} \mathbf{w}_i)^\top \right] \mathbf{K} \\ &= \mathbf{K}^\top \mathbf{R}_{t-1} \mathbf{K} - \mathbf{K}^\top \text{sign}(\mathbf{K} \mathbf{w}_{t-1}) \text{sign}(\mathbf{K} \mathbf{w}_{t-1})^\top \mathbf{K} \\ &= \mathbf{A}_{t-1} - \mathbf{K}^\top \text{sign}(\mathbf{K} \mathbf{w}_{t-1}) \text{sign}(\mathbf{K} \mathbf{w}_{t-1})^\top \mathbf{K}, \end{aligned}$$

在等式 (3-11) 中, 如果首先计算 $\hat{\mathbf{X}}\bar{\mathbf{X}}^\top$, 再计算 $c\mathbf{K}^\top$ 、 $\hat{\mathbf{X}}\bar{\mathbf{X}}^\top$ 和 \mathbf{K} 的乘积, 时间复杂度为 $\mathcal{O}(n^2(n_k + d + 2) + nn_k^2)$ 。然而如果分别计算 $\mathbf{K}^\top\hat{\mathbf{X}}$ 和 $\bar{\mathbf{X}}^\top\mathbf{K}$, 再计算 $\mathbf{K}^\top\hat{\mathbf{X}}$ 和 $\bar{\mathbf{X}}^\top\mathbf{K}$ 的乘积, 时间复杂度将降低至 $\mathcal{O}(2nn_k(d + 2) + n_k^2(d + 2))$ 。现实应用中, $n \gg n_k, d$ 。因此, 可以采用分别计算 $\mathbf{K}^\top\hat{\mathbf{X}}$ 和 $\bar{\mathbf{X}}^\top\mathbf{K}$ 的方法来避免平方级别的复杂度。

当 t 从 1 循环到 c 轮后, 将学习得到参数 $\mathbf{W} = \{\mathbf{w}_i\}_{i=1}^c$ 。通过重新定义残差矩阵如下:

$$\mathbf{R}_t = c\tilde{\mathbf{S}} - \sum_{i=1, i \neq t}^c \text{sign}(\mathbf{K}\mathbf{w}_i)\text{sign}(\mathbf{K}\mathbf{w}_i)^\top,$$

可以进一步进行序列学习。通过进一步的序列学习, 本章提出的 SGH 方法能达到更好的检索精度, 但进一步的学习需要花费更多的时间来进行训练。因此, 实际应用中, 是否进行进一步学习可以根据对检索精度和计算开销的需求来决定。

虽然上述残差优化过程中采用了松弛策略近似地学习了二值哈希编码, 在进行下一轮残差学习时, 上述算法已经将实值编码量化到二值空间并指导下一轮残差学习, 因此上述算法仍然是离散优化算法。

本章提出的 SGH 模型的算法可以简单地概括在算法 3.1 中。

3.3.3 样本外扩展

训练结束后, 可以使用学习得到的哈希函数来为不属于训练的样本生成二值哈希编码。具体来说, 给定样本 $\mathbf{x}_q \notin \mathbf{X}$, 使用如下的公式来生成二值哈希编码: $\mathbf{b}_q = \text{sign}(\mathbf{W}\mathbf{k}(\mathbf{x}_q))$ 。

3.3.4 复杂度分析

SGH 算法的复杂度可以分为两部分考虑, 即: 初始化过程和学习过程。初始化过程中, $\hat{\mathbf{X}}$ 和 $\bar{\mathbf{X}}$, \mathbf{K} 的构造, \mathbf{A}_0 及 \mathbf{Z} 的初始化的复杂度为: $\mathcal{O}(dn_k + dnn_k + nn_k + (n_k^2 + nn_k)(d + 2) + nn_k^2)$ 。学习过程的复杂度为: $\mathcal{O}(c(nn_k + n_k^2) + n_k^3)$ 。 n_k, d, c 通常远小于 n , 因此, 整个算法的时间复杂度为 $\mathcal{O}(n)$ 。

算法 3.1 SGH 模型学习算法

输入:

训练集特征 $\mathbf{X} \in \mathbb{R}^{n \times d}$, 二值哈希编码长度 c , 核数目 n_k 。

输出:

哈希函数参数 \mathbf{W} 。

- 1: 根据等式 (3-5) 构造 $\hat{\mathbf{X}}$ 与 $\bar{\mathbf{X}}$;
- 2: 从训练集中随机选择 n_k 个样本作为核函数的基, 根据核函数的定义构造核化特征矩阵 \mathbf{K} ;
- 3: 初始化: $\mathbf{A}_1 = c[\mathbf{K}^\top \hat{\mathbf{X}}][\bar{\mathbf{X}}^\top \mathbf{K}]$, $\mathbf{Z} = \mathbf{K}^\top \mathbf{K} + \gamma \mathbf{I}_{n_k}$;
- 4: **for** $t = 1 \mapsto c$ **do**
- 5: 解广义特征值问题: $\mathbf{A}_t \mathbf{v} = \lambda \mathbf{Z} \mathbf{v}$;
- 6: 更新 $\mathbf{w}_t = \mathbf{v}$;
- 7: $\mathbf{U} = [\mathbf{K}^\top \text{sign}(\mathbf{K} \mathbf{w}_t)][\mathbf{K}^\top \text{sign}(\mathbf{K} \mathbf{w}_t)]^\top$;
- 8: $\mathbf{A}_{t+1} = \mathbf{A}_t - \mathbf{U}$;
- 9: **end for**
- 10: $\hat{\mathbf{A}}_0 = \mathbf{A}_{c+1}$;
- 11: **for** $s = 1 \mapsto T$ **do**
- 12: 构造从 1 到 c 的随机索引集 $\Omega = \{\pi_1, \dots, \pi_c\}$ 。
- 13: **for** $t = 1 \mapsto c$ **do**
- 14: $\hat{t} = \pi_t$;
- 15: $\hat{\mathbf{A}}_t = \hat{\mathbf{A}}_0 + \mathbf{K}^\top \text{sign}(\mathbf{K} \mathbf{w}_{\hat{t}}) \text{sign}(\mathbf{K} \mathbf{w}_{\hat{t}})^\top \mathbf{K}$;
- 16: 解广义特征值问题: $\hat{\mathbf{A}}_0 \mathbf{v} = \lambda \mathbf{Z} \mathbf{v}$;
- 17: 更新 $\hat{\mathbf{w}}_t = \mathbf{v}$;
- 18: $\hat{\mathbf{A}}_t = \hat{\mathbf{A}}_0 - \mathbf{K}^\top \text{sign}(\mathbf{K} \mathbf{w}_{\hat{t}}) \text{sign}(\mathbf{K} \mathbf{w}_{\hat{t}})^\top \mathbf{K}$;
- 19: **end for**
- 20: **end for**

3.4 实验验证

本节通过在两个大规模数据集的实验来验证本章提出的 SGH 方法的有效性。所有的实验都在一台包含 Intel(R) CPU E5-2620V2@2.1G 12 cores 和 64G RAM 的服务器上完成。

3.4.1 实验设置

3.4.1.1 数据集

本章使用 TINY1M^①和 FLICKR1M^[113] 两个数据集进行实验。数据集的介绍可以参考第二章第 2.2 节。

^①<http://horatio.cs.nyu.edu/mit/tiny/data/index.html>

3.4.1.2 评价标准与对比方法

对于 TINY1M 数据集和 FLICKR1M 数据集，本章随机选择 5000 个样本作为查询集合，剩下的样本作为数据库集合。同时使用整个数据库集合作为训练集。根据每个查询样本与数据库样本之间的欧氏距离来定义相似样本。具体来说，对于每个查询样本，定义数据库中与之最近的 2% 的样本为相似样本。因此，每个查询样本具有 19900 个相似样本。

本章选择了六个具有代表性的方法进行对比。其中包括一个数据独立哈希方法，即：LSH^[16]；两个非图哈希学习方法，即：PCAH^[21] 和 ITQ^[21]；三个图哈希学习方法，即：AGH^[30]、DGH-I^[38] 和 DGH-R^[38]。其中，DGH-I 与 DGH-R 表示 DGH 方法的两个变体。具体来说，DGH-I 与 DGH-R 表示采用了不同初始化策略进行初始化的 DGH 算法^[38]。由于 Liu 等人^[32,38] 已经证明 AGH 和 DGH 方法能达到比 SH^[30] 更好的检索精度，本章没有采用 SH 作为基准方法。对于 SGH 方法，实验中 $\phi(\cdot, \cdot)$ 定义为径向基函数（Radial Basis Function，简称 RBF），并从训练集中随机选择 300 个样本作为核基。设置算法 3.1 中的迭代次数 $T = 1$ ，设置超参数 $\rho = 2$ 。对于 AGH 和 DGH，使用 K-Means 方法来学习聚类中心，然后构造锚图。聚类中心数目设置为 300。对于其他对比方法，实验中根据原论文作者的建议选择超参数。

为验证本章提出的 SGH 方法的有效性，本章使用在返回序列的截断位置 K 的查准率（ $\text{cPre}(K)$ ）作为评价标准。查准率 $\text{cPre}(K)$ 的计算方法参见第二章第 2.2 节。

3.4.2 性能对比

3.4.2.1 精度对比

表 3-1 和表 3-2 分别汇报了本章提出的 SGH 方法和基准方法在 TINY1M 数据集和 FLICKR1M 数据集上的 $\text{cPre}(K)$ ，这里 $K = 1000$ ，二值哈希编码长度设置为：32 比特，64 比特，96 比特和 128 比特。在表 3-1 和表 3-2 中，最好的 $\text{cPre}(K)$ 使用加粗字体标注。从表 3-1 和表 3-2 中可以看到，随着二值哈希编码的长度增加，除 PCAH 之外的所有方法的检索精度越来越好。通过对比 ITQ 和 PCAH，可以发现基于离散优化的非图哈希学习方法能达到更好的检索精度。通过对比 SGH、DGH-I、DGH-R 和 AGH 方法，可以发现基于离散优化

的图哈希学习方法能达到更好的检索精度。与基准方法相比，SGH 在几乎所有情况下都达到了最好的检索精度。在 FLICKR1M 数据集上，当比特长度为 32 时，ITQ 方法的检索精度比 SGH 方法的检索精度稍好。

表 3-1: SGH 方法和基准方法在 TINY1M 数据集上的查准率

方法	比特长度			
	32bits	64bits	96bits	128bits
LSH	0.2507	0.3575	0.4122	0.4529
PCAH	0.2457	0.2203	0.2000	0.1836
ITQ	0.4298	0.4782	0.4947	0.4986
AGH	0.3973	0.4402	0.4577	0.4654
DGH-I	0.3974	0.4536	0.4737	0.4874
DGH-R	0.3793	0.4554	0.4871	0.4989
SGH	0.4696	0.5742	0.6299	0.6737

表 3-2: SGH 方法和基准方法在 FLICKR1M 数据集上的查准率

方法	比特长度			
	32bits	64bits	96bits	128bits
LSH	0.2597	0.3995	0.4660	0.5160
PCAH	0.2702	0.2384	0.2141	0.1950
ITQ	0.5177	0.5776	0.5999	0.6096
AGH	0.4299	0.4741	0.4911	0.4998
DGH-I	0.4299	0.4806	0.5001	0.5111
DGH-R	0.4121	0.4776	0.5054	0.5196
SGH	0.4919	0.6041	0.6677	0.6985

为进一步验证本章提出的 SGH 方法的有效性，图 3-2 中给出了本章提出的 SGH 方法和基准方法在 TINY1M 和 FLICKR1M 数据集上在不同 K 值时对应的

cPre(K) 的结果。在图 3-2 中, K 值的取值范围为: $100 \leq K \leq 1000$, 比特长度设置为 64 比特和 128 比特, 其余比特的实验结果类似。从图 3-2 中可以看出, 与所有基准方法相比, SGH 方法在所有情况能达到最好的检索精度。

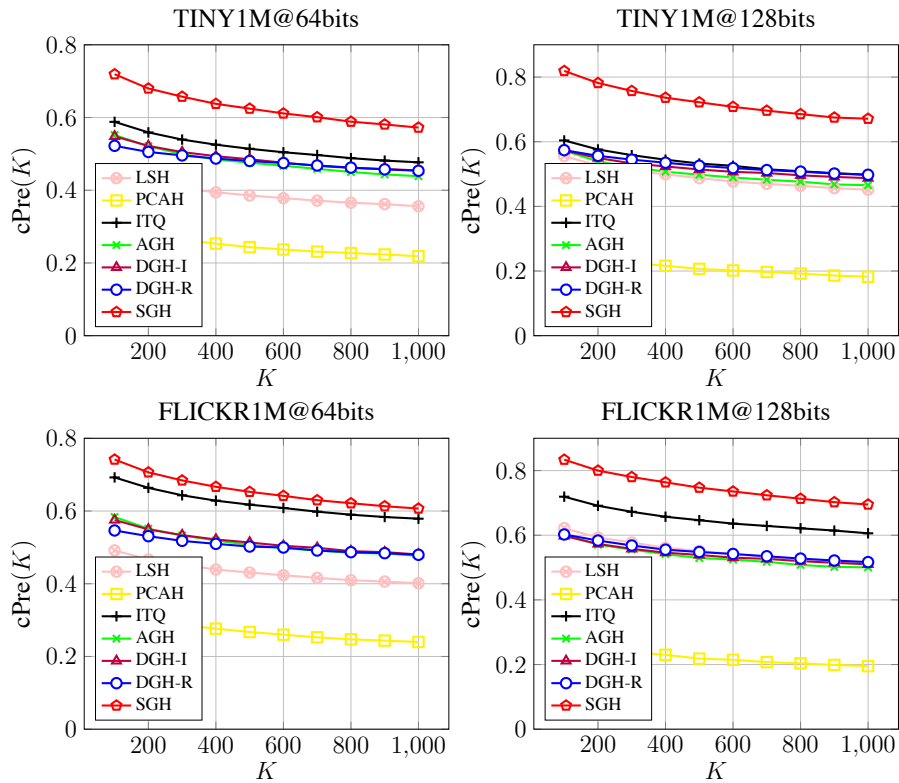


图 3-2: SGH 方法和基准方法在两个数据集上的查准率

3.4.2.2 训练效率对比

本节汇报了所有方法在 TINY1M 和 FLICKR1M 数据集上的哈希函数的生成时间和二值哈希编码的生成时间以验证本文提出的 SGH 的训练效率, 结果展现在表 3-3 和表 3-4 中。对于 LSH 方法, 哈希函数生成的时间表示随机采样生成哈希函数的时间。对于除 LSH 之外的方法, 哈希函数生成的时间表示哈希模型学习的时间, 包括初始化变量的时间和模型训练的时间。在表 3-3 和表 3-4 中, 时间单位为秒。对于 AGH 方法和 DGH 方法, t_0 表示使用 K-Means 聚类方法构造锚图的时间。对于 TINY1M 数据集, $t_0 = 1438.60$, 对于 FLICKR1M 数据集, $t_0 = 1564.86$ 。从表 3-3 和表 3-4 中可以看到, 本章提出的 SGH 方法在所有情况下要快于 AGH、DGH-I 和 DGH-R 方法。在绝大部分情况下, 本章提出的 SGH 方法要快于 ITQ 方法。尽管 LSH、PCAH 方法的哈希函

数的生成时间和二值哈希编码的生成时间比 SGH 更短，但 LSH 和 PCAH 无法达到令人满意的检索精度，因此在现实场景中并不实用。

表 3-3: SGH 方法和基准方法在 TINY1M 数据集上的训练时间

方法	比特长度				
	32bits	64bits	96bits	128bits	256bits
LSH	1.68	1.77	1.84	2.55	3.76
PCAH	4.29	4.54	4.75	5.85	6.49
ITQ	31.72	60.62	89.01	149.18	322.06
AGH	$18.60+t_0$	$19.40+t_0$	$20.08+t_0$	$22.48+t_0$	$25.09+t_0$
DGH-I	$187.57+t_0$	$296.99+t_0$	$518.57+t_0$	$924.08+t_0$	$1838.30+t_0$
DGH-R	$217.06+t_0$	$360.18+t_0$	$615.74+t_0$	$1089.10+t_0$	$2300.10+t_0$
SGH	34.49	52.37	71.53	89.65	174.23

表 3-4: SGH 方法和基准方法在 FLICKR1M 数据集上的训练时间

方法	比特长度				
	32bits	64bits	96bits	128bits	256bits
LSH	2.44	2.43	2.71	3.38	4.21
PCAH	7.65	7.90	8.47	9.23	10.42
ITQ	36.17	64.61	89.50	132.71	285.10
AGH	$17.99+t_0$	$18.80+t_0$	$20.30+t_0$	$19.87+t_0$	$21.60+t_0$
DGH-I	$85.81+t_0$	$143.68+t_0$	$215.41+t_0$	$352.73+t_0$	$739.56+t_0$
DGH-R	$116.25+t_0$	$206.24+t_0$	$308.32+t_0$	$517.97+t_0$	$1199.44+t_0$
SGH	41.51	59.02	74.86	97.25	168.25

3.4.3 超参数实验

本节讨论了 SGH 方法两个最重要的超参数，即： ρ 和核数目 n_k ，的敏感性。图 3-3 汇报了超参数 ρ 对 SGH 方法的影响。从图 3-3 可以看到，随着超参

数 ρ 的增大, 检索精度先变好后变差。当 $1 \leq \rho \leq 4$, SGH 方法对超参数 ρ 并不敏感。图 3-4 汇报了超参数 n_k 对 SGH 方法的影响。从图 3-4 可以看到, 随着 n_k 的增大, 检索精度变得越来越好。然而, 核数目 n_k 的取值越大, 训练过程中所需求的计算开销就越大。实际应用中, n_k 的选择可以根据对检索精度和计算开销的需求来决定。

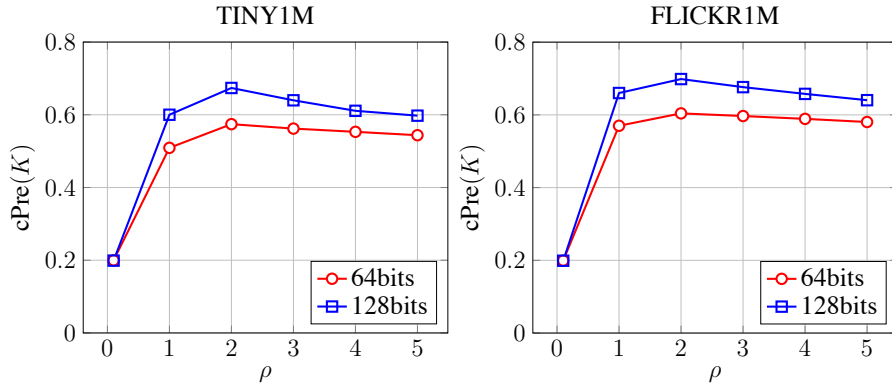


图 3-3: SGH 方法对超参数 ρ 的敏感性实验

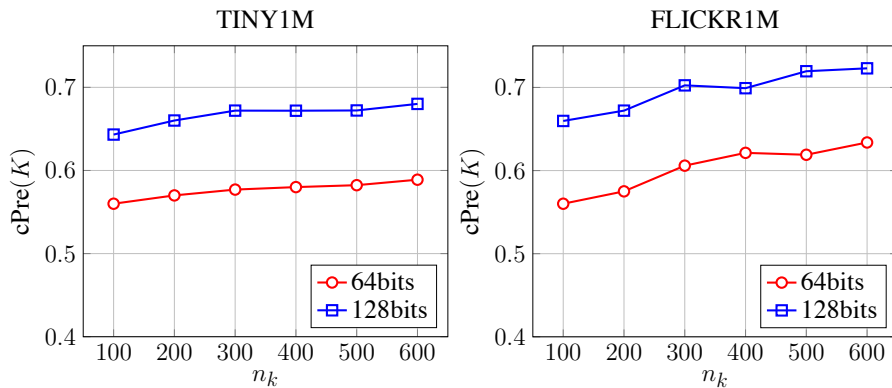


图 3-4: SGH 方法对超参数 n_k 的敏感性实验

3.5 本章小结

本章提出一种基于特征变换的可扩展图哈希学习方法 SGH。SGH 方法可以隐式地利用整个相似度图的监督信息来指导哈希函数的学习。同时, 在学习过程中, SGH 方法避免了平方级别的复杂度, 提高了离散图哈希算法的训练效率。本章同时设计了一种高效的逐比特离散优化的二值哈希编码学习算法。该学习算法通过优化残差损失来学习图哈希模型。实验表明, 本章提出的 SGH 能

达到比现有离散图哈希学习方法更好的检索精度。同时，SGH 方法的训练也比现有的离散图哈希学习方法高效。

本章的工作已总结成文并发表：

- JIANG Q-Y, LI W-J. Scalable Graph Hashing with Feature Transformation[C] // Proceedings of the International Joint Conference on Artificial Intelligence. 2015: 2248 – 2254. (CCF-A 类会议)

第四章 深度单模态离散哈希

4.1 引言

近年来，随着深度学习的迅速兴起，深度特征学习^[61]的思想被引入到很多领域中。Xia 等人^[62]提出了 CNNH，首次将深度特征学习的思想引入哈希学习领域。深度哈希使用深度神经网络将数据投影到低维空间。通过利用深度特征学习，CNNH 取得了比无法进行深度特征学习的哈希学习方法更高的检索精度。CNNH 首先利用监督信息学习样本的实值编码，并将实值编码量化为二值哈希编码，然后使用卷积神经网络来拟合学习得到的二值哈希编码。随后，研究者们提出了很多深度哈希学习方法。代表性的深度哈希学习方法包括 NINH^[63]、DSRH^[92]、DRSCH^[91]、DPSH^[22]、DHN^[65] 和 DSH^[70] 等方法。在现有的深度哈希学习方法中，大部分方法为避免二值哈希编码带来的困难，都采用了直接丢弃二值约束的松弛策略。另外一些方法则设计了离散优化方法。DPSH 是第一个端到端的深度单模态离散哈希学习方法。DPSH 和 DSH 均采用了神经网络学习数据的实值表示，同时尽量降低二值哈希编码和实值表示之间的误差。然而，DPSH、DSH 不能使用监督信息直接指导二值哈希编码的学习，这里，直接指导二值哈希编码的学习的含义为监督信息和二值哈希编码出现在损失函数的同一项中。同时，现有的基于标签对信息或三元组监督信息的深度单模态哈希学习方法都是基于对称哈希的，需要拟合平方级别以上的相似度信息，因此，这类深度哈希学习方法的训练过程低效。

为了解决现有深度单模态离散哈希无法同时直接监督深度特征学习和二值哈希编码学习的问题，本文提出一种深度离散监督哈希学习方法 (Deep Discrete Supervised Hashing, 简称 DDSH)。同时，针对现有的深度哈希学习方法训练低效的问题，本章还提出一种非对称深度监督哈希学习方法 (Asymmetric Deep Supervised Hashing, 简称 ADSH)。本章工作的主要贡献包括以下五个方面:

- 本章提出的深度离散监督哈希 DDSH 首次设计了一种使用监督信息直接指导二值哈希编码学习和深度特征学习的方法。DDSH 方法可以使用监督信

息在同一框架中同时完成监督深度特征的学习和二值哈希编码的学习，通过这种方法可以加强二值哈希编码学习和深度特征学习之间的相互反馈作用。

- 本章提出的 DDSH 方法设计了一种基于 BQP 的离散优化算法来直接学习数据的二值哈希编码表示，并提高哈希学习方法的检索精度。
- 本章还提出一种非对称深度监督哈希学习方法 ADSH。ADSH 方法使用非对称哈希^[118]建模，以非对称的方式来处理针对查询样本的深度特征学习和针对数据库样本的二值哈希编码学习。
- 本章提出的 ADSH 方法设计了一种高效的离散优化算法，该算法能同时完成查询样本的深度特征学习和数据库样本的二值哈希编码学习。与对称深度哈希学习方法相比，本章提出的 ADSH 的训练过程更加高效。
- 实验证明与之前的深度哈希学习方法相比，DDSH 通过使用直接监督二值哈希编码学习和深度特征学习，能达到更高的检索精度。与除 DDSH 方法外的对称深度哈希学习方法相比，ADSH 方法能在最短时间内达到更高的检索精度。与 DDSH 方法相比，ADSH 方法的训练过程更加高效。

本章剩余部分组织如下：第 4.2 节中介绍问题定义；第 4.3 介绍本章提出的 DDSH 方法；第 4.4 介绍本章提出的 ADSH 方法；第 4.5 节通过实验验证 DDSH 方法的有效性；第 4.6 节通过实验验证 ADSH 方法的有效性；最后，第 4.7 节对本章进行总结。

4.2 问题定义

假设训练集为 $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ ，其中 n 表示训练集的大小。对于对称哈希学习方法，定义哈希函数为 $h(\cdot) \in \{-1, +1\}^c$ ，其中 c 表示二值哈希编码长度。对于样本 \mathbf{x}_i ，其二值哈希编码^①为 $\mathbf{b}_i = h(\mathbf{x}_i)$ 。对于监督哈希学习方法，训练样本间的相似度 $\mathbf{S} = \{S_{ij}\}_{ij=1}^n \in \{-1, +1\}^{n \times n}$ 也已给定，其中， S_{ij} 表示样本 \mathbf{x}_i 与样本 \mathbf{x}_j 之间的是否相似。当 $S_{ij} = 1$ 时，表示两个样本相似，当 $S_{ij} = -1$ 时，表示两个样本不相似。对于非对称哈希学习方法，定义哈希函数为 $f(\cdot), g(\cdot) \in \{-1, +1\}^c$ 。在非对称哈希模型中，每个样本均有两个二值哈希编码表示。实际应用场景中，查询样本的编码需要用哈希函数生成，而数据库样本的编码不需要用哈希函数生成。因此，在训练过程中，本章将训练集划分为

^①这里，二值哈希编码被表示成 -1 与 $+1$ 。学习结束后， -1 被替换成 0 ，得到用 0 和 1 表示的二值哈希编码。

查询集 $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{n_q}$ 和数据库样本集 $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^{n_d}$ ，其中 n_q 表示查询集中样本的数目， n_d 表示数据库集中样本的数目。并且 DDSH 仅为查询样本学习哈希函数 $f(\cdot)$ ，仅为数据库样本学习由哈希函数 $g(\cdot)$ 生成的二值哈希编码表示。定义数据库样本的二值哈希编码为 $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_{n_d}]^\top \in \{-1, +1\}^{n_d \times c}$ 。对于监督哈希学习方法，使用 $\mathbf{A} \in \{-1, +1\}^{n_q \times n_d}$ 表示查询样本集和数据库样本集之间的相似度。对于任意的 $A_{ij} \in \mathbf{A}$ ，当 $A_{ij} = 1$ 时，表示查询样本 \mathbf{y}_i 和数据库样本 \mathbf{z}_j 相似，否则，当 $A_{ij} = -1$ 时，表示查询样本 \mathbf{y}_i 和数据库样本 \mathbf{z}_j 不相似。

哈希学习的目标是使用哈希函数将数据从原始空间投影到二值空间中以得到数据的二值哈希编码表示，同时希望二值哈希编码尽可能的保持其在原始空间中的相似度。如果两个数据样本在原始空间相似，那么希望他们的二值哈希编码之间的海明距离尽可能小，否则，如果两个数据样本在原始空间不相似，希望他们之间的海明距离尽可能大。

4.3 深度离散监督哈希学习方法 DDSH

本节从模型、学习算法和样本外扩展三个方面介绍本章提出的深度离散监督哈希学习方法 DDSH。

4.3.1 模型

对于任意二值哈希编码对 $(\mathbf{b}_i, \mathbf{b}_j)$ ，可以通过优化他们之间的内积和其相似度 S_{ij} 之间的 L_2 损失来学习保相似性的二值哈希编码。具体来说，基于 L_2 损失的目标函数定义为：

$$J(\mathbf{b}_i, \mathbf{b}_j; S_{ij}) = (cS_{ij} - \mathbf{b}_i^\top \mathbf{b}_j)^2.$$

给定训练集 $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ 和监督信息 \mathbf{S} ，哈希的目标是学习哈希函数 $h(\cdot)$ 将样本表示为二值哈希编码 \mathbf{B} 。因此优化问题可写为如下的形式：

$$\min_{\mathbf{B}, \theta} \mathcal{J}(\mathbf{B}, \theta) = \sum_{i=1}^n \sum_{j=1}^n J(\mathbf{b}_i, \mathbf{b}_j; S_{ij}) = \sum_{i=1}^n \sum_{j=1}^n (cS_{ij} - \mathbf{b}_i^\top \mathbf{b}_j)^2 \quad (4-1)$$

$$\text{subject to: } \mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_n]^\top \in \{-1, +1\}^{n \times c},$$

$$\forall i \in \{1, \dots, n\}, \mathbf{b}_i = h(\mathbf{x}_i) = \text{sign}(e(\mathbf{x}_i; \theta)).$$

这里, $e(\cdot)$ 定义为将样本投影到 \mathbb{R}^c 空间的投影函数, θ 表示函数 $e(\cdot)$ 的参数。问题 (4-1) 中使用了全部相似度信息来建模。受到 COSDISH^[44] 的启发, DDSH 采用列采样的方式将监督信息划分成两个部分。具体来说, 假设下标集合 $\Phi = \{1, \dots, n\}$, 随机生成一个 Φ 的子集 Ω , 定义 $\Gamma = \{i \mid \forall i \in \Phi, i \notin \Omega\}$, 其中, 假设 $|\Gamma| \gg |\Omega|$ 。根据上述定义, 重新定义优化问题如下:

$$\begin{aligned} \min_{\theta, \mathbf{B}} \mathcal{J}(\theta, \mathbf{B}) &= \sum_{i \in \Omega} \sum_{j=1}^n (cS_{ij} - \mathbf{b}_i^\top \mathbf{b}_j)^2 \\ &= \sum_{i \in \Omega} \sum_{j \in \Gamma} (cS_{ij} - \mathbf{b}_i^\top \mathbf{b}_j)^2 + \sum_{i \in \Omega} \sum_{j \in \Omega} (cS_{ij} - \mathbf{b}_i^\top \mathbf{b}_j)^2 \\ \text{subject to: } \mathbf{B} &= [\mathbf{b}_1, \dots, \mathbf{b}_n]^\top \in \{-1, +1\}^{n \times c}, \\ \forall j \in \Gamma, \mathbf{b}_j &= h(\mathbf{x}_j) = \mathbf{sign}(e(\mathbf{x}_j; \theta)). \end{aligned} \quad (4-2)$$

定义 $\mathbf{B}^\Omega \triangleq \{\mathbf{b}_i \mid \forall i \in \Omega, \mathbf{b}_i = \mathbf{sign}(e(\mathbf{x}_i; \theta))\}$, 然后将优化问题 (4-2) 重写为:

$$\begin{aligned} \min_{\theta, \mathbf{B}^\Omega} \mathcal{J}(\theta, \mathbf{B}^\Omega) &= \sum_{i \in \Omega} \sum_{j \in \Gamma} (cS_{ij} - \mathbf{b}_i^\top \mathbf{sign}(e(\mathbf{x}_j; \theta)))^2 \\ &\quad + \sum_{i, j \in \Omega} (cS_{ij} - \mathbf{b}_i^\top \mathbf{b}_j)^2 \\ \text{subject to: } \forall j \in \Gamma, \mathbf{b}_j &= \mathbf{sign}(e(\mathbf{x}_j; \theta)) \in \{-1, +1\}^c. \end{aligned} \quad (4-3)$$

