Single and Multiple Instance Learning for Visual Categorisation

Ruo Du

Faculty of Engineering and Information Technology

University of Technology, Sydney

A thesis submitted for the degree of

Doctor of Philosophy

2013

CERTIFICATE OF
AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Author

Production Note:
Signature removed prior to publication.

_____ ____

This thesis is dedicated to my wife and my parents.

For their endless support and encouragement

Acknowledgements

First and foremost I would like to sincerely thank my principal supervisor, Prof. Xiangjian HE, for his guidance, understanding, patience and help on scholarship to make my PhD experience such a rewarding and exciting journey.

I also want to express my utmost gratitude to my co-supervisor, Dr. Qiang Wu, who spent enormous time and energy on me as a mentor and a friend. His guidance was embedded in every step of my studies.

Special thanks also to Dr. Wenjing Jia, Dr. Min Xu, Prof. Massimo Piccardi and Dr. Richard Xu for their great suggestions, knowledge sharing and invaluable assistance.

I wish to thank my fellow research students of our team, Muhammad Hasan, Man Wong, Chao Zeng, Sheng Wang, Minqi Li, Thomas Tan, Aruna Jamdagni, Ying Wan and Mohammed A AmbuSaid, for their assistance and friendships.

Finally, and most importantly, I would like to thank my wife Ting Zhou. Her faith in me, support, encouragement and quite patience made it possible that I could even continue to pursue my PhD after working in IT for ten years. I thank my parents, Changyou Du and Shenghua Liu, and my parents in law, Ruguo Zhou and Shijie Li, for their help and support as always.

Abstract

Nowadays, huge amounts of visual data, e.g., videos and images, have become widely accessible. Therefore, intelligently categorizing the large and growing collections of data for access convenience has been a central goal for modern computer vision research. In this thesis, we describe several newly-developed approaches for visual categorization upon the single and multiple instance learning cases.

In single-instance learning (SIL), each of the training instances has been labeled. Here, we focus on a challenging task of facial expressions recognition where manually labeling each training instance, i.e., face video, is handy. To get the distinct features of expressions, we propose a novel feature representation, Histogram Variances Face (HVF), which integrates dynamic expression information into a static image being invariant to illumination and in-plane rotation. Through HVFs, the facial expression recognition can be cast as a facial recognition problem. We have applied our approach on the well-known Cohn-Kanade AU-Coded Facial Expression database, and then those extracted HVFs are classified by using facial recognition technology, i.e., Eigenfaces and Support Vector Machines (SVMs). The recognition accuracy is very encouraging. We further propose an extension of HVFs, Hexagonal Histogram Variance Faces (HHVFs), which applies HVFs on a hexagonal structure. Comparing to HVFs, HHVFs not only greatly reduce the computation costs but also improve the recognition accuracy.

In multiple-instance learning (MIL), the training instances are divided into groups and the instances in the same group share only one label. MIL arises from many applications where individually labeling training instances is expensive. In this case, we propose a novel algorithm,

multiple-instance learning with a supervised kernel density estimation (MIL-SKDE), to tackle the labeling ambiguity. Our algorithm extends the twin technologies, kernel density estimation (SKDE) and mean shift, to their supervised versions in which the labels of data points will affect the mode seeking. We apply MIL-SKDE in several applications of visual categorization, e.g., image and object categorization, and our algorithm performs superiorly comparing to other state-of-the-art methods. Furthermore, to address the complexity issue of MIL-SKDE, we propose MIL-SS (MIL with speed-up SKDE) to speed up the training process. Experiments shows that it has comparable performances to MIL-SKDE but is much more efficient in training stage.

Finally, we apply MIL-SS in a "bag-of-words" (BoW) system to learn the visual codebook for object categorization on a more comprehensive dataset. Our system consists of four steps: codebook generation, feature coding, feature pooling and classification. Unlike conventional BoW methods that learn codebook from the whole image areas, our method can learn codebook just from the areas of target objects, which significantly improves classification accuracy.

Author's publications for the Ph.D

Journal paper:

1. **Ruo Du**, Qiang Wu, Xiangjian He, and Jie Yang. "MIL-SKDE: Multiple-instance learning with supervised kernel density estimation". *Signal Processing*, Volume 93, Issue 6, June 2013, Pages 1471-1484

Conference papers:

2. **Ruo Du**, Qiang Wu, XiangJian HE, and Jie Yang. "Object categorisation based on a supervised mean shift algorithm". In *12th European Conference on Computer Vision (ECCV) Demos*, Part III, LNCS 7585. Springer, Heidelberg, 2012.

3. **Ruo Du**, Qiang Wu, XiangJian HE, and Jie Yang. "Multi-instance learning with an extended kernel density estimation for object categorisation". In *IEEE International Conference on Multimedia and Expo (ICME) Workshops*, pp.477-482, 9-13 July 2012.

4. Lin Wang, Xiangjian He, **Ruo Du**, Wenjing Jia, Qiang Wu, and Wei-Chang Yeh. "Facial expression recognition on hexagonal structure using lbp-based histogram variances". In *17th International MultiMedia Modeling Conference (MMM)*, pages 35 - 45, 2011.

5. **Ruo Du**, Sheng Wang, Qiang Wu, and Xiangjian He. "Learn concepts in multiple-instance learning with diverse density framework using supervised mean shift". In *Digital Image Computing: Techniques and Applications (DICTA) - Oral presentation*, pages 643-648, 2010.

6. Sheng Wang, **Ruo Du**, Qiang Wu, and Xiangjian He. "Adaptive stick-like features for human detection based on multi-scale feature fusion scheme". In *Digital Image Computing: Techniques and Applications (DICTA)*, pages 375-380, 2010.

7. **Ruo Du**, Qiang Wu, Xiangjian He, Wenjing Jia, and Daming Wei. "Facial expression recognition using histogram variances faces". In *Workshop on Applications of Computer Vision (WACV)*, pages 1-7, 2009.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The proliferation of digital imaging products like mobile phones and digital cameras is keeping producing a large number of video and image collections. Such collections are vital information resources for our daily lives. Those resources can only be well exploited when they have been properly organised according to their categories. Given visual data with categorisation, one can easily access the wanted instances via high-level semantic search. Otherwise, seeking desired videos or images in a massive repository will be difficult. However, manually assigning the categories to those overwhelming and ever-growing visual data is extremely time-consuming and infeasible. We are thus confronted with an issue of automatic visual categorisation which has been a central goal for the modern computer vision research.

Typically, a system of visual categorisation with supervised learning includes two stages: training stage and recognition stage. In training stage, the representative features must be extracted from the manually labeled training data. Then, the classifiers need to be learned by using machine learning and pattern recognition algorithms in the feature space. In recognition stage, the same feature extraction method will first be applied on testing data, then the feature representation of test data will be classified by the classifiers learned in training process. The general training and recognition process are illustrated in Figure

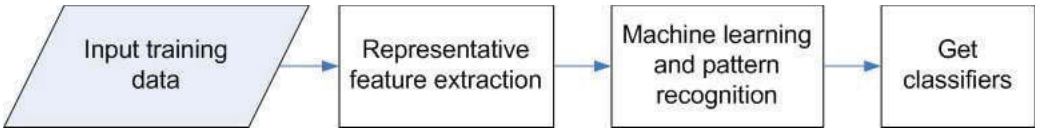


Figure 1.1: The training process for visual categorisation.

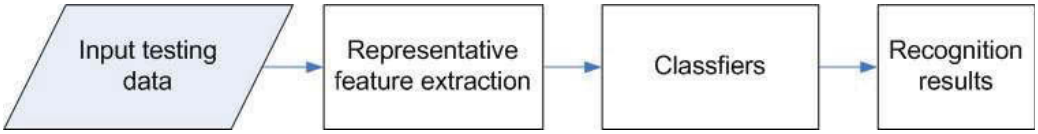


Figure 1.2: The testing process for visual categorisation.

1.1 and Figure 1.2 respectively.

The performances of recognition heavily depends on the classifiers learned from training data. How to label the training data will significantly affect the classifier training. Basically, in terms of different labeling on training data, the supervised learning can be divided into three categories: single-instance (fully supervised), multiple-instance (weakly supervised) and semi-supervised learning.

In single-instance learning (SIL), the instances in training data have been individually labeled, i.e., each instance has its own label. This is an ideal situation occurring when labeling individual training instance is handy. However, in quite a few cases, individually labeling each instance is too expensive. In that case, labeling grouped instances or a small part of instances are often applied. In multiple-instance learning (MIL), the training data are divided into groups and the instances in the same group share only one label. In semi-supervised learning, only a part of the training data are labeled, typically a small amount of data is labeled and a large amount of data is unlabeled, and semi-supervised learning makes use of both labeled and unlabeled data for training classifiers.

In this thesis, we explore the single-instance learning and multiple-instance learning for visual categorisation. The semi-supervised learning for visual categorisation is beyond our scope and interested readers can refer to the related literature.