本章使用深度神经网络来完成深度特征学习。具体来说, DDSH 首先使用一个卷积神经网络作为基础网络架构, 然后在基础网络架构的基础上增加了一层全连接层组成新的深度神经网络, 并使用这个深度神经网络完成深度特征学习。实际中, 基础网络架构可以自行设计或者从任意一种已知的深度神经网络修改得到, 例如, 可以将 AlexNet^[61]、CNNF^[119] 或者 VGGNet^[120] 的分类层去掉, 得到的网络就可以作为基础网络架构。由于本章提出的方法重点在研究如何设计一种可以同时直接监督深度特征学习和二值哈希编码学习的哈希模型, 因此这里不指定某种具体的基础网络架构, 仅假设基础网络架构的输出维度为 d_o 。得到 d_o 维的神经网络输出后, 使用全连接层将基础网络架构的输出投影到 \mathbb{R}^c 空间。使用 $e(\mathbf{x}; \theta)$ 表示深度特征学习架构, 其中 θ 表示深度特征学习架构的所有待学习参数。

从优化问题 (4-3) 可以看到, 二值哈希编码表示 \mathbf{B}^Ω 、哈希函数的参数 θ 都

和监督信息 \mathbf{S} 在同一个目标函数项中，没有使用中间变量来建立与监督信息的联系，这意味着 DDSH 采用了一种直接监督的方式来完成深度特征学习和二值哈希编码学习。

4.3.2 学习算法

本节设计了交替优化算法来学习参数 $\{\theta, \mathbf{B}^\Omega\}$ 。具体来说，交替优化算法依次优化其中的一个参数，固定其他参数不变。

4.3.2.1 固定参数 θ ，学习参数 \mathbf{B}^Ω

当参数 θ 固定时，将问题 (4-3) 重写为矩阵形式，可得：

$$\begin{aligned} \min_{\mathbf{B}^\Omega} \mathcal{J}(\mathbf{B}^\Omega) &= \|\mathbf{c}\mathbf{S}^\Omega - \mathbf{B}^\Omega[\mathbf{B}^\Omega]^\top\|_F^2 + \|\mathbf{c}\mathbf{S}^\Gamma - \mathbf{B}^\Gamma[\mathbf{B}^\Omega]^\top\|_F^2 \quad (4-4) \\ \text{subject to: } \mathbf{B}^\Omega &\in \{-1, +1\}^{|\Omega| \times c}, \end{aligned}$$

其中， $\mathbf{B}^\Gamma \triangleq \{\mathbf{b}_j \mid \forall j \in \Gamma, \mathbf{b}_j = \text{sign}(e(\mathbf{x}_j; \theta))\}$ ， $\mathbf{S}^\Omega \in \{-1, +1\}^{|\Omega| \times |\Omega|}$ 表示由以 Ω 索引的数据样本集之间的相似度组成的相似度矩阵， $\mathbf{S}^\Gamma \in \{-1, +1\}^{|\Gamma| \times |\Omega|}$ 表示由以 Γ 索引的数据样本集与由以 Ω 索引的数据样本集之间的相似度组成的相似度矩阵。

本章采用依次优化 \mathbf{B}^Ω 的某一行 \mathbf{B}_{*k}^Ω 的方式来学习 \mathbf{B}^Ω 。具体来说，当优化第 k 列二值哈希编码 \mathbf{B}_{*k}^Ω 时，固定其余列对应的二值哈希编码不变。因此可将问题 (4-4) 中目标函数 $\mathcal{J}(\mathbf{B}^\Omega)$ 关于第 k 个比特的问題改写为如下形式：

$$\begin{aligned} \min_{\mathbf{B}_{*k}^\Omega} \mathcal{J}(\mathbf{B}_{*k}^\Omega) &= [\mathbf{B}_{*k}^\Omega]^\top \mathbf{Q}^k \mathbf{B}_{*k}^\Omega + [\mathbf{B}_{*k}^\Omega]^\top \mathbf{p}^k \\ \text{subject to: } \mathbf{B}_{*k}^\Omega &\in \{-1, +1\}^{|\Omega|}, \quad (4-5) \end{aligned}$$

其中：

$$\begin{aligned} \forall i \neq j, Q_{ij}^k &= -2(cS_{ij}^\Omega - \sum_{m=1}^{k-1} B_{im}^\Omega B_{jm}^\Omega), \\ Q_{ii}^k &= 0, \\ p_i^k &= -2 \sum_{l=1}^{|\Gamma|} B_{lk}^\Gamma (cS_{li}^\Gamma - \sum_{m=1}^{k-1} B_{lm}^\Gamma B_{im}^\Omega). \end{aligned}$$

类似 COSDISH 方法^[44]，可以将离散二次规划问题 (4-5) 转化为聚类问题^[103] 来得到 B_{*k}^Ω 的解。

4.3.2.2 固定 B^Ω ，学习参数 θ

当二值哈希编码 B^Ω 固定时，首先将优化问题 (4-3) 重写为目标函数关于参数 θ 的形式，即：

$$\begin{aligned} \min_{\theta} \quad \mathcal{J}(\theta) &= \sum_{i \in \Omega} \sum_{j \in \Gamma} (cS_{ij} - \mathbf{b}_i^\top \mathbf{b}_j)^2 \\ \text{subject to: } \forall j \in \Gamma, \mathbf{b}_j &= h(\mathbf{x}_j) = \mathbf{sign}(e(\mathbf{x}_j; \theta)). \end{aligned} \quad (4-6)$$

由于问题 (4-6) 中存在取符号函数 $\mathbf{sign}(\cdot)$ ，该函数在非零点的梯度处处为零。因此，在学习过程中无法直接使用反向传播算法优化模型。因此，本章中采用 $\mathbf{tanh}(\cdot)$ 函数代替 $\mathbf{sign}(\cdot)$ 函数的松弛策略，将优化问题 (4-6) 改写为如下形式：

$$\begin{aligned} \min_{\theta} \quad \hat{\mathcal{J}}(\theta) &= \sum_{i \in \Omega} \sum_{j \in \Gamma} (cS_{ij} - \mathbf{b}_i^\top \tilde{\mathbf{b}}_j)^2 \\ \text{subject to: } \forall j \in \Gamma, \tilde{\mathbf{b}}_j &= \tilde{h}(\mathbf{x}_j) = \mathbf{tanh}(e(\mathbf{x}_j; \theta)). \end{aligned}$$

DDSH 使用反向传播算法来学习参数 θ 。具体来说，本章随机选择包含 n_b 个样本的小批量样本集合来构造小批量集合。根据这个样本集合，计算损失函数 $\hat{\mathcal{J}}(\theta)$ 关于 θ 的梯度。给定样本 \mathbf{x}_{j_p} ，损失函数关于神经网络输出 $\mathbf{o}_{j_p} = e(\mathbf{x}_{j_p}; \theta)$ 的梯度的计算公式为：

$$\frac{\partial \hat{\mathcal{J}}}{\partial \tilde{\mathbf{b}}_{j_p}} = \sum_{i \in \Omega} 2(\tilde{\mathbf{b}}_{j_p}^\top \mathbf{b}_i - cS_{ij_p}) \mathbf{b}_i, \quad (4-7)$$

$$\begin{aligned} \frac{\partial \hat{\mathcal{J}}}{\partial \mathbf{o}_{j_p}} &= \frac{\partial \hat{\mathcal{J}}}{\partial \tilde{\mathbf{b}}_{j_p}} \odot (1 - \tilde{\mathbf{b}}_{j_p}^2) \\ &= \sum_{i \in \Omega} 2(\tilde{\mathbf{b}}_{j_p}^\top \mathbf{b}_i - cS_{ij_p}) \mathbf{b}_i \odot (1 - \tilde{\mathbf{b}}_{j_p}^2), \end{aligned} \quad (4-8)$$

其中，符号 \odot 表示向量点乘操作。根据 $\frac{\partial \hat{\mathcal{J}}}{\partial \tilde{\mathbf{b}}_{j_p}}$ 和 $\frac{\partial \hat{\mathcal{J}}}{\partial \mathbf{o}_{j_p}}$ ，可以使用链式法则来计算 $\frac{\partial \hat{\mathcal{J}}}{\partial \theta}$ 。

本章提出的 DDSH 模型的算法可以简单地概括在算法 4.1 中。

算法 4.1 DDSH 模型学习算法

输入:

图片训练集 $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$, 相似度矩阵 \mathbf{S} , 二值哈希编码长度 c 。

输出:

深度神经网络参数 θ , 二值哈希编码 \mathbf{B} 。

- 1: **初始化**: 初始化深度神经网络参数 θ , 小批量样本大小 n_b , 迭代轮数 T_{out} , T_{in} , 二值哈希编码 \mathbf{B} 。
- 2: **for** $iteration = 1 \mapsto T_{out}$ **do**
- 3: 从 Φ 中随机采子集 Ω , 同时令: $\Gamma = \Phi \setminus \Omega$;
- 4: 将训练集 \mathbf{X} 划分为 \mathbf{X}^Ω 与 \mathbf{X}^Γ ;
- 5: 将二值哈希编码 \mathbf{B} 划分为 \mathbf{B}^Ω 与 \mathbf{B}^Γ ;
- 6: **for** $epoch = 1 \mapsto T_{in}$ **do**
- 7: **for** $k = 1 \mapsto c$ **do**
- 8: 以 \mathbf{b}_k 为变量, 使用公式 (4-5) 构造离散二次优化问题;
- 9: 根据构造的离散二次优化问题构造等价的聚类问题并求解;
- 10: **end for**
- 11: **for** $t = 1 \mapsto \lceil \frac{|\Gamma|}{n_b} \rceil$ **do**
- 12: 从训练集子集 \mathbf{X}^Γ 中随机采 n_b 个样本构造小批量样本集合;
- 13: 使用前馈神经网络为小批量样本集中的样本计算 $e(\mathbf{x}_j; \theta)$;
- 14: 根据公式 (4-7) 和公式 (4-8) 计算梯度;
- 15: 使用反向传播算法更新深度神经网络参数 θ ;
- 16: 更新二值哈希编码 $\mathbf{b}_j = \text{sign}(e(\mathbf{x}_j; \theta))$;
- 17: **end for**
- 18: **end for**
- 19: **end for**

4.3.3 样本外扩展

训练结束后, 可以使用学习得到的深度神经网络来为任一不属于训练集中的样本生成二值哈希编码表示。具体来说, 给定的数据样本不属于训练集时, 即: $\mathbf{x}_q \notin \mathbf{X}$, 使用如下的公式来生成二值哈希编码:

$$\mathbf{b}_q = \text{sign}(e(\mathbf{x}_q; \theta)).$$

4.4 非对称深度监督哈希学习方法 AD SH

由于需要使用全部监督信息来指导模型学习, 大部分现有的深度哈希学习算法和上一节提出的 DDSH 方法存在训练效率低的问题。本节提出一种非对称深度监督哈希学习方法 AD SH, 以非对称的方式来处理针对查询样本的深度特

征学习和针对数据库样本的二值哈希编码学习。本节从模型、学习算法、样本外扩展和复杂度分析四个方面详细地介绍本章提出的 ADSH 方法。

4.4.1 模型

本节使用与 DDSH 相同的 L_2 损失函数来学习查询样本的哈希函数和数据库样本的二值哈希编码。具体来说，优化问题可写为如下的形式：

$$\begin{aligned} \min_{\mathbf{V}, \phi} \mathcal{J}(\mathbf{V}, \phi) &= \sum_{i=1}^{n_q} \sum_{j=1}^{n_d} (cA_{ij} - \mathbf{u}_i^\top \mathbf{v}_j)^2 \\ \text{subject to: } \mathbf{V} &\in \{-1, +1\}^{n_d \times c}, \mathbf{u}_i = f(\mathbf{y}_i) = \mathbf{sign}(e(\mathbf{y}_i; \phi)). \end{aligned} \quad (4-9)$$

这里， \mathbf{u}_i 表示查询样本 \mathbf{y}_i 由哈希函数 $f(\cdot)$ 生成的二值哈希编码， \mathbf{v}_j 表示数据库样本 \mathbf{z}_j 由哈希函数 $g(\cdot)$ 生成的二值哈希编码，由于函数 $e(\cdot)$ 功能和 DDSH 中相同，均为将数据样本投影到 \mathbb{R}^c 空间的投影函数，这里仍使用 $e(\cdot)$ 表示深度特征学习函数，同时使用 ϕ 表示哈希函数 $f(\cdot)$ 的参数。

由于取符号函数 $\mathbf{sign}(\cdot)$ 的存在，问题 (4-9) 是一个 NP 难的问题。因此，训练过程中，ADSH 使用 $\mathbf{tanh}(\cdot)$ 替代取符号函数 $\mathbf{sign}(\cdot)$ ，并尝试解如下的优化问题：

$$\begin{aligned} \min_{\mathbf{V}, \phi} \mathcal{J}(\mathbf{V}, \phi) &= \sum_{i=1}^{n_q} \sum_{j=1}^{n_d} (cA_{ij} - \tilde{\mathbf{u}}_i^\top \mathbf{v}_j)^2 \\ \text{subject to: } \mathbf{V} &= [\mathbf{v}_1, \dots, \mathbf{v}_{n_d}]^\top \in \{-1, +1\}^{n_d \times c}, \\ \tilde{\mathbf{u}}_i &= \mathbf{tanh}(e(\mathbf{y}_i; \phi)). \end{aligned} \quad (4-10)$$

与 DDSH 方法相同，本章提出的 ADSH 首先使用一个卷积神经网络作为基础网络架构，并在这个基础网络架构的基础上增加了一层全连接层作为深度特征学习架构以完成深度特征表示学习。深度神经网络的设计策略和 DDSH 的设计策略相同，不再赘述。

现实场景中，训练过程中可能只能获得一个数据库样本集 \mathbf{Z} 。在这种情况下，ADSH 从数据库样本集中随机选择一个子集来构造查询样本集。具体来说，令 $\mathbf{Y} = \mathbf{Z}^\Psi$ ，其中， Ψ 表示索引集合， \mathbf{Z}^Ψ 表示由 Ψ 索引的数据库样本集合，即： $\mathbf{Z}^\Psi = \{\mathbf{z}_i \mid \forall i \in \Psi, \mathbf{z}_i \in \mathbf{Z}\}$ 。由于 $\mathbf{Z}^\Psi \subset \mathbf{Z}$ ， \mathbf{Z}^Ψ 中的数据样本有两种编码表示，即：对于查询样本 $\mathbf{y}_j \in \mathbf{Y}$ ， \mathbf{v}_j 和 $\mathbf{tanh}(e(\mathbf{y}_j; \phi))$ 均为查询样本 \mathbf{y}_j

的编码表示。因此，对于具有两种表示的查询样本，ADSH 考虑增加一个两种编码表示的量化损失项，将优化问题 (4-10) 重写为：

$$\begin{aligned} \min_{\mathbf{V}, \phi} \mathcal{J}(\mathbf{V}, \phi) &= \sum_{i \in \Psi} \sum_{j=1}^{n_d} (cA_{ij} - \tilde{\mathbf{u}}_i \mathbf{v}_j)^2 \\ &\quad + \gamma \sum_{i \in \Psi} (\mathbf{v}_i - \tilde{\mathbf{u}}_i)^2 \\ \text{subject to: } \mathbf{V} &= [\mathbf{v}_1, \dots, \mathbf{v}_{n_d}]^\top \in \{-1, +1\}^{n_d \times c}, \\ \forall i \in \Psi, \tilde{\mathbf{u}}_i &= \mathbf{tanh}(\mathbf{y}_i; \theta). \end{aligned} \quad (4-11)$$

这里， $\gamma > 0$ 表示超参数。

实际中，如果给定了查询样本集和数据库样本集，可使用优化问题 (4-10) 建模，否则，如果只给定了数据库样本集，可从数据库样本集中采样构成查询集并使用优化问题 (4-11) 建模。

4.4.2 学习算法

本节只给出了优化问题 (4-11) 的学习算法，优化问题 (4-10) 的学习算法可根据本节提出的算法并设置 $\gamma = 0$ 得到。具体来说，ADSH 设计了交替优化算法来学习参数 $\{\phi, \mathbf{V}\}$ 。学习过程中，依次优化其中的一个参数，固定其他参数不变。

4.4.2.1 固定参数 ϕ ，学习二值哈希编码 \mathbf{V}

首先将优化问题 (4-11) 重写为矩阵形式，可得：

$$\begin{aligned} \min_{\mathbf{V}} \mathcal{J}(\mathbf{V}) &= \|c\mathbf{A} - \tilde{\mathbf{U}}\mathbf{V}^\top\|_F^2 + \gamma\|\mathbf{V}^\Psi - \tilde{\mathbf{U}}\|_F^2 \\ &= \|\tilde{\mathbf{U}}\mathbf{V}^\top\|_F^2 - 2\text{tr}(\mathbf{V}^\top \mathbf{A}^\top \tilde{\mathbf{U}}) + c^2\|\mathbf{A}\|_F^2 \\ &\quad + \gamma(\|\mathbf{V}^\Psi\|_F^2 - 2\text{tr}(\mathbf{V}^\Psi \tilde{\mathbf{U}}^\top) + \|\tilde{\mathbf{U}}\|_F^2) \\ &= \|\tilde{\mathbf{U}}\mathbf{V}^\top\|_F^2 - 2\text{tr}(\mathbf{V}^\top \mathbf{A}^\top \tilde{\mathbf{U}}) - 2\gamma\text{tr}(\mathbf{V}^\Psi \tilde{\mathbf{U}}^\top) \\ &\quad + c^2\|\mathbf{A}\|_F^2 + \gamma c|\Psi| + \gamma\|\tilde{\mathbf{U}}\|_F^2 \\ &= \|\tilde{\mathbf{U}}\mathbf{V}^\top\|_F^2 - 2\text{tr}(\mathbf{V}^\top \mathbf{A}^\top \tilde{\mathbf{U}}) - 2\gamma\text{tr}(\mathbf{V}^\Psi \tilde{\mathbf{U}}^\top) + \text{const} \\ \text{subject to: } \mathbf{V} &\in \{-1, +1\}^{n_d \times c}. \end{aligned} \quad (4-12)$$

这里, \mathbf{const} 表示与 \mathbf{V} 无关的常量, $\Psi = \{i_1, \dots, i_{n_q}\}$, $\tilde{\mathbf{U}} = [\tilde{\mathbf{u}}_{i_1}, \dots, \tilde{\mathbf{u}}_{i_{n_q}}]^\top \in \{-1, +1\}^{n_q \times c}$, \mathbf{V}^Ψ 表示由索引集合 Ψ 索引的数据库样本的二值哈希编码构成的矩阵, 即: $\mathbf{V}^\Psi = [\mathbf{v}_{i_1}, \dots, \mathbf{v}_{i_{n_q}}]^\top$.

定义 $\bar{\mathbf{U}} = [\bar{\mathbf{u}}_1, \dots, \bar{\mathbf{u}}_{n_d}]^\top \in \mathbb{R}^{n_d \times c}$, 其中 $\bar{\mathbf{u}}_i$ 定义如下:

$$\bar{\mathbf{u}}_i = \begin{cases} \tilde{\mathbf{u}}_i, & \text{if } i \in \Psi, \\ \mathbf{0}_c, & \text{otherwise.} \end{cases} \quad (4-13)$$

这里, $\mathbf{0}_c$ 表示元素全为 0 的 c 维向量。然后, 问题 (4-11) 可重写为:

$$\begin{aligned} \min_{\mathbf{V}} \mathcal{J}(\mathbf{V}) &= \|\tilde{\mathbf{U}}\mathbf{V}^\top\|_F^2 - 2\text{tr}(\mathbf{V}[c\tilde{\mathbf{U}}^\top\mathbf{A} + \gamma\bar{\mathbf{U}}^\top]) + \mathbf{const} \\ &= \|\tilde{\mathbf{U}}\mathbf{V}^\top\|_F^2 + \text{tr}(\mathbf{V}\mathbf{Q}^\top) + \mathbf{const} \\ \text{subject to: } \mathbf{V} &\in \{-1, +1\}^{n_d \times c}. \end{aligned} \quad (4-14)$$

这里, $\mathbf{Q} = -2c\mathbf{A}^\top\tilde{\mathbf{U}} - 2\gamma\bar{\mathbf{U}}$ 。

然后, 本节设计了一种逐比特优化的策略来学习二值哈希编码 \mathbf{V} 。具体来说, 依次优化 \mathbf{V} 的某一行二值哈希编码时, 固定其他比特不变。这里, 使用 \mathbf{V}_{*k} 表示 \mathbf{V} 的第 k 列, 使用 $\hat{\mathbf{V}}_k$ 表示矩阵 \mathbf{V} 除去第 k 后剩余的列组成的矩阵。使用 \mathbf{Q}_{*k} 表示 \mathbf{Q} 的第 k 列, 使用 $\hat{\mathbf{Q}}_k$ 表示矩阵 \mathbf{Q} 除去第 k 后剩余的列组成的矩阵。使用 $\tilde{\mathbf{U}}_{*k}$ 表示 $\tilde{\mathbf{U}}$ 的第 k 列, 使用 $\hat{\mathbf{U}}_k$ 表示矩阵 $\tilde{\mathbf{U}}$ 除去第 k 后剩余的列组成的矩阵。为了学习参数 \mathbf{V}_{*k} , 将 $\mathcal{J}(\mathbf{V}_{*k})$ 重写如下:

$$\begin{aligned} \mathcal{J}(\mathbf{V}_{*k}) &= \|\mathbf{V}\tilde{\mathbf{U}}^\top\|_F^2 + \text{tr}(\mathbf{V}\mathbf{Q}^\top) + \mathbf{const} \\ &= \left\| \begin{bmatrix} \mathbf{V}_{*k}, \hat{\mathbf{V}}_k \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{U}}_{*k}^\top \\ \hat{\mathbf{U}}_k^\top \end{bmatrix} \right\|_F^2 + \text{tr} \left(\begin{bmatrix} \mathbf{V}_{*k}, \hat{\mathbf{V}}_k \end{bmatrix} \begin{bmatrix} \mathbf{Q}_{*k}^\top \\ \hat{\mathbf{Q}}_k^\top \end{bmatrix} \right) + \mathbf{const} \\ &= \|\mathbf{V}_{*k}\tilde{\mathbf{U}}_{*k}^\top + \hat{\mathbf{V}}_k\hat{\mathbf{U}}_k^\top\|_F^2 + \text{tr}(\mathbf{V}_{*k}\mathbf{Q}_{*k}^\top + \hat{\mathbf{V}}_k\hat{\mathbf{Q}}_k^\top) + \mathbf{const} \\ &= \text{tr}(\mathbf{V}_{*k}[2\tilde{\mathbf{U}}_{*k}^\top\hat{\mathbf{U}}_k\hat{\mathbf{V}}_k^\top + \mathbf{Q}_{*k}^\top]) + n\text{tr}(\tilde{\mathbf{U}}_{*k}^\top\tilde{\mathbf{U}}_{*k}) + \|\hat{\mathbf{V}}_k\hat{\mathbf{U}}_k^\top\|_F^2 + \text{tr}(\hat{\mathbf{V}}_k\hat{\mathbf{Q}}_k^\top) \\ &= \text{tr}(\mathbf{V}_{*k}[2\tilde{\mathbf{U}}_{*k}^\top\hat{\mathbf{U}}_k\hat{\mathbf{V}}_k^\top + \mathbf{Q}_{*k}^\top]) + \mathbf{const}. \end{aligned}$$

然后可得损失函数 $\mathcal{J}(\mathbf{V}_{*k})$ 关于参数 \mathbf{V}_{*k} 的优化问题:

$$\begin{aligned} \min_{\mathbf{V}_{*k}} \mathcal{J}(\mathbf{V}_{*k}) &= \text{tr}(\mathbf{V}_{*k}[2\tilde{\mathbf{U}}_{*k}^\top\hat{\mathbf{U}}_k\hat{\mathbf{V}}_k^\top + \mathbf{Q}_{*k}^\top]) + \mathbf{const} \\ \text{subject to: } \mathbf{V}_{*k} &\in \{-1, +1\}^{n_d}. \end{aligned} \quad (4-15)$$

由此可得，优化问题 (4-15) 的最优解为：

$$\mathbf{V}_{*k} = -\text{sign}(2\hat{\mathbf{V}}_k\hat{\mathbf{U}}_k^\top\tilde{\mathbf{U}}_{*k}^\top + \mathbf{Q}_{*k}). \quad (4-16)$$

4.4.2.2 固定二值哈希编码 \mathbf{V} ，学习参数 ϕ

ADSH 使用反向传播算法来学习参数 ϕ 。具体来说，随机选择包含 n_b 个样本的小批量样本集合来构造小批量样本集合。根据这个样本集合，计算损失函数 $\mathcal{J}(\phi)$ 关于参数 ϕ 的梯度。给定查询样本 \mathbf{y}_{i_p} ，损失函数关于神经网络输出 $\mathbf{o}_{i_p} = e(\mathbf{y}_{i_p}; \phi)$ 的梯度为：

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial \tilde{\mathbf{u}}_{i_p}} &= 2 \sum_{j=1}^{n_d} [\tilde{\mathbf{u}}_{i_p}^\top \mathbf{v}_j - cA_{i_p j}] \mathbf{v}_j + 2\gamma(\tilde{\mathbf{u}}_{i_p} - \mathbf{v}_{i_p}) \\ \frac{\partial \mathcal{J}}{\partial \mathbf{o}_{i_p}} &= \frac{\partial \mathcal{J}}{\partial \tilde{\mathbf{u}}_{i_p}} \odot (1 - \tilde{\mathbf{u}}_{i_p}^2) \\ &= (1 - \tilde{\mathbf{u}}_{i_p}^2) \odot \left[2 \sum_{j=1}^{n_d} (\tilde{\mathbf{u}}_{i_p}^\top \mathbf{v}_j - cA_{i_p j}) \mathbf{v}_j + 2\gamma(\tilde{\mathbf{u}}_{i_p} - \mathbf{v}_{i_p}) \right]. \end{aligned} \quad (4-17)$$

根据 $\frac{\partial \mathcal{J}}{\partial \tilde{\mathbf{u}}_{i_p}}$ 和 $\frac{\partial \mathcal{J}}{\partial \mathbf{o}_{i_p}}$ ，可以使用链式法则来计算 $\frac{\partial \mathcal{J}}{\partial \phi}$ 。

本章提出的 ADSH 模型的算法可以简单地概括在算法 4.2 中。

4.4.3 样本外扩展

训练结束后，可以使用学习得到的深度神经网络来为任一不属于训练集中的查询样本生成二值哈希编码表示。具体来说，当给定的查询样本不属于训练集时，即： $\mathbf{y}_q \notin \mathbf{Y}$ ，使用如下的公式来生成二值哈希编码：

$$\mathbf{u}_q = \text{sign}(e(\mathbf{y}_q; \phi)).$$

4.4.4 复杂度分析

本章提出的 ADSH 方法的复杂度为： $\mathcal{O}(S_{out}S_{in}[(n_d + 2)n_q c + (n_q + 1)n_d c^2 + (c(n_d + n_q) - n_q)c])$ 。现实应用中， S_{out} 、 S_{in} 、 n_q 和 c 的值通常远小于数据库集大小 n_d 。因此，ADSH 方法的复杂度为 $\mathcal{O}(n_d)$ 。对于对称深度哈希学习方法，如果全部数据库样本都被用于训练，则其复杂度通常为 $\mathcal{O}(n_d^2)$ 。因此，对称深

算法 4.2 AD SH 模型学习算法

输入:

数据库样本集合 $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^{n_d}$, 相似度矩阵 $\mathbf{A} \in \{-1, 1\}^{n_d \times n_d}$, 二值哈希编码长度 c 。

输出:

神经网络参数 ϕ , 数据库二值哈希编码 \mathbf{V} 。

- 1: **初始化:** 初始化深度神经网络参数 ϕ , \mathbf{V} , n_b , S_{out} 和 S_{in} 。
- 2: **for** $w = 1 \rightarrow S_{out}$ **do**
- 3: 随机采样生成 Ψ , 以此构建 \mathbf{Y} 和相似度矩阵 $\mathbf{A} = \mathbf{A}^\Psi$;
- 4: **for** $t = 1 \rightarrow S_{in}$ **do**
- 5: **for** $s = 1, 2, \dots, \lceil n_q/n_b \rceil$ **do**
- 6: 从 \mathbf{Y} 采 n_b 个样本构成小批量集合;
- 7: 通过正向传播为小批量集合中的所有样本计算 $e(\mathbf{y}_i; \phi)$;
- 8: 根据公式 (4-17) 计算梯度;
- 9: 使用反向传播算法更新参数 ϕ ;
- 10: **end for**
- 11: **for** $k = 1 \rightarrow c$ **do**
- 12: 根据公式 (4-16) 更新 \mathbf{V}_{*k} 。
- 13: **end for**
- 14: **end for**
- 15: **end for**

度哈希学习方法的训练过程效率过低, 通常需要在采样的训练集上进行训练。在每次迭代中, 对称哈希学习方法需要使用 n_d 个样本通过神经网络, 而 AD SH 方法仅需要使用 n_q 个样本通过神经网络。因此, AD SH 方法比对称深度哈希学习方法高效。

4.5 DDSH 方法的实验验证

本节通过在两个图片数据集上的实验验证 DDSH 方法的有效性。所有的实验均在一台包含 NVIDIA M40 显卡的服务器上完成。该服务器的 CPU 型号为: Intel(R) Xeon(R) CPU E5-2680V4@2.4G 14 cores, 内存为 746G。

4.5.1 实验设置

4.5.1.1 数据集

本章使用 CIFAR10^[108] 和 NUSWIDE^[109] 数据集进行实验。两个数据集的介绍可以参考第二章第 2.2 节。对于 NUSWIDE 数据集, 本章仅使用图片数据

及其标签进行实验。并选择类别样本最多的 10 个类别对应的样本进行实验，对应的样本总数为 186,577。

4.5.1.2 评价标准与对比方法

对于 CIFAR10 数据集，实验中随机选择每类 100 张一共 1000 张图片样本作为查询样本集，剩余的数据样本作为数据库样本。另外，实验中从数据库样本集中选择每类 500 张一共 5000 张图片作为采样训练样本集。同时，如果两个图片属于同一个类别，则定义他们相似，否则，如果两张图片不属于同一个类别，则定义他们不相似。对于 NUSWIDE 数据集，实验中随机选择 1876 张图片样本作为查询样本集，剩余的样本作为数据库样本。另外，实验中从数据库样本集中随机选择了 5000 个图片样本作为采样训练样本集。同时，如果两张图片共享至少一个类别标签，则定义他们相似，否则，如果两张图片没有共享的类别标签，则定义他们不相似。

本章设计了一个实验来证明本文提出的 DDSH 方法的有效性。该实验选取了八个哈希学习方法作为基准方法，包括一个无监督哈希学习方法，即：ITQ^[21]；四个非深度监督哈希学习方法，即：LFH^[75]、FastH^[60]、SDH^[85]和 COSDISH^[44]；三个深度单模态哈希学习方法，即：DHN^[65]、DSH^[70]和 DPSH^[22]。除了 ITQ 之外的所有方法均使用采样训练集进行训练。对于 DDSH 方法，设置小批量集合大小 $n_b = 128$ ，采样集大小 $|\Omega| = 100$ ，迭代次数 $T_{in} = 50$ ， $T_{out} = 3$ 。对于其他对比方法，实验中根据原论文作者的建议选择超参数。对于 SDH 方法，从训练集中随机选择了 1000 个样本作为核基。对于 LFH、FastH 和 COSDISH 方法，实验中根据作者建议使用提升决策树来学习哈希函数。对于所有的深度哈希学习方法，使用原始图片作为输入，并将图片大小调整到 224×224 。同时，为了对比公平，实验中采用 CNNF^[119] 作为基础网络架构。在 CNNF 网络的基础上，增加一层全连接层作为新的深度神经网络来完成深度特征学习，该网络结构的配置细节展示在表 4-1 中。从表 4-1 中可以看到，该深度神经网络结构包含 5 个卷积层和 3 个全连接层。其中，卷积层标记为：“conv1”-“conv5”，全连接层标记为：“full6”-“full8”。同时，表 4-1 中使用以下几个方面对深度神经网络进行描述：