For visual categorisation using single-instance learning, we focus on a challenging task of recognition of human facial expressions in which the training instances, i.e., expression videos, are handy to be labeled, and our major goal

is to extract representative features from videos. For multiple-instance learning, we propose a novel algorithm, multiple-instance learning with a supervised kernel density estimation (MIL-SKDE), and apply it object categorisation. Both expression recognition and object categorisation are challenging and active in current computer vision research and their backgrounds will be described in Section 1.2 and 1.3.

1.2 Facial expression recognition

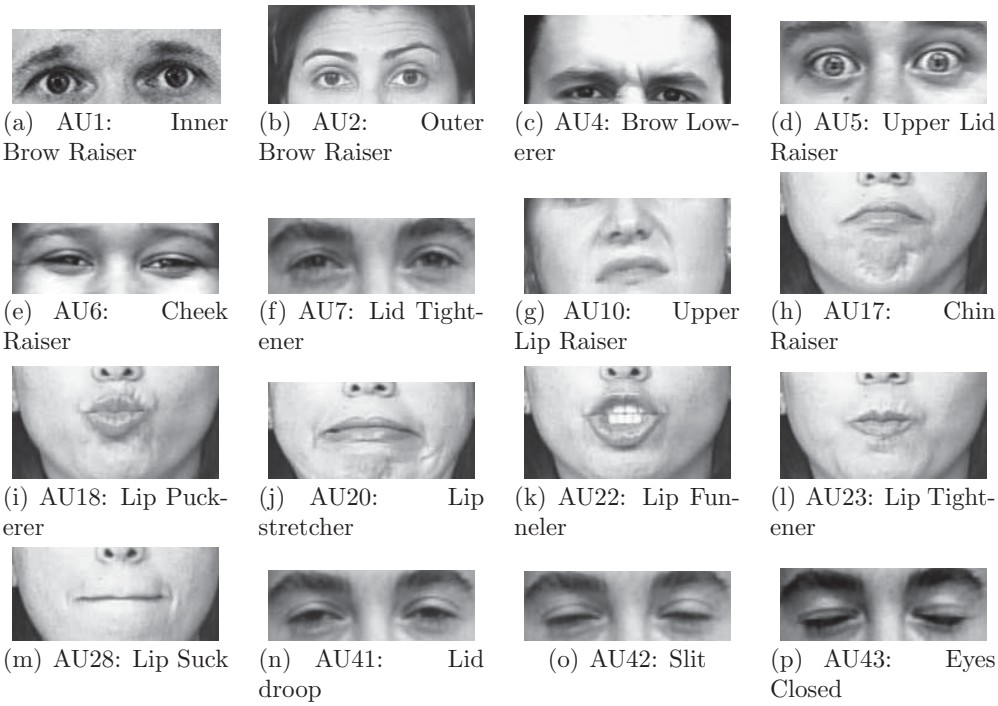
Human facial expressions are the visible representations of human's emotions, mental activity, social interaction and physiological signals. Recognizing such facial expressions can help computer to better understand human's inward activities and react more sophisticatedly. Therefore, it has enormous potentials in human-computer interaction (HCI).

In literature (e.g., [1], [2], [3] and [4]), the approaches of facial expression recognition are summarised into two categories:

1. Action Unit (Sign) based recognition.

2. Emotion (Judgment) based recognition.

In the first category, a set of basic facial Action Units (AU) are firstly carefully defined. Each AU represents a certain deformation of a specific set of facial muscles. Then, facial expressions are treated as the combinations of such basic AUs in a coded way. The key task of this category is to detect the AUs or combinations of AUs from the expression imagery. Finally, the interpretation of human's mental activities is determined by the pre-defined templates of AU combination, i.e., the codes of AU. Currently, the Facial Action Coding System (FACS) [5] is the most commonly used AU code system developed by Ekman et al. in 1978. In total, there are 46 AUs defined in FACS. Fig.1.3 shows the examples of some action units extracted from Cohn-Kanade database [6]. By using AU, the facial expressions can be translated into different combinations (codes) of AU. For example, AU6+12 represents "happiness" and AU1+2+4+5+20+26 means "fear". Some expression codes are shown in Fig.1.4. Note that the same type of facial expression may have multiple AU combinations (codes).



(a) AU1: Inner Brow Raiser (b) AU2: Outer Brow Raiser (c) AU4: Brow Lowerer (d) AU5: Upper Lid Raiser

(e) AU6: Cheek Raiser (f) AU7: Lid Tightener (g) AU10: Upper Lip Raiser (h) AU17: Chin Raiser

(i) AU18: Lip Puckerer (j) AU20: Lip stretcher (k) AU22: Lip Funneler (l) AU23: Lip Tightener

(m) AU28: Lip Suck (n) AU41: Lid droop (o) AU42: Slit (p) AU43: Eyes Closed

Figure 1.3: Examples of some Action Units extracted from Cohn-Kanade database [6].

The approaches in the second category do not code the facial expressions. Instead, they try to learn the expression patterns directly from imagery associated with human's emotions, e.g., happiness, sadness, fear, disgust, surprise and anger. The key task of this category is to obtain the representative features of a specific facial expression.

Figure 1.4: Examples of some combinations of Action Units. [7]

1.2.1 Action unit based approaches

A number of approaches in computer vision and pattern recognition have been developed to recognise AUs and their combinations from video sequences or static images by extracting and classifying features related to the corresponding AUs.

To get the expression features, [8] and [9] used dense optical flow to detect the direction and magnitude of the facial skin motion related to some AUs by extracting a motion field for the whole face area in each frame. However, these methods based on dense optical flow are sensitive to image alignment, motion smoothness and illumination variation. Furthermore, the computation of dense optical flow is expensive. To be more efficient, some other methods, e.g., [10], [11] and [12], extracted facial expression motion through the difference between the face with expression and the reference neutral face. These methods need less computation but only obtain limited expression dynamics. Besides the difference-between-images methods mentioned above, some single-image-based methods are employed to recognise AUs from the static image. BarBartlett et al. [13] extracted expression features from the still image by using Gabor wavelet filters and Lanitis et al. [14] utilised statistical learning by an Active Appearance model for feature extraction. This category of methods can efficiently detect geometrical information of facial expressions, however, they have the limitation that only static clues are used and the information of temporal evolution of facial expressions has been ignored.

Another group of methods [15] [16] [7] [17] [18] [19] [20] observe the displacement of feature points around the local permanent facial components, e.g., eyebrows, eyes and mouth for AU detection. These local approaches are sensitive to subtle changes in small areas and more computationally efficient than the above mentioned approaches. However, detecting and tracking the feature points on the human face are often unreliable.

Recently, Jiang et al. [21] detected AUs in the spatial-temporal video volumes. First, the Local Binary Pattern (LBP) [22] descriptor is computed for every pixel in the face. Then, the face is broken down into sub-regions and the histograms of those sub-regions are summarised to reduce the dimensionality and the sensitivity to alignment of the face. Aside from LBP, other appearance descriptors, e.g., Ga-

bor Wavelet filter [23][24][25][26], Haar-like features [27], Free-Form Deformations and Motion History Images [28], have also been used in AU detection.

For those approaches based on AU detection, although there are only a small number of distinctive AUs, i.e., 46 AUs, which are defined in FACS, the number of different AU combinations that depict the facial expressions is large. For example, in [29], there are over 7,000 AU combinations that have been observed. As AUs and their combinations are independent of any interpretation on human's emotion, such interpretation will be left for the higher level decision making process. However, accurately mapping the huge number of AU combinations to real facial expressions is tricky and arduous. In addition, when several AUs occur together, their features often change significantly comparing to when only a single AU occurs. This phenomenon makes detecting individual AU of human face unstable.

1.2.2 Emotion based approaches

The aim of emotion based approaches is to directly acquire the patterns that underlie the displayed facial expressions. Therefore, this category of approaches attempts to recognise the prototypes of emotional facial expressions, e.g., happiness, sadness, fear, disgust, surprise and anger.

As the appearance features usually have distinctive patterns to describe the texture of the face caused by expression, many appearance features have been employed for expression recognition. They include LBP operator [30] [31], Local Gabor Binary Pattern [32], Local Phase Quantisation (LPQ) and Histogram of Oriented Gradients (HOG) [33]. Zafeiriou et al. [34] proposed a method based on non-negative matrix factorisation (NMF) for data representation and classification. This method achieved 83.5% recognition rate over the six basic expressions on the Cohn-Kanade dataset [35]. Zhi et al. [36] proposed a method called Graph-preserving sparse nonnegative matrix factorisation (GNSMF). This method takes into account the occlusion in the expression recognition and achieved recognition rate between 91.4% and 94% on the occluded images.

Aside from appearance based methods, another group of approaches utilise the geometric features to get the expression patterns by tracking a set of critical

points around the facial components such as mouth and eyes. In this sort of methods, most of them use Active Appearance Models (AAMs) or its variances for point tracking. Then, the displacements and locations of the facial points are later used to classify the expressions. Asthana et al. [37] compared different AAM based algorithms and evaluated their performances on the Cohn-Kanade dataset. Sung et al. [38] utilised AAMs to track critical facial points in 3-D videos and proposed Stereo Active Appearance Models (STAAM) which improved the fitting and tracking of AAMs by using multiple cameras to model the 3D shape and rigid motion parameters. Sebe et al. [39] proposed another approach that used geometric features to detection expressions. They first manually located a number of critical points of face, then used Piece-wise Bézier volume deformation to track those points. They implemented the algorithms with several machine learning classification methods and found that the k-Nearest Neighbor attained the best result with 93% recognition rate.

The emotion based approaches do not rely on the pre-defined decomposition of facial expression (e.g., AUs) and have more freedom to model the expression dynamics. However, seeking the good features that are able to robustly grasp the distinction of expression dynamics is still challenging. To address this problem, we propose a new feature, Histogram Variances Face, which can grasp the expression dynamic features well and is easy to be calculated.

1.2.3 Our approach for expression recognition

We develop a new feature, *Histogram Variances Face(HVF)*, to represent the facial expression. Our method belongs to emotion-based category. It extracts expression dynamic features and stores the extracted features into an image. Each pixel of HVF represents the variance of sub-region texture among the frames of a face video. In our method, the Local Binary Pattern (LBP) [40] is employed to extract the face texture for making the histogram variances be immune to illumination interference. Furthermore, the Earth Movers's Distance (EMD) [41] is used to measure the histogram distance for ensuring that the histogram variances are consistent with human's vision. The finally obtained HVF images will be similar if they belong to the same expression, so that static facial recognition methods

can be utilised for the dynamic expression recognition. We test the HVFs classification by Support Vector Machines (SVMs) after Principal Component Analysis (PCA) dimensionality reduction. The accuracy of HVFs classification is very encouraging.

1.3 Object categorisation

Object categorisation involves determining whether or not an image contains some specific categories of objects as shown in Figure 1.5.

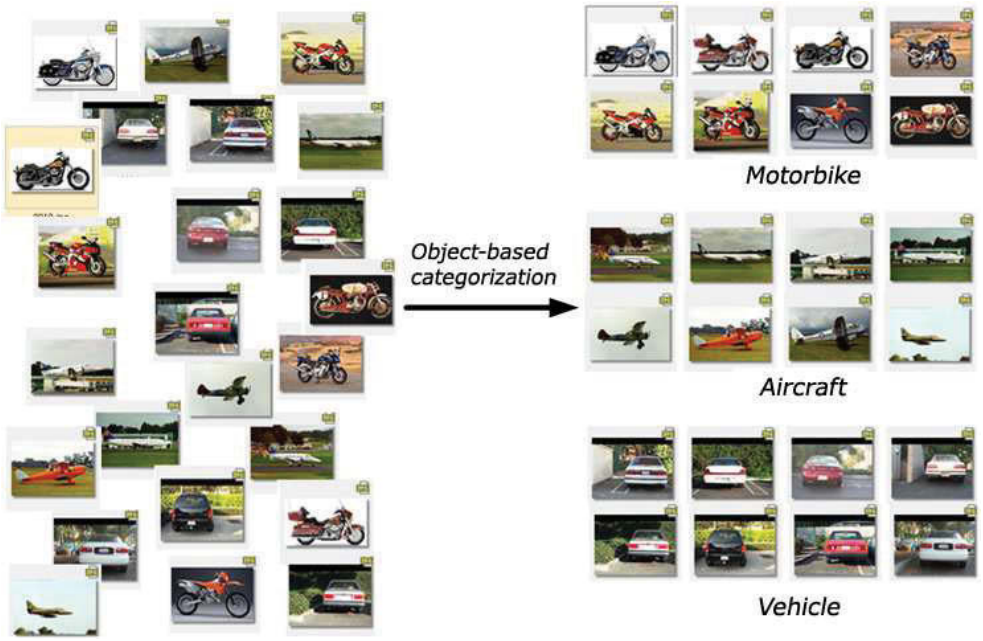


Figure 1.5: The task of object-based visual categorisation.

Object categorisation is still a challenging task in computer vision so far. One of the reasons is that the high intra-class variability means the global appearance of the objects within the same category may be quite different. It demands that the category models should be representative to the objects that belong to the same category and be flexible enough to accommodate intra-class variability. Another major disturbance for object categorisation is irrelevant background which

hampers learning category models by using clustering techniques, especially the backgrounds across training samples are similar.

There is a broad agreement that one category is modeled as a collection of local salient parts (i.e., stable regions representing the category) and these salient parts will be used for classifier training. Figure 1.6 shows a few examples of salient parts for car category. However, manually labeling the salient parts in a large number of training samples will be arduous and often too subjective.

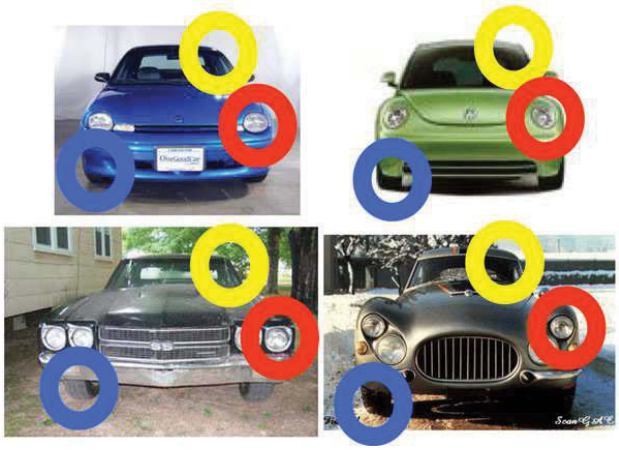


Figure 1.6: Examples of salient parts (colored circles) that form the model of car category [42].

Here, the major task we are targeting is to automatically find the salient parts for arbitrary object categories and then recognise the categories based on the salient parts. In practice, the sample images can be easily labeled to indicate whether or not the target objects are in the image. But annotating (locating) and aligning such objects is often infeasible. In this case, we assume that the training data are images containing target objects with different scales, rotations and positions. This assumption naturally poses a multiple-instance learning (MIL) [43] (or called weakly supervised learning) problem. Unlike standard supervised learning which receives a set of instances labeled positive or negative, the MIL learner receives a set of bags that are labeled positive or negative. Each bag contains many instances. A bag is labeled negative if all the instances in it are negative. On the other hand, a bag is labeled positive if there is at least one

instance in it which is positive. From a collection of labeled bags, the learner tries to induce concepts that will label individual bag or instances correctly.

1.3.1 Multiple-instance learning

In standard supervised (or single-instance) learning, the training set is given by $\mathcal{D}' = \{\langle \mathbf{x}_i, y_i \rangle\}_{i=1}^{n}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is an instance, and for each $\mathbf{x}_i$, there is a known label $y_i \in \mathbb{Y} = \{0, 1\}$. The task is to learn a classifier $f : \mathbb{R}^d \to \mathbb{Y}$. This learning usually requires much manual labor on labeling. In contrast, the multiple-instance learning (MIL) avoids labeling the individual instances and assigns only one label to a collection of instances instead. Such a collection of instances is called a bag. More formally, the training set in MIL is given by $\mathcal{D} = \{\langle B_i, y_i \rangle\}_{i=1}^{m}$, where bag $B_i = \{\mathbf{x}_{i,j}\}_{j=1}^{|B_i|}$, $\mathbf{x}_{i,j} \in \mathbb{R}^d$ is an instance and $|B_i|$ is the number of instances in B_i. Let $y_{i,j} \in \{0, 1\}$ be the latent label of instance $\mathbf{x}_{i,j} \in B_i$. Then, the label of B_i is known as $y_i = \max\{y_{i,j}\}_{j=1}^{|B_i|}$. The aim of MIL is to learn a classifier that is able to classify new bags or new instances.

Many tasks in computer vision and machine learning can be naturally cast as an MIL problem. For instance, object categorisation which involves determining whether or not an image contains a certain category of objects [44][45][46]. To tackle the intra-class variability, e.g., appearance diversity of vehicles according to different makes and models, MIL treats each image as a set of local regions, but only those regions that carry category-specific information are regions of interest (ROIs) for purposes of classification. For example, all the wheel regions of vehicles have common circular shapes, so the wheel regions are ROIs. Other regions have random features and possess no discriminative power. From this viewpoint, an image is a bag and the image regions are instances of the image categorisation problem. Likewise, many other applications are also able to be treated as MIL problems, e.g., content-based image retrieval [47][48][49], segmentation [50], object tracking [51], human detection [52] and computer aided diagnosis [53][54].

Loosely speaking, if an instance repeatedly occurs in different positive bags, this instance is called a concept (concepts for negative bags are not considered here because negative instances are usually randomly distributed) and the weights of concepts indicate the frequencies of which concepts occur in positive bags.