- “filter”：卷积核，表示为“卷积核个数”×“卷积核大小”×“卷积核大小”；
- “stride”：表示卷积核在移动时的移动幅度；
- “padding”：表示进行卷积操作时，在图片四周添加的像素数目；

- “LRN”：表示是否进行局部归一化操作^[61]；
- “pooling”：表示池化区域大小；
- 全连接层参数，如“4096”等，表示该全连接层的隐层单元数目，同时该数也是该全连接层的输出的维度；

此外，该深度神经网络的前 7 层的激活函数均使用修正线性单元^[61]（Rectified Linear Unit, 简称 ReLU）。在初始化过程中，该深度神经网络前 7 层的参数都使用已经在 ImageNet 数据集上进行过预训练的深度神经网络的参数进行初始化。

表 4-1: DDSH 方法中使用的深度神经网络配置细节

网络层	网络结构配置
conv1	filter: $64 \times 11 \times 11$, stride: 4, padding: 0, LRN, pooling: $\times 2$
conv2	filter: $256 \times 5 \times 5$, stride: 1, padding: 2, LRN, pooling: $\times 2$
conv3	filter: $256 \times 3 \times 3$, stride: 1, padding: 1
conv4	filter: $256 \times 3 \times 3$, stride: 1, padding: 1
conv5	filter: $256 \times 3 \times 3$, stride: 1, padding: 1, pooling: $\times 2$
full6	4096
full7	4096
full8	二值哈希编码长度: c

为了验证本章提出的 DDSH 方法的有效性，本章汇报了 DDSH 方法和所有基准方法在返回序列截断位置 K 的平均查准率均值 $cMAP(K)$ 。平均查准率均值 $cMAP(K)$ 的计算参见第二章第 2.2 小节。

4.5.2 性能对比

表 4-2 和表 4-3 分别汇报了在 CIFAR10 数据集和 NUSWIDE 数据集上本章提出的 DDSH 方法与选取的八个基准方法的平均查准率均值 $cMAP(K)$ ，其中， K 的值设定为整个数据库样本集的大小。在表 4-2 和表 4-3 中，最好的 $cMAP(K)$ 使用加粗字体标注。从表 4-2 和表 4-3 可以看出，随着二值哈希编码长度的增加，DDSH 方法的检索精度越来越高。在 CIFAR10 数据集和 NUSWIDE 数据集上，本章提出的 DDSH 方法在所有的情况下都取得了更高的检索精度。

表 4-2: DDSH 方法和基准方法在 CIFAR10 数据集上的平均查准率均值

方法	比特长度			
	12bits	24bits	32bits	48bits
ITQ	0.2580	0.2725	0.2834	0.2936
LFH	0.4009	0.6047	0.6571	0.6995
FastH	0.6202	0.6731	0.6870	0.7163
SDH	0.5200	0.6416	0.6577	0.6688
COSDISH	0.6085	0.6827	0.6959	0.7158
DHN	0.6725	0.7107	0.7045	0.7135
DSH	0.6457	0.7492	0.7857	0.8113
DPSH	0.6844	0.7225	0.7396	0.7460
DDSH	0.7695	0.8289	0.8352	0.8194

表 4-3: DDSH 方法和基准方法在 NUSWIDE 数据集上的平均查准率均值

方法	比特长度			
	12bits	24bits	32bits	48bits
ITQ	0.5053	0.5037	0.5031	0.5054
LFH	0.7049	0.7549	0.7778	0.7936
FastH	0.7412	0.7830	0.7948	0.8085
SDH	0.7385	0.7616	0.7697	0.7720
COSDISH	0.7303	0.7643	0.7868	0.7993
DHN	0.7900	0.8101	0.8092	0.8180
DSH	0.7622	0.7940	0.7968	0.8081
DPSH	0.7882	0.8085	0.8167	0.8234
DDSH	0.7911	0.8165	0.8217	0.8259

4.5.3 有效性验证实验

本节进一步讨论了监督策略对深度哈希学习方法检索精度的影响。具体来说，本章设计了四个方法来验证 DDSH 的有效性和不同监督策略对最终检索精度的影响。第一个方法为 DDSH 的一个变体，记为 DDSH0，表示使用与 DDSH 方法同样的模型与优化算法，与 DDSH 不同的是，在进行深度特征学习时，DDSH0 仅更新最后一层全连接层的参数。这意味着 DDSH0 方法不进行深度特征学习。第二个方法为 COSDISH-Linear，这个方法首先使用 COSDISH^[44] 模型学习二值哈希编码 $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^n$ ，然后使用线性回归函数学习哈希函数。第三个方法为 COSDISH-CNN，这个方法同样首先使用 COSDISH 模型学习二值哈希编码，然后使用与 DDSH 模型的深度特征学习架构相同的网络 $e(\cdot)$ 将数据投影到 \mathbb{R}^c 空间，并优化如下的损失函数来学习哈希函数：

$$\mathcal{J}(\theta) = \frac{1}{n} \sum_{i=1}^n (\mathbf{b}_i - e(\mathbf{x}_i; \theta))^2.$$

最后一个方法为 DDSH-MAC，表示使用 Raziperchikolaei 等人^[121] 提出的辅助坐标方法（Method of Auxiliary Coordinate, 简称 MAC）方法解问题 (4-2)。具体来说，DDSH-MAC 尝试解如下优化问题：

$$\begin{aligned} \min_{\theta, \mathbf{B}} \mathcal{J}(\theta, \mathbf{B}) &= \sum_{i \in \Omega} \sum_{j=1}^n (\mathbf{b}_i^\top \mathbf{b}_j - cS_{ij})^2 + \nu \sum_{i=1}^n (\mathbf{b}_i - e(\mathbf{x}_i; \theta))^2 \\ &= \sum_{i \in \Omega} \sum_{j \in \Gamma} (\mathbf{b}_i^\top \mathbf{b}_j - cS_{ij})^2 + \sum_{i, j \in \Omega} (\mathbf{b}_i^\top \mathbf{b}_j - cS_{ij})^2 \\ &\quad + \nu \sum_{i=1}^n (\mathbf{b}_i - e(\mathbf{x}_i; \theta))^2 \end{aligned} \quad (4-18)$$

$$\text{subject to: } \mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_n]^\top \in \{-1, +1\}^{n \times c}.$$

实验结果展现在表 4-4 中。在表 4-4 中，最好的检索精度使用加粗字体标注。从表 4-4 中可以看到，与 DDSH0 相比，DDSH 方法能达到更高的检索精度，由此证明了哈希学习中深度特征学习的重要性。与 COSDISH-Linear 相比，COSDISH-CNN 方法能达到更高的检索精度，由此证明了使用非线性哈希函数能提升哈希学习模型的检索精度。与 COSDISH-CNN 和 DDSH-MAC 相比，DDSH 方法能达到更高的检索精度，由此证明了同时直接监督深度特征学习过程和二值哈希编码学习过程能达到更高的检索精度。

表 4-4: DDSH 方法有效性验证实验

方法	比特长度			
	12bits	24bits	32bits	48bits
DDSH0	0.5793	0.6387	0.6536	0.6800
COSDISH-Linear	0.2123	0.2345	0.2578	0.2723
COSDISH-CNN	0.3742	0.4773	0.4678	0.5153
DDSH-MAC	0.4121	0.5060	0.5276	0.5335
DDSH	0.7695	0.8289	0.8352	0.8194

4.6 ADSSH 方法的实验验证

本节通过在两个图片数据集上的实验验证 ADSSH 方法的有效性。实验中所使用的硬件配置与 DDSH 方法的实验设置相同。

4.6.1 实验设置

4.6.1.1 数据集

本章使用 CIFAR10^[108] 和 MSCOCO^[144] 数据集进行实验。两个数据集的介绍可以参考第二章第 2.2 节。

4.6.1.2 评价标准与对比方法

在这个实验中，CIFAR10 数据集的划分和相似样本对的定义与 DDSH 方法的实验设置相同。对于 MSCOCO 数据集，实验中选择从类别样本最多的 20 个类别对应的样本集中选择每类 250 张一共 5000 张图片样本作为查询集，剩下的图片样本作为数据库样本集。另外，实验中从数据库样本随机选择 10000 张图片作为采样训练样本集。同时，如果两张图片共享至少一个类别标签，则定义他们相似，否则，如果两张图片没有共享的类别标签，则定义他们不相似。

本章设计了两个实验来证明 ADSSH 方法的有效性。第一个实验选取了六个哈希学习方法作为基准方法，包括：ITQ^[21]、LFH^[75]、SDH^[85]、COSDISH^[44]、核化非对称离散图哈希^[122] (Kernelized Asymmetric Discrete Graph

Hashing, 简称 KADGH) 和 DPSH^[22]。其中, KADGH 是基于非对称哈希的非深度单模态哈希学习方法。由于 ADSH 复杂度为 $\mathcal{O}(n_d)$, 所以实验中选取了全部数据库样本进行实验。为了与 DDSH 实验中的符号区分, 使用全部数据库样本进行训练的监督哈希学习方法使用后缀“-D”标记, 例如: LFH-D 等。为了对比公平, 不再选取复杂度为 $\mathcal{O}(n_d^2)$ 的 FastH 作为基准方法。对于 ADSH, 设置 $\gamma = 200$, $S_{out} = 50$, $S_{in} = 3$, $|\Psi| = 2000$ 。对于其他对比方法, 实验中根据原论文作者的建议选择超参数。对于深度单模态哈希学习方法, 由于训练低效, 第一个实验仅选取了代表性的深度哈希学习方法 DPSH 进行对比。所有方法均使用全部数据库样本进行训练。对于 SDH 和 KADGH 方法, 选择了 1000 个样本作为核基。由于 KADGH 无法使用多标签数据集进行训练, KADGH 仅使用 CIFAR10 进行实验。对于 DPSH, 为了对比公平, 实验中采用在 ImageNet 数据集上预训练过的 CNNF^[119] 作为基础网络架构。在 CNNF 网络的基础上, 增加一层全连接层作为新的深度神经网络来完成深度特征学习。该网络结构与 DDSH 方法中采用的深度神经网络结构相同, 同时网络结构的前 7 层初始化同样使用了已经在 ImageNet 数据集上进行预训练的网络进行初始化。对于所有的深度哈希学习方法, 使用原始图片作为输入, 并将图片大小调整到 224×224 。

同时本章设计了另外一个实验来证明 ADSH 训练的高效性。这个实验选取了三个深度哈希学习方法作为基准方法, 包括: DHN^[65]、DSH^[70]、DPSH^[22]。这个实验中不仅使用数据库样本进行训练, 还使用了采样训练集进行训练。为了区分这两种场景, 使用形如“DHN”和“DHN-D”这样的标记来区分两种不同的训练策略, 前者表示使用采样训练集进行训练, 后者表示使用数据库样本进行训练。对于基准方法, 由于使用全部数据库样本进行训练低效, 因此本实验仅在 MSCOCO 数据集上进行。

本章汇报了在返回序列截断位置 K 的平均查准率均值 $\text{cMAP}(K)$ 以验证 ADSH 方法的有效性。同时, 为了验证 ADSH 方法训练的高效性, 本节汇报了随着训练的进行, 平均查准率均值 $\text{cMAP}(K)$ 的变化。

4.6.2 性能对比

4.6.2.1 ADSH 与基准方法精度对比

表 4-5 和表 4-6 分别汇报了在 CIFAR10 和 MSCOCO 数据集上本章提出的 ADSH 方法与选取的六个基准方法的平均查准率均值, 其中, K 的值设定

为整个数据库样本集的大小。在表 4-5 和表 4-6 中，最好的平均查准率均值 $cMAP(K)$ 使用加粗字体标注。从表 4-5 和表 4-6 可以看出，随着比特长度的增加，本章提出的 ADSSH 方法的检索精度越来越高。此外，从表 4-5 和表 4-6 还可以看出，在 CIFAR10 数据集和 MSCOCO 数据集上，本章提出的 ADSSH 方法在所有的情况下都取得了更高的检索精度。

表 4-5: ADSSH 方法和基准方法在 CIFAR10 数据集上的平均查准率均值

方法	比特长度			
	12bits	24bits	32bits	48bits
ITQ	0.2580	0.2725	0.2834	0.2936
LFH-D	0.5091	0.7267	0.7712	0.8333
SDH-D	0.6616	0.8466	0.8501	0.8501
COSDISH-D	0.8544	0.8700	0.8802	0.8771
KADGH-D	0.8606	0.8710	0.8759	0.8749
DPSH-D	0.8899	0.9078	0.9060	0.9141
ADSSH	0.9034	0.9164	0.9205	0.9212

表 4-6: ADSSH 方法和基准方法在 MSCOCO 数据集上的平均查准率均值

方法	比特长度			
	12bits	24bits	32bits	48bits
ITQ	0.6338	0.6326	0.6308	0.6338
LFH-D	0.7408	0.7729	0.8063	0.8165
SDH-D	0.7644	0.7724	0.7708	0.7711
COSDISH-D	0.6976	0.7685	0.8052	0.7943
DPSH-D	0.8194	0.8434	0.8476	0.8379
ADSSH	0.8388	0.8590	0.8633	0.8651

4.6.2.2 ADSSH 与基准方法训练效率对比

实验中进一步对比了随着训练的进行，平均查准率均值的变化情况。图4-1给出了在 MSCOCO 数据集上的实验结果。从图4-1中可以看出，对于同一种深度哈希学习方法，与使用采样训练集进行训练相比，使用数据库集合训练能达到更高的检索精度，但训练过程更低效。与现有的深度哈希学习方法相比，ADSSH 方法能在最短时间内达到更高的检索精度。

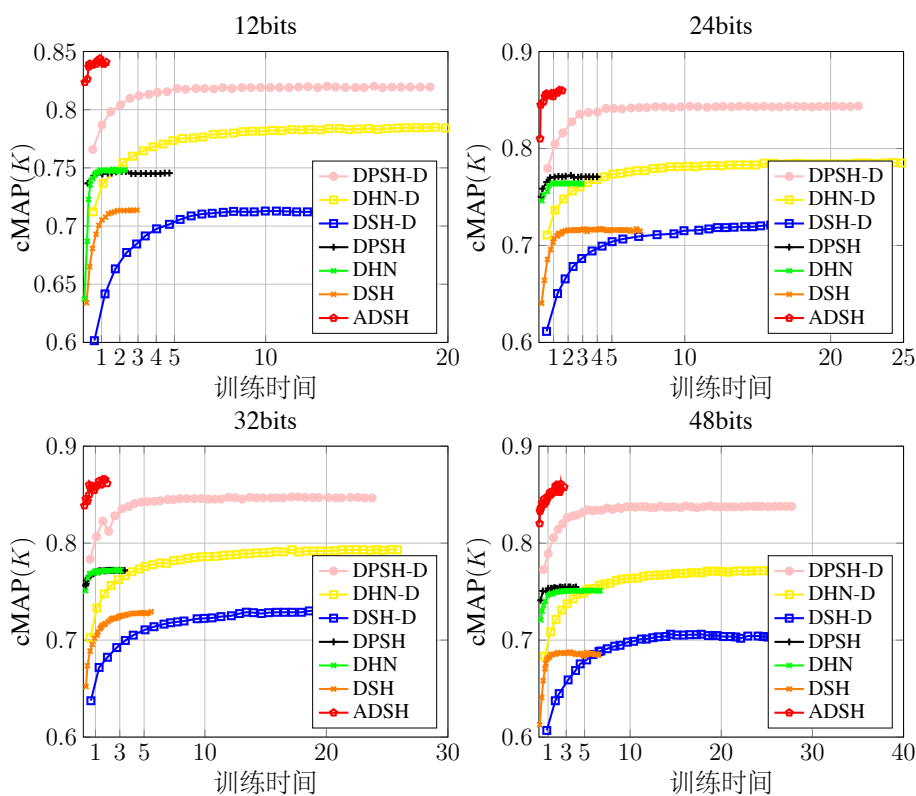


图 4-1: ADSSH 方法和基准方法在 MSCOCO 数据集上的训练时间

4.6.2.3 ADSSH 和 DDSH 的对比

本节在 CIFAR10 数据集上对比了本章提出的两个方法 ADSSH 方法和 DDSH 方法。实验结果展示在表4-7中。在表4-7中，最好的平均查准率均值 $cMAP(K)$ 使用加粗字体标注，训练时间的单位为秒。从表4-7中可以看到，当 DDSH 方法在采样训练集上进行训练时，虽然 DDSH 方法的训练时间和 ADSSH 方法的训练时间接近，但 DDSH 方法的检索精度低于 ADSSH 方法的检索精度。当 DDSH 方法采用全部数据库样本进行训练时，DDSSH 方法的检索精度在 12

比特时低于 ADSH 方法的检索精度，在其余比特时高于 ADSH 方法的检索精度。同时，当 DDSH 方法采用全部数据库样本进行训练时，DDSH 的训练过程没有 ADSH 方法高效。在实际应用中，如果要求高效地训练，可以选择 ADSH 方法建模。如果有足够资源进行建模或者对训练效率没有太高的要求，可以使用 DDSH 方法进行建模并使用足够的训练集进行训练。

表 4-7: ADSH 方法和 DDSH 方法在 CIFAR10 数据集上的对比

	方法	DDSH	DDSH-D	ADSH
	训练集大小	5000	59000	59000
12bit	cMAP	0.7695	0.8938	0.9034
	训练时间	7.47×10^3	3.76×10^5	2.36×10^3
24bit	cMAP	0.8289	0.9379	0.9164
	训练时间	7.77×10^3	3.79×10^5	3.63×10^3
32bit	cMAP	0.8352	0.9463	0.9205
	训练时间	1.01×10^4	3.83×10^5	4.60×10^3
48bit	cMAP	0.8194	0.9435	0.9212
	训练时间	1.34×10^4	3.83×10^5	6.31×10^3

4.6.3 超参数实验

本节讨论了 ADSH 方法的超参数敏感性。对于 ADSH 方法而言，最重要的超参数为 γ 和采样集合大小 $|\Psi|$ 。图 4-2 (a) 中给出了超参数 γ 敏感性实验。随着超参数 γ 的增大，检索精度先变好后变差。当 $1 \leq \gamma \leq 500$ ，ADSH 方法对超参数 γ 不敏感。图 4-2 (b) 汇报了超参数 $|\Psi|$ 的影响。从图 4-2 可以看到，随着 $|\Psi|$ 的增大，检索精度变得越来越好。然而， $|\Psi|$ 越大，训练过程中所需求的计算开销就越大。实际应用中， $|\Psi|$ 的选择可以根据对检索精度和计算开销的需求来决定。

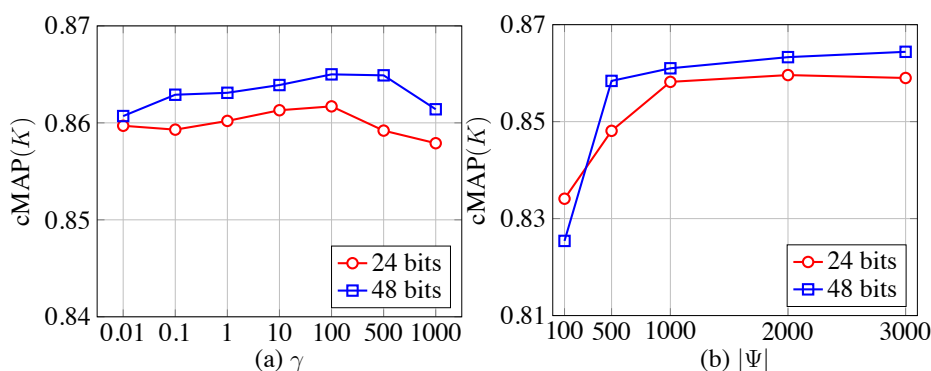


图 4-2: ADSH 方法对超参数 γ 和 $|\Psi|$ 的敏感性实验

4.7 本章小结

本章提出一种深度离散监督哈希 DDSH。DDSH 方法可以使用监督信息同时直接监督二值哈希编码学习和深度特征学习。针对现有的深度哈希学习方法训练低效的问题，本章还提出一种非对称深度监督哈希学习方法 ADSH。通过使用非对称哈希建模，ADSH 设计了一种高效的学习算法，该算法能同时完成查询样本的深度特征学习和数据库样本的二值哈希编码学习。实验证明了与现有的深度哈希学习方法相比，DDSH 方法能达到更高的检索精度。与除 DDSH 方法外的对称深度哈希学习方法相比，ADSH 方法能在最短时间内达到更高的检索精度。与 DDSH 方法相比，ADSH 方法的训练更加高效。

本章的工作已总结成文并发表：

- JIANG Q-Y, CUI X, LI W-J. Deep Discrete Supervised Hashing[J]. IEEE Transactions on Image Processing, 2018, 27(12): 5996 – 6009. (CCF-A 类期刊)
- JIANG Q-Y, LI W-J. Asymmetric Deep Supervised Hashing[C] //Proceedings of the AAAI Conference on Artificial Intelligence. 2018: 3342 – 3349. (CCF-A 类会议)

第五章 非深度多模态离散哈希

5.1 引言

第三章和第四章介绍了本文在单模态哈希中提出的一些新型离散哈希学习方法。现实场景中，数据可能以多模态的形式出现。在多模态数据的检索任务中，多模态哈希学习方法^[48,56,69]得到了广泛的应用。在多模态哈希学习方法中，跨模态哈希学习方法^[48,53,54,57,69]的应用场景比多源哈希学习方法^[66,99]的应用场景更广阔，因此近年来受到了越来越多的关注。

在跨模态哈希中，为了避免由二值约束带来的困难，部分跨模态哈希学习方法选择了在训练过程中直接丢弃二值哈希编码的松弛策略。这类方法的检索精度也因采用松弛策略受损。另外一些方法采用了离散哈希学习策略。代表性的跨模态离散哈希学习方法包括：CCAITQ^[21]、ACQ^[69]、QCH^[52]、SCM^[51]和SePH^[53]等方法。CCAITQ、ACQ和QCH都是无监督跨模态离散哈希学习方法。CCAITQ是ITQ算法在跨模态哈希学习中的扩展。ACQ^[69]将降维过程和二值哈希编码学习过程融合到同一个框架中。ACQ采用邻域保护嵌入算法^[123]和CCA两种方法进行降维，同时优化降维后的实值特征和二值哈希编码的误差。QCH^[52]采用多模态数据降维后的特征间的余弦相似度来定义跨模态相似度，设计了一种能同时逼近跨模态相似度和学习二值哈希编码的跨模态哈希学习方法。SCM和SePH是监督跨模态离散哈希学习方法。SCM^[51]使用类别标签信息来逼近跨模态标签对相似度，设计了一种逐比特优化的离散哈希学习算法。SePH^[53]首先将跨模态相似度信息和二值哈希编码间的距离分别投影为概率分布，并通过优化两个分布之间的KL散度来学习二值哈希编码。由于要利用跨模态标签对相似度信息，SePH的复杂度达到了 $\mathcal{O}(n^2)$ ，其中， n 表示训练集样本数。计算资源有限时，SePH通常只能使用采样的训练集进行训练，其检索精度也因此受限。同时，由于高复杂度，SePH的训练十分低效。

为了提高非深度跨模态哈希学习方法检索精度，并解决其训练低效问题，本章提出一种基于离散隐因子模型的跨模态哈希学习方法（**Discrete Latent Factor Modal based Cross-Modal Hashing**，简称DLFH）。本章的主要贡献包括以

下四个方面:

- 本章提出的 DLFH 方法使用离散隐因子模型建模跨模态哈希学习问题。
- 本章提出的 DLFH 方法设计了一种直接学习二值哈希编码的离散优化算法。该离散优化算法可以在不丢弃二值约束的情况直接学习多模态数据的二值哈希编码。同时, 本章给出了所提出的离散优化算法的收敛性证明。
- 本章提出的 DLFH 方法设计了一种基于随机采样的离散优化方法, 使得 DLFH 方法的训练过程更加高效。
- 实验证明与之前的非深度跨模态哈希学习方法相比, 本章提出的 DLFH 能达到更高的检索精度。同时, DLFH 模型的训练也更加高效。

本章剩余部分组织如下: 第 5.2 节中介绍问题定义; 第 5.3 介绍本章提出的 DLFH 方法; 第 5.4 节通过在三个数据集上的实验验证本章提出的 DLFH 方法的有效性; 最后, 第 5.5 节对本章进行总结。

5.2 问题定义

本章使用图片模态和文本模态来阐述本章提出的方法。值得一提的是, 本章提出的方法可以很容易地应用到数据模态数目大于两个的场景中。本章分别使用 $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d_x}$ 和用 $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_n]^\top \in \mathbb{R}^{n \times d_y}$ 表示图片和文本模态的特征, 其中, d_x 和 d_y 分别表示图片和文本模态的特征维度。这里假设训练集中所有样本的两个模态的数据都已观测到。对于监督跨模态哈希学习方法, 跨模态相似度 $\mathbf{S} = \{S_{ij}\}_{i,j=1}^n$ 也已给定, 其中, $S_{ij} \in \{0, 1\}$, $S_{ij} = 0$ 表示图片 \mathbf{x}_i 和文本 \mathbf{y}_j 不相似, $S_{ij} = 1$ 表示图片 \mathbf{x}_i 和文本 \mathbf{y}_j 相似。

本章使用 $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_n]^\top \in \{-1, +1\}^{n \times c}$ 表示图片数据的二值哈希编码^①, 使用 $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_n]^\top \in \{-1, +1\}^{n \times c}$ 表示文本数据的二值哈希编码, 使用 $h(\cdot)$ 表示图片模态的哈希函数, 使用 $g(\cdot)$ 表示文本模态的哈希函数。跨模态哈希的目标是将图片数据 \mathbf{X} 和文本数据 \mathbf{Y} 表示成为二值哈希编码, 同时希望得到的二值哈希编码尽可能保持原始空间中的跨模态相似度。具体来说, 假如两个数据点在原始空间中相似, 即 $S_{ij} = 1$, 那么希望他们的二值哈希编码的海明距离尽量小, 反之, 如果两个数据点在原始空间中不相似, 即 $S_{ij} = 0$, 那么希望他们的二值哈希编码之间的海明距离尽量大。

^①这里, 二值哈希编码被表示成 -1 与 $+1$ 。学习结束后, -1 被替换成 0 , 得到用 0 和 1 表示的二值哈希编码。

5.3 基于离散隐因子模型的跨模态哈希学习方法 DLFH

本节从模型、学习算法、随机学习策略、样本外扩展和复杂度分析五个方面介绍本章提出的 DLFH 方法。

5.3.1 模型

给定图片数据样本 \mathbf{x}_i 和文本数据样本 \mathbf{y}_j 的二值哈希编码对 $(\mathbf{u}_i, \mathbf{v}_j)$ ，首先定义 Ω_{ij} 如下：

$$\Omega_{ij} = \frac{\lambda}{c} \mathbf{u}_i^\top \mathbf{v}_j,$$

其中， c 表示二值哈希编码长度， $\lambda > 0$ 是一个超参数。

基于 Ω_{ij} ，定义 A_{ij} 如下：

$$A_{ij} = \frac{1}{1 + e^{-\Omega_{ij}}}.$$

根据 A_{ij} 的定义，定义跨模态相似度 \mathbf{S} 的似然如下：

$$p(\mathbf{S} | \mathbf{U}, \mathbf{V}) = \prod_{i,j=1}^n p(S_{ij} | \mathbf{U}, \mathbf{V}),$$

其中， $p(S_{ij} | \mathbf{U}, \mathbf{V})$ 定义如下：

$$p(S_{ij} | \mathbf{U}, \mathbf{V}) = \begin{cases} A_{ij}, & \text{if } S_{ij} = 1, \\ 1 - A_{ij}, & \text{otherwise.} \end{cases}$$

因此，跨模态相似度 \mathbf{S} 的对数似然可以按如下方式推导：

$$\begin{aligned} \log p(\mathbf{S} | \mathbf{U}, \mathbf{V}) &= \log \left(\prod_{i,j=1}^n A_{ij}^{S_{ij}} (1 - A_{ij})^{1-S_{ij}} \right) \\ &= \sum_{i,j=1}^n [S_{ij} \log A_{ij} + (1 - S_{ij}) \log(1 - A_{ij})] \\ &= \sum_{i,j=1}^n [S_{ij} \Omega_{ij} - \log(1 + e^{\Omega_{ij}})]. \end{aligned}$$

DLFH 的目标为最小化跨模态相似度的负对数似然。因此，DLFH 尝试解如下优化问题：

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V}} \mathcal{J}(\mathbf{U}, \mathbf{V}) &= - \sum_{i,j=1}^n [S_{ij}\Omega_{ij} - \log(1 + e^{\Omega_{ij}})] \quad (5-1) \\ \text{subject to: } \mathbf{U}, \mathbf{V} &\in \{-1, +1\}^{n \times c}. \end{aligned}$$

首先，最小化跨模态相似度的负对数似然等价于最大化跨模态相似度的似然。当 $S_{ij} = 1$ ，最大化跨模态相似度的似然可以使得 \mathbf{u}_i 和 \mathbf{v}_j 之间的内积尽量大。根据海明距离和二值哈希编码内积的关系， \mathbf{u}_i 和 \mathbf{v}_j 之间的内积变大， \mathbf{u}_i 和 \mathbf{v}_j 之间的海明距离将变小，反之亦然。因此，通过最小化跨模态相似度的负对数似然，可以得到尽可能保持原始空间中跨模态相似度的二值哈希编码。

由于二值哈希编码 $\{\mathbf{U}, \mathbf{V}\}$ 定义在二值空间上，问题 (5-1) 是一个非凸的优化问题。由于丢弃二值约束将导致模型的检索精度变差，本章提出一种基于二值哈希编码优化的交替优化方法来直接学习二值哈希编码。

5.3.2 学习算法

本章在待学习变量 $\{\mathbf{U}, \mathbf{V}\}$ 之间采用交替优化的方法来解问题 (5-1)。在每个变量的优化过程中，首先固定其他变量，优化该变量。

5.3.2.1 优化 \mathbf{U} ，固定 \mathbf{V}

本文采用逐列优化的方式来优化 \mathbf{U} 。具体来说，算法依次优化 \mathbf{U} 中的某一行 \mathbf{U}_{*k} 。

首先，目标函数关于变量 \mathbf{U}_{*k} 的梯度及海森 (Hessian) 矩阵可根据如下公式计算：

$$\begin{aligned} \frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}} &= \frac{\lambda}{c} \sum_{j=1}^n [\mathbf{A}_{*j} - \mathbf{S}_{*j}] \mathbf{V}_{jk}, \quad (5-2) \\ \frac{\partial^2 \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k} \partial \mathbf{U}_{*k}^\top} &= \frac{\lambda^2}{c^2} \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_n \end{bmatrix}. \end{aligned}$$