Learning concepts is critical for MIL algorithms. However, MIL algorithms often confront a problem of inducing weighted concepts from a large instance space (large value of $n = \sum_{i=1}^{m} |B_i|$ as each bag may contain many instances) and a large feature space (feature vectors extracted from instances are high-dimensional. E.g., SIFT features [55] are usually of 128-D). How to efficiently induce important concepts from a large instance+feature space is still challenging for MIL. Another issue is that many existing MIL algorithms follow a multiple-instance setting such that a positive bag must contain at least one true positive instance, whereas a negative bag contains negative instances only. This setting is often not true in computer vision tasks because the labeling processing is quite subjective and visual features extracted from instances may be distorted due to changes of scale, illumination and view angle etc.. Those bags labeled positive but containing no true positive instances and bags labeled negative but containing true positive instances are called labeling noise.

MIL is firstly proposed in the context of drug activity prediction [43]. Since then, many efforts have been endeavored to address this learning with ambiguous labeling. Maron et al. [56] proposed the Diverse Density to estimate the instance distribution in the feature space. Specifically, let $\left\{B_i^+\right\}_{i=1}^{m^+}$ and $\left\{B_i^-\right\}_{i=1}^{m^-}$ denote positive and negative bags respectively and $\mathbf{x}$ be a concept, the Diverse Density estimator (DDE) of $\mathbf{x}$ is defined as:

$$\hat{f}_{dde}\left(\mathbf{x}\right) = \prod_{i=1}^{m^+} Pr\left(\mathbf{x}|B_i^+\right) \prod_{i=1}^{m^-} Pr\left(\mathbf{x}|B_i^-\right). \tag{1.1}$$

The $Pr\left(\mathbf{x}|B_i^+\right)$ and $Pr\left(\mathbf{x}|B_i^-\right)$ in (1.1) are defined using a *noisy-or* model as below,

$$Pr\left(\mathbf{x}|B_i^+\right) = 1 - \prod_j (1 - Pr(\mathbf{x}|\mathbf{x}_{i,j}^+)) \tag{1.2}$$

$$Pr\left(\mathbf{x}|B_i^-\right) = \prod_j (1 - Pr(\mathbf{x}|\mathbf{x}_{i,j}^-)) \tag{1.3}$$

where

$$Pr\left(\mathbf{x}|\mathbf{x}_{i,j}\right) = \exp(-\left\|\mathbf{x}_{i,j} - \mathbf{x}\right\|^2). \tag{1.4}$$

The target concept can be found by maximizing DDE (1.1), i.e.,

$$\mathbf{x} = \arg\max_{\mathbf{x}} \hat{f}_{dde}(\mathbf{x}) . \tag{1.5}$$

Based on the Diverse Density (DD), EM-DD [57] utilised an Expectation Maximisation framework to improve the concept learning. Since learning a single concept might be insufficient to capture the multi-modal distribution, GEM-DD [58] and DD-SVM [59] iteratively located multiple concepts from various initial locations using Quasi-Newton methods.

The MIL problem has also been tackled by some variations of the standard supervised learning. mi/MI SVM [60] treated the multiple-instance setting as extra constraints of Support Vector Machine (SVM) optimisation and learned hyperplanes to separate bags or instances. Many more MIL approaches, such as MILES [46], MIL for sparse positive bags [61], IS-MIL [44] and MILIS [62], were also based on SVM. Aside from SVM, there are some other supervised learning methods that have also been modified to suit the MIL. These methods include the nearest neighbor [63], decision tree [64], Bayesian [53], Boosting [65], randomised trees [66] and logistic regression [67].

Many current MIL approaches are sensitive to labeling noise (e.g., DD-based approaches such as GEM-DD [58] and DD-SVM [59]). To our knowledge, MILES [46] is the only one that is reportedly robust to labeling noise and capable of learning multiple concepts. However, MILES maps every bag into an instance space whose dimensionality is given by the total number of instances across all bags. Then, the 1-norm SVM is used to select concepts and to train classifiers. The high-dimensional issue will lead to high complexity for both bag mapping and 1-norm SVM optimisation even for a moderate scale dataset. MILIS [62] later addressed the high-dimensional issue by initially selecting a concept from each bag and all the selected concepts were iteratively optimised by an EM-like framework. Although it is much more efficient, MILIS becomes vulnerable to labeling noise according to our experiments and often learns too few concepts.

1.3.2 Our approach for object categorisation

In our application, an image is treated as an MIL bag and the local regions of the image are treated as instances. As shown in Figure 1.6, the salient parts should occur in every positive bag (image), so each of the salient parts is actually a concept to be sought. We propose a novel MIL algorithm, MIL-SKDE (multiple-instance learning with supervised kernel density estimation), which is able to conveniently learn concepts in a large feature space and is robust to labeling noise. Firstly, we advance a modified version of kernel density estimation (KDE) function to estimate the instance distribution. The modified KDE is named supervised kernel density estimation (SKDE) as it considers class labels of data points. SKDE is an alternative to the well-known diverse density estimation (DDE) [56] and more robust to the labeling noise. The same as DDE, the modes (local maxima) of SKDE are the concepts to be learned. As mean shift is widely used to locate modes of KDE, we extend mean shift to its supervised version (named supervised mean shift) to adapt it to SKDE. Similar to mean shift, the supervised mean shift is also steepest ascent (and it only computes the first order gradient) with a varying step sise that is the magnitude of the gradient. Supervised mean shift is handy to seek modes in a large feature space because, from an initial point, it can quickly converge to a mode without expensively computing Hessian matrix (a second-order gradient) in the high-dimensional space.

1.4 Contributions

The contributions of this thesis are summarised as follows.

- For facial expression recognition, we develop a new feature representation, Histogram Variances Faces (HVFs), to grasp the dynamic feature, and the expression recognition rate is quite encouraging by using HVFs.
- An update of the Histogram Variances Faces, which is called Hexagonal Histogram Variances Faces (HHVFs), is later proposed for better efficiency and performances in facial expression recognition.

- We develop a novel multi-instance learning (MIL) algorithm, Multiple-instance Learning with Supervised Kernel Density Estimation (MIL-SKDE), for object categorisation and our method outperforms other state-of-the-art approaches.
- We develop a fast version of MIL-SKDE called multiple-instance learning with a speed-up supervised kernel density estimation (MIL-SS). Compared to MIL-SKDE, MIL-SS significantly reduces the expensiveness of training process meanwhile maintaining comparable performance.
- Finally, we apply MIL-SS in a bag-of-words model and develop an object-categorisation software system which consists of four components, i.e., codebook generation, feature coding, feature pooling and classification. The whole system is modularised and each of the components can be easily modified.

1.5 Outline of this thesis

The organisation of the rest of the thesis is:

- In Chapter 2, we briefly describe some related work on feature extraction and machine learning methods. Those methods will be utilised in our work.
- In Chapter 3, we propose a dynamic feature presentation, Histogram Variances Faces (HVFs), for facial expression recognition.
- We extend HVFs to a hexagonal structure and develop the Hexagonal Histogram Variances Faces (HHVFs) for facial expression recognition in Chapter 4 .
- In Chapter 5, we describe an innovative MIL algorithm, Multiple-instance Learning with Supervised Kernel Density Estimation (MIL-SKDE), and its applications for image classification and object categorisation.
- In Chapter 6, we speed up the training process of MIL-SKDE and propose MIL-SS. Then, we apply MIL-SS in a bag-of-words model and develop a software demo for object categorisation.

- Chapter 7 is to draw conclusions.

Chapter 2

Related feature extraction and pattern recognition methods

In this section, we briefly describe the related existing algorithms that will be utilised in the rest sections on feature extraction and pattern recognition. Here, we only give the the review of related technologies. For more details of such approaches, please refer to the references listed in the corresponding reviewed papers.

2.1 Feature extraction

Feature extraction is an essential pre-processing step for pattern recognition and machine learning problems. It transforms the input graphic data into a reduced representation set of features (also named feature vector) that truly maintains the distinct information of the original data. Then, the extracted features, instead of the original data, will be treated as the input for the subsequent pattern recognition algorithms. Note that here we refer the term "feature extraction" to not only feature extraction but also feature detection as feature detection is a pre-requisite to get the features in most cases.

The feature extraction methods for videos and images can be summarised as follows[1].

[1]Modified from Wikipedia - *http* : *//en.wikipedia.org/wiki/Feature_extraction*

- Extraction of low-level features
  - Edge detection
  - Corner detection
  - Blob detection
  - Ridge detection
  - Scale-invariant feature transform
  - Curvature.
  - Motion detection.
- Extraction of shape features
  - Thresholding
  - Blob extraction
  - Template matching
  - Hough transform
- Others
  - Extraction of deformable, parameterised shapes
  - Extraction of active contours (snakes)

The computer vision community has advanced considerable feature extraction methods. In this section, we review those approaches that are closely related to our work.