这里, $\mathbf{A} = [A_{ij}]_{i,j=1}^n$, $a_i = \sum_{j=1}^n A_{ij}(1 - A_{ij})$ 。

定义 $\mathbf{U}_{*k}(t)$ 表示第 t 轮迭代时 \mathbf{U}_{*k} 的值。同时定义 $\frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t)$ 表示第 t 轮迭代时梯度 (5-2) 的值。定义如下矩阵:

$$\mathbf{H} = \frac{n\lambda^2}{4c^2} \mathbf{I}_n,$$

其中, \mathbf{I}_n 表示维度为 $n \times n$ 的单位矩阵。定义函数 $\tilde{\mathcal{J}}(\mathbf{U}_{*k})$ 为:

$$\begin{aligned} \tilde{\mathcal{J}}(\mathbf{U}_{*k}) &= \mathcal{J}(\mathbf{U}_{*k}(t)) + [\mathbf{U}_{*k} - \mathbf{U}_{*k}(t)]^\top \frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t) \\ &\quad + \frac{1}{2} [\mathbf{U}_{*k} - \mathbf{U}_{*k}(t)]^\top \mathbf{H} [\mathbf{U}_{*k} - \mathbf{U}_{*k}(t)] \\ &= \mathbf{U}_{*k}^\top \left[\frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H} \mathbf{U}_{*k}(t) \right] - \mathbf{U}_{*k}(t)^\top \frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t) + \mathcal{J}(\mathbf{U}_{*k}(t)) \\ &\quad + \frac{[\mathbf{U}_{*k}(t)]^\top \mathbf{H} [\mathbf{U}_{*k}(t)]}{2} - \frac{\lambda^2 n^3}{8c^2} \\ &= \mathbf{U}_{*k}^\top \left[\frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H} \mathbf{U}_{*k}(t) \right] + \text{const}, \end{aligned}$$

其中, const 表示与变量 \mathbf{U}_{*k} 无关的变量。

根据上述定义, 可得如下定理:

定理 5-1 函数 $\tilde{\mathcal{J}}(\mathbf{U}_{*k})$ 是函数 $\mathcal{J}(\mathbf{U}_{*k})$ 的一个上界。

为了证明上述定理, 首先引入如下引理:

引理 5-2 对于一个曲率有界的凸函数 $f(\mathbf{x})$, 如果对于一个矩阵 $\mathbf{D} \succ \mathbf{0}_{n \times n}$, 函数 $f(\mathbf{x})$ 的海森矩阵 $\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^\top}$ 满足:

$$\mathbf{D} \succ \frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x} \partial \mathbf{x}^\top},$$

则有:

$$f(\mathbf{y}) \leq f(\mathbf{x}) + (\mathbf{y} - \mathbf{x})^\top \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} + \frac{1}{2} (\mathbf{y} - \mathbf{x})^\top \mathbf{D} (\mathbf{y} - \mathbf{x}).$$

这里, $\mathbf{D} \succ \mathbf{0}_{n \times n}$ 表示矩阵 \mathbf{D} 是一个正定矩阵, $\mathbf{E} \succ \mathbf{F}$ 表示矩阵 $\mathbf{E} - \mathbf{F}$ 是一个正定矩阵。引理 (5-2) 的证明可参见 Kenneth 等人的论文^[124]。

下面根据引理 (5-2) 给出定理 (5-1) 的证明。

定理 (5-1) 的证明: 首先, 函数 $\tilde{\mathcal{J}}(\mathbf{U}_{*k})$ 是一个曲率有界的凸函数。由于 A_{ij} 满

足 $0 < A_{ij} < 1$, 能得到: $0 \leq A_{ij}(1 - A_{ij}) \leq \frac{1}{4}$, 于是可得:

$$\mathbf{H} \succeq \frac{\partial^2 \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k} \partial \mathbf{U}_{*k}^\top}.$$

根据引理 (5-2), 可得: $\mathcal{J}(\mathbf{U}_{*k}) \leq \tilde{\mathcal{J}}(\mathbf{U}_{*k})$, 即: 函数 $\tilde{\mathcal{J}}(\mathbf{U}_{*k})$ 是函数 $\mathcal{J}(\mathbf{U}_{*k})$ 的上界。□

根据定理 (5-1), 可得如下的优化问题:

$$\begin{aligned} \min_{\mathbf{U}_{*k}} \tilde{\mathcal{J}}(\mathbf{U}_{*k}) &= \mathbf{U}_{*k}^\top \left[\frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H} \mathbf{U}_{*k}(t) \right] \\ \text{subject to: } \mathbf{U}_{*k} &\in \{-1, +1\}^n. \end{aligned} \quad (5-3)$$

定义向量 $\mathbf{p} = \frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H} \mathbf{U}_{*k}(t)$ 。对于任意 l , $U_{lk} \in \{-1, +1\}$, 为了最小化 $\tilde{\mathcal{J}}(\mathbf{U}_{*k})$, 只需要在 $p_l \geq 0$ 时令 $U_{lk} = -1$, 在 $p_l < 0$ 时令 $U_{lk} = 1$ 。因此可得问题 (5-3) 的最优解为:

$$\begin{aligned} \mathbf{U}_{*k} &= \text{sign}(\mathbf{H} \mathbf{U}_{*k}(t) - \frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t)). \\ &= \text{sign}(\mathbf{U}_{*k}(t) - \gamma \frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t)). \end{aligned}$$

这里, $\gamma = \frac{4c^2}{n\lambda^2}$ 。然后, 使用这个最优解去得到下一轮迭代的 \mathbf{U}_{*k} 值, 即:

$$\mathbf{U}_{*k}(t+1) = \text{sign}(\mathbf{U}_{*k}(t) - \gamma \frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t)). \quad (5-4)$$

5.3.2.2 优化 \mathbf{V} , 固定 \mathbf{U}

当参数 \mathbf{U} 固定时, 使用与优化参数 \mathbf{U} 类似的策略来优化参数 \mathbf{V} 。具体来说, 优化参数 \mathbf{V}_{*k} 的某一行时, 参数 \mathbf{V}_{*k} 的闭式解为:

$$\mathbf{V}_{*k}(t+1) = \text{sign}(\mathbf{V}_{*k}(t) - \gamma \frac{\partial \mathcal{J}(\mathbf{V}_{*k})}{\partial \mathbf{V}_{*k}}(t)). \quad (5-5)$$

这里, $\gamma = \frac{4c^2}{n\lambda^2}$, $\frac{\partial \mathcal{J}(\mathbf{V}_{*k})}{\partial \mathbf{V}_{*k}}$ 的定义为:

$$\frac{\partial \mathcal{J}(\mathbf{V}_{*k})}{\partial \mathbf{V}_{*k}} = \frac{\lambda}{c} \sum_{i=1}^n [\mathbf{A}_{i*}^\top - \mathbf{S}_{i*}^\top] U_{ik}. \quad (5-6)$$

DLFH 模型的算法可简单地概括在算法 5.1 中。

算法 5.1 DLFH 模型学习算法

输入：

跨模态相似度 $\mathbf{S} \in \{0, 1\}^{n \times n}$ ，二值哈希编码长度 c 。

输出：

二值哈希编码 \mathbf{U} 和 \mathbf{V} 。

- 1: **初始化**：初始化 \mathbf{U} 和 \mathbf{V} 。
 - 2: **repeat**
 - 3: **for** $k = 1 \mapsto c$ **do**
 - 4: 根据公式 (5-2) 计算梯度 $\frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}$ ；
 - 5: 根据公式 (5-4) 更新 \mathbf{U}_{*k} ；
 - 6: **end for**
 - 7: **for** $k = 1 \mapsto c$ **do**
 - 8: 根据公式 (5-6) 计算梯度 $\frac{\partial \mathcal{J}(\mathbf{V}_{*k})}{\partial \mathbf{V}_{*k}}$ ；
 - 9: 根据公式 (5-5) 更新 \mathbf{V}_{*k} ；
 - 10: **end for**
 - 11: **until** 达到指定的循环次数。
-

通过算法 5.1 可以学习得到二值哈希编码 \mathbf{U} 和 \mathbf{V} 。基于算法 5.1 可得如下定理：

定理 5-3 DLFH 模型的优化算法 5.1 是收敛的。

定理 (5-3) 的证明：首先，根据定理 (5-1)，可得：

$$\mathcal{J}(\mathbf{U}_{*k}) \leq \tilde{\mathcal{J}}(\mathbf{U}_{*k}).$$

此外，由于解 (5-4) 是优化问题 (5-3) 的最优解，因此有：

$$\tilde{\mathcal{J}}(\mathbf{U}_{*k}(t+1)) \leq \tilde{\mathcal{J}}(\mathbf{U}_{*k}(t)).$$

同时根据 $\tilde{\mathcal{J}}(\mathbf{U}_{*k})$ 的定义可得： $\tilde{\mathcal{J}}(\mathbf{U}_{*k}(t)) = \mathcal{J}(\mathbf{U}_{*k}(t))$ 。于是能得到：

$$\mathcal{J}(\mathbf{U}_{*k}(t+1)) \leq \tilde{\mathcal{J}}(\mathbf{U}_{*k}(t+1)) \leq \tilde{\mathcal{J}}(\mathbf{U}_{*k}(t)) = \mathcal{J}(\mathbf{U}_{*k}(t)).$$

因此在优化过程中，总能保证 $\mathcal{J}(\mathbf{U}_{*k}(t+1)) \leq \mathcal{J}(\mathbf{U}_{*k}(t))$ 。类似地，能得到保证： $\mathcal{J}(\mathbf{V}_{*k}(t+1)) \leq \mathcal{J}(\mathbf{V}_{*k}(t))$ 。这意味着在学习过程中，可以保证算法 5.1 中目标函数不会出现上升的情况。结合目标函数的下界为 0，可以得出结论：算法 5.1 是收敛的。由于目标函数 $\mathcal{J}(\mathbf{U}, \mathbf{V})$ 关于参数 \mathbf{U} 和 \mathbf{V} 是非凸的，

算法 5.1 仅能得到一个局部最优解。 □

5.3.3 随机学习策略

算法 5.1 中，优化 \mathbf{U}_{*k} 和 \mathbf{V}_{*k} 的算法复杂度为 $\mathcal{O}(n^2)$ ，因此该算法的扩展性很差。为了解决算法的可扩展性问题，本章设计了一种随机学习策略来避免算法 5.1 中的高复杂度。

根据公式 (5-2) 和 (5-6) 可知，算法 5.1 的复杂度为 $\mathcal{O}(n^2)$ 主要原因是在计算目标函数 $\mathcal{J}(\mathbf{U}, \mathbf{V})$ 关于 \mathbf{U} 和 \mathbf{V} 的梯度时使用了全部的相似度信息。为了降低复杂度，这里使用随机选择的 m 列和 m 行的相似度来分别计算目标函数 $\mathcal{J}(\mathbf{U}, \mathbf{V})$ 关于 \mathbf{U} 和 \mathbf{V} 的梯度。基于这种采样策略，可将梯度的公式重写如下：

$$\begin{aligned}\frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t) &= \frac{\lambda}{c} \sum_{p=1}^m [\mathbf{A}_{*j_p} - \mathbf{S}_{*j_p}] U_{j_p k}, \\ \frac{\partial \mathcal{J}(\mathbf{V}_{*k})}{\partial \mathbf{V}_{*k}}(t) &= \frac{\lambda}{c} \sum_{p=1}^m [\mathbf{A}_{i_p *}^\top - \mathbf{S}_{i_p *}^\top] V_{i_p k}.\end{aligned}$$

使用基于采样策略的梯度计算复杂度为 $\mathcal{O}(n)$ 。为了得到新的算法，仅需要把上述公式替换算法 5.1 中损失函数关于 \mathbf{U} 和 \mathbf{V} 的梯度计算公式即可。

5.3.4 样本外扩展

对于训练集以外的数据，需要学习哈希函数 $h(\cdot)$ 和 $g(\cdot)$ 将数据表示成二值哈希编码，其中， $h(\cdot)$ 表示图片模态的哈希函数， $g(\cdot)$ 表示文本模态的哈希函数。本章设计了两种哈希函数的学习策略，即基于线性回归的方法和基于核逻辑回归的方法。这里以图片模态为例说明哈希函数的学习过程。

5.3.4.1 基于线性回归的方法

给定已学习到的二值哈希编码 \mathbf{U} 和数据特征 \mathbf{X} ，基于线性回归的方法的优化问题定义如下：

$$\min_{\mathbf{W}} \mathcal{L}_l(\mathbf{W}) = \|\mathbf{U} - \mathbf{X}\mathbf{W}\|_F^2 + \gamma_x \|\mathbf{W}\|_F^2, \quad (5-7)$$

其中， $\mathbf{W} \in \mathbb{R}^{d_x \times c}$ ， γ_x 表示正则项的超参数。

问题 (5-7) 是一个凸优化问题, 为了得到最优解, 仅需将目标函数 $\mathcal{L}_l(\mathbf{W})$ 关于变量 \mathbf{W} 的一阶导数置为 $\mathbf{0}_{d_x \times c}$ 即可。具体来说, 首先根据目标函数推导变量 \mathbf{W} 的梯度为:

$$\frac{\partial \mathcal{L}_l(\mathbf{W})}{\partial \mathbf{W}} = 2\mathbf{X}^\top (\mathbf{X}\mathbf{W} - \mathbf{U}) + 2\gamma_x \mathbf{W}.$$

令: $\frac{\partial \mathcal{L}_l(\mathbf{W})}{\partial \mathbf{W}} = \mathbf{0}_{d_x \times c}$, 可得最优解为:

$$\mathbf{W} = (\mathbf{X}^\top \mathbf{X} + \gamma_x \mathbf{I}_{d_x})^{-1} \mathbf{X}^\top \mathbf{U}.$$

然后可得哈希函数为: $h(\mathbf{x}_q) = \text{sign}(\mathbf{W}^\top \mathbf{x}_q)$, 这里, $\mathbf{x}_q \notin \mathbf{X}$ 表示不在训练集 \mathbf{X} 中的样本。文本模态的哈希函数 $g(\cdot)$ 学习策略类似, 不再赘述。简单起见, 这个方法简记为 DLFH。

5.3.4.2 基于核逻辑回归的方法

给定已学习到的二值哈希编码 \mathbf{U} 和数据特征 \mathbf{X} , 将二值哈希编码看做数据样本的 c 个类别信息, 然后学习 c 个分类器将数据投影到类别空间。这 c 个分类器可以作为哈希函数。本章使用核逻辑回归方法进行分类。对于第 k 个比特, 优化问题定义如下:

$$\min_{\mathbf{M}_{*k}} \mathcal{L}_k(\mathbf{M}_{*k}) = \sum_{i=1}^n \log(1 + e^{-U_{ik} \phi(\mathbf{x}_i)^\top \mathbf{M}_{*k}}) + \eta_x \|\mathbf{M}_{*k}\|_F^2. \quad (5-8)$$

这里, $\phi(\cdot) \in \mathbb{R}^{n_k}$ 表示核基数目为 n_k 的 RBF 核化表示, $\mathbf{M}_{*k} \in \mathbb{R}^{n_k}$ 表示第 k 个比特的分类器参数, $\eta_x > 0$ 表示正则项的超参数。

本章使用随机梯度下降算法 (Stochastic Gradient Descent, 简称 SGD) 求解问题 (5-8)。然后可得哈希函数为: $h(\mathbf{x}_q) = \text{sign}(\mathbf{M}^\top \mathbf{x}_q)$, 其中, $\mathbf{x}_q \notin \mathbf{X}$ 表示不在训练集 \mathbf{X} 中的样本。文本模态的哈希函数 $g(\cdot)$ 学习策略类似, 不再赘述。简单起见, 这个方法简记为 Kernelized DLFH (KDLFH)。

5.3.5 复杂度分析

本章将未使用随机采样版本的 DLFH 方法记为 DLFH_f , 使用随机采样版本的 DLFH 方法记为 DLFH_s 。DLFH_f 的时间复杂度为 $\mathcal{O}(Tcn^2)$, 其中, T 表示迭代次数, T 和 c 通常是较小的常量。DLFH_s 的时间复杂度为 $\mathcal{O}(Tcnm)$ 。当

$m \ll n$, DLFH_s 的时间复杂度比 DLFH_f 的时间复杂度低很多。

5.4 实验验证

本章使用三个数据集验证了 DLFH 方法的有效性。由于实验中的基准方法包含非深度跨模态哈希学习方法和深度跨模态哈希学习方法, 本章使用了两个服务器进行实验。对于本章提出的 DLFH 、 KDLFH 方法和非深度跨模态哈希基准方法, 使用了一台 12 核心及 96GB 内存, CPU 为 Intel (R) CPU E5-2620V2@2.1G 的服务器。对于深度跨模态哈希基准方法, 使用了一台 14 核心及 746G 内存, CPU 为 Intel (R) CPU E5-2860V4@2.4G, 显卡为 NVIDIA M40 的服务器。

5.4.1 实验设置

5.4.1.1 数据集

本章使用 IAPRTC12^[116]、FLICKR25K^[110] 和 NUSWIDE^[109] 三个数据集进行实验。三个数据集的介绍可以参考第二章第 2.2 节。在本章中, 跨模态检索的实验使用了三个数据集图片和文本模态的数据。对于 NUSWIDE 数据集, 本章选择类别样本最多的 10 个类别对应的样本进行实验, 对应的样本总数为 186577。

5.4.1.2 评价标准与对比方法

本章使用了十个现有的跨模态哈希学习方法进行了对比, 包括三个无监督跨模态哈希学习方法, 即: CCAITQ ^[21]、 CMFH ^[50] 和无监督生成对抗跨模态哈希^[125] (Unsupervised Generative Adversarial Cross-Modal Hashing, 简称 UGACH) ; 七个监督跨模态哈希学习方法, 即: MLBE ^[68]、 SCM ^[51]、 SePH_{rnd} ^[53]、 SePH_{km} ^[53]、 GSPH ^[54]、 DCMH ^[56] 和跨模态海明哈希^[126] (Cross-Modal Hamming Hashing, 简称 CMHH)。这些方法中, UGACH 、 DCMH 和 CMHH 是深度跨模态哈希学习方法, 使用在 ImageNet ^[61] 数据集上预训练的 CNNF 网络来初始化用于深度特征学习的深度神经网络。 SePH_{rnd} 和 SePH_{km} 表示 SePH ^[53] 方法的两个变体, SePH_{rnd} 表示使用随机初始化作为核基的方法, SePH_{km} 表示使用 K-Means 方法来生成核基的方法。对于 SePH_{rnd} 、 SePH_{km} 、

GSPH 和 KDLFH，根据 SePH^[53] 作者的建议，设置核基的数目为 500。对于其他基准方法，实验中根据原论文作者的建议选择超参数。对于 DLFH 方法，设置 $\lambda = 8$ ， $T = 30$ 。设置 $m = c$ 。初始化过程中，首先从均匀分布中随机生成 $\mathbb{R}^{n \times c}$ 的矩阵，然后使用取符号函数来初始化变量 U 和 V 。

对于 IAPRTC12 数据集，实验中随机选择了 2000 个样本构造查询集，剩余的所有样本作为数据库集。对于 FLICKR25K 数据集，实验中随机选择了 2000 个样本构造查询集，剩余的所有样本作为数据库集。对于 NUSWIDE 数据集，随机选择了 1867 个样本作为查询集，剩余的所有样本作为数据库集。由于 SePH_{rnd}、SePH_{km}、GSPH、DCMH、CMHH 和 MLBE 方法的复杂度太高，本章采用采样的训练集进行训练。具体来说，对于 SePH_{rnd}、SePH_{km}、DCMH、CMHH 和 GSPH，本章从数据库集中选择 5000 个样本作为采样训练集，对于 MLBE，本章从数据库集中选择 1000 个样本作为采样训练集。除去这几个方法，其他所有方法均使用整个数据库集进行训练。

实验中使用了海明排序和哈希表查询来验证本章提出的 DLFH 方法的有效性。对于海明排序，本章对比了在返回序列截断位置的平均查准率均值 $cMAP(K)$ 。对于哈希表查询，本章汇报了在海明球 R 的查准-查全率曲线。 $cMAP(K)$ 、在海明球 R 的查准率 $rPre(R)$ 和在海明球 R 的查全率 $rRec(R)$ 的计算参见第二章第 2.2 小节。

5.4.2 收敛性分析

本节从实验角度验证了 DLFH 方法的收敛性。具体来说，从 FLICKR25K 数据集中随机选择 5000 个样本进行实验，其中 2000 个样本作为查询集，3000 个样本作为数据库集，同时使用全部数据库集作为训练集。图 5-1 汇报了随着实验的进行，损失函数值的变化。从图 5-1 可以看出， $DLFH_f$ 的损失函数值随着训练的进行严格下降。 $DLFH_s$ 的损失函数值随着训练的进行虽然有一定震荡，但整体呈下降的趋势，并且最终的目标函数值接近 $DLFH_f$ 的目标函数值。因此证明了 DLFH 方法收敛。同时，图 5-2 汇报了随着实验的进行，平均查准率均值 $cMAP(K)$ 的变化，其中， $I \rightarrow T$ 表示以图搜文， $T \rightarrow I$ 表示以文搜图。从图 5-2 中可以看出，平均查准率均值整体趋势与损失函数值变化趋势一致，即： $DLFH_f$ 的平均查准率均值随着训练的进行严格上升。 $DLFH_s$ 的平均查准率均值随着训练的进行虽然有一定震荡，但整体呈上升的趋势，并且最终的平均查准率均值接近 $DLFH_f$ 的平均查准率均值。

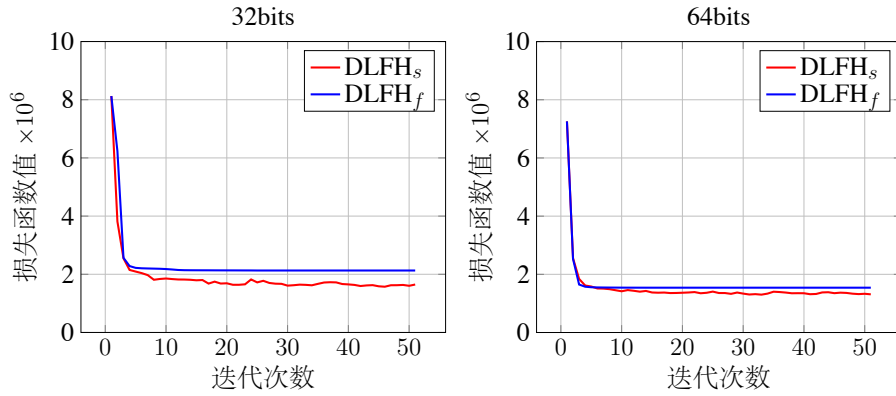


图 5-1: DLFH 方法损失函数值随训练变化趋势

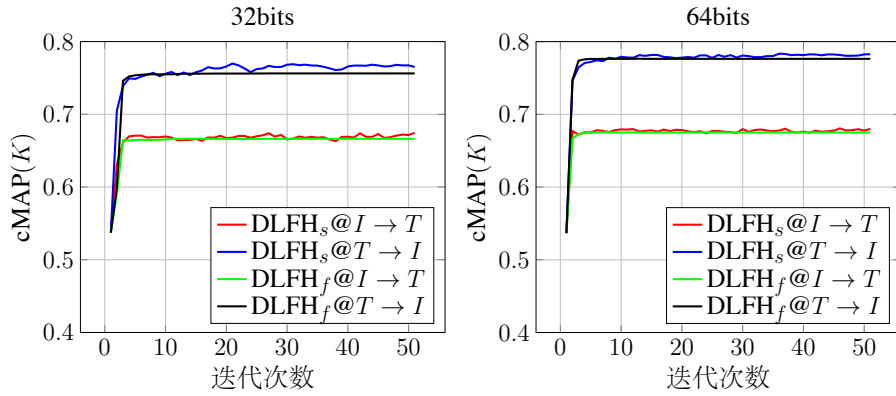


图 5-2: DLFH 方法平均查准率均值随训练变化趋势

5.4.3 性能对比

5.4.3.1 海明排序

表5-1、表5-2和表5-3中分别展示了在 IAPRTC12 数据集、FLICKR25K 数据集和 NUSWIDE 数据集上的平均查准率均值 $cMAP(K)$ 。为了对比公平，在表5-1、表5-2和表5-3中，跨模态哈希学习方法根据是否使用基于分类的样本外扩展方法被分为两组。具体来说， $SePH_{rnd}$ 、 $SePH_{km}$ 、GSPH 和 KDLFH 方法均使用了非线性的分类方法来建模样本外扩展方法以学习哈希函数，而 SCM、CMFH、CCAITQ、MLBE 和 DLFH 方法则没有使用这种策略来学习哈希函数。同时最好的平均查准率均值 $cMAP(K)$ 使用加粗字体标注。从表5-1、表5-2和表5-3中可以看出，与基准方法 CCAITQ、CMFH、MLBE 和 SCM 方法相比，本章提出的 DLFH 方法在所有情况下能达到更高的检索精度。与基准方法 $SePH_{rnd}$ 、 $SePH_{km}$ 和 GSPH 方法相比，KDLFH 方法在所有情况下能达

到更高的检索精度。与 DLFH 方法相比，KDLFH 方法在大部分情况下能达到更高的检索精度，原因是 KDLFH 方法采用了非线性的分类方法来学习哈希函数。

表 5-1: DLFH 方法和基准方法在 IAPRTC12 数据集上的平均查准率均值

方法	$I \rightarrow T$			$T \rightarrow I$		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
SCM	0.3874	0.3972	0.4102	0.3857	0.3981	0.4082
CMFH	0.3091	0.3170	0.3076	0.3167	0.3242	0.3171
CCAITQ	0.3417	0.3309	0.3236	0.3419	0.3318	0.3244
MLBE	0.3052	0.3036	0.3046	0.3044	0.3035	0.3049
DLFH	0.4558	0.5200	0.5484	0.4847	0.5713	0.6243
SePH _{rnd}	0.4147	0.4227	0.4289	0.4431	0.4562	0.4666
SePH _{km}	0.4141	0.4207	0.4265	0.4377	0.4476	0.4567
GSPH	0.4003	0.4167	0.4285	0.4230	0.4477	0.4662
KDLFH	0.4702	0.5261	0.5654	0.5073	0.5810	0.6393

表 5-2: DLFH 方法和基准方法在 FLICKR25K 数据集上的平均查准率均值

方法	$I \rightarrow T$			$T \rightarrow I$		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
SCM	0.6389	0.6506	0.6576	0.6237	0.6337	0.6425
CMFH	0.5738	0.5732	0.5687	0.5752	0.5771	0.5769
CCAITQ	0.5762	0.5711	0.5677	0.5759	0.5712	0.5682
MLBE	0.5581	0.5583	0.5583	0.5579	0.5579	0.5604
DLFH	0.7584	0.7802	0.7892	0.8239	0.8501	0.8605
SePH _{rnd}	0.6560	0.6596	0.6623	0.6866	0.6942	0.6979
SePH _{km}	0.6606	0.6640	0.6666	0.6972	0.7039	0.7083
GSPH	0.6462	0.6577	0.6640	0.6847	0.6993	0.7121
KDLFH	0.7707	0.7927	0.8008	0.8214	0.8502	0.8583

表 5-3: DLFH 方法和基准方法在 NUSWIDE 数据集上的平均查准率均值

方法	$I \rightarrow T$			$T \rightarrow I$		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
SCM	0.5219	0.5481	0.5558	0.4925	0.5153	0.5226
CMFH	0.3812	0.3862	0.4003	0.3687	0.3748	0.3824
CCAITQ	0.3955	0.3823	0.3703	0.3898	0.3781	0.3679
MLBE	0.3451	0.3446	0.3486	0.3453	0.3453	0.3453
DLFH	0.6700	0.6927	0.7033	0.7850	0.8203	0.8378
SePH _{rnd}	0.5396	0.5485	0.5524	0.6179	0.6275	0.6340
SePH _{km}	0.5441	0.5554	0.5564	0.6263	0.6414	0.6447
GSPH	0.5419	0.5529	0.5602	0.6269	0.6395	0.6513
KDLFH	0.6847	0.7019	0.7112	0.7827	0.8178	0.8325

此外, 表 5-4 中给出了与深度跨模态哈希学习方法的检索精度对比。在表 5-4 中, 比特长度设置为 16, 其余比特的实验结果类似。最好的结果使用加粗字体标注, 次好的结果使用下划线标注。在这个实验中, 为了对比公平, 对于图片模态的特征, DLFH 方法和 KDLFH 方法使用在 ImageNet 数据集^[61]上预训练的深度神经网络 CNNF 提取的 4096 维的深度特征。由于深度跨模态哈希学习方法训练过程耗时, 因此这个实验中深度哈希学习方法使用了采样训练集进行训练。表 5-4 中, UGACH 方法的结果直接引自原始论文, 因此该方法只汇报了在 NUSWIDE 数据集上的检索精度。

从表 5-4 中可以看到, 得益于可以使用全部数据库集合进行高效训练, DLFH 和 KDLFH 方法的检索精度在大部分情况下比深度跨模态哈希学习方法要好。此外, 尽管 DLFH 方法和 KDLFH 方法使用了全部训练集进行训练, DLFH 方法和 KDLFH 方法的训练过程也比深度跨模态方法高效。在第六章中, 将进一步讨论 DLFH 方法和深度跨模态哈希学习方法的对比。

5.4.3.2 哈希表查询

对于哈希表查询, 本章通过实验汇报了在 IAPRTC12、FLICKR25K 和 NUSWIDE 数据集上的查准-查全率曲线。这里, 本章仅汇报了比特长度为 16 比特的查准-查全率曲线, 其余比特长度的结果类似。

表 5-4: DLFH 方法和深度跨模态基准方法的对比

数据集	方法	检索精度		训练时间 (秒)
		$I \rightarrow T$	$T \rightarrow I$	
IAPRTC12	CMHH	<u>0.4881</u>	0.4946	45468.5
	DCMH	0.4526	<u>0.5185</u>	33090.2
	DLFH	0.4377	0.5267	5.85
	KDLFH	0.5048	0.4991	741.75
FLICKR25K	CMHH	0.7537	0.7684	44358.6
	DCMH	0.7441	0.7848	33736.7
	DLFH	<u>0.7794</u>	<u>0.7900</u>	4.46
	KDLFH	0.8591	0.8203	777.59
NUSWIDE	UGACH	0.613	0.603	N/A
	CMHH	0.6682	0.6735	48128.5
	DCMH	0.6479	0.6884	35777.2
	DLFH	<u>0.8067</u>	<u>0.7576</u>	36.01
	KDLFH	0.8332	0.7604	6651.90

图 5-3 给出了查准-查全率曲线的结果。从图 5-3 中可以看到，与没有采用离散优化学习的跨模态哈希学习方法相比，SCM、SePH_{rnd} 和 SePH_{km} 在大多数情况下取得了更高的检索精度，DLFH 和 KDLFH 在所有情况下都达到了更高的检索精度，由此证明跨模态哈希学习中离散哈希学习的重要性。与基准方法中所有非深度跨模态哈希学习方法相比，DLFH 和 KDLFH 在所有的情况下达到了更高的查准-查全率。由于采用了非线性的分类方法来学习哈希函数，在大多数情况下，KDLFH 方法的检索精度要优于 DLFH 方法的检索精度。

5.4.3.3 训练速度对比

为了验证本章提出的 DLFH 方法和 KDLFH 方法训练的高效性，本节在最大规模的 NUSWIDE 数据集上进行了实验。具体来说，本文从 NUSWIDE 数据集的数据库集合中选取大小不同的子集来构成采样训练集进行训练，然后汇报

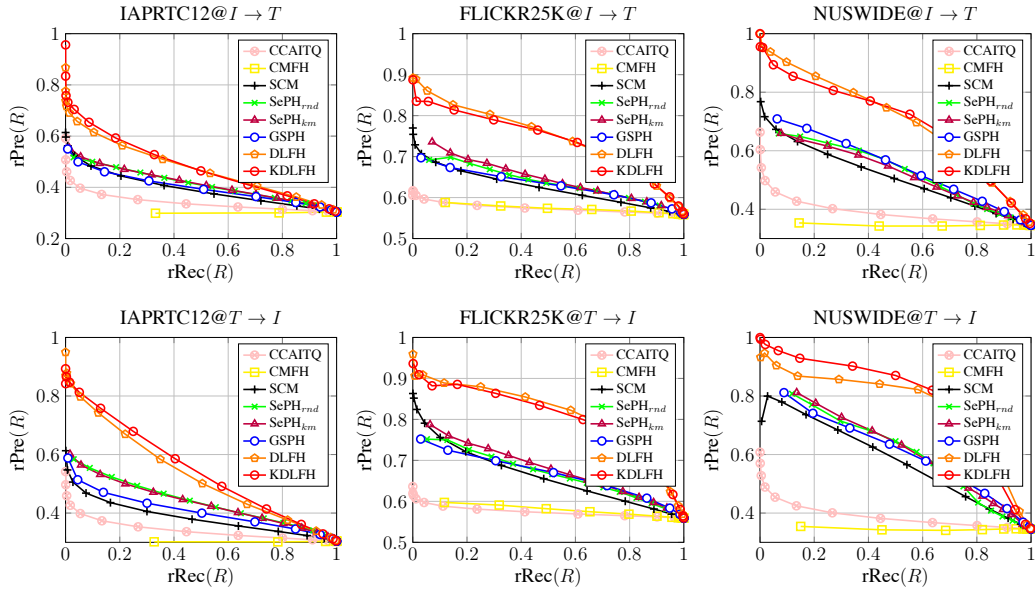


图 5-3: DLFH 方法和基准方法在三个数据集上的查准-查全率曲线

了本文提出的 DLFH 方法、KDLFH 方法和基准方法中所有非深度跨模态哈希学习方法在不同大小的采样训练集上的训练时间。

表 5-5 汇报了在不同的采样训练集下进行训练所花费的训练时间。在表 5-5 中，符号“-”表示由于内存溢出等问题无法进行实验，符号“~184K”表示使用全部样本进行训练。本实验中按照跨模态哈希学习方法根据是否使用基于分类的样本外扩展方法被分为两组。从表 5-5 中可以看到，DLFH 方法的训练没有 CCAITQ 方法高效，但 CCAITQ 方法无法达到令人满意的检索精度。与跨模态离散哈希学习方法 SePH_{rnd} 和 SePH_{km} 相比，DLFH 方法的训练更高效。当训练集的样本数目增大时， SePH_{rnd} 和 SePH_{km} 由于内存溢出等问题无法训练，而 DLFH 方法则不存在这个问题。与非深度跨模态离散哈希学习方法 SCM 相比，虽然 DLFH 方法的训练没有 SCM 高效，但 SCM 是基于类别标签的，其应用场景与 DLFH 方法不同。由于采用了基于分类的非线性哈希函数，KDLFH 方法的训练没有 DLFH 方法高效，但 KDLFH 方法能达到更高的检索精度。实际应用中可根据对检索精度和训练耗时的不同需要选取不同的方法。

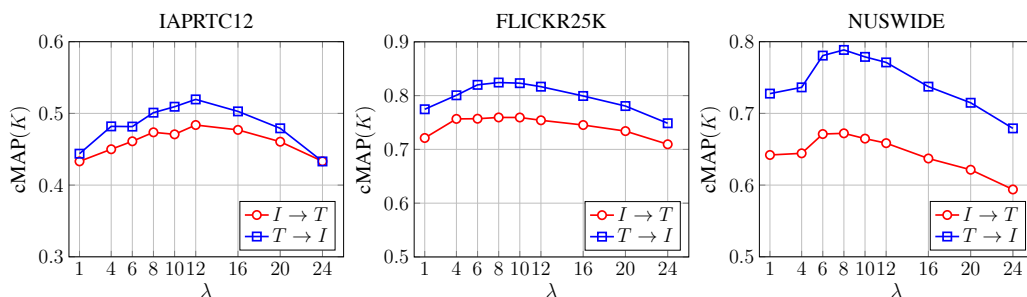
5.4.4 超参数实验

DLFH 方法中， λ 和采样集大小 m 是最重要的超参数。本节验证了 DLFH 方法关于超参数 λ 和 m 的敏感性。这个实验在 IAPRTC12、FLICKR25K 和

表 5-5: DLFH 方法和基准方法在 NUSWIDE 数据集上的训练时间

方法	训练集大小				
	1K	5K	10K	50K	~184K
SCM	10.41	11.13	11.09	11.34	12.30
CMFH	4.20	12.01	20.65	84.00	305.51
CCAITQ	0.56	0.59	0.69	1.51	4.25
MLBE	998.98	-	-	-	-
DLFH	1.26	3.84	6.61	34.88	112.88
SePH _{rnd}	80.54	606.18	-	-	-
SePH _{km}	83.81	705.39	-	-	-
GSPH	126.43	981.35	-	-	-
KDLFH	56.58	214.65	479.46	2097.52	7341.05

NUSWIDE 三个数据集上进行, 并设置比特长度为 16 比特, 其余比特长度的实验结果类似。图 5-4 中给出了在超参数 λ 不同取值下, DLFH 方法在三个数据集上的检索精度。从图 5-4 中可以看到, 随着超参数 λ 的增大, 检索精度先变好后变差。当 $6 \leq \lambda \leq 10$, DLFH 方法对超参数 λ 不敏感。图 5-5 给出了在超参数 m 不同取值下, DLFH 方法在三个数据集上的检索精度。从图 5-5 可以看到, 随着 m 的增大, 检索精度变得越来越好。然而, m 越大, 所需求的计算开销就越大。实际应用中, m 的选择可以根据对检索精度和计算开销的需求来决定。

图 5-4: DLFH 方法对超参数 λ 的敏感性实验

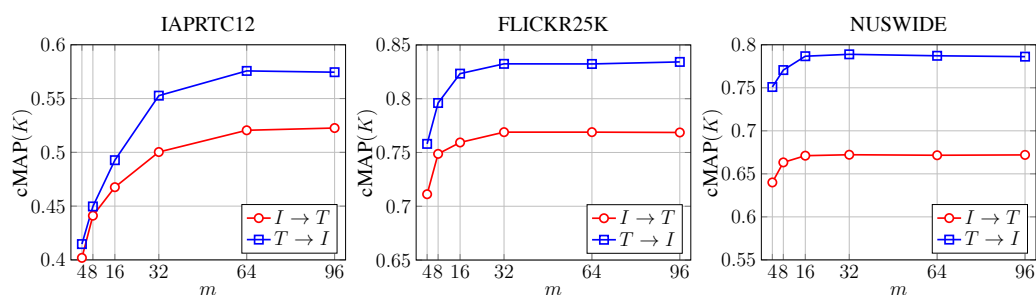


图 5-5: DLFH 方法对超参数 m 的敏感性实验

5.5 本章小结

本章提出一种新型非深度跨模态离散哈希学习方法 DLFH。DLFH 使用离散隐因子模型对跨模态哈希学习方法建模，并设计了一种高效的二值哈希编码学习算法。与现有的非深度跨模态哈希学习方法和部分深度哈希学习方法相比，DLFH 方法能达到更高的检索精度，同时 DLFH 方法的训练过程更加高效。

本章的工作已总结成文并发表：

- JIANG Q-Y, LI W-J. Discrete Latent Factor Model for Cross-Modal Hashing[J]. IEEE Transactions on Image Processing, 2019, 28(7): 3490 – 3501. (CCF-A 类期刊)

第六章 深度多模态离散哈希

6.1 引言

现有的大部分跨模态哈希学习方法^[21,53,69]都属于非深度跨模态哈希学习方法。代表性的非深度跨模态方法包括 BRE^[100]、CRH^[48]、CCAITQ^[21]、ACQ^[69]、QCH^[52]、SCM^[51]、SePH^[53]和第五章提出的 DLFH^[57]等。这些方法通常使用已经提取好的数据特征进行建模和学习，无法进行深度多模态特征学习。因此，这些方法的检索精度往往会受限于所采用的多模态数据的特征的表达能力。当数据的特征表达能力较差时，这些方法通常无法达到令人满意的检索精度。近年来，深度学习也已经在多模态领域取得了进展^[127]。本章首先尝试将深度多模态特征学习引入到跨模态哈希学习中，并首次提出一种深度跨模态哈希学习方法（**Deep Cross-Modal Hashing**, 简称 DCMH）。DCMH 方法利用一个深度卷积神经网络来完成对图片模态数据的深度特征学习，利用一个多层全连接网络构成的深度神经网络来完成文本模态的深度特征学习。同时，DCMH 设计了一种可以学习多模态数据的二值哈希编码表示的离散优化算法，将二值哈希编码学习和多模态深度特征学习整合到一个统一的学习框架。DCMH 方法是第一个^①使用深度神经网络来完成多模态深度特征学习的跨模态哈希学习方法。在第五章中，DLFH 方法使用监督信息直接指导二值哈希编码的学习，提高了现有非深度跨模态哈希学习方法的检索精度。然而，DLFH 是一种非深度跨模态哈希学习方法，无法进行深度特征学习。因此，DLFH 的检索精度仍有可提升的空间。本章还提出一种基于深度离散隐因子模型的跨模态哈希学习方法（**Deep Discrete Latent Factor Model for Cross-Modal Hashing**, 简称 DDLFH）。DDLFH 使用深度离散隐因子模型建模跨模态哈希学习问题，并设计了一种直接学习二值哈希编码的算法。DDLFH 方法也可以在学习二值哈希编码的同时完成深度特征学习过程。本章的主要贡献包括以下五个方面：

- 本章提出的 DCMH 方法是首次将深度特征学习引入到跨模态哈希学习中的

^①本文提出的 DCMH 方法发表于 2017 年。此后，出现了一系列深度跨模态哈希学习方法，包括基于标签对关系的深度哈希^[128]（Pairwise Relationship Deep Hashing, 简称 PRDH）、CMHH^[126]等方法。

跨模态哈希学习方法。DCMH 方法将二值哈希编码学习和深度特征学习整合到一个统一的学习框架中。

- 本章提出的 DCMH 方法设计了一种可同时完成二值哈希编码学习和深度多模态特征学习的离散优化算法。
- 本章还提出一种基于深度离散隐因子模型的跨模态哈希学习方法 DDLFH。DDLFH 将 DLFH 的二值哈希编码学习能力和深度学习的特征学习能力整合到一个统一的学习框架，能实现两种能力的相互反馈。
- 本章提出的 DDLFH 设计了一种可以直接学习二值哈希编码的逐比特离散编码算法 (Bitwise Discrete Coding, 简称 BDC) 来学习二值哈希编码。本章给出了 BDC 算法的收敛性证明。
- 实验表明与非深度跨模态哈希学习方法相比，本章提出的 DCMH 方法能达到更高的检索精度。同时，通过结合 DLFH 方法的二值哈希编码学习能力和深度特征学习能力，DDLFH 能达到比现有的非深度跨模态哈希、深度跨模态哈希学习方法更高的检索精度。

本章剩余部分组织如下：第 6.2 节中介绍问题定义；第 6.3 介绍 DCMH 方法；第 6.4 介绍 DDLFH 方法；第 6.5 节通过实验验证 DCMH 方法有效性；第 6.6 节通过实验验证 DDLFH 方法有效性；最后，第 6.7 节对本章进行总结。

6.2 问题定义

本章使用图片模态和文本模态来阐述本章提出的方法。值得一提的是，本章提出的方法可以很容易地应用到数据模态数目大于两个的场景中。使用 $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ 表示图片样本集，使用 $\mathbf{Y} = \{\mathbf{y}_j\}_{j=1}^n$ 表示文本样本集。为了完成深度特征学习，本章假设样本为未经过特征提取的原始数据。对于跨模态哈希学习方法，本章中假设训练集中所有样本的两个模态的数据都已观测到。值得一提的是，本章提出的方法可以很容易地应用到训练集中某个模态仅有部分数据被观测到的场景中。同时，这个假设仅是针对训练集而言，测试过程中，不要求所有模态的所有数据样本都被观测到。对于监督跨模态哈希学习方法，跨模态相似度 $\mathbf{S} = \{S_{ij}\}_{i,j=1}^n \in \{0,1\}^{n \times n}$ 也已给定。这里， $S_{ij} \in \{0,1\}$ ， $S_{ij} = 1$ 表示图片 \mathbf{x}_i 和文本 \mathbf{y}_j 相似， $S_{ij} = 0$ 表示图片 \mathbf{x}_i 和文本 \mathbf{y}_j 不相似。

给定上述训练集信息 \mathbf{X} ， \mathbf{Y} 和 \mathbf{S} ，跨模态哈希的目标是学习哈希函数将多

模态数据表示成具有保相似性的二值哈希编码形式。本章使用 $h(\cdot)$ 和 $g(\cdot)$ 分别表示图片模态和文本模态的哈希函数。使用 $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_n]^\top \in \{-1, +1\}^{n \times c}$ 和 $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_n]^\top \in \{-1, +1\}^{n \times c}$ 分别表示图片模态和文本模态的二值哈希编码^①，其中， c 表示二值哈希编码长度。跨模态哈希学习的目标是为多模态数据学习能尽量保持原始空间中跨模态相似度的二值哈希编码表示。具体来说，如果 $S_{ij} = 1$ ，那么希望二值哈希编码 \mathbf{u}_i 和二值哈希编码 \mathbf{v}_j 之间的海明距离尽量小，否则，如果 $S_{ij} = 0$ ，那么希望二值哈希编码 \mathbf{u}_i 和二值哈希编码 \mathbf{v}_j 之间的海明距离尽量大。

6.3 深度跨模态哈希学习方法 DCMH

本节从模型、学习算法和样本外扩展三个方面介绍本章提出的深度跨模态哈希学习方法 DCMH。

6.3.1 模型

给定图片样本 \mathbf{x}_i 和文本样本 \mathbf{y}_j 的二值哈希编码对 $(\mathbf{u}_i, \mathbf{v}_j)$ ，定义 Ω_{ij} 如下：

$$\Omega_{ij} = \frac{\lambda}{c} \mathbf{u}_i^\top \mathbf{v}_j.$$

这里， $\lambda > 0$ 表示超参数。根据 Ω_{ij} ，定义 A_{ij} 为： $A_{ij} = \frac{1}{1+e^{-\Omega_{ij}}}$ 。

然后，定义跨模态相似度 S_{ij} 的似然为：

$$p(S_{ij} | \mathbf{U}, \mathbf{V}) = \begin{cases} A_{ij}, & \text{if } S_{ij} = 1, \\ 1 - A_{ij}, & \text{otherwise.} \end{cases}$$

然后，跨模态相似度 \mathbf{S} 的对数似然可根据如下公式推导：

$$\begin{aligned} \log p(\mathbf{S} | \mathbf{U}, \mathbf{V}) &= \log \left(\prod_{i,j=1}^n p(S_{ij} | \mathbf{U}, \mathbf{V}) \right) \\ &= \sum_{i,j=1}^n [S_{ij} \Omega_{ij} - \log(1 + e^{\Omega_{ij}})]. \end{aligned}$$

^①这里，二值哈希编码被表示成 -1 与 $+1$ 。学习结束后， -1 被替换成 0 ，得到用 0 和 1 表示的二值哈希编码。

DCMH 的目标为最小化跨模态相似度 \mathbf{S} 的负对数似然。同时，DCMH 模型中使用哈希函数 $h(\cdot)$ 将图片数据从原始空间投影到二值空间，使用哈希函数 $g(\cdot)$ 将文本数据从原始空间投影到二值空间。因此，DCMH 尝试解如下优化问题：

$$\begin{aligned} \min \mathcal{J}(\mathbf{U}, \mathbf{V}, \theta, \phi) &= - \sum_{i,j=1}^n [S_{ij}\Omega_{ij} - \log(1 + e^{\Omega_{ij}})] \\ \text{subject to: } \mathbf{U}, \mathbf{V} &\in \{-1, +1\}^{n \times c}, \\ \forall i \in \{1, \dots, n\}, \mathbf{u}_i &= h(\mathbf{x}_i) = \mathbf{sign}(f(\mathbf{x}_i; \theta)), \\ \forall j \in \{1, \dots, n\}, \mathbf{v}_j &= g(\mathbf{y}_j) = \mathbf{sign}(e(\mathbf{y}_j; \phi)), \end{aligned} \quad (6-1)$$

其中， $f(\cdot)$ 表示将图片数据从原始空间投影到 \mathbb{R}^c 空间的函数， θ 表示函数 $f(\cdot)$ 的参数， $e(\cdot)$ 表示将文本数据从原始空间投影到 \mathbb{R}^c 空间的函数， ϕ 表示函数 $e(\cdot)$ 的参数。

定义向量 $\mathbf{f}_i = f(\mathbf{x}_i)$ ， $\mathbf{e}_j = e(\mathbf{y}_j)$ ，并定义矩阵 $\mathbf{F} = [\mathbf{f}_1, \dots, \mathbf{f}_n]^\top \in \mathbb{R}^{n \times c}$ ， $\mathbf{E} = [\mathbf{e}_1, \dots, \mathbf{e}_n]^\top \in \mathbb{R}^{n \times c}$ 。由于二值约束的存在，问题 (6-1) 是一个 NP 难的问题。因此，DCMH 将问题 (6-1) 中的部分二值约束松弛到连续空间。具体来说，定义 Φ_{ij} 为：

$$\Phi_{ij} = \frac{\lambda}{c} \mathbf{f}_i^\top \mathbf{e}_j,$$

并使用 Φ_{ij} 替代问题 (6-1) 中的 Ω_{ij} 。为了降低二值哈希编码和实值表示之间的误差，在原始目标函数中添加二值哈希编码和实值表示之间的量化损失。同时，考虑到比特平衡的问题，在目标函数中添加施加在实值表示上的平衡约束。因此，问题 (6-1) 可重写为如下形式：

$$\begin{aligned} \min \mathcal{J}(\mathbf{U}, \mathbf{V}, \theta, \phi) &= - \sum_{i,j=1}^n [S_{ij}\Phi_{ij} - \log(1 + e^{\Phi_{ij}})] \\ &\quad + \nu(\|\mathbf{1}_n \mathbf{F}\|_F^2 + \|\mathbf{1}_n \mathbf{E}\|_F^2) \\ &\quad + \gamma(\|\mathbf{U} - \mathbf{F}\|_F^2 + \|\mathbf{V} - \mathbf{E}\|_F^2) \\ \text{subject to: } \forall i \in \{1, \dots, n\}, \mathbf{f}_i &= f(\mathbf{x}_i; \theta), \\ \forall j \in \{1, \dots, n\}, \mathbf{e}_j &= e(\mathbf{y}_j; \phi), \\ \mathbf{U}, \mathbf{V} &\in \{-1, +1\}^{n \times c}, \end{aligned} \quad (6-2)$$

其中, $\mathbf{1}_n$ 表示元素全部为 1 的 n 维向量, $\nu > 0$, $\gamma > 0$ 表示超参数。

同时, DCMH 方法假设多模态数据被投影到同一个二值子空间中。因此, DCMH 模型在问题 (6-2) 的基础上引入二值变量 \mathbf{B} 并添加约束: $\mathbf{B} = \mathbf{U} = \mathbf{V}$ 。问题 (6-2) 可进一步化简如下:

$$\begin{aligned} \min_{\mathbf{B}, \theta, \phi} \mathcal{J}(\mathbf{B}, \theta, \phi) = & - \sum_{i,j=1}^n [S_{ij} \Phi_{ij} - \log(1 + e^{\Phi_{ij}})] \\ & + \nu (\|\mathbf{1}_n \mathbf{F}\|_F^2 + \|\mathbf{1}_n \mathbf{E}\|_F^2) \\ & + \gamma (\|\mathbf{B} - \mathbf{F}\|_F^2 + \|\mathbf{B} - \mathbf{E}\|_F^2) \quad (6-3) \\ \text{subject to: } & \forall i \in \{1, \dots, n\}, \mathbf{f}_i = f(\mathbf{x}_i; \theta), \\ & \forall j \in \{1, \dots, n\}, \mathbf{e}_j = e(\mathbf{y}_j; \phi), \\ & \mathbf{B} \in \{-1, +1\}^{n \times c}. \end{aligned}$$

本章使用两个深度神经网络来完成深度特征学习。具体来说, 对于图片模态, DCMH 首先使用一个卷积神经网络作为基础网络架构, 然后在基础网络架构的基础上增加了一层全连接层组成新的深度神经网络, 并使用这个深度神经网络完成图片模态的深度特征学习。实际中, 基础网络架构可以自行设计或者从任意一种已知的深度神经网络修改得到。例如, 可以将 AlexNet^[61]、CNNF^[119] 或者 VGGNet^[120] 的分类层去掉, 得到的网络就可以作为基础网络架构。由于本章提出的方法重点在研究监督信息的监督策略, 因此这里不指定某种具体的基础网络架构。使用 $f(\mathbf{x}; \theta)$ 表示用于图片模态的深度特征学习架构, 其中 θ 表示图片模态深度特征学习架构的所有待学习参数。同理, 对于文本模态, DCMH 首先将每个文本数据点 \mathbf{y}_i 表示成为 BOW 特征。然后本章设计了一个以词袋表示为输入, 由两层全连接网络构成的深度神经网络结构, 以学习文本模态的数据特征。使用 $e(\mathbf{y}; \phi)$ 表示用于文本模态的深度特征学习架构, 其中 ϕ 表示文本模态深度特征学习架构的所有待学习参数。

从问题 (6-3) 中可以看到, DCMH 方法可以学习两个数据模态对应的深度神经网络参数, 同时还可以学习二值哈希编码 \mathbf{B} 。

6.3.2 学习算法

为了学习问题 (6-3) 中的参数, 本文在待学习参数 $\{\mathbf{B}, \theta, \phi\}$ 之间采用交替优化的方法来解问题 (6-3)。在每个参数的优化过程中, 首先固定其他参数,

优化该参数。

6.3.2.1 固定参数 $\{\theta, \phi\}$, 学习参数 B

当参数 $\{\theta, \phi\}$ 固定时, 为了优化参数 B , 首先将优化问题 (6-3) 改写成如下形式:

$$\begin{aligned} \min_B \mathcal{J}(B) &= \|B - F\|_F^2 + \|B - E\|_F^2 \\ &= -2\text{tr}(B^\top(F + E)) + \text{const} \\ \text{subject to: } B &\in \{-1, +1\}^{n \times c}. \end{aligned} \quad (6-4)$$

为了解问题 (6-4), 仅需使得参数 B 与矩阵 $F + E$ 保持相同的符号即可。因此可得:

$$B = \text{sign}(F + E). \quad (6-5)$$

6.3.2.2 固定参数 $\{B, \phi\}$, 学习参数 θ

DCMH 使用反向传播算法来学习卷积神经网络的参数 θ 。首先从训练集合中随机选择包含 n_b 个样本的小批量的样本集合, 根据这个样本集合更新 θ 。具体来说, 对于训练集中的第 j_p 个样本, 使用如下公式计算损失函数关于 e_{j_p} 的梯度:

$$\frac{\partial \mathcal{J}(f_{i_p})}{\partial f_{i_p}} = \frac{\lambda}{c} \sum_{j=1}^n (\tilde{A}_{i_p j} e_j - S_{i_p j} e_j) + 2\gamma(f_{i_p} - b_{i_p}) + 2\nu F^\top \mathbf{1}_n. \quad (6-6)$$

这里, $\tilde{A}_{i_p j} = \frac{1}{1 + e^{-\Phi_{i_p j}}}$ 。

根据上述梯度和链式法则, 可以计算损失函数关于参数 θ 的梯度。然后, 使用反向传播算法来更新参数 θ 。

6.3.2.3 固定参数 $\{B, \theta\}$, 学习参数 ϕ

与 θ 的学习方法类似, DCMH 使用反向传播算法来学习用于文本模态的深度神经网络的参数 ϕ 。首先从训练集合中随机选择包含 n_b 个样本的小批量的样本集合, 根据这个样本集合更新 ϕ 。具体来说, 对于训练集中的第 j_p 个样本,

使用如下公式计算损失函数关于 \mathbf{e}_{j_p} 的梯度：

$$\frac{\partial \mathcal{J}(\mathbf{e}_{j_p})}{\partial \mathbf{e}_{j_p}} = \frac{\lambda}{c} \sum_{i=1}^n (\tilde{A}_{ij_p} \mathbf{f}_i - S_{ij_p} \mathbf{f}_i) + 2\gamma(\mathbf{e}_{j_p} - \mathbf{b}_{j_p}) + 2\nu \mathbf{E}^\top \mathbf{1}_n. \quad (6-7)$$

这里， $\tilde{A}_{ij_p} = \frac{1}{1+e^{-\Phi_{ij_p}}}$ 。

根据上述梯度和链式法则，可以计算损失函数关于参数 ϕ 的梯度。然后，使用反向传播算法来更新参数 ϕ 。

DCMH 模型的优化算法可以简单地概括在算法 6.1 中。

算法 6.1 DCMH 模型学习算法

输入：

图片训练集 \mathbf{X} ，文本训练集 \mathbf{Y} ，跨模态相似度 \mathbf{S} 。

输出：

二值哈希编码 \mathbf{B} ，图片模态的卷积神经网络参数 θ ，文本模态的深度神经网络参数 ϕ 。

- 1: **初始化**：初始化深度神经网络参数 θ 和 ϕ ，批量大小 $n_b = 128$ ，迭代次数： $t_x = \lceil \frac{n}{n_b} \rceil, t_y = \lceil \frac{n}{n_b} \rceil$ 。
 - 2: **repeat**
 - 3: 根据式 (6-5) 更新二值哈希编码 \mathbf{B} ；
 - 4: **for** $t = 1 \mapsto t_x$ **do**
 - 5: 从图片模态的训练集 \mathbf{X} 中随机采 n_b 个样本组成批量集合；
 - 6: 对于批量集合中的所有样本，通过正向传播计算 $\mathbf{f}_i = f(\mathbf{x}_i; \theta)$ ；
 - 7: 根据式 (6-6) 计算参数 \mathbf{f}_i 和 θ 梯度；
 - 8: 使用反向传播算法更新参数 θ ；
 - 9: **end for**
 - 10: **for** $t = 1 \mapsto t_y$ **do**
 - 11: 从文本模态的训练集 \mathbf{Y} 中随机采 n_b 个样本组成批量集合；
 - 12: 对于批量集合中的所有样本，通过正向传播计算 $\mathbf{e}_j = e(\mathbf{y}_j; \phi)$ ；
 - 13: 根据式 (6-7) 计算参数 \mathbf{e}_j 和 ϕ 梯度；
 - 14: 使用反向传播算法更新参数 ϕ ；
 - 15: **end for**
 - 16: **until** 达到指定的循环次数
-

6.3.3 样本外扩展

训练结束后，可以使用学习得到的深度神经网络来为任一不属于训练集中的数据样本生成二值哈希编码表示。具体来说，当给定的图片或文本数据不属于训练集时，使用学习得到的哈希函数来生成二值哈希编码。对于图片 $\mathbf{x}_q \notin \mathbf{X}$ ，使用如下的公式生成二值哈希编码： $\mathbf{u}_q = \text{sign}(f(\mathbf{x}_q; \theta))$ 。

类似地，对于文本 $\mathbf{y}_q \notin \mathbf{Y}$ ，使用如下的公式生成二值哈希编码： $\mathbf{v}_q = \text{sign}(e(\mathbf{y}_q; \phi))$ 。

6.4 基于深度离散隐因子模型的跨模态哈希学习方法 DDLFH

本节从模型、学习算法、随机学习策略和样本外扩展四个方面介绍本章提出的基于深度离散隐因子模型的跨模态哈希 DDLFH。

6.4.1 模型

为了学习能尽量保持原始空间相似性的二值哈希编码，DDLFH 采用与 DCMH 方法相同的目标函数。具体来说，DDLFH 的优化问题定义如下：

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V}, \theta, \phi} \mathcal{J}(\mathbf{U}, \mathbf{V}, \theta, \phi) &= - \sum_{i,j=1}^n [S_{ij} \Omega_{ij} - \log(1 + e^{\Omega_{ij}})] \\ \text{subject to: } \mathbf{U}, \mathbf{V} &\in \{-1, +1\}^{n \times c}, \\ \forall i \in \{1, \dots, n\}, \mathbf{u}_i &= h(\mathbf{x}_i) = \text{sign}(f(\mathbf{x}_i; \theta)), \\ \forall j \in \{1, \dots, n\}, \mathbf{v}_j &= g(\mathbf{y}_j) = \text{sign}(e(\mathbf{y}_j; \phi)). \end{aligned} \quad (6-8)$$

这里，符号定义与 DCMH 方法中符号定义相同，不再赘述。

为了避免二值约束带来的困难，DCMH 方法将损失函数中第一项松弛到实值空间。与 DCMH 方法不同，DDLFH 使用 MAC^[121] 方法，将优化问题改写为如下形式：

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V}, \theta, \phi} \mathcal{J}(\mathbf{U}, \mathbf{V}, \theta, \phi) &= - \sum_{i,j=1}^n [S_{ij} \Omega_{ij} - \log(1 + e^{\Omega_{ij}})] \\ &\quad + \eta(\|\mathbf{U} - \mathbf{F}\|_F^2 + \|\mathbf{V} - \mathbf{E}\|_F^2) \\ \text{subject to: } \mathbf{U}, \mathbf{V} &\in \{-1, +1\}^{n \times c}, \\ \forall i \in \{1, \dots, n\}, \mathbf{f}_i &= f(\mathbf{x}_i; \theta), \\ \forall j \in \{1, \dots, n\}, \mathbf{e}_j &= e(\mathbf{y}_j; \phi). \end{aligned} \quad (6-9)$$

与 DCMH 方法类似，DDLFH 方法也使用两个深度神经网络来完成特征学

习过程。DDL FH 方法中深度神经网络的构造过程与 DCMH 方法的构造方法相同，不再赘述。

对比 DCMH 的优化问题 (6-3) 和 DDLFH 的优化问题 (6-8)，可以看到两个方法均可以在同一个端到端的框架中完成深度特征学习和二值哈希编码学习。不同的是，DCMH 使用跨模态监督信息来直接指导深度特征学习，而 DDLFH 方法则使用跨模态监督信息来直接指导二值哈希编码的学习。

6.4.2 学习算法

本文在待学习参数 $\{U, V, \theta, \phi\}$ 之间采用交替优化的方法来解问题 (6-9)。具体来说，在每个变量的优化过程中，首先固定其他变量，优化该变量。为了学习二值哈希编码 U 和 V ，本章提出一种具有收敛保证的逐比特离散编码算法 BDC。BDC 算法可以在不丢弃二值约束的情况下，直接学习二值哈希编码 U 和 V 。本节同时给出了 BDC 算法的收敛性证明。

6.4.2.1 固定参数 $\{V, \theta, \phi\}$ ，优化参数 U

本节设计了一种逐比特离散编码算法 BDC 来学习二值哈希编码 U 。BDC 算法与第五章中二值哈希编码的学习策略类似，具体来说，BDC 算法依次固定参数 U 中的 $c-1$ 列，优化其中一列二值哈希编码。

首先，目标函数关于变量 U_{*k} 的梯度可根据如下公式计算：

$$\frac{\partial \mathcal{J}(U_{*k})}{\partial U_{*k}} = \frac{\lambda}{c} \sum_{j=1}^n [A_{*j} - S_{*j}] V_{jk} + 2\eta(U_{*k} - F_{*k}), \quad (6-10)$$

这里， $A = [A_{ij}]_{i,j=1}^n$ 。

同时，目标函数关于变量 U_{*k} 的海森矩阵可根据如下公式计算：

$$\frac{\partial^2 \mathcal{J}(U_{*k})}{\partial U_{*k} \partial U_{*k}^\top} = \frac{\lambda^2}{c^2} \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_n \end{bmatrix} + 2\eta I_n.$$