2.1.1 Haar-like features

Haar-like features are image features used in object recognition. They owe their name to their intuitive similarity with Haar wavelets and were used in the first real-time face detector proposed by Viola et al. [68]. A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. Figure 2.1 shows some samples of Haar-like features.

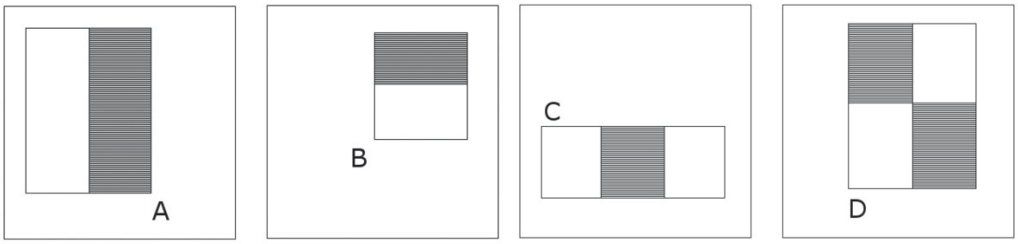


Figure 2.1: Examples of calculation of Haar-like features within the detection window. The intensity sum of the pixels within the white rectangles is subtracted by the density sum of pixels in the grey rectangles. Such pixel difference is a Haar-like feature value. Two-rectangle features are shown in (A) and (B). (C) shows a three-rectangle feature, and (D) shows a four-rectangle feature. [68]

Because of its simplicity, a Haar-like feature has the key advantage of lower computation over most other features. A fast way to calculate Haar-like feature is to use the integral images (or called summed area tables). The integral image at location (x, y) contains the sum of the pixels above and to the left of (x, y),

$$I(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'), \tag{2.1}$$

where $I(x, y)$ is the density of integral image and $i(x', y')$ is the density of original image. Once the integral image has been computed based on the original image, the task of calculating sum of any rectangle on original image can be accomplished in constant time. Specifically, using Figure 2.2 [1] as an example, the sum value is just

$$\sum_{\substack{A(x) < x' \leq C(x) \\ A(y) < y' \leq C(y)}} i(x', y') = I(C) + I(A) - I(B) - I(D). \tag{2.2}$$

The Haar-like features are then used to categorise object regions of an image. For instance, suppose we have a dataset of human face images. It is a common observation that among all faces the region of the eyes is darker than the region of the cheeks. Therefore a common haar-like feature for face detection is a set of two adjacent rectangles that lie above the eye and the cheek region as the feature

[1] Figure 2.2 is from http://upload.wikimedia.org/wikipedia/commons/e/ee/Prm_VJ_fig3_computeRectangleWithAlpha.png

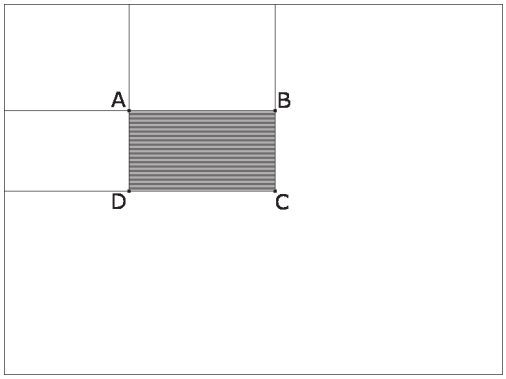


Figure 2.2: Computing the sum of a rectangular area using integral image.

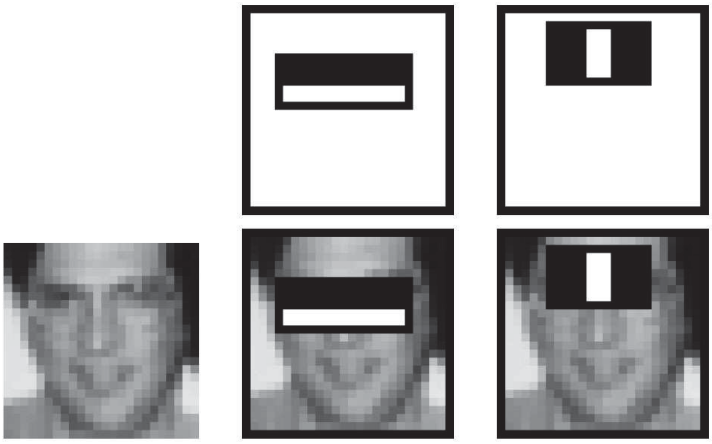


Figure 2.3: Typical Haar-like features for face detection [68].

values are quite stable for such face region. The position of these rectangles is defined relative to a detection window that acts like a bounding box to the target object (e.g., the face in this case) as shown in Figure 2.3.

Haar-like features combining with Adaboost (described in Section 2.2.3) algorithm will be used to detect the human face regions from videos at the early stage of our facial expression recognition. Since describing features is the emphasis of this section, please see [68] for more details of facial detection using Haar-like features and Adaboost.

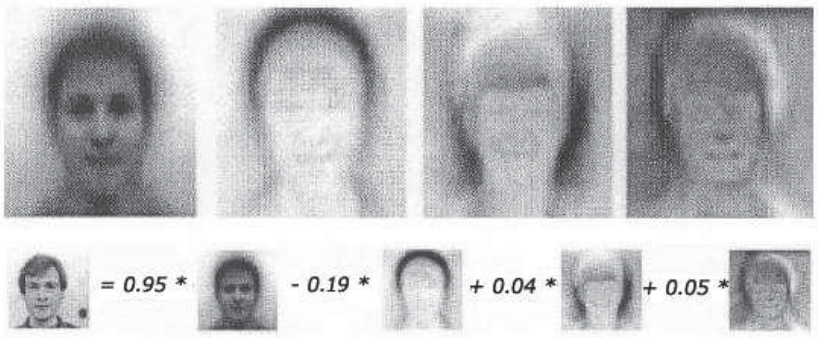


Figure 2.4: Visualisation of the eigenface approach: The first row shows a set of eigenfaces derived from many face samples. The second line shows that a new face can be represented as a linear combination of the eigenfaces. The new feature vector of the test face formed by the coefficients of the linear combination, which is $(0.95, -0.19, 0.04, 0.05)$. By representing a face image with the coefficients, the dimensionality of face data can be greatly reduced.

2.1.2 Eigenface

Eigenfaces are commonly used in the computer vision problem of human face recognition. Generally, eigenfaces [69] can be seen as a set of "standardised face ingredients" derived statistically from many face sample images. Any human face can be considered to be a linear combination of these eigenfaces. For example, one's face might be composed of 95% the first eigenface plus -19% the second eigenface, and plus 4% the third eigenface. Remarkably, it does not take many eigenfaces combined together to achieve a fair approximation of most faces. Figure 2.4 illustrates a visualisation of the eigenface approach. The first row shows a set of eigenfaces. The second row shows a new face that is represented as a linear combination of the eigenfaces.

Formally, suppose that column vectors $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{R}^n$ represent m face images, and the matrix $\mathbf{T} = [\mathbf{a}_1 - \mathbf{u}, \ldots, \mathbf{a}_m - \mathbf{u}] \in \mathbb{R}^{n \times m}$ denotes the data set with each column representing a preprocessed training image, where $\mathbf{u} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{a}_i$ is the mean. The eigenfaces are actually a set of principal eigenvectors of covariance matrix $\mathbf{S} = \mathbf{T}\mathbf{T}^T$. The eigenfaces are usually generated by a mathematical procedure called principal component analysis (PCA) summarised as follows.

1. **Prepare a training set of face images.** The pictures constituting the

training set should be aligned (e.g., eyes have the same positions) across all images. They must also have the same size, e.g., r by c pixels. Each image is treated as one vector $\mathbf{a}_i \in \mathbb{R}^n$, simply by concatenating the rows (or columns) of pixels in the original image, resulting in a single vector with $n = r \times c$ elements. Suppose that there are m images in the training set and they are stored in a single matrix $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_m] \in \mathbb{R}^{n \times m}$, where each column of the matrix is an image.

2. **Subtract the mean.** The average image $\mathbf{u}$ needs to be computed and then subtracted from each original image in $\mathbf{A}$. The new training set is denoted as $\mathbf{T} = \mathbf{A} - \mathbf{uh}$, where $\mathbf{h}$ is a $1 \times N$ row vector of all 1s, i.e., $h_i = 1$ for $i = 1, \ldots, N$ and $\mathbf{u} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{a}_i$.