这里， $a_i = \sum_{j=1}^n A_{ij}(1 - A_{ij})$ 。

构造 $\mathcal{J}(\mathbf{U}_{*k})$ 的一个上界为:

$$\begin{aligned}
\tilde{\mathcal{J}}(\mathbf{U}_{*k}) &= \mathcal{J}(\mathbf{U}_{*k}(t)) + [\mathbf{U}_{*k} - \mathbf{U}_{*k}(t)]^\top \frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t) \\
&\quad + \frac{1}{2} [\mathbf{U}_{*k} - \mathbf{U}_{*k}(t)]^\top \mathbf{H} [\mathbf{U}_{*k} - \mathbf{U}_{*k}(t)] \\
&= \mathbf{U}_{*k}^\top \left[\frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H} \mathbf{U}_{*k}(t) \right] + \mathcal{J}(\mathbf{U}_{*k}(t)) - [\mathbf{U}_{*k}(t)]^\top \frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t) \\
&\quad + \frac{1}{2} [\mathbf{U}_{*k}]^\top \mathbf{H} \mathbf{U}_{*k} + [\mathbf{U}_{*k}(t)]^\top \mathbf{H} \mathbf{U}_{*k}(t) \\
&= \mathbf{U}_{*k}^\top \left[\frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H} \mathbf{U}_{*k}(t) \right] + \text{const},
\end{aligned}$$

其中, $\mathbf{U}_{*k}(t)$ 和 $\frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t)$ 分别表示变量 \mathbf{U}_{*k} 和变量 $\frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}$ 在第 t 轮迭代时的取值, 矩阵 \mathbf{H} 的定义为:

$$\mathbf{H} = \left(\frac{\lambda^2 n}{4c^2} + 2\eta \right) \mathbf{I}_n.$$

根据上述定义, 可得如下定理:

定理 6-1 函数 $\tilde{\mathcal{J}}(\mathbf{U}_{*k})$ 是函数 $\mathcal{J}(\mathbf{U}_{*k})$ 的一个上界。

定理 (6-1) 的证明: 首先, 函数 $\tilde{\mathcal{J}}(\mathbf{U}_{*k})$ 是一个曲率有界的凸函数。由于 A_{ij} 满足 $0 < A_{ij} < 1$, 能得到:

$$0 \leq A_{ij}(1 - A_{ij}) \leq \frac{1}{4}.$$

于是可得:

$$\mathbf{H} \succeq \frac{\partial^2 \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k} \partial \mathbf{U}_{*k}^\top}.$$

根据引理 (5-2), 可得:

$$\mathcal{J}(\mathbf{U}_{*k}) \leq \tilde{\mathcal{J}}(\mathbf{U}_{*k}),$$

上式意味着, 函数 $\tilde{\mathcal{J}}(\mathbf{U}_{*k})$ 是函数 $\mathcal{J}(\mathbf{U}_{*k})$ 的上界。 □

根据定理 (6-1), 可得如下的优化问题:

$$\begin{aligned}
\min_{\mathbf{U}_{*k}} \tilde{\mathcal{J}}(\mathbf{U}_{*k}) &= \mathbf{U}_{*k}^\top \left[\frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H} \mathbf{U}_{*k}(t) \right] \\
\text{subject to: } \mathbf{U}_{*k} &\in \{-1, +1\}^n
\end{aligned} \tag{6-11}$$

定义向量 $\mathbf{p} = \frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H}\mathbf{U}_{*k}(t)$ 。对于任意 l , $U_{lk} \in \{-1, +1\}$, 为了最小化 $\tilde{\mathcal{J}}(\mathbf{U}_{*k})$, 只需要在 $p_l \geq 0$ 时令 $U_{lk} = -1$, 在 $p_l < 0$ 时令 $U_{lk} = 1$ 。因此可得问题 (6-11) 的最优解为:

$$\begin{aligned} \mathbf{U}_{*k} &= \mathbf{sign}\left(\mathbf{H}\mathbf{U}_{*k}(t) - \frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t)\right). \\ &= \mathbf{sign}\left(\mathbf{U}_{*k}(t) - \gamma \frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t)\right). \end{aligned} \quad (6-12)$$

这里, $\gamma = \frac{1}{\frac{\lambda^2 n}{4c^2} + 2\eta}$ 。然后, 使用这个最优解去得到下一轮迭代的 \mathbf{U}_{*k} 值, 即:

$$\mathbf{U}_{*k}(t+1) = \mathbf{sign}\left(\mathbf{U}_{*k}(t) - \gamma \frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t)\right). \quad (6-13)$$

逐比特离散编码算法 BDC 总结在算法 6.2 中。这里将第 t 轮二值哈希编码 $\mathbf{U}(t)$ 的更新过程记为如下形式:

$$\mathbf{U}(t+1) = \text{BDC}(t, \mathbf{U}(t), \mathbf{S}, \mathbf{V}, \mathbf{F}).$$

算法 6.2 BDC 算法

输入:

迭代轮数 t , 跨模态相似度 \mathbf{S} , 文本模态的二值哈希编码 \mathbf{V} , 图片模态的二值哈希编码 $\mathbf{U}(t)$, 图片模态的实值特征 \mathbf{F} 。

输出:

更新后的二值哈希编码 $\mathbf{U}(t+1)$ 。

- 1: **初始化:** $\gamma = \frac{1}{\frac{\lambda^2 n}{4c^2} + 2\eta}$ 。
 - 2: **for** $k = 1 \mapsto c$ **do**
 - 3: 根据公式 (6-10) 计算梯度 $\frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}}(t)$;
 - 4: 根据公式 (6-13) 更新 $\mathbf{U}_{*k}(t+1)$;
 - 5: **end for**
-

然后有如下的定理:

定理 6-2 关于二值变量 \mathbf{U} 的逐比特离散编码算法 BDC 收敛。

定理 (6-2) 的证明: 由定理 (6-1), 可得:

$$\mathcal{J}(\mathbf{U}_{*k}) \leq \tilde{\mathcal{J}}(\mathbf{U}_{*k}).$$

同时根据 $\tilde{\mathcal{J}}(\mathbf{U}_{*k})$ 的定义, 有:

$$\tilde{\mathcal{J}}(\mathbf{U}_{*k}(t)) = \mathcal{J}(\mathbf{U}_{*k}(t)).$$

由于 $\mathbf{U}_{*k}(t+1)$ 是问题 (6-11) 的最优解, 可得:

$$\tilde{\mathcal{J}}(\mathbf{U}_{*k}(t+1)) \leq \tilde{\mathcal{J}}(\mathbf{U}_{*k}(t)).$$

由此可得:

$$\mathcal{J}(\mathbf{U}_{*k}(t+1)) \leq \tilde{\mathcal{J}}(\mathbf{U}_{*k}(t+1)) \leq \tilde{\mathcal{J}}(\mathbf{U}_{*k}(t)) = \mathcal{J}(\mathbf{U}_{*k}(t)).$$

因此在优化过程中, 总能保证目标函数的值不会上升, 即: $\mathcal{J}(\mathbf{U}_{*k}(t+1)) \leq \mathcal{J}(\mathbf{U}_{*k}(t))$ 。同时考虑到目标函数 $\mathcal{J}(\mathbf{U})$ 的值大于 0, 可得逐比特离散编码算法 BDC 是收敛的。□

6.4.2.2 固定参数 $\{\mathbf{U}, \theta, \phi\}$, 优化参数 \mathbf{V}

\mathbf{V} 的优化和 \mathbf{U} 的优化类似, 即, 采用一种逐列优化的方法来学习二值哈希编码 \mathbf{V} , 当优化二值哈希编码的第 k 列 \mathbf{V}_{*k} 时, 固定 \mathbf{V} 的其他列。具体来说, 使用逐比特离散编码算法 BDC 来学习二值哈希编码 \mathbf{V} 。为了学习二值哈希编码 \mathbf{V} , 仅需将 BDC 算法的输入改为: 迭代轮次 t , 跨模态相似度 \mathbf{S} , 图片模态的二值哈希编码 \mathbf{U} , 文本模态的二值哈希编码 $\mathbf{V}(t)$, 文本模态的实值特征 \mathbf{E} 。并将梯度计算公式替换为目标函数 $\mathcal{J}(\mathbf{V}_{*k})$ 关于变量 \mathbf{V}_{*k} 的梯度, 即:

$$\frac{\partial \mathcal{J}(\mathbf{V}_{*k})}{\partial \mathbf{V}_{*k}} = \frac{\lambda}{c} \sum_{i=1}^n [\mathbf{A}_{i*}^\top - \mathbf{S}_{i*}^\top] U_{ik} + 2\eta(\mathbf{V}_{*k} - \mathbf{E}_{*k}). \quad (6-14)$$

同时, 二值哈希编码 $\mathbf{V}(t+1)$ 的更新公式替换为:

$$\mathbf{V}_{*k}(t+1) = \mathbf{sign}(\mathbf{V}_{*k}(t) - \gamma \frac{\partial \mathcal{J}(\mathbf{V}_{*k})}{\partial \mathbf{V}_{*k}}(t)). \quad (6-15)$$

这里, $\gamma = \frac{1}{\frac{\lambda^2 n}{4c^2} + 2\eta}$ 。并将第 t 轮二值哈希编码 $\mathbf{V}(t)$ 的更新过程记为如下形式:

$$\mathbf{V}(t+1) = \text{BDC}(t, \mathbf{V}(t), \mathbf{S}, \mathbf{U}, \mathbf{E}).$$

根据 BDC 算法的收敛性，可得：

$$\mathcal{J}(\mathbf{V}(t+1)) \leq \mathcal{J}(\mathbf{V}(t)).$$

6.4.2.3 固定参数 $\{U, V, \phi\}$ ，优化参数 θ

首先将目标函数重写为：

$$\begin{aligned} \min_{\theta} \mathcal{J}(\theta) &= \eta \|\mathbf{F} - \mathbf{U}\|_F^2 = \eta \sum_{i=1}^n \|\mathbf{f}_i - \mathbf{u}_i\|_2^2 \\ \text{subject to: } &\forall i \in \{1, \dots, n\}, \mathbf{f}_i = f(\mathbf{x}_i; \theta). \end{aligned} \quad (6-16)$$

DDL FH 使用反向传播算法来优化变量 θ 。具体来说，为了优化问题 (6-16)，随机从训练集中选择包含 n_b 个训练样本的小批量样本集。根据这个样本集合，计算损失函数 $\mathcal{J}(\theta)$ 关于参数 θ 的梯度。对于样本集合中的第 i_p 个样本，计算损失函数关于 \mathbf{f}_{i_p} 的梯度：

$$\frac{\partial \mathcal{J}(\mathbf{f}_{i_p})}{\partial \mathbf{f}_{i_p}} = 2\eta(\mathbf{f}_{i_p} - \mathbf{u}_{i_p}). \quad (6-17)$$

根据上述梯度和链式法则，可以计算损失函数关于参数 θ 的梯度。然后，使用反向传播算法来更新参数 θ 。

6.4.2.4 固定 $\{U, V, \theta\}$ ，优化参数 ϕ

DDL FH 使用与参数 θ 的优化类似的算法来优化变量 ϕ 。首先将目标函数重写为：

$$\begin{aligned} \min_{\phi} \mathcal{J}(\phi) &= \eta \|\mathbf{E} - \mathbf{V}\|_F^2 = \eta \sum_{j=1}^n \|\mathbf{e}_j - \mathbf{v}_j\|_2^2 \\ \text{subject to: } &\forall j \in \{1, \dots, n\}, \mathbf{e}_j = e(\mathbf{y}_j; \phi). \end{aligned} \quad (6-18)$$

使用反向传播算法来优化变量 ϕ 。具体来说，为了优化问题 (6-18)，随机从训练集中选择包含 n_b 个训练样本的小批量样本集。根据这个样本集合，计算损失函数 $\mathcal{J}(\phi)$ 关于参数 ϕ 的梯度。对于样本集合中的第 j_p 个样本，计算损失

函数关于 e_{j_p} 的梯度:

$$\frac{\partial \mathcal{J}(e_{j_p})}{\partial e_{j_p}} = 2\eta(e_{j_p} - v_{j_p}). \quad (6-19)$$

根据上述梯度和链式法则, 可以计算损失函数关于参数 ϕ 的梯度。然后, 使用反向传播算法来更新参数 ϕ 。

DDL FH 模型的算法可简单地概括在算法 6.3 中。

算法 6.3 DDL FH 模型学习算法

输入:

图片模态的训练样本集 $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$, 文本模态的训练样本集 $\mathbf{Y} = \{\mathbf{y}_j\}_{j=1}^n$, 跨模态相似度 \mathbf{S} , 二值哈希编码长度 c 。

输出:

二值哈希编码 \mathbf{U} 和 \mathbf{V} , 神经网络参数 θ 和 ϕ 。

- 1: **初始化:** 初始化 \mathbf{U} 和 \mathbf{V} , 初始化神经网络参数 θ 和 ϕ , 初始化批量样本集大小 n_b , 迭代次数: T_{out} , $t_x = \lceil \frac{n}{n_b} \rceil$, $t_y = \lceil \frac{n}{n_b} \rceil$ 。
 - 2: **for** $t = 1 \mapsto T_{out}$ **do**
 - 3: 使用 BDC 算法更新二值哈希编码 $\mathbf{U}(t)$ 。
 - 4: 使用 BDC 算法更新二值哈希编码 $\mathbf{V}(t)$ 。
 - 5: **for** $s = 1 \mapsto t_x$ **do**
 - 6: 从图片模态的训练集 \mathbf{X} 中随机采 n_b 个样本组成批量集合;
 - 7: 对于批量集合中的所有样本, 通过正向传播计算 $\mathbf{f}_i = f(\mathbf{x}_i; \theta)$;
 - 8: 根据式 (6-17) 计算参数 \mathbf{f}_i 和 θ 的梯度;
 - 9: 使用反向传播算法更新参数 θ ;
 - 10: **end for**
 - 11: **for** $s = 1 \mapsto t_y$ **do**
 - 12: 从文本模态的训练集 \mathbf{Y} 中随机采 n_b 个样本组成批量集合;
 - 13: 对于批量集合中的所有样本, 通过正向传播计算 $\mathbf{e}_j = e(\mathbf{y}_j; \phi)$;
 - 14: 根据式 (6-19) 计算参数 \mathbf{e}_j 和 ϕ 的梯度;
 - 15: 使用反向传播算法更新参数 ϕ ;
 - 16: **end for**
 - 17: **end for**
-

6.4.3 随机学习策略

根据目标函数关于 \mathbf{U}_{*k} 的梯度计算公式 (6-10) 和关于 \mathbf{V}_{*k} 的梯度计算公式 (6-14), 可以看到算法 6.3 的计算复杂度为 $\mathcal{O}(n^2)$ 。为了避免平方级别的复杂度, 本章采用与 DLFH 类似的随机采样策略设计了新的算法。具体来说, 使用随机选择的 m 列和 m 行的跨模态相似度 \mathbf{S} 来计算目标函数关于 \mathbf{U}_{*k} 和 \mathbf{V}_{*k} 的

梯度。基于这种采样策略，可将梯度的公式重写如下：

$$\begin{aligned}\frac{\partial \mathcal{J}(\mathbf{U}_{*k})}{\partial \mathbf{U}_{*k}} &= \frac{\lambda}{c} \sum_{p=1}^m [\mathbf{A}_{*j_p} - \mathbf{S}_{*j_p}] U_{j_pk} + 2\eta(\mathbf{U}_{*k} - \mathbf{F}_{*k}), \\ \frac{\partial \mathcal{J}(\mathbf{V}_{*k})}{\partial \mathbf{V}_{*k}} &= \frac{\lambda}{c} \sum_{p=1}^m [\mathbf{A}_{i_p*}^\top - \mathbf{S}_{i_p*}^\top] V_{i_pk} + 2\eta(\mathbf{V}_{*k} - \mathbf{E}_{*k}).\end{aligned}$$

使用上式计算梯度的复杂度为 $\mathcal{O}(n)$ 。为了得到基于采样策略的算法，仅需要把上述公式替换算法 6.3 中损失函数关于 \mathbf{U}_{*k} 和 \mathbf{V}_{*k} 的梯度计算公式即可。

6.4.4 样本外扩展

训练结束后，可以使用学习得到的神经网络来为任一不属于训练集中的数据样本生成二值哈希编码表示。具体来说，当给定的图片或者文本数据不属于训练集时，使用哈希函数来生成二值哈希编码。对于图片 $\mathbf{x}_q \notin \mathbf{X}$ ，使用如下的公式来生成二值哈希编码：

$$\mathbf{u}_q = \text{sign}(f(\mathbf{x}_q; \theta)).$$

类似地，对于文本 $\mathbf{y}_q \notin \mathbf{Y}$ ，使用如下的公式来生成二值哈希编码：

$$\mathbf{v}_q = \text{sign}(e(\mathbf{y}_q; \phi)).$$

6.5 DCMH 方法的实验验证

本章使用两个跨模态数据集验证了本章提出的 DCMH 方法的有效性。实验使用的服务器配置为：Intel(R) Xeon(R) CPU E5-2620 v4@2.1GHz，8 核心；128G 内存；4 张 Titan Xp 显卡。所有深度跨模态哈希学习方法均使用显卡进行实验。

6.5.1 实验设置

6.5.1.1 数据集

本章使用 IAPRTC12^[116]、FLICKR25K^[110] 数据集进行实验。IAPRTC12 和 FLICKR25K 数据集的介绍可以参考第二章第 2.2 节。

对于所有的数据集，如果图片 \mathbf{x}_i 和文本 \mathbf{y}_j 至少有一个共享的类别标签，定义他们相似，即 $S_{ij} = 1$ ，否则，如果图片 \mathbf{x}_i 和文本 \mathbf{y}_j 没有一个共享的类别标签，定义他们不相似，即 $S_{ij} = 0$ 。

6.5.1.2 评价标准与对比方法

本节设计了实验来验证本章提出的 DCMH 方法的有效性。具体来说，本节选择了五个基准方法来进行实验，包括：两个无监督跨模态哈希学习方法，即：CCAITQ^[21] 和 CMFH^[50]；三个监督跨模态哈希学习方法，即：SePH_{km}^[53]^①、DLFH^[57] 和 KDLFH^[57]。对于 SePH_{km} 和 KDLFH 方法，设置核基的数目为 500。对于所有基于深度特征学习的算法，为了完成图片模态的深度特征学习，实验中采用在 ImageNet 数据集上预训练过的 CNNF^[119] 作为图片模态的基础网络架构。在 CNNF 网络的基础上，增加一层全连接层作为新的深度神经网络来完成深度特征学习。该深度神经网络与第四章中的深度神经网络架构类似，其详细配置如表 4-1 所示。为了完成文本模态的深度特征学习，实验中采用一个两层的多层感知机网络作为文本模态的基础网络架构，其中中间层的隐变量数目为 4096，同时，使用 ReLU 函数作为激活函数^[61]。该深度神经网络的详细配置如表 6-1 所示。对于 DCMH 方法，由于使用松弛策略来学习具有较强的表达能力的实值表示，因此设置 DCMH 方法的超参数 $\lambda = \frac{1}{2c}$ 。同时设置超参数 $\gamma = 1$ ，设置超参数 $\nu = 1$ 。第 6.5.4 节给出了 DCMH 方法的超参数 γ 和超参数 ν 的敏感性实验。此外，设置 DCMH 方法的小批量样本集大小 $n_b = 128$ ，使用 SGD 方法来优化深度神经网络参数，设置学习率的范围为： $[10^{-6}, 10^{-1}]$ ，迭代轮数为 500。

表 6-1: DCMH 方法中用于文本数据深度特征学习的深度神经网络配置细节

网络层	网络结构配置
full1	4096
full2	二值哈希编码长度: c

对于 IAPRTC12 数据集，随机选择了 2000 个样本构造查询样本集，剩余的所有样本作为数据库样本集。对于 FLICKR25K 数据集，随机选择了 2000 个样

^①与第五章不同，这里不再选择 SePH_{rnd} 作为基准方法，原因是实验证明，在大多数情况下，SePH_{km} 的检索精度优于 SePH_{rnd}。

本构造查询样本集，剩余的所有样本作为数据库样本集。对于 IAPRCTC12 数据集和 FLICKR25K 数据集，DCMH 方法的验证实验使用了采样的策略来构造训练集。具体来说，实验中从两个数据集的数据库集随机选择 10000 个样本作为训练集。本章提出的 DCMH 方法和所有基准方法均使用这个采样的训练集进行训练。

本章使用海明排序和哈希表查询来评价本章提出的 DCMH 方法。对于海明排序，本章汇报了在返回序列截断位置的平均查准率均值 $cMAP(K)$ 。对于哈希表查询，汇报了在海明球 R 的查准-查全率曲线。平均查准率均值 $cMAP(K)$ 、在海明球 R 的查准率 ($rPre(R)$) 和查全率 ($rRec(R)$) 的计算参见第二章第 2.2 小节。

6.5.2 性能对比

6.5.2.1 海明排序

表 6-2 和表 6-3 中分别给出了使用采样训练集进行训练时，DCMH 方法和所有基准方法在 IAPRCTC12 数据集和 FLICKR25K 数据集上的实验结果。在表 6-2 和表 6-3 中，最好的结果使用加粗字体标注。从表 6-2 和表 6-3 中可以看到，当使用采样训练集进行训练时，与非深度跨模态哈希学习方法相比，DCMH 方法在大多数情况下都达到了更高的检索精度。

表 6-2: DCMH 方法和基准方法在 IAPRCTC12 数据集上的平均查准率均值

方法	$I \rightarrow T$			$T \rightarrow I$		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
CCAITQ	0.3034	0.3079	0.3111	0.3035	0.3082	0.3120
CMFH	0.3931	0.3996	0.3966	0.3923	0.3977	0.3978
SePH _{km}	0.4549	0.4664	0.4774	0.4511	0.4641	0.4740
DLFH	0.4061	0.4328	0.4551	0.4091	0.4514	0.4757
KDLFH	0.4450	0.4738	0.5048	0.4403	0.4814	0.5058
DCMH	0.4526	0.4732	0.4844	0.5185	0.5378	0.5468

表 6-3: DCMH 方法和基准方法在 FLICKR25K 数据集上的平均查准率均值

方法	$I \rightarrow T$			$T \rightarrow I$		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
CCAITQ	0.5568	0.5609	0.5620	0.5578	0.5618	0.5618
CMFH	0.5922	0.5990	0.6045	0.5910	0.5981	0.6037
SePH _{km}	0.7052	0.7092	0.7119	0.7099	0.7124	0.7165
DLFH	0.6701	0.6932	0.7000	0.7161	0.7385	0.7503
KDLFH	0.6957	0.7162	0.7286	0.7174	0.7352	0.7418
DCMH	0.7410	0.7465	0.7485	0.7827	0.7900	0.7932

6.5.2.2 哈希表查询

图 6-1 中展示了使用采样训练集进行训练时，在 IAPRTC12 和 FLICKR25K 数据集上的查准-查全率曲线的结果。这个实验中，二值哈希编码长度设置为 16 比特，其余比特长度的结果类似。从图 6-1 中可以看到，当使用采样训练集进行训练时，在大部分情况下 DCMH 方法取得了最高的检索精度。

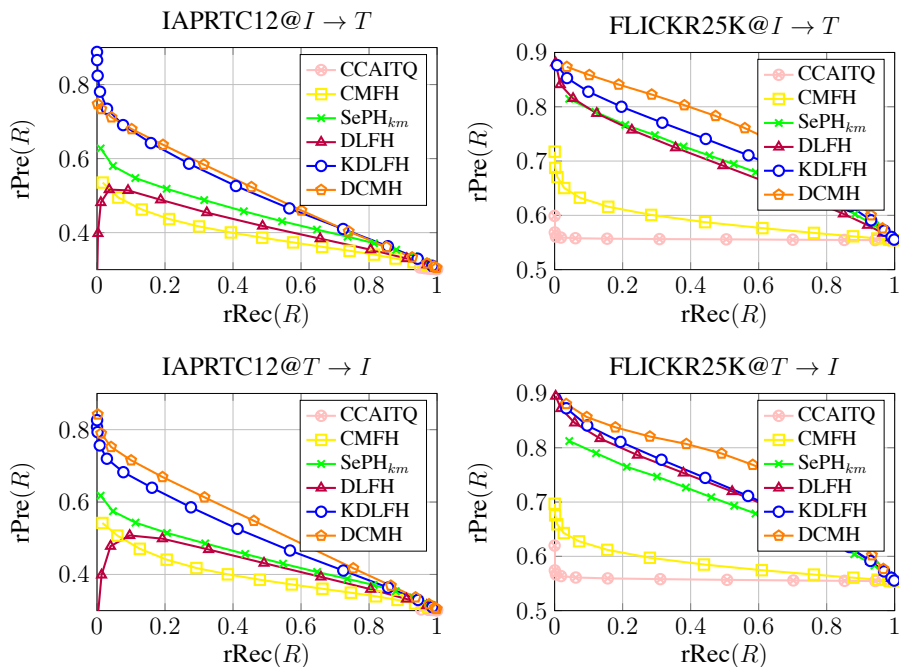


图 6-1: DCMH 方法和基准方法的查准-查全率曲线（采样训练）

6.5.3 有效性验证实验

本节通过有效性验证实验证明了 DCMH 方法的有效性。对于 DCMH 方法，本节设计了三个 DCMH 方法的变体来证明深度特征学习的重要性。第一个方法为 DCMH-I，表示使用和 DCMH 方法相同的模型和算法，但训练过程中固定图片模态的深度神经网络基础网络架构的参数。第二个方法为 DCMH-T，DCMH-T 表示文本模态的多层感知机模型被替换为一个线性投影模型，图片模态的模型和优化算法与 DCMH 方法相同。DCMH-IT 表示文本模态的多层感知机模型被替换为一个线性投影模型，同时训练过程中固定图片模态的深度神经网络基础网络架构的参数。

图 6-2 中给出了这几个方法的实验结果。从结果中可以看到，DCMH-I 和 DCMH-T 方法要优于 DCMH-IT，而 DCMH 方法要优于 DCMH-I、DCMH-T 和 DCMH-IT 方法。这证明了深度特征学习在跨模态哈希中的重要性。

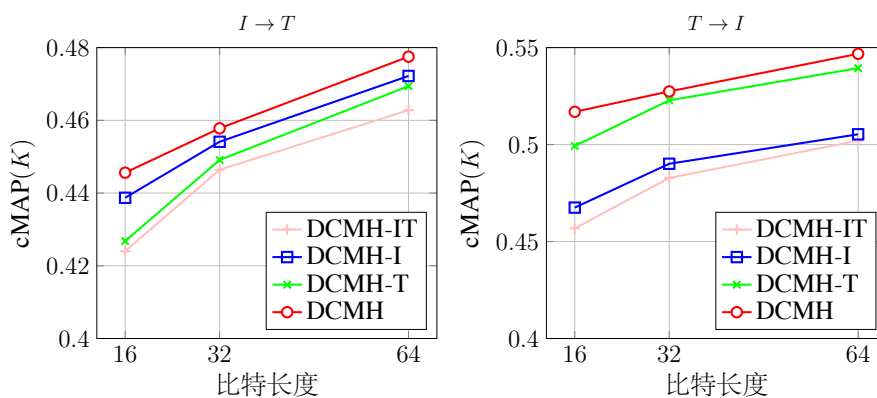
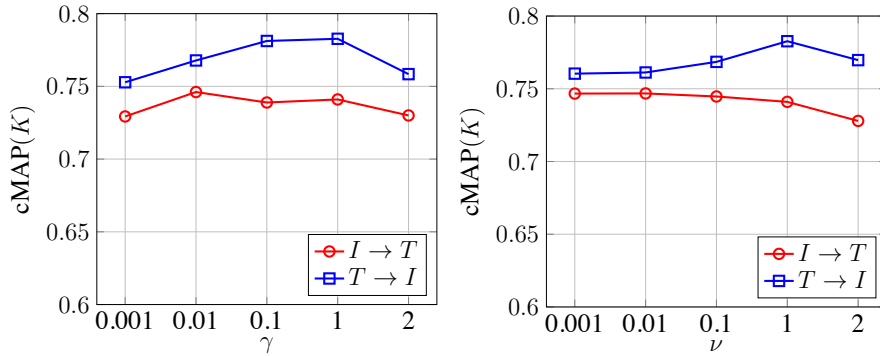


图 6-2: DCMH 方法有效性验证实验

6.5.4 超参数实验

本节通过实验验证了 DCMH 的超参数 γ 、 ν 的敏感性。具体来说， γ 和 ν 是 DCMH 方法最重要的超参数。图 6-3 给出了 DCMH 方法的超参数 γ 和 ν 的敏感性实验。从图 6-3 中可以看到，随着 γ 和 ν 增加，几乎所有情况下检索精度都是先变好后变差。对于超参数 γ ，当 $0.01 \leq \gamma \leq 2$ 时，DCMH 方法对于超参数 γ 不敏感。对于超参数 ν ，当 $0.01 \leq \nu \leq 2$ 时，DCMH 方法对于超参数 ν 不敏感。

图 6-3: DCMH 方法对超参数 γ 和 ν 的敏感性实验

6.6 DDLFH 方法的实验验证

本章使用两个跨模态数据集验证本章提出的 DDLFH 方法的有效性。实验中所使用的硬件配置与 DCMH 方法的实验设置相同。

6.6.1 实验设置

6.6.1.1 数据集

本章使用 IAPRTC12^[116]、FLICKR25K^[110] 数据集进行实验。IAPRTC12 和 FLICKR25K 数据集的介绍可以参考第二章第 2.2 节。相似样本对的定义与 DCMH 实验中定义相同。

6.6.1.2 评价标准与对比方法

本节设计了实验来验证 DDLFH 方法的有效性。基准方法的选择与 DCMH 实验中的选择相同，即本节选择了 CCAITQ^[21]、CMFH^[50]、SePH_{km}^[53]、DLFH^[57] 和 KDLFH^[57] 作为基准方法。这些方法的设置与 DCMH 实验中设置相同，不再赘述。对于 DDLFH 方法，当训练集大小不同时，考虑到目标函数中两项损失中求和项的数目不同，本章采用参数正则化方法^[75] 将超参数重定义为： $\eta_m = \frac{n}{|\mathcal{S}|} \eta$ 。实验中，超参数 η_m 的值从区间 $[0.001, 0.01]$ 中选取。第 6.6.4 节给出了 DDLFH 方法的超参数 η_m 的敏感性实验。此外，设置参数 $\lambda = 8$ ，采样数目 $m = c$ ，迭代轮数 $T_{out} = 100$ ，小批量样本集大小 $n_b = 200$ ，初始学习率为 0.0001，并在每 20 轮后将学习率降低为当前学习率的 0.1。DDL FH 方法使用 ADAM 算法来学习深度神经网络的参数。对于二值哈希编码

变量 U 和 V ，首先从均匀分布中采样生成随机矩阵，然后对该矩阵进行取符号操作来得到初始化的二值哈希编码变量 U 和 V 。

数据集的划分与 DCMH 方法中的划分相同。由于部分基准方法的复杂度较高，本节使用两种策略来构造训练集。首先从数据库集随机选择 10000 个样本作为训练集，并在所有方法上进行了测试。然后使用全部数据库样本作为训练集，在除去 SePH_{km} 的所有方法上进行了测试。这里，不在 SePH_{km} 上使用全部训练集进行训练的原因是该方法的复杂度均为 $\mathcal{O}(n^2)$ 。当使用全部数据库样本进行训练时， SePH 方法会遇到内存溢出等错误。在实际应用中，这个方法不适合使用较大训练集进行训练。

本章使用海明排序和哈希表查询来评价 DDLFH 方法。对于海明排序，本章汇报了在返回序列截断位置的平均查准率均值 $\text{cMAP}(K)$ 。对于哈希表查询，汇报了在海明球 R 的查准-查全率曲线。

6.6.2 性能对比

6.6.2.1 海明排序

表 6-4 和表 6-5 中分别给出了使用采样训练集进行训练时，DDL FH 方法和所有基准方法在 IAPRTC12 数据集和 FLICKR25K 数据集上的实验结果。在表 6-4 和表 6-5 中，最好的结果使用加粗字体标注。从表 6-4 和表 6-5 中可以看到，当使用采样训练集进行训练时，与所有基准方法相比，DDL FH 方法都达到了更高的检索精度。