3. **Calculate the eigenvectors and eigenvalues of the covariance matrix $\mathbf{S} = \mathbf{T}\mathbf{T}^T \in \mathbb{R}^{n \times n}$.** Each eigenvector has the same dimensionality $(n \times 1)$ as the original images, and thus can itself be seen as an image. The eigenvectors of this covariance matrix are therefore called eigenfaces. Note that because image size n is normally a large value, directly calculating eigenvectors of $\mathbf{S}$ is extremely expensive. Alternatively, the eigenvectors of $\mathbf{S}$ are computed by the singular value decomposition (SVD) on matrix $\mathbf{T}$, i.e., $\mathbf{T} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ is an $n \times n$ unitary matrix, $\mathbf{\Sigma}$ is an $n \times m$ rectangular diagonal matrix with nonnegative real numbers on the diagonal, and $\mathbf{V}^T$ is an $m \times m$ unitary matrix. The columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{S} = \mathbf{T}\mathbf{T}^T$.

4. **Choose the principal eigenvectors.** The $n \times n$ covariance matrix will result in n eigenvectors. Here, only those eigenvectors (eigenfaces) with the largest associated eigenvalue are kept.

These eigenfaces can now be utilised to represent both existing and new faces. Suppose that $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k \in \mathbb{R}^n$ denote the k chosen eigenfaces, and $\mathbf{x}_i \in \mathbb{R}^n$ denotes a face image. Then, to represent $\mathbf{x}_i$ using eigenfaces, we need only to

compute the corresponding vector

$$\mathbf{h}_i = \begin{bmatrix} \mathbf{z}_1^T \mathbf{x}_i \\ \mathbf{z}_2^T \mathbf{x}_i \\ \vdots \\ \mathbf{z}_k^T \mathbf{x}_i \end{bmatrix} \in \mathbb{R}^k. \tag{2.3}$$

Thus, whereas $\mathbf{x}_i \in \mathbb{R}^n$, the vector $\mathbf{h}_i$ now gives a lower k-dimensional approximation/representation for $\mathbf{x}_i$ as $k << n$. Eigenfaces or PCA is therefore also referred to as a dimensionality reduction algorithm. Finally, the vectors $\mathbf{h}_i$'s, rather than $\mathbf{x}_i$'s are used for classification such as facial recognition.

Eigenfaces will be used in our work for human expression recognition.

2.1.3 Scale-invariant feature transform

Scale-invariant feature transform (SIFT) [70][55] is a method for detecting and extracting distinctive invariant features from images that can be later used to perform reliable matching between different views of an object or scene. Because a SIFT feature descriptor is invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes, it has been widely utilised in object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, and match moving.

Basically, SIFT features can be obtained via four steps described as below.

1. **Scale-space extrema detection**. This step is to detect the interest points, which are called keypoints in the SIFT framework, from an image. To do so, the image is firstly convolved with Gaussian filters $G(x, y, \sigma_i) = \frac{1}{2\pi\sigma_i^2} e^{-(x^2+y^2)/2\sigma_i^2}$ at different scales $\sigma_i, i = 1, \ldots, n$, and then the difference of successive Gaussian-blurred images are taken. Keypoints are finally taken as maxima/minima of the Difference of Gaussians (DoG) that occur at multiple scales. Specifically, let $D(x, y, \sigma)$ be a DoG image,

$$D(x, y, \sigma_i) = L(x, y, \sigma_i) - L(x, y, \sigma_{i+1}), \tag{2.4}$$

where the Gaussian-blurred image $L(x, y, \sigma)$ is the convolution of the orig-

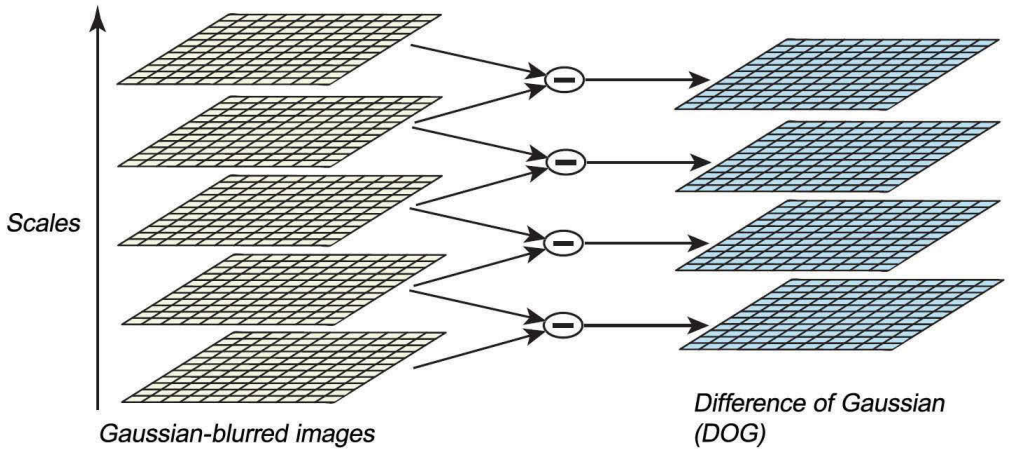


Figure 2.5: Left column is the Gaussian-blurred images with different scales σ_i's. Right column is the DoG images obtained by subtracting adjacent Gaussian-blurred images.

inal image $I(x, y)$ with the Gaussian blur $G(x, y, \sigma)$ at scale σ, i.e.,

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.5}$$

Hence a DoG image $D(x, y, \sigma_i)$ between scales σ_i and σ_{i-1} is just the difference of the Gaussian-blurred images at scales σ_i and σ_{i-1}. Figure 2.5 illustrates the generation of DoG. Once DoG images have been obtained, keypoints are identified as local minima/maxima of the DoG images across scales. This is done by comparing each pixel in the DoG images to its eight neighbors at the same scale and nine corresponding neighboring pixels in each of the neighboring scales. Altogether, it compares with the nearest 26 neighbors as shown in Figure 2.6. If the pixel value is the maximum or minimum among all compared pixels, it is selected as a candidate keypoint.

2. **Keypoints localisation.** DoG extrema detection produces too many candidate keypoints, of which some are unstable or do not contribute to feature description (e.g., the candidates having low-contrast or localizing along an edge responses are often vulnerable to noise). To discard those unstable

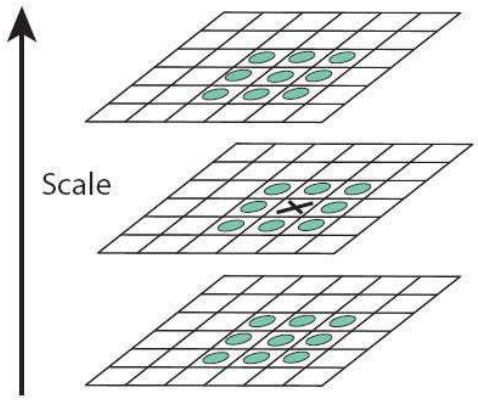


Figure 2.6: Maxima and minima of the difference-of-Gaussian images are detected by comparing the values of a pixel (marked with X) with the values of its 26 neighbors in a 3x3 regions at the current and adjacent scales (marked with circles) [55]

.

candidates, this step performs a detailed fitting to the nearby data for location, scale, and ratio of principal curvatures. Then, this information is used for candidate selection.

Firstly, for each candidate keypoint, its nearby data are interpolated so that the accurate position of keypoint can be determined. Unlike the previous approach that simply locates each keypoint at the location and scale of the extremum [70], the updated approach computes the interpolated location of the extremum, so it significantly improves matching and stability [55]. The interpolation uses the quadratic Taylor expansion of the DoG function (2.4) with the candidate keypoint as the origin, i.e., $\mathbf{0}$. This Taylor expansion is given by:

$$D(\mathbf{x}) = D(\mathbf{0}) + \frac{\partial D(\mathbf{0})^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D(\mathbf{0})}{\partial \mathbf{x}^2}\mathbf{x} \tag{2.6}$$

where $\mathbf{x} = (x, y, \sigma)$ is the offset from the candidate keypoint. The location of the extremum $\hat{\mathbf{x}}$ is determined by taking the derivative of (2.6) with respect to $\mathbf{x}$ and setting it to zero. If the offset $\hat{\mathbf{x}} > 0.5$ in any dimension, it indicates that the extremum lies closer to another candidate keypoint. In

this case, the candidate keypoint is changed and the interpolation performed instead about that point. Otherwise, the offset is added to its candidate keypoint to get the interpolated estimate for the location of the extremum.

Then, the keypoints with low contrast need to be discarded. To do that, the value of the second-order Taylor expansion $D(\mathbf{x})$ in (2.6) is computed at the offset $\hat{\mathbf{x}}$. In [55], if this value is less than 0.03, the candidate keypoint is discarded. Otherwise, it is kept, with final location $\mathbf{y} + \hat{\mathbf{x}}$ and scale σ, where $\mathbf{y}$ is the original location of the keypoint at scale σ.

Finally, those high edge responses need to be eliminated. This is because the DoG function (2.4) will have strong responses along edges, even if the candidate keypoint is not robust to small amounts of noise. If a keypoint poorly locates along an edge, in the DoG function (2.4), the principal curvature across the edge would be much larger than the principal curvature along it. Getting these principal curvatures requires solving for the eigenvalues of the second-order Hessian matrix, $\mathbf{H}$:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix},$$

where D is defined in (2.4), and the eigenvalues of $\mathbf{H}$ are proportional to the principal curvatures of D. Suppose k_1 is the larger eigenvalue and k_2 the smaller eigenvalue, explicitly computing the eigenvalues is actually unnecessary as the ratio $r = k_1/k_2$ is sufficient for SIFT's purposes. The sum $k_1 + k_2$ and product $k_1 k_2$ can be computed from the trace and determinant of $\mathbf{H}$ respectively,

$$\mathrm{Tr}\,(\mathbf{H}) = D_{xx} + D_{yy} = k_1 + k_2$$

$$\mathrm{Det}\,(\mathbf{H}) = D_{xx}D_{yy} - D_{xy}^2 = k_1 k_2.$$

The ratio $\mathbf{R} = \mathrm{Tr}\,(\mathbf{H})^2 / \mathrm{Det}\,(\mathbf{H})$ can be shown to be equal to $(r+1)^2/r$, which depends only on r rather than individual values of k_1 and k_2. $\mathbf{R}$ is minimum when $r = 1$, i.e., $k_1 = k_2$, and monotonously increases when $r \geq 1$. Therefore, the higher value of $\mathbf{R}$ implies a higher absolute difference between the two principal curvatures of (2.4). It follows that, for some threshold

eigenvalue ratio $r_{\mathbf{th}}$, if $\mathbf{R} > (r_{\text{th}}+1)^2/r_{\text{th}}$ for a candidate keypoint, that keypoint is poorly localised and hence rejected. Empirically, $r_{\text{th}} = 10$ [55].

3. **Orientation assignment**. In this step, each keypoint is assigned one or more orientations (called dominant orientations) based on local image gradient directions. Such orientation(s) will be used to rotate corresponding keypoint in order to achieve orientation invariance. To get the dominant orientations, the Gaussian-blurred image $L(x, y, \sigma)$ in (2.5), at the keypoint's scale σ is taken so that all computations are performed in a scale-invariant manner. For an image sample $L(x, y)$ at scale σ, the gradient magnitude, $m(x, y)$, and orientation, $\theta(x, y)$, are precomputed by:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{2.7}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \tag{2.8}$$

(2.7) and (2.8) are computed for every pixel in a neighboring region around the keypoint in the Gaussian-blurred image. An orientation histogram with 36 bins is formed, with each bin covering 10 degrees. Each sample in the neighboring window added to a histogram bin is weighted by its gradient magnitude and by a Gaussian-weighted circular window with 1.5σ (σ is the scale of the keypoint). The peaks in this histogram correspond to dominant orientations. Once the histogram is filled, the orientations corresponding to the highest peak and local peaks that are within 80% of the highest peaks are assigned to the keypoint. In the case of multiple orientations being assigned, an additional keypoint is created having the same location and scale as the original keypoint for each additional orientation. According to [55], only about 15% of points are assigned multiple orientations, but these contribute significantly to the stability of matching.

4. **Keypoint descriptor**. Previous steps have found positions of keypoints at particular scales and assigned orientations to such keypoints. Those works ensure invariance to image location, scale and rotation. Now, a descriptor

vector for each keypoint needs to be computed such that the descriptor is highly distinctive and partially invariant to the remaining variations such as illumination, 3D viewpoint, etc. This step is performed on the image closest in scale to the keypoint's scale.

First, a set of orientation histograms are created on a 4x4 pixel neighborhood with 8 bins each. These histograms are computed from magnitude and orientation values of samples in a 16 x 16 region around the keypoint such that each histogram contains samples from a 4 x 4 subregion of the original neighborhood region. The magnitudes are further weighted by a Gaussian function with σ equal to one half the width of the descriptor window. The descriptor then becomes a vector of all the values of these histograms. Since there are 4 x 4 = 16 histograms each with 8 bins, the vector has 128 elements. This vector is then normalised to unit length in order to enhance invariance to affine changes in illumination. To reduce the effects of non-linear illumination, a threshold of 0.2 is applied and the vector is again normalised. Figure 2.7 [55] illustrates the generation of SIFT descriptor.

There are also some variants of SIFT that have been proposed recently. For instance, SURF (Speeded Up Robust Features) [71] greatly accelerates the processing of keypoint detection with little performance loss. HOG (Histograms of Oriented Gradients) [72] adopts a similar way to generate the feature descriptor and has excellent performance in human detection. GLOH (Gradient Location and Orientation Histogram) [73] extends SIFT by changing the location grid and using PCA to reduce the size. GLOG was designed to increase robustness and distinctiveness of SIFT descriptor. In the comprehensive test within [73], GLOG has the best performance in the most tests comparing with other common descriptors.

2.1.4 Speeded up robust feature

Speeded-Up Robust Features (SURF) [71] can be seen as a speed-up revision of SIFT features. The procedures of SURF are very similar to SIFT and are described below.

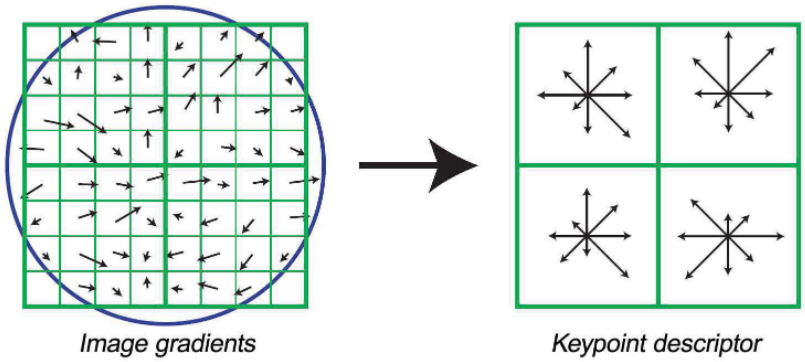


Figure 2.7: To get a SIFT descriptor, the gradient magnitudes and orientations in a region around the keypoint are computed as shown on the left. Those magnitudes and orientations are weighted by a Gaussian window indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in [55] use 4x4 descriptors computed from a 16x16 sample array.

1. Find the interest points. SURF utilises the maxima of the determinant of the Hessian matrix whereas SIFT uses extreme values in Difference of Gaussians (DoG), but both of them are to find extrema in scale space.

2. Interest points orientation assignment. SIFT uses the histogram of gradient in a rectangular area and SURF calculates the Haar wavelet responses in x and y direction within a circular neighborhood around the interest point.

3. Point descriptor. SIFT uses an 8 directions histogram for one of the 4×4 cells and gets a 128 dimensional vector; SURF descriptor is based on sum of Haar Wavelet responses $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$ for one of the 4×4 cells and get a 64 dimensional vector.

2.1.5 Histogram of oriented gradients

Histogram of oriented gradients (HOG) descriptors [72] are feature descriptors used in object detection of computer vision. The basic idea behind the HOG

descriptors is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The algorithm is shown as follows.

1. Gradient computation. For each pixel, compute the horizontal and vertical gradient values. The best masks in experiments are $[-1, 0, 1]$ (horizontal) and $[-1, 0, 1]^T$ (vertical). In this step, the image preprocessing e.g. normalisation is not necessary due to the descriptor normalisation essentially achieves the same result. Also the Gassian smoothing would not perform better in practice.

2. Orientation bining. Generating orientation histogram using above gradient values, 9 histogram channels performed best in human detection experiments.

3. Descriptor blocks. There are two main block geometries - rectangular R-HOG blocks and circular C-HOG blocks. R-HOG are commonly square grids, parametrised by the number of cells per block, the number of pixels per cell, and the number of channels per cell histogram.

4. Block normalisation. Some normalisation factors which provide similar performance are listed below, where v is vector and e is a constant.

$$L2 - norm : f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$$

$$L1 - norm : f = \frac{v}{\|v\|_1 + e}$$

$$L1 - sqrt : f = \sqrt{\frac{v}{\|v\|_1 + e}}$$

2.2 Machine learning and pattern recognition

In machine learning, the purpose of pattern recognition is to assign a label to a given input instance. An example of pattern recognition is classification, which attempts to map each input instance to one of a given set of class labels (for

example, determine whether a given face video indicates a certain expression, e.g., happiness, sadness, fear, disgust, surprise and anger). The term "instance" refers to a piece of input data for which an output label (value) is generated. An instance is formally described by a vector of features, which together constitute a description of all known characteristics of the instance via feature extraction. These feature vectors can be seen as defining points in an appropriate multi-dimensional feature space, and pattern recognition methods can be applied to manipulate those points in the feature space, such as separating feature points according to their class labels (classification), mapping each feature point to a value (regression) and grouping similar feature points together (clustering).