表 6-4: DDLFH 方法和基准方法在 IAPRTC12 数据集上的平均查准率均值（采样训练）

方法	$I \rightarrow T$			$T \rightarrow I$		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
CCAITQ	0.3034	0.3079	0.3111	0.3035	0.3082	0.3120
CMFH	0.3931	0.3996	0.3966	0.3923	0.3977	0.3978
SePH_{km}	0.4549	0.4664	0.4774	0.4511	0.4641	0.4740
DLFH	0.4061	0.4328	0.4551	0.4091	0.4514	0.4757
KDLFH	0.4450	0.4738	0.5048	0.4403	0.4814	0.5058
DDL FH	0.5589	0.5677	0.5921	0.5806	0.5800	0.6181

表 6-5: DDLFH 方法和基准方法在 FLICKR25K 数据集上的平均查准率均值 (采样训练)

方法	$I \rightarrow T$			$T \rightarrow I$		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
CCAITQ	0.5568	0.5609	0.5620	0.5578	0.5618	0.5618
CMFH	0.5922	0.5990	0.6045	0.5910	0.5981	0.6037
SePH _{km}	0.7052	0.7092	0.7119	0.7099	0.7124	0.7165
DLFH	0.6701	0.6932	0.7000	0.7161	0.7385	0.7503
KDLFH	0.6957	0.7162	0.7286	0.7174	0.7352	0.7418
DDLFH	0.7712	0.7840	0.7892	0.7995	0.8008	0.8136

表 6-6 和表 6-7 中分别给出了使用全部数据库样本进行训练时, DDLFH 方法和所有基准方法在 IAPRTC12 数据集和 FLICKR25K 数据集上的实验结果。在表 6-6 和表 6-7 中, 最好的结果使用加粗字体标注。从表 6-6 和表 6-7 中可以看到, 当使用全部数据库样本进行训练时, 与所有的基准方法相比, DDLFH 方法在大部分情况下能达到更高的检索精度。

表 6-6: DDLFH 方法和基准方法在 IAPRTC12 数据集平均查准率均值 (全集训练)

方法	$I \rightarrow T$			$T \rightarrow I$		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
CCAITQ	0.3329	0.3597	0.3700	0.3320	0.3613	0.3749
CMFH	0.4102	0.4208	0.4180	0.4063	0.4210	0.4156
DLFH	0.4252	0.4876	0.5320	0.4336	0.5012	0.5617
KDLFH	0.4775	0.5555	0.6048	0.4876	0.5661	0.6267
DDLFH	0.5488	0.6118	0.6524	0.5386	0.5984	0.6579

6.6.2.2 哈希表查询

图 6-4 中展示了使用采样训练集进行训练时, 在 IAPRTC12 数据集和 FLICKR25K 数据集上的查准-查全率曲线的结果。图 6-5 中展示了使用全部数据库样本进行训练时, 在 IAPRTC12 数据集和 FLICKR25K 数据集上的查

表 6-7: DDLFH 方法和基准方法在 FLICKR25K 数据集平均查准率均值 (全集训练)

方法	$I \rightarrow T$			$T \rightarrow I$		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
CCAITQ	0.5855	0.5893	0.5787	0.5902	0.5875	0.5821
CMFH	0.5937	0.6067	0.6101	0.5934	0.6065	0.6083
DLFH	0.7794	0.7944	0.7986	0.7900	0.8201	0.8314
KDLFH	0.8591	0.8566	0.8659	0.8203	0.8605	0.8692
DDLFH	0.8538	0.8738	0.8820	0.8468	0.8627	0.8700

准-查全率曲线的结果。这个实验中，二值哈希编码长度设置为 16 比特，其余比特长度的结果类似。从图 6-4 和图 6-5 中可以看到，当使用采样训练集进行训练时，在两个数据集上 DDLFH 方法均取得了最高的检索精度。当使用全部数据库集合进行训练时，在 IAPRTC12 数据集上，本章提出的 DDLFH 方法取得了最高的检索精度，而在 FLICKR25K 数据集，DDLFH 方法的检索精度稍低于 KDLFH 方法的检索精度。

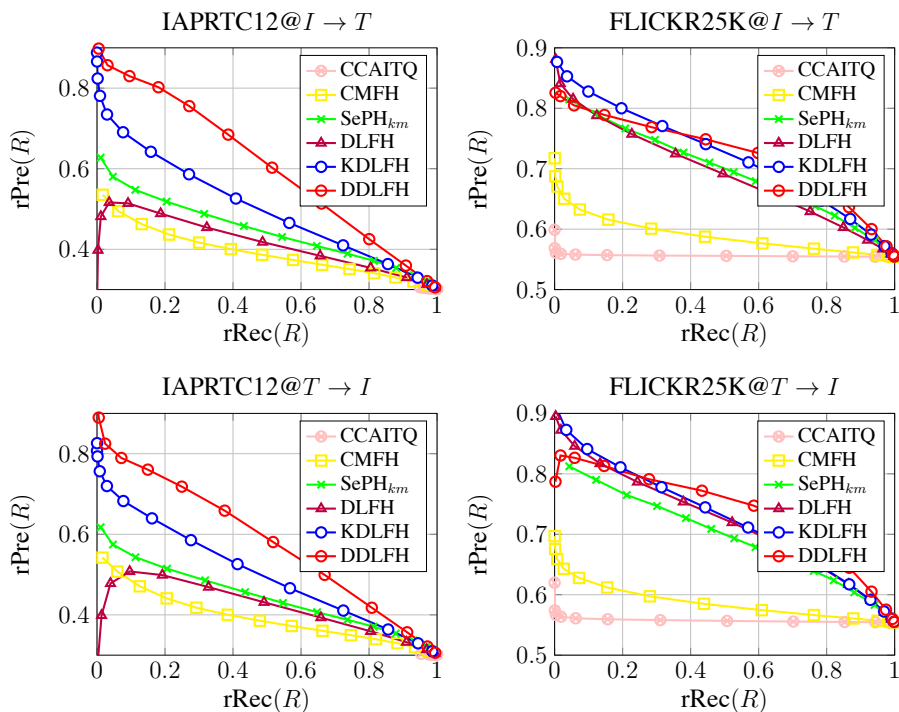


图 6-4: DDLFH 方法和基准方法的查准-查全率曲线 (采样训练)

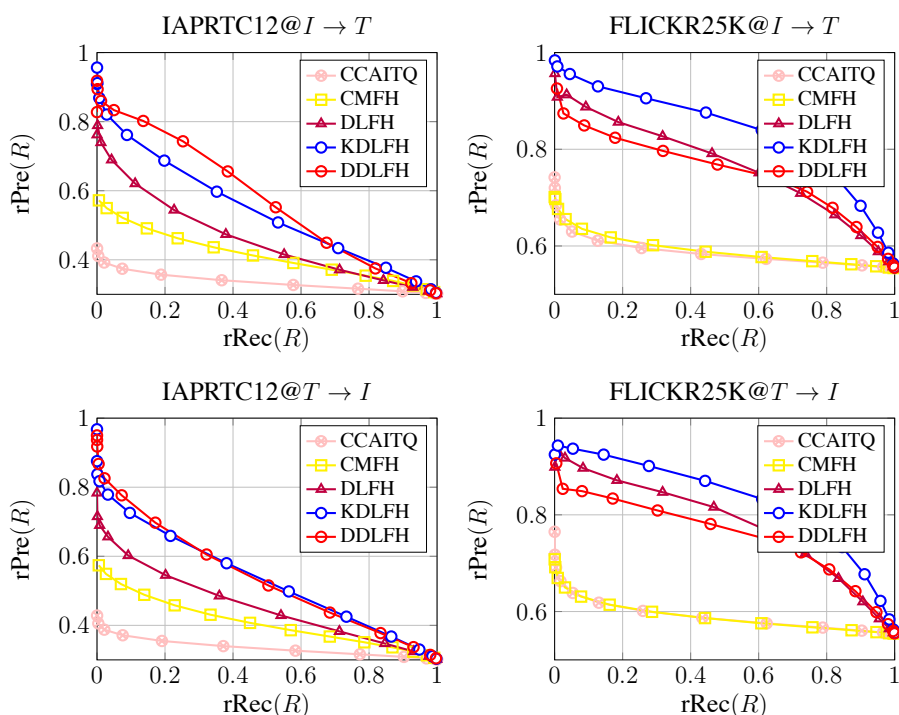


图 6-5: DDLFH 方法和基准方法的查准-查全率曲线 (全集训练)

6.6.2.3 DLFH、DCMH 和 DDLFH 的对比

本节进一步深入对比了第五章提出的 DLFH、本章提出的 DCMH 和 DDLFH 方法。在表 6-8 中列出了三个数据集在 FLICKR25K 数据集上的实验结果，最好的结果使用加粗字体标注。

表 6-8 中按照是否对训练集进行采样将结果分为两个部分。从表 6-8 中可以看到，当使用采样数据集训练时，DCMH 方法能达到比 DLFH、KDLFH 方法更高的检索精度。同时，DDLHF 方法能达到比 DCMH、DLFH 和 DDLFH 方法更高的检索精度。当使用全部数据集进行训练时，DDLHF 方法能达到比 DLFH 和 DDLHF 方法更高的检索精度。此外，DLFH、KDLFH 使用全部数据库进行训练时能达到比 DCMH 使用采样训练集训练时更高的精度。同时，尽管 DLFH、KDLFH 方法使用全部数据库集合进行训练，但 DLFH、KDLFH 仍然能达到比采用采样训练集进行训练的 DCMH 方法更高的检索精度。因此，现实应用中，如果要求高效地训练，可以选择 DLFH、KDLFH 等方法建模。如果有足够的资源进行训练，或者对训练效率没有太高的要求，可以选择 DCMH 或者 DDLHF 方法进行建模。

表 6-8: DLFH、DCMH 和 DDLFH 在 FLICKR25K 数据集上的对比

方法	训练集大小	训练时间	检索精度	
			$I \rightarrow T$	$T \rightarrow I$
DCMH	10000	33736.7	0.7441	0.7848
DLFH	10000	2.96	0.6701	0.7161
KDLFH	10000	400.83	0.6957	0.7174
DDLFH	10000	15868.9	0.7712	0.7995
DLFH	18015	4.46	0.7794	0.7900
KDLFH	18015	777.59	0.8591	0.8203
DDLFH	18015	30521.3	0.8538	0.8468

6.6.3 有效性验证实验

本节通过有效性验证实验证明了 DDLFH 方法的有效性。具体来说，本节使用 DCMH 方法、DLFH 方法、KDLFH 方法和一个 DDLFH 方法的变体来验证 DDLFH 的有效性。DDLFH 的变体是一个使用两步学习策略来实现的跨模态哈希学习方法。具体来说，与 DLFH 方法相同的算法来学习多模态数据的二值哈希编码 \mathbf{U} 和 \mathbf{V} 。学习得到二值哈希编码 \mathbf{U} 和 \mathbf{V} 后，使用两个与 DDLFH 模型中相同的神经网络来作为哈希函数，并优化神经网络输出与二值哈希编码 \mathbf{U} 和 \mathbf{V} 之间的平方损失来分别拟合学习得到的二值哈希编码。具体来说，通过优化如下的两个目标函数将数据投影到二值空间：

$$\mathcal{J}_x(\theta) = \sum_{i=1}^n (\mathbf{u}_i - f(\mathbf{x}_i; \theta))^2,$$

$$\mathcal{J}_y(\phi) = \sum_{i=1}^n (\mathbf{v}_i - e(\mathbf{y}_i; \phi))^2,$$

其中， θ 和 ϕ 表示两个神经网络的参数，神经网络的结构与 DDLFH 中两个神经网络的结构相同，分别用 $f(\cdot)$ 和 $e(\cdot)$ 表示神经网络。这个方法被称为两步学习 DDLFH（Two-step DDLFH, 简称 TDDL FH）。

实验结果如图 6-6 所示。从图 6-6 的结果可以看到，本文提出的 DDLFH 方法在所有情形下都达到了更高的检索精度。这证明 DDLFH 方法将 DLFH 方

法的二值哈希编码学习能力和深度学习的特征学习能力整合到一个统一的学习框架中是 DDLFH 能达到更高的检索精度的关键。

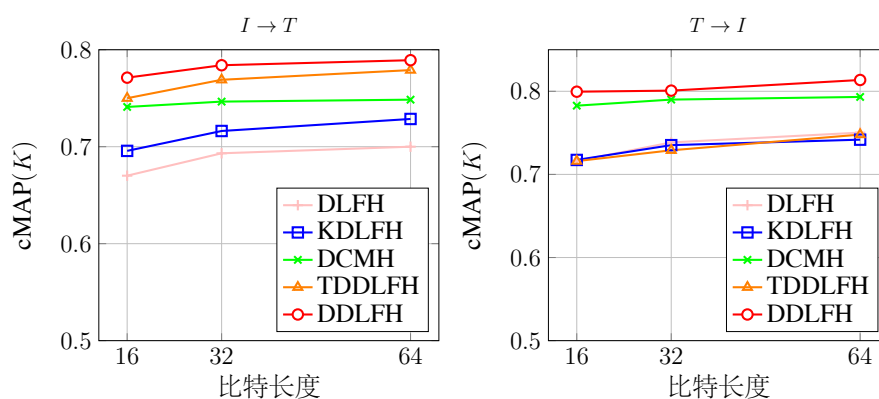


图 6-6: DDLFH 方法有效性验证实验

6.6.4 超参数实验

本节通过实验验证了 DDLFH 的超参数 η_n 的敏感性。图 6-7 给出了 DDLFH 方法关于超参数 η_n 的敏感性实验。从图 6-7 中可以看到，随着 η_n 的增加，DDL FH 方法的检索精度先变好后变差。当 $10^{-4} \leq \eta_n \leq 0.02$ 时，DDL FH 方法对于超参数 η_n 不敏感。

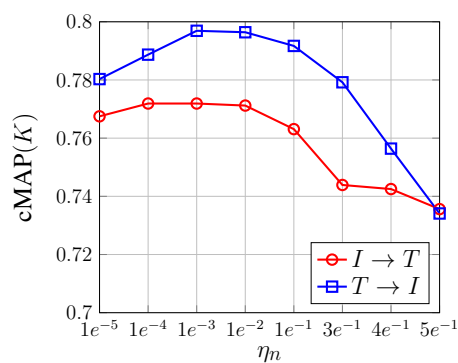


图 6-7: DDLFH 方法对超参数 η_n 的敏感性实验

6.7 本章小结

本章提出一种深度离散跨模态哈希 DCMH 方法。DCMH 方法首次将深度特征学习引入到跨模态哈希领域。该方法将二值哈希编码学习和深度特征学习

整合到一个统一框架中，并设计了一种离散优化算法来完成二值哈希编码学习和深度特征学习。本章还提出一种基于深度离散隐因子模型的跨模态哈希学习方法 DDLFH。DDL FH 将 DLFH 的二值哈希编码学习能力和深度学习的特征学习能力整合到一个统一的学习框架，能实现两种能力的相互反馈。DDL FH 设计了一种可直接学习二值哈希编码的优化算法。该算法可以在不丢弃二值约束的情况下，同时完成二值哈希编码的学习和深度特征的学习。实验证明，与非深度跨模态哈希学习方法相比，DCMH 能达到更高的检索精度。与非深度跨模态哈希学习方法和深度哈希学习方法相比，DDL FH 方法能达到更高的检索精度。

本章提出的 DCMH 方法已总结成文并发表：

- JIANG Q-Y, LI W-J. Deep Cross-Modal Hashing[C] //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 3270 – 3278. (CCF-A 类会议)

本章提出的 DDLFH 方法已总结成文并投稿到计算机学报^①：

- 蒋庆远, 李武军. 基于深度离散隐因子模型的跨模态哈希.

^①<http://cjc.ict.ac.cn/>

第七章 总结与展望

7.1 本文总结

哈希学习由于具有降低存储开销与加速数据查询过程的优点，近年来受到了越来越多的关注，并且已经被广泛应用在大规模检索的实际场景中。由于二值约束的存在，哈希学习模型的优化变得十分困难。直接丢弃二值约束的松弛策略通常会导致模型的检索精度受损，因此，近年来研究者们提出了一些直接学习二值哈希编码的离散哈希学习方法。根据不同的应用场景，本文从离散哈希学习建模与算法设计的角度展开研究，提出一些新的离散哈希学习方法。本文的工作可以概括为如下：

- 在非深度单模态场景中，图哈希学习是非深度单模态哈希学习方法的重要算法之一。现有的离散图哈希学习方法在训练过程中无法使用全部图相似度信息，这些方法的检索精度也因此受损。同时，现有的离散图哈希学习方法的训练过程低效。本文提出一种基于特征变换的可扩展图哈希学习方法 **SGH**。**SGH** 通过使用基于特征变换的方法，不需要显式的计算整个相似度图。通过这种方式，**SGH** 方法可以避免平方级别的复杂度，并提高离散图哈希学习方法的训练效率。同时，**SGH** 设计了一种逐比特离散优化算法来学习二值哈希编码。实验证明，与现有的离散图哈希学习方法相比，本文提出的 **SGH** 方法能达到更高的检索精度，同时训练也更加高效。
- 在深度单模态场景中，一方面，现有的深度哈希学习方法无法使用监督信息同时直接监督二值哈希编码学习和深度特征学习。另一方面，现有的监督哈希学习方法都是基于对称哈希框架的，训练过程低效。本文首先提出一种深度离散监督哈希学习方法 **DDSH**。**DDSH** 首次设计了一种使用监督信息直接同时指导二值哈希编码学习和深度特征学习的策略。通过这种策略可以加强二值哈希编码学习和深度特征学习之间的反馈作用。本文还提出一种非对称深度监督哈希学习方法 **ADSH**。**ADSH** 方法使用非对称哈希建模，并设计了一种高效的二值哈希编码学习算法，该算法能同时完成查询样本的深度特征学习和数据库样本的二值哈希编码学习。实验证明，与现

有的深度哈希学习方法相比，DDSH 通过使用直接监督二值哈希编码学习和深度特征学习的策略，能达到更高的检索精度。同时，与除 DDSH 方法外的对称深度哈希学习方法相比，ADSH 能在最短时间内达到更高的检索精度。与 DDSH 方法相比，ADSH 方法的训练过程更加高效。

- 在非深度跨模态场景中，现有的基于标签对信息的非深度跨模态离散哈希学习方法的复杂度为训练集大小的平方。当实际中计算资源有限时，这些方法只能使用采样的训练集完成训练过程。由于复杂度高，即便使用采样的训练集，这些方法的训练过程仍旧低效。本文提出一种基于离散隐因子模型的跨模态哈希学习方法 DLFH。DLFH 方法使用离散隐因子模型对跨模态哈希学习问题建模，并设计了一种高效的二值哈希编码学习算法。实验证明，与之前的非深度跨模态哈希学习方法相比，本文提出的 DLFH 能达到更高的检索精度。同时，与之前的跨模态离散哈希学习方法相比，DLFH 方法的训练更高效。
- 在深度多模态场景中，本文主要探讨深度跨模态哈希学习方法中的问题。本文首次提出一种深度跨模态哈希学习方法 DCMH。DCMH 方法利用深度神经网络来完成深度特征学习，同时设计了一种可以学习二值哈希编码的离散优化算法，首次将二值哈希编码学习和深度特征学习整合到一个统一的学习框架中。此外，本文还提出一种基于深度离散隐因子模型的跨模态哈希学习方法 DDLFH。DDLFH 将 DLFH 的二值哈希编码学习能力和深度学习的特征学习能力整合到一个统一的学习框架中，能实现两种能力的相互反馈。实验证明，与现有的非深度跨模态哈希学习方法相比，本文提出的 DCMH 方法能达到更高的检索精度。通过结合 DLFH 方法的二值哈希编码学习能力和深度学习的深度特征学习能力，DDLFH 能达到比现有的非深度跨模态哈希学习方法、深度跨模态哈希学习方法更高的检索精度。

7.2 未来工作展望

哈希学习问题的本质是一个离散优化问题，因此，离散哈希学习是哈希学习的核心问题。离散哈希学习在将来的工作中主要有以下有待进一步研究的问题：

- 本文提出的算法主要使用标签对的监督信息来指导模型学习。当监督信息以其他形式出现时，例如：类别标签信息^[85]、三元组标签信息^[91,92]和序列

标签信息^[89]等，哈希学习模型将会遇到不同的问题。例如，当监督信息为三元组标签信息或序列标签信息时，需要拟合的监督信息的复杂度可能达到训练数据量的三次方甚至更高。因此，在这些场景中，是否需要设计离散优化算法学习二值哈希编码，如何设计离散优化算法来学习二值哈希编码是一个值得研究的课题。

- 目前的离散哈希学习致力于解决由于二值哈希编码的离散性带来的困难，对于取符号函数带来的困难的研究工作相对较少。然而取符号函数同样会带来一些困难，例如，在深度哈希学习，取符号函数的存在和反向传播算法通常相互矛盾。如果使用反向传播算法进行优化，往往只能采取一些松弛的策略。在某些场景中，由于取符号函数的存在带来的困难同样会造成最终模型的检索精度受损。研究如何避免取符号函数带来的困难也是一个值得研究的课题。
- 基于哈希的检索过程中，二值哈希编码可以通过海明排序和哈希表查询两种策略来加速检索。目前的哈希学习方法更多的侧重于学习保相似性的二值哈希编码，然而，考虑到不同检索场景中对编码的要求不同，哈希学习模型应该被重新考虑，例如，海明排序中，只要要求二值哈希编码的相对顺序正确即可提高实际检索精度，而哈希表查询则对海明半径较小的桶中的相似样本的数目要求较高。哈希表查询过程同时还存在着一些不同的策略，例如单表查询、多表查询等。为了解决这些新场景中的问题，也需要设计在不同场景中离散优化算法。
- 哈希不仅仅可以用在大规模检索任务中，一些别的应用场景中，例如，大规模推荐系统^[49,129-131]、神经网络压缩^[72,73]、数据压缩、分布式系统加速^[132]等中，同样对存储开销或者运算速度敏感。在这些新的场景中，是否需要设计离散优化算法来学习二值哈希编码同样是值得研究的课题。

参考文献

- [1] 李武军, 周志华. 大数据哈希学习: 现状与趋势 [J]. 科学通报, 2015, 60(5-6): 485–490.
- [2] 李武军, 蒋庆远. 哈希学习 [J]. 中国人工智能学会通讯, 2016, 9: 9–15.
- [3] 李武军. 大数据机器学习 [J]. 中国人工智能学会通讯, 2018, 12: 49–53.
- [4] BISHOP C M. Pattern Recognition and Machine Learning[M]. [S.l.]: Springer, 2006.
- [5] 周志华. 机器学习 [M]. [S.l.]: 清华大学出版社, 2016.
- [6] KNUTH D E. The Art of Computer Programming, Volume III: Sorting and Searching[M]. [S.l.]: Addison-Wesley, 1973.
- [7] CLARKSON K L. Fast Algorithms for the All Nearest Neighbors Problem[C] // Proceedings of the Annual IEEE Symposium on Foundations of Computer Science. 1983: 226–232.
- [8] CAYTON L. Fast Nearest Neighbor Retrieval for Bregman Divergences[C] // Proceedings of the International Conference on Machine Learning. 2008: 112–119.
- [9] ROUSSOPOULOS N, KELLEY S, VINCENT F. Nearest Neighbor Queries[C] // Proceedings of the ACM SIGMOD International Conference on Management of Data. 1995: 71–79.
- [10] MANNING C D, RAGHAVAN P, SCHUTZE H. Introduction to Information Retrieval[M]. [S.l.]: Cambridge University Press, 2008.
- [11] HAN J, PEI J, KAMBER M. Data Mining: Concepts and Techniques[M]. [S.l.]: Elsevier, 2011.

- [12] ARYA S, MOUNT D M, NETANYAHU N S, et al. An Optimal Algorithm for Approximate Nearest Neighbor Searching[C] // Proceedings of the ACM-SIAM Symposium on Discrete Algorithms. 1994 : 573 – 582.
- [13] BENTLEY J L. Multidimensional Binary Search Trees Used for Associative Searching[J]. Communication ACM, 1975, 18(9) : 509 – 517.
- [14] DOLATSHAH M, HADIAN A, MINAEI-BIDGOLI B. Ball*-tree: Efficient Spatial Indexing for Constrained Nearest-Neighbor Search in Metric Spaces[J]. arXiv preprint arXiv:1511.00628, 2015.
- [15] JeGOU H, DOUZE M, SCHMID C. Product Quantization for Nearest Neighbor Search[J]. IEEE Transactions on Pattern Analysis Machine Intelligence, 2011, 33(1) : 117 – 128.
- [16] DATAR M, IMMORLICA N, INDYK P, et al. Locality-Sensitive Hashing Scheme based on p -Stable Distributions[C] // Proceedings of the ACM Symposium on Computational Geometry. 2004 : 253 – 262.
- [17] GE T, HE K, KE Q, et al. Optimized Product Quantization for Approximate Nearest Neighbor Search[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013 : 2946 – 2953.
- [18] GIONIS A, INDYK P, MOTWANI R. Similarity Search in High Dimensions via Hashing[C] // Proceedings of 25th International Conference on Very Large Data Bases. [S.l.] : Morgan Kaufmann, 1999 : 518 – 529.
- [19] ANDONI A, INDYK P, LAARHOVEN T, et al. Practical and Optimal LSH for Angular Distance[C] // Advances in Neural Information Processing Systems. 2015 : 1225 – 1233.
- [20] DASGUPTA S, STEVENS C F, NAVLAKHA S. A Neural Algorithm for A Fundamental Computing Problem[J]. Science, 2017, 358(6364) : 793 – 796.
- [21] GONG Y, LAZEBNIK S. Iterative Quantization: A Procrustean Approach to Learning Binary Codes[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2011 : 817 – 824.

-
- [22] LI W-J, WANG S, KANG W. Feature Learning based Deep Supervised Hashing with Pairwise Labels[C] // Proceedings of the International Joint Conference on Artificial Intelligence. 2016: 1711 – 1717.
- [23] LOWE D G. Object Recognition from Local Scale-Invariant Features[C] // Proceedings of the IEEE International Conference on Computer Vision. 1999: 1150 – 1157.
- [24] ANDONI A, INDYK P. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions[C] // Proceedings of the Annual IEEE Symposium on Foundations of Computer Science. 2006: 459 – 468.
- [25] LI P, SHRIVASTAVA A, MOORE J L, et al. Hashing Algorithms for Large-Scale Learning[C] // Advances in Neural Information Processing Systems. 2011: 2672 – 2680.
- [26] KULIS B, GRAUMAN K. Kernelized Locality-Sensitive Hashing for Scalable Image Search[C] // Proceedings of the IEEE International Conference on Computer Vision. 2009: 2130 – 2137.
- [27] SHRIVASTAVA A. Simple and Efficient Weighted Minwise Hashing[C] // Advances in Neural Information Processing Systems. 2016: 1498 – 1506.
- [28] MORAN S, LAVRENKO V, OSBORNE M. Variable Bit Quantisation for LSH[C] // Proceedings of the Annual Meeting of the Association for Computational Linguistics. 2013: 753 – 758.
- [29] CAKIR F, SCLAROFF S. Adaptive Hashing for Fast Similarity Search[C] // Proceedings of the IEEE International Conference on Computer Vision. 2015: 1044 – 1052.
- [30] WEISS Y, TORRALBA A, FERGUS R. Spectral Hashing[C] // Advances in Neural Information Processing Systems. 2008: 1753 – 1760.
- [31] ZHANG D, WANG J, CAI D, et al. Self-Taught Hashing for Fast Similarity Search[C] // Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval. 2010: 18 – 25.

-
- [32] LIU W, WANG J, KUMAR S, et al. Hashing with Graphs[C] // Proceedings of the International Conference on Machine Learning. 2011 : 1 – 8.
- [33] NOROUZI M, FLEET D J. Minimal Loss Hashing for Compact Binary Codes[C] // Proceedings of the International Conference on Machine Learning. 2011 : 353 – 360.
- [34] KONG W, LI W-J. Isotropic Hashing[C] // Advances in Neural Information Processing Systems. 2012 : 1655 – 1663.
- [35] HEO J, LEE Y, HE J, et al. Spherical hashing[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2012 : 2957 – 2964.
- [36] JIN Z, HU Y, LIN Y, et al. Complementary Projection Hashing[C] // Proceedings of the IEEE International Conference on Computer Vision. 2013 : 257 – 264.
- [37] HE K, WEN F, SUN J. K-Means Hashing: An Affinity-Preserving Quantization Method for Learning Binary Compact Codes[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013 : 2938 – 2945.
- [38] LIU W, MU C, KUMAR S, et al. Discrete Graph Hashing[C] // Advances in Neural Information Processing Systems. 2014 : 3419 – 3427.
- [39] ZHANG L, ZHANG Y, GU X, et al. Scalable Similarity Search With Topology Preserving Hashing[J]. IEEE Transactions on Image Processing, 2014, 23(7): 3025 – 3039.
- [40] YU F X, KUMAR S, GONG Y, et al. Circulant Binary Embedding[C] // Proceedings of the International Conference on Machine Learning. 2014 : 946 – 954.
- [41] MUKHERJEE L, RAVIS N, ITHAPU V K, et al. An NMF Perspective on Binary Hashing[C] // Proceedings of the IEEE International Conference on Computer Vision. 2015 : 4184 – 4192.
- [42] JIANG Q-Y, LI W-J. Scalable Graph Hashing with Feature Transformation[C] // Proceedings of the International Joint Conference on Artificial Intelligence. 2015 : 2248 – 2254.

-
- [43] TANG J, LI Z, WANG M, et al. Neighborhood Discriminant Hashing for Large-Scale Image Retrieval[J]. IEEE Transactions on Image Processing, 2015, 24(9): 2827–2840.
- [44] KANG W, LI W-J, ZHOU Z. Column Sampling Based Discrete Supervised Hashing[C] // Proceedings of the AAAI Conference on Artificial Intelligence. 2016: 1230–1236.
- [45] DAI B, GUO R, KUMAR S, et al. Stochastic Generative Hashing[C] // Proceedings of the International Conference on Machine Learning. 2017: 913–922.
- [46] CUI Q, JIANG Q-Y, WEI X-S, et al. ExchNet: A Unified Hashing Network for Large-Scale Fine-Grained Image Retrieval[C] // Proceedings of the European Conference on Computer Vision. 2020.
- [47] BRONSTEIN M M, BRONSTEIN A M, MICHEL F, et al. Data Fusion Through Cross-Modality Metric Learning Using Similarity-Sensitive Hashing[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2010: 3594–3601.
- [48] ZHEN Y, YEUNG D. Co-Regularized Hashing for Multimodal Data[C] // Advances in Neural Information Processing Systems. 2012: 1385–1393.
- [49] LIU X, HE J, DENG C, et al. Collaborative Hashing[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014: 2147–2154.
- [50] DING G, GUO Y, ZHOU J. Collective Matrix Factorization Hashing for Multimodal Data[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014: 2083–2090.
- [51] ZHANG D, LI W-J. Large-Scale Supervised Multimodal Hashing with Semantic Correlation Maximization[C] // Proceedings of the AAAI Conference on Artificial Intelligence. 2014: 2177–2183.

- [52] WU B, YANG Q, ZHENG W, et al. Quantized Correlation Hashing for Fast Cross-Modal Search[C] // Proceedings of the International Joint Conference on Artificial Intelligence. 2015 : 3946–3952.
- [53] LIN Z, DING G, HU M, et al. Semantics-Preserving Hashing for Cross-View Retrieval[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015 : 3864–3872.
- [54] MANDAL D, CHAUDHURY K N, BISWAS S. Generalized Semantic Preserving Hashing for N-Label Cross-Modal Retrieval[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017 : 2633–2641.
- [55] CAO Y, LONG M, WANG J, et al. Deep Visual-Semantic Hashing for Cross-Modal Retrieval[C] // Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016 : 1445–1454.
- [56] JIANG Q-Y, LI W-J. Deep Cross-Modal Hashing[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017 : 3270–3278.
- [57] JIANG Q-Y, LI W-J. Discrete Latent Factor Model for Cross-Modal Hashing[J]. IEEE Transactions on Image Processing, 2019, 28(7) : 3490–3501.
- [58] JIN L, LI K, LI Z, et al. Deep Semantic-Preserving Ordinal Hashing for Cross-Modal Similarity Search[J]. IEEE Transactions on Neural Networks and Learning Systems, 2019, 30(5) : 1429–1440.
- [59] LIU X, YU G, DOMENICONI C, et al. Ranking-Based Deep Cross-Modal Hashing[C] // Proceedings of the AAAI Conference on Artificial Intelligence. 2019 : 4400–4407.
- [60] LIN G, SHEN C, SHI Q, et al. Fast Supervised Hashing with Decision Trees for High-Dimensional Data[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014 : 1971–1978.
- [61] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet Classification with Deep Convolutional Neural Networks[C] // Advances in Neural Information Processing Systems. 2012 : 1106–1114.

-
- [62] XIA R, PAN Y, LAI H, et al. Supervised Hashing for Image Retrieval via Image Representation Learning[C] // Proceedings of the AAAI Conference on Artificial Intelligence. 2014 : 2156 – 2162.
- [63] LAI H, PAN Y, LIU Y, et al. Simultaneous Feature Learning and Hash Coding with Deep Neural Networks[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015 : 3270 – 3278.
- [64] LIONG V E, LU J, WANG G, et al. Deep Hashing for Compact Binary Codes Learning[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015 : 2475 – 2483.
- [65] ZHU H, LONG M, WANG J, et al. Deep Hashing Network for Efficient Similarity Retrieval[C] // Proceedings of the AAAI Conference on Artificial Intelligence. 2016 : 2415 – 2421.
- [66] ZHANG D, WANG F, SI L. Composite Hashing with Multiple Information Sources[C] // Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval. 2011 : 225 – 234.
- [67] MORAN S, LAVRENKO V. Regularised Cross-Modal Hashing[C] // Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval. 2015 : 907 – 910.
- [68] ZHEN Y, YEUNG D. A Probabilistic Model for Multimodal Hash Function Learning[C] // Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2012 : 940 – 948.
- [69] IRIE G, ARAI H, TANIGUCHI Y. Alternating Co-Quantization for Cross-Modal Hashing[C] // Proceedings of the IEEE International Conference on Computer Vision. 2015 : 1886 – 1894.
- [70] LIU H, WANG R, SHAN S, et al. Deep Supervised Hashing for Fast Image Retrieval[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016 : 2064 – 2072.

- [71] LIU W, HE J, CHANG S. Large Graph Construction for Scalable Semi-Supervised Learning[C] // Proceedings of the International Conference on Machine Learning. 2010 : 679 – 686.
- [72] HU Q, WANG P, CHENG J. From Hashing to CNNs: Training Binary Weight Networks via Hashing[C] // Proceedings of the AAAI Conference on Artificial Intelligence. 2018 : 3247 – 3254.
- [73] COURBARIAUX M, BENGIO Y, DAVID J. BinaryConnect: Training Deep Neural Networks with binary weights during propagations[C] // Advances in Neural Information Processing Systems. 2015 : 3123 – 3131.
- [74] SALAKHUTDINOV R, HINTON G E. Semantic Hashing[J]. International Journal of Approximate Reasoning, 2009, 50(7) : 969 – 978.
- [75] ZHANG P, ZHANG W, LI W-J, et al. Supervised Hashing with Latent Factor Models[C] // Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval. 2014 : 173 – 182.
- [76] XU J, WANG P, TIAN G, et al. Convolutional Neural Networks for Text Hashing[C] // Proceedings of the International Joint Conference on Artificial Intelligence. 2015 : 1369 – 1375.
- [77] CHAIDAROON S, FANG Y. Variational Deep Semantic Hashing for Text Documents[C] // Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval. 2017 : 75 – 84.
- [78] CHAIDAROON S, EBESU T, FANG Y. Deep Semantic Text Hashing with Weak Supervision[C] // Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval. 2018 : 1109 – 1112.
- [79] FAN L, JIANG Q-Y, YU Y-Q, et al. Deep Hashing for Speaker Identification and Retrieval[C] // Proceedings of the Annual Conference of the International Speech Communication Association. 2019 : 2908 – 2912.

-
- [80] JIANG Q-Y, HE Y, LI G, et al. SVD: A Large-Scale Short Video Dataset for Near-Duplicate Video Retrieval[C] // Proceedings of the IEEE International Conference on Computer Vision. 2019 : 5280 – 5288.
- [81] IRIE G, LI Z, WU X, et al. Locally Linear Hashing for Extracting Non-linear Manifolds[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014 : 2123 – 2130.
- [82] CARREIRA-PERPIÑÁN M Á, RAZIPERCHIKOLAEI R. Hashing with Binary Autoencoders[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015 : 557 – 566.
- [83] WANG J, KUMAR S, CHANG S. Semi-Supervised Hashing for Large-Scale Search[J]. IEEE Transactions on Pattern Analysis Machine Intelligence, 2012, 34(12): 2393 – 2406.
- [84] LIN G, SHEN C, SUTER D, et al. A General Two-Step Approach to Learning-Based Hashing[C] // Proceedings of the IEEE International Conference on Computer Vision. 2013 : 2552 – 2559.
- [85] SHEN F, SHEN C, LIU W, et al. Supervised Discrete Hashing[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015 : 37 – 45.
- [86] LIU W, WANG J, JI R, et al. Supervised Hashing with Kernels[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2012 : 2074 – 2081.
- [87] NOROUZI M, FLEET D J, SALAKHUTDINOV R. Hamming Distance Metric Learning[C] // Advances in Neural Information Processing Systems. 2012 : 1070 – 1078.
- [88] LI X, LIN G, SHEN C, et al. Learning Hash Functions Using Column Generation[C] // Proceedings of the International Conference on Machine Learning. 2013 : 142 – 150.

- [89] WANG J, LIU W, SUN A X, et al. Learning Hash Codes with Listwise Supervision[C] // Proceedings of the IEEE International Conference on Computer Vision. 2013 : 3032 – 3039.
- [90] WANG Q, ZHANG Z, SI L. Ranking Preserving Hashing for Fast Similarity Search[C] // Proceedings of the International Joint Conference on Artificial Intelligence. 2015 : 3911 – 3917.
- [91] ZHANG R, LIN L, ZHANG R, et al. Bit-Scalable Deep Hashing With Regularized Similarity Learning for Image Retrieval and Person Re-Identification[J]. IEEE Transactions on Image Processing, 2015, 24(12) : 4766 – 4779.
- [92] ZHAO F, HUANG Y, WANG L, et al. Deep Semantic Ranking based Hashing for Multi-label Image Retrieval[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015 : 1556 – 1564.
- [93] WANG X, SHI Y, KITANI K M. Deep Supervised Hashing with Triplet Labels[C] // Proceedings of the Asian Conference on Computer Vision. 2016 : 70 – 84.
- [94] LIN K, LU J, CHEN C, et al. Learning Compact Binary Descriptors with Unsupervised Deep Neural Networks[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016 : 1183 – 1192.
- [95] ZIEBA M, SEMBERECKI P, EL-GAALY T, et al. BinGAN: Learning Compact Binary Descriptors with a Regularized GAN[C] // Advances in Neural Information Processing Systems. 2018 : 3612 – 3622.
- [96] DIZAJI K G, ZHENG F, SADOUGHI N, et al. Unsupervised Deep Generative Adversarial Hashing Network[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018 : 3664 – 3673.
- [97] KANG Y, KIM S, CHOI S. Deep Learning to Hash with Multiple Representations[C] // Proceedings of the International Conference on Data Mining. 2012 : 930 – 935.

-
- [98] SONG J, YANG Y, HUANG Z, et al. Multiple Feature Hashing for Real-Time Large Scale Near-Duplicate Video Retrieval[C] // Proceedings of the Annual ACM Conference on Multimedia Conference. 2011 : 423 – 432.
- [99] WANG D, CUI P, OU M, et al. Deep Multimodal Hashing with Orthogonal Regularization[C] // Proceedings of the International Joint Conference on Artificial Intelligence. 2015 : 2291 – 2297.
- [100] KULIS B, DARRELL T. Learning to Hash with Binary Reconstructive Embeddings[C] // Advances in Neural Information Processing Systems. 2009 : 1042 – 1050.
- [101] WANG D, GAO X, WANG X, et al. Semantic Topic Multimodal Hashing for Cross-Media Retrieval[C] // Proceedings of the International Joint Conference on Artificial Intelligence. 2015 : 3890 – 3896.
- [102] LIU H, JI R, WU Y, et al. Supervised Matrix Factorization for Cross-Modality Hashing[C] // Proceedings of the International Joint Conference on Artificial Intelligence. 2016 : 1767 – 1773.
- [103] YANG R. New Results on Some Quadratic Programming Problems[D]. [S.l.] : University of Illinois at Urbana-Champaign, 2013.
- [104] HADSELL R, CHOPRA S, LECUN Y. Dimensionality Reduction by Learning an Invariant Mapping[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2006 : 1735 – 1742.
- [105] ZOBEL J, MOFFAT A, RAMAMOZHANARAO K. Inverted Files Versus Signature Files for Text Indexing[J]. ACM Transactions on Database System, 1998, 23(4): 453 – 490.
- [106] JIANG Q-Y, CUI X, LI W-J. Deep Discrete Supervised Hashing[J]. IEEE Transactions on Image Processing, 2018, 27(12): 5996 – 6009.
- [107] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-based Learning Applied to Document Recognition[J]. IEEE, 1998, 86(11): 2278 – 2324.

- [108] KRIZHEVSKY A. Learning Multiple Layers of Features from Tiny Images[D]. [S.l.]: University of Toronto, 2009.
- [109] CHUA T, TANG J, HONG R, et al. NUS-WIDE: A Real-World Web Image Database from National University of Singapore[C] // Proceedings of the ACM International Conference on Image and Video Retrieval. 2009.
- [110] HUISKES M J, LEW M S. The MIR flickr Retrieval Evaluation[C] // Proceedings of the ACM SIGMM International Conference on Multimedia Information Retrieval. 2008 : 39–43.
- [111] NAGRANI A, CHUNG J S, ZISSERMAN A. VoxCeleb: A Large-Scale Speaker Identification Dataset[C] // Proceedings of the Annual Conference of the International Speech Communication Association. 2017 : 2616–2620.
- [112] CHUNG J S, NAGRANI A, ZISSERMAN A. VoxCeleb2: Deep Speaker Recognition[C] // Proceedings of the Annual Conference of the International Speech Communication Association. 2018 : 1086–1090.
- [113] HUISKES M J, THOMEE B, LEW M S. New Trends and Ideas in Visual Concept Detection: the MIR flickr Retrieval Evaluation Initiative[C] // Proceedings of the Annual ACM Conference on Multimedia Conference. 2010 : 527–536.
- [114] LIN T, MAIRE M, BELONGIE S J, et al. Microsoft COCO: Common Objects in Context[C] // Proceedings of the European Conference on Computer Vision. 2014 : 740–755.
- [115] RASIWASIA N, PEREIRA J C, COVIELLO E, et al. A New Approach to Cross-Modal Multimedia Retrieval[C] // Proceedings of the Annual ACM Conference on Multimedia Conference. 2010 : 251–260.
- [116] ESCALANTE H J, HERNANDEZ C A, GONZALEZ J A, et al. The Segmented and Annotated IAPR TC-12 Benchmark[J]. Computer Vision and Image Understanding, 2010, 114(4): 419–428.
- [117] BLEI D M, NG A Y, JORDAN M I. Latent Dirichlet Allocation[C] // Advances in Neural Information Processing Systems. 2001 : 601–608.

-
- [118] NEYSHABUR B, SREBRO N, SALAKHUTDINOV R, et al. The Power of Asymmetry in Binary Hashing[C] // Advances in Neural Information Processing Systems. 2013 : 2823 – 2831.
- [119] CHATFIELD K, SIMONYAN K, VEDALDI A, et al. Return of the Devil in the Details: Delving Deep into Convolutional Nets[C] // Proceedings of the British Machine Vision Conference. 2014.
- [120] SIMONYAN K, ZISSERMAN A. Very Deep Convolutional Networks for Large-Scale Image Recognition[C] // Proceedings of the International Conference on Learning Representations. 2015.
- [121] RAZIPERCHIKOLAEI R, CARREIRA-PERPIÑÁN M Á. Optimizing Affinity-based Binary Hashing Using Auxiliary Coordinates[C] // Advances in Neural Information Processing Systems. 2016 : 640 – 648.
- [122] SHI X, XING F, XU K, et al. Asymmetric Discrete Graph Hashing[C] // Proceedings of the AAAI Conference on Artificial Intelligence. 2017 : 2541 – 2547.
- [123] HE X, CAI D, YAN S, et al. Neighborhood Preserving Embedding[C] // Proceedings of the IEEE International Conference on Computer Vision. 2005 : 1208 – 1213.
- [124] LANGE K, HUNTER D R, YANG I. Optimization Transfer Using Surrogate Objective Functions[J]. Journal of Computational and Graphical Statistics, 2000, 9(1): 1 – 20.
- [125] ZHANG J, PENG Y, YUAN M. Unsupervised Generative Adversarial Cross-Modal Hashing[C] // Proceedings of the AAAI Conference on Artificial Intelligence. 2018 : 539 – 546.
- [126] CAO Y, LIU B, LONG M, et al. Cross-Modal Hamming Hashing[C] // Proceedings of the European Conference on Computer Vision. 2018 : 207 – 223.

- [127] NGIAM J, KHOSLA A, KIM M, et al. Multimodal Deep Learning[C] // Proceedings of the International Conference on Machine Learning. 2011: 689–696.
- [128] YANG E, DENG C, LIU W, et al. Pairwise Relationship Guided Deep Hashing for Cross-Modal Retrieval[C] // Proceedings of the AAAI Conference on Artificial Intelligence. 2017: 1618–1625.
- [129] ZHOU K, ZHA H. Learning Binary Codes for Collaborative Filtering[C] // Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2012: 498–506.
- [130] ZHANG H, SHEN F, LIU W, et al. Discrete Collaborative Filtering[C] // Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval. 2016: 325–334.
- [131] HANSEN C, HANSEN C, SIMONSEN J G, et al. Content-aware Neural Hashing for Cold-start Recommendation[C] // Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020: 971–980.
- [132] ZHENG S, HUANG Z, KWOK J T. Communication-Efficient Distributed Blockwise Momentum SGD with Error-Feedback[C] // Advances in Neural Information Processing Systems. 2019: 11446–11456.
- [133] JIANG Q-Y, LI W-J. Asymmetric Deep Supervised Hashing[C] // Proceedings of the AAAI Conference on Artificial Intelligence. 2018: 3342–3349.

附录 A 符号及简称说明

A.1 本文使用的通用符号

表 A-1 中列出了一些本文使用的通用符号。

表 A-1: 本文使用的通用符号

符号	说明
\emptyset	空集
\mathbb{R}	实空间
\mathbb{R}^c	c 维实空间
$\{-1, +1\}^c$	c 维二值空间
\mathbf{A}	大写黑体字母, 矩阵
\mathbf{A}_{i*}	矩阵 \mathbf{A} 的第 i 行
\mathbf{A}_{*j}	矩阵 \mathbf{A} 的第 j 列
A_{ij}	矩阵 \mathbf{A} 中行标为 i , 列标为 j 的元素
\mathbf{a}	小写黑体字母, 列向量
a_i	向量 \mathbf{a} 的第 i 个元素
\mathbf{I}_n	维度为 $n \times n$ 的单位矩阵
$\mathbf{1}_d$	元素全部为 1 的 d 维向量
$\mathbf{1}_{m \times n}$	元素全部为 1 的 $m \times n$ 的矩阵
$\mathbf{0}_d$	元素全部为 0 的 d 维向量
$\mathbf{0}_{m \times n}$	元素全部为 0 的 $m \times n$ 的矩阵
\mathbf{A}^\top	矩阵 \mathbf{A} 的转置
$\mathbf{A} \succeq \mathbf{0}$	方阵 \mathbf{A} 为半正定矩阵
$\mathbf{A} \succeq \mathbf{B}$	方阵 $\mathbf{A} - \mathbf{B}$ 为半正定矩阵
$\mathbf{A} \succ \mathbf{0}$	方阵 \mathbf{A} 为正定矩阵
$\mathbf{A} \succ \mathbf{B}$	方阵 $\mathbf{A} - \mathbf{B}$ 为正定矩阵

此外，一些函数定义如下。

- 向量拼接函数。给定向量 $\mathbf{a} \in \mathbb{R}^{n_a}$, $\mathbf{b} \in \mathbb{R}^{n_b}$, 向量拼接函数定义如下:

$$\mathbf{concat}(\mathbf{a}, \mathbf{b}) = [\mathbf{a}; \mathbf{b}] = [a_1, \dots, a_{n_a}, b_1, \dots, b_{n_b}]^\top \in \mathbb{R}^{n_a+n_b}. \quad (\text{A-1})$$

这里使用两个向量作为例子。本文假设 $\mathbf{concat}(\cdot, \dots)$ 函数可以接受任意多个向量作为参数。

- 方阵的迹。给定方阵 $\mathbf{A} \in \mathbb{R}^{n \times n}$, \mathbf{A} 的迹定义为:

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^n A_{ii}. \quad (\text{A-2})$$

- 向量的 2 范数。给定向量 $\mathbf{a} \in \mathbb{R}^n$, \mathbf{a} 的 2 范数定义为:

$$\|\mathbf{a}\|_2 = \sqrt{\sum_{i=1}^n a_i^2}. \quad (\text{A-3})$$

- 矩阵的 Frobenius 范数。给定矩阵 $\mathbf{A} \in \mathbb{R}^{m \times n}$, \mathbf{A} 的 Frobenius 范数定义为:

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n A_{ij}^2}. \quad (\text{A-4})$$

- 矩阵的 1 范数。给定矩阵 $\mathbf{A} \in \mathbb{R}^{m \times n}$, \mathbf{A} 的 1 范数定义为:

$$\|\mathbf{A}\|_1 = \sum_{i=1}^m \sum_{j=1}^n |A_{ij}|. \quad (\text{A-5})$$

- 逐元素取符号函数 $\mathbf{sign}(\cdot)$ 。给定输入 $a \in \mathbb{R}$, 定义:

$$\mathbf{sign}(a) = \begin{cases} 1 & \text{if } a \geq 0, \\ -1 & \text{otherwise.} \end{cases} \quad (\text{A-6})$$

当输入为向量 $\mathbf{a} \in \mathbb{R}^n$ 时, $\mathbf{sign}(\mathbf{a})$ 定义如下:

$$\mathbf{sign}(\mathbf{a}) = [\mathbf{sign}(a_1), \dots, \mathbf{sign}(a_n)]^\top. \quad (\text{A-7})$$

输入为矩阵时同理。

- $\tanh(\cdot)$ 函数。给定输入 $a \in \mathbb{R}$ ，定义：

$$\tanh(a) = \frac{1 - e^{-a}}{1 + e^{-a}}. \quad (\text{A-8})$$

当输入为向量 $\mathbf{a} \in \mathbb{R}^n$ 时， $\tanh(\mathbf{a})$ 定义如下：

$$\tanh(\mathbf{a}) = [\tanh(a_1), \dots, \tanh(a_n)]^\top. \quad (\text{A-9})$$

输入为矩阵时同理。

- 指示函数。给定条件 *condition*，定义：

$$\mathbb{1}(\text{condition}) = \begin{cases} 1 & \text{if } \text{condition} \text{ is true,} \\ 0 & \text{otherwise.} \end{cases} \quad (\text{A-10})$$

- 向量点乘函数。给定向量 $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ ，向量点乘操作定义为：

$$\mathbf{a} \odot \mathbf{b} = [a_1 b_1, \dots, a_n b_n]^\top. \quad (\text{A-11})$$

A.2 本文使用的通用简称

表 A-2 列出了一些本文使用的通用简称的定义。

表 A-2: 本文使用的通用简称

简称	全称
ACQ	Alternating Co-Quantization, 交替协同量化 ^[69]
ADSH	Asymmetric Deep Supervised Hashing, 非对称深度监督哈希 ^[133]
AGH	Anchor Graph Hashing, 锚图哈希 ^[32]
ANNS	Approximate Nearest Neighbor Search, 近似最近邻检索
binGAN	Binary Generative Adversarial Network, 二值生成对抗神经网络 ^[95]
BA	Binary Autoencoder, 二值自动编码器 ^[82]
BDC	Bitwise Discrete Coding, 逐比特离散编码算法
BOW	Bag-of-Words, 词袋特征
BOVW	Bag-of-Visual Words, 视觉词袋特征
BQP	Binary Quadratic Programming, 离散二次规划
BRE	Binary Reconstructive Embedding, 二值哈希编码重构嵌入 ^[100]

cMAP	Mean Average Precision at Cutoff K , 在返回序列的截断位置 K 的平均查准率均值
cPre	Precision at Cutoff K , 在返回序列的截断位置 K 的查准率
cRec	Recall at Cutoff K , 在返回序列的截断位置 K 的查全率
CCAITQ	Canonical Correlation Analysis based Iterative Quantization, 基于典型相关分析的矢量迭代 ^[21]
CGH	Column Generation Hashing, 列生成哈希 ^[88]
CMFH	Collective Matrix Factorization Hashing, 协同矩阵哈希 ^[50]
CMHH	Cross-Modal Hamming Hashing, 跨模态海明哈希 ^[126]
CNNH	Convolutional Neural Network Hashing, 卷积神经网络哈希 ^[62]
COSDISH	Column Sampling based Discrete Supervised Hashing, 基于列采样的离散监督哈希 ^[44]
CRH	Co-Regularized Hashing, 协同量化哈希 ^[48]
DCMH	Deep Cross-Modal Hashing, 深度跨模态哈希 ^[56]
DDLFFH	Deep Discrete Latent Factor Modal for Cross-Modal Hashing, 基于深度离散隐因子模型的跨模态哈希
DDSH	Deep Discrete Supervised Hashing, 深度离散监督哈希 ^[106]
DeepBit	Deep Bit, 深度比特哈希 ^[94]
DGH	Discrete Graph Hashing, 离散锚图哈希 ^[38]
DH	Deep Hashing, 深度哈希 ^[64]
DHN	Deep Hashing Network, 深度哈希网络 ^[65]
DLFH	Discrete Latent Factor Modal based Cross-Modal Hashing, 基于离散隐因子模型的跨模态哈希 ^[57]
DPSH	Deep Pairwise Supervised Hashing, 基于标签对的深度监督哈希
DRSCH	Deep Regularized Similarity Comparison Hashing, 深度正则相似度比较哈希 ^[91]
DSH	Deep Supervised Hashing, 深度监督哈希 ^[70]

DSRH	Deep Semantic Ranking Hashing, 深度语义排序哈希 ^[92]
DTSH	Deep Triplet Supervised Hashing, 深度三元组监督哈希 ^[93]
FastH	Fast Hashing, 快速哈希 ^[60]
GSPH	Generalized Semantic Supervised Hashing, 广义语义保持哈希 ^[54]
hashGAN	Hashing for Generative Adversarial Network, 生成对抗神经网络哈希 ^[96]
HDML	Hamming Distance Metric Learning, 海明距离度量学习 ^[87]
ITQ	Iterative Quantization, 迭代量化 ^[21]
IsoHash	Isotropic Hashing, 等方差哈希 ^[34]
KADGH	Kernelized Asymmetric Discrete Graph Hashing, 核化非对称离散图哈希 ^[122]
KH	K-Means Hashing, K 聚类哈希 ^[37]
KLSH	Kernelized Locality Sensitive Hashing, 核局部敏感哈希 ^[26]
KSH	Kernelized Supervised Hashing, 基于核的监督哈希 ^[86]
LFH	Latent Factor Hashing, 隐因子模型哈希 ^[75]
LLH	Locally Linear Hashing, 局部线性哈希 ^[81]
LSH	Locality Sensitive Hashing, 局部敏感哈希
MAC	Method of Auxiliary Coordinate, 辅助坐标方法 ^[121]
MLBE	Multimodal Latent Binary Embedding, 多模态隐二值嵌入 ^[68]
NINH	Network in Network Hashing, 内建网络哈希 ^[63]
NNS	Nearest Neighbor Search, 最近邻检索
PCAH	Principal Component Analysis Hashing, 主成分分析哈希
PQ	Product Quantization, 乘积量化
QCH	Quantized Correlation Hashing, 相关性量化哈希 ^[52]
rPre	Precision at Radius R , 在海明球 R 的查准率
rRec	Recall at Radius R , 在海明球 R 的查全率
RBF	Radial Basis Function, 径向基函数
ReLU	Rectified Linear Unit, 修正线性单元 ^[61]
RPH	Ranking Preserving Hashing, 保序哈希 ^[90]
RSH	Ranking-based Supervised Hashing, 基于排序的监督哈希 ^[89]

SCM	Semantic Correlation Maximization, 语义相关最大化 ^[51]
SDH	Supervised Discrete Hashing, 离散监督哈希 ^[85]
SePH	Semantic-Preserving Hashing, 语义保持哈希 ^[53]
SGD	Stochastic Gradient Descent, 随机梯度下降
SGH	Scalable Graph Hashing with Feature Transformation, 基于特征变换的可扩展图哈希 ^[42]
SH	Spectral Hashing, 谱哈希 ^[30]
SIFT	Scale-Invariant Feature Transform, 尺度不变特征变换
SMFH	Supervised Matrix Factorization Hashing, 监督矩阵分解哈希 ^[102]
SPLH	Sequential Projection Learning Hashing, 序列投影学习哈希 ^[83]
STH	Self-Taught Hashing, 自学习哈希 ^[31]
STMH	Semantic Topic Multimodal Hashing, 语义主题多模态哈希 ^[101]
TSH	Two-Step Hashing, 两步走哈希 ^[84]
UGACH	Unsupervised Generative Adversarial Cross-Modal Hashing, 无监督生成对抗跨模态哈希 ^[125]

致 谢

岁月勿错，转眼在南京大学的十年就要过去。在南大的十年是我人生中充实而又有意义的十年。伴随着仙林校区越来越繁华，伴随着同窗们毕业、告别，我终于也到了要离开的时刻，人生如逆旅，我亦是行人。在这里我要衷心感谢十年来所有关心支持我的人。

首先要感谢我的导师李武军老师。感谢李老师六年来对我的谆谆教导和悉心培养。李老师严谨的治学态度和渊博的知识让我受益匪浅。李老师是我科研道路的领路人，从科研选题、研究思路、论文组织等方方面面的耐心教导到一字一句帮我修改论文，李老师都精益求精。在生活中，李老师平易近人，并且在我感到迷茫时给我指引方向。李老师的教诲我将铭记于心。

其次，感谢周志华老师和 LAMDA 组所有老师。周老师渊博的学识和广阔的视野令我折服。在 LAMDA 的六年里，我有幸参加了一些周老师和各位老师的讨论班，从中我学到了很多。感谢周老师、各位老师和 LAMDA 研究所给我的指导、教育。

同时，感谢 LAMDA 组所有的同学们。在 LAMDA 的六年，我和组里的同学们一起生活、学习、科研，让我的博士生活充满乐趣。感谢项如、王盛、赵申宜、高鹏同学，我们一起在小组里学习的时光让人难忘。感谢钱超老师、李博同学、庞明同学、姚开浪师弟、房康师弟、蔡文朴师弟、胡毅奇师弟、宛袁玉师弟，我们一起打球时光青春飞扬。感谢康望程同学、刘弘师兄、李明威师弟、冯心月师妹，我们一起在哈希学习方向上的研究讨论让我获益良多。感谢刘冲师弟、王涵师妹、马啸师妹，我们一起参加的课外活动让我的生活平添精彩。因为你们，我的研究生生涯充满阳光。

最后，感谢十年以来一直支持我的家人。感谢我的爸爸妈妈和我的妻子杨斐女士，你们是我坚强的后盾。十年时光漫漫，我无法想象你们在我的身后为我撑起了怎样的蓝天。今天我终于要毕业了，也终于要为你们撑起一片蓝天。

简历与科研成果

基本信息

蒋庆远，男，汉族，1991年12月出生，云南省腾冲市人。

教育背景

2010年9月 — 2014年6月 南京大学计算机科学与技术系 本科
2014年9月 — 至今 南京大学计算机科学与技术系 博士在读

获奖情况

1. 南京大学优秀博士研究生创新能力提升计划 A 类，2019 年。
2. 博士研究生国家奖学金，2018 年。
3. 英才奖学金，2017 年。
4. 硕士研究生国家奖学金，2015 年。

攻读博士学位期间完成的学术成果

1. JIANG Q-Y, HE Y, LI G, LIN J, LI L, LI W-J. SVD: A Large-Scale Short Video Dataset for Near-Duplicate Video Retrieval[C] //Proceedings of the IEEE International Conference on Computer Vision. 2019: 5280 – 5288. (CCF-A 类会议)
2. JIANG Q-Y, LI W-J. Discrete Latent Factor Model for Cross-Modal Hashing[J]. IEEE Transactions on Image Processing, 2019, 28(7): 3490 – 3501. (CCF-A 类期刊)
3. JIANG Q-Y, LI W-J. Asymmetric Deep Supervised Hashing[C] //Proceedings of the AAAI Conference on Artificial Intelligence. 2018: 3342 – 3349. (CCF-A 类会议)
4. JIANG Q-Y, CUI X, LI W-J. Deep Discrete Supervised Hashing[J]. IEEE Transactions on Image Processing, 2018, 27(12): 5996 – 6009. (CCF-A 类期刊)
5. JIANG Q-Y, LI W-J. Deep Cross-Modal Hashing[C] //Proceedings of the IEEE

- Conference on Computer Vision and Pattern Recognition. 2017: 3270 – 3278. (CCF-A 类会议)
6. JIANG Q-Y, LI W-J. Scalable Graph Hashing with Feature Transformation[C] //Proceedings of the International Joint Conference on Artificial Intelligence. 2015: 2248 – 2254. (CCF-A 类会议)
 7. CUI Q, JIANG Q-Y^①, WEI X-S, LI W-J, O. YOSHIE. ExchNet: A Unified Hashing Network for Large-Scale Fine-Grained Image Retrieval[C] //Proceedings of the European Conference on Computer Vision. 2020. (CCF-B 类会议)
 8. FAN L, JIANG Q-Y, YU Y-Q, LI W-J. Deep Hashing for Speaker Identification and Retrieval[C] //Proceedings of the Annual Conference of the International Speech Communication Association. 2019: 2908 – 2912. (CCF-C 类会议)

已申请国家发明专利

1. 李武军, 李明威, 蒋庆远。一种基于深度多索引哈希的行人重识别方法。专利申请号: 201910166071.1
2. 李武军, 樊磊, 蒋庆远, 余亚奇。一种基于深度哈希的声纹检索方法。专利申请号: 201910574215.7

攻读博士学位期间参与的科研课题

1. 国家自然科学基金国际合作与交流项目 (项目编号: 61861146001): “大数据机器学习及多模态应用”。课题年限: 2019 年 1 月 – 2021 年 12 月。
2. 国家自然科学基金面上项目 (项目编号: 61472182): “面向大数据的哈希学习理论与应用”。课题年限: 2015 年 1 月 – 2018 年 12 月。
3. 国家电网公司科学技术项目 (项目编号: SGGR0000XTJS1900448): “面向业务 SQL 交互特征的信息内外网边界安全技术研究”。课题年限: 2019 年 1 月 – 2020 年 12 月。

^①共同一作