Traditionally, the algorithms of pattern recognition are divided into two categories: supervised learning or unsupervised learning. In supervised learning, each feature vector in training data is paired with a given label. For unsupervised learning, only the feature vectors are known and labels are not given. Formally, the problem of supervised pattern recognition can be stated as follows: Suppose an unknown function $g : \mathcal{X} \to \mathcal{Y}$ is the "ground truth" that maps input instances $\boldsymbol{x} \in \mathcal{X}$ to output labels $y \in \mathcal{Y}$, along with training data $\mathbf{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ assumed to represent accurate examples of the mapping, produce a function $h : \mathcal{X} \to \mathcal{Y}$ that approximates as closely as possible the correct mapping g. (For example, if the problem is facial expression recognition, then $\boldsymbol{x}_i$ is some representation of a face video and y is an expression like "happiness" or "sadness" etc.). In contrast, unsupervised learning is not trying to find the mapping between feature vectors and labels but to find hidden structure in unlabeled training data $\mathbf{D}' = \{(\boldsymbol{x}_1), \ldots, (\boldsymbol{x}_n)\}$.

In this section, we will review those widely-used algorithms of pattern recognition that are related to our work.

2.2.1 K-means

K-means is a simple and widely-used clustering algorithm which aims to partition n observations into k ($k \leq n$) clusters in which each observation belongs to the cluster with the nearest mean. Formally, Given a set of observations $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, where each observation $x_i \in \mathbb{R}^d$ is a d-dimensional real vec-

tor, k-means clustering aims to partition the n observations into k sets $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to:

$$\mathbf{S} = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2,$$

where $\boldsymbol{\mu}_i = \frac{1}{|S_i|} \sum_{\mathbf{x}_j \in S_i} \mathbf{x}_j$ is the mean of the vectors in set S_i.

For K-means, the most common algorithm utilises an iterative refinement technique to obtain $\mathbf{S}$. Given an initial set of k means $\boldsymbol{\mu}_1^{(1)}, \ldots, \boldsymbol{\mu}_k^{(1)}$, the algorithm proceeds by alternating between two steps[1]:

1. **Assignment step:** Assign each observation to the cluster with the closest mean.
$$S_i^{(t)} = \left\{\mathbf{x}_p : \left\|\mathbf{x}_p - \boldsymbol{\mu}_i^{(t)}\right\| \leq \left\|\mathbf{x}_p - \boldsymbol{\mu}_j^{(t)}\right\| \ \forall\ 1 \leq j \leq k\right\},$$
Where each $\mathbf{x}_p$ goes into exactly one $S_i^{(t)}$, even if it could go in more than one of them.

2. **Update step:** Calculate the new means to be the centroid of the observations in the cluster.
$$\boldsymbol{\mu}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j.$$
Then, go back to step one.

The algorithm is deemed to have converged when the assignments no longer change, i.e., $\boldsymbol{\mu}_i^{(t+1)} = \boldsymbol{\mu}_i^{(t)}, \forall i = 1, \ldots, k$.

As K-means is a heuristic algorithm, there is no guarantee that it will converge to the global optimum, and the result may depend on the initial clusters. However, as it is simple and usually very fast, K-means is nearly the most popular algorithm for clustering in computer vision.

2.2.2 Kernel density estimation and mean shift

Kernel density estimation (KDE) [74], also termed the Parzen-Rosenblatt window method [75][76], is a non-parametric way to estimate the probability density

[1]From wikipedia: http://en.wikipedia.org/wiki/K-means_clustering

function of a random variable based on a finite amount of data samples. Supposing that $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ are i.i.d. data drawn from an unknown density $f(\mathbf{x})$, then the KDE, denoted by $\hat{f}_{kde}$, can be used to approximate the original unknown density in the way of:

$$f(\mathbf{x}) \approx \hat{f}_{kde}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h_i^d} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right), \tag{2.9}$$

where h_i is a variable bandwidth associating with data point $\mathbf{x}_i$ for all $i = 1 \ldots n$ [77]. Normally, h_i is estimated beforehand. And function $K(x)$ is a bounded d-variate Kernel function with compact support satisfying the following properties.

- Normalized: $\int_{\mathbb{R}^d} K(\mathbf{x})\, d\mathbf{x} = 1$.
- Symmetric: $\int_{\mathbb{R}^d} \mathbf{x} K(\mathbf{x})\, d\mathbf{x} = 0$.
- Exponential weight decay: $\lim_{\mathbf{x}\to\infty} \|\mathbf{x}\|^d K(\mathbf{x}) = 0$.
- Uncorrelated: $\int_{\mathbb{R}^d} \mathbf{x}\mathbf{x}^T K(\mathbf{x})\, d\mathbf{x} = c\mathbf{I}$.

We are interested in special class of radially symmetric kernels satisfying:

$$K(\mathbf{x}) = c \cdot k\left(\|\mathbf{x}\|^2\right), \mathbf{x} \in \mathbb{R}^d, \tag{2.10}$$

where the function $k(x)$ is called the profile of kernel $K(\mathbf{x})$ (defined only for $x \in \mathbb{R}^+$). c is the normalisation constant, which makes $K(\mathbf{x})$ be integrated to one, is assumed strictly positive.

One of the commonly-used kernels is the multivariate Gaussian kernel $K_N = (2\pi)^{-d/2} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right)$, whose profile is

$$k_N(x) = \exp\left(-\frac{1}{2}x\right), x \geq 0. \tag{2.11}$$

Another example is the Epanechnikov profile

$$k_E(x) = \begin{cases} 1-x & 0 \leq x \leq 1 \\ 0 & x > 1, \end{cases} \tag{2.12}$$

which yields the Epanechnikov kernel

$$K_E(\mathbf{x}) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)\left(1-\|\mathbf{x}\|^2\right) & \|\mathbf{x}\|^2 \leq 1 \\ 0 & \text{otherwise,} \end{cases}$$

where c_d is the volume of the unit $d-dimensional$ sphere. These two profiles of kernels, i.e., (2.11) and (2.12), will suffice for most applications. Mostly, the type of kernel is not critical for the kernel density estimation. Actually, different kernels may have similar performances in most cases.

In many applications of KDE, the key purpose is to find the modes (local maxima) of the density function (2.9) located at the zeros of the gradient function. Mean shift algorithm is an elegant way to locate the modes. Putting (2.10) into (2.9), we yield:

$$\hat{f}_{kde}(\mathbf{x}) = \frac{c}{n}\sum_{i=1}^{n}\frac{1}{h_i^d}K\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right), \tag{2.13}$$

Taking the gradient of (2.13) and setting the gradient to zero yields,

$$\begin{aligned} \nabla\hat{f}_{kde}(\mathbf{x}) &= \frac{2c}{n}\sum_{i=1}^{n}\frac{\mathbf{x}-\mathbf{x}_i}{h_i^{d+2}}k'\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right) \\ &= \frac{2c}{n}\sum_{i=1}^{n}\frac{\mathbf{x}_i-\mathbf{x}}{h_i^{d+2}}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right) \\ &= \frac{2c}{n}\left[\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right)\right]\left[\underbrace{\frac{\sum_{i=1}^{n}\frac{\mathbf{x}_i}{h_i^{d+2}}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right)}{\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right)}}_{\text{mean vector}} - \mathbf{x}\right] \\ &= 0, \end{aligned} \tag{2.14}$$

where

$$g(x) = -k'(x). \tag{2.15}$$

Mean shift [78][79] is an iterative procedure to locate the KDE local maxima $\mathbf{x}'s$ that satisfy (2.14). Generally, mean shift procedure can be described as follows. Let $\mathbf{x}$ be one of the initial points. Then, $\mathbf{x}$ will gradually converge to a maximum of (2.9) via iteratively setting $\mathbf{x} = \bar{\mathbf{x}}$, where $\bar{\mathbf{x}}$ is the mean vector calculated by

$$\bar{\mathbf{x}} = \frac{\sum_{i=1}^{n} \frac{\mathbf{x}_i}{h_i^{d+2}} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right)}{\sum_{i=1}^{n} \frac{1}{h_i^{d+2}} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right)}. \tag{2.16}$$

2.2.3 Adaboost

AdaBoost, short for Adaptive Boosting formulated by Yoav Freund and Robert Schapire [80], is a classification algorithm for constructing a "strong" classifier as linear combination of "weak" classifiers:

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

where $h_t(x)$ is a "weak" classifer, hypothesis $H(x) = sign(f(x))$. AdaBoost generates and calls a "best" weak classifier in each of a series of rounds $t = 1, \ldots, T$. For each call, a distribution of weights D_t is updated that indicates the importance of training samples for the classification. On each round, the weights of each incorrectly classified sample are increased, and the weights of each correctly classified sample are decreased, so the new classifier focuses on the samples which have been incorrectly classified so far. The Algorithm 1 shows the Adaboost for the binary classification. Some examples of weak classifier $h_t \in \mathcal{H}$:

- Decision tree builder, perceptron learning rule - $\mathcal{H}$ infinite
- Selecting the best one from given finite set $\mathcal{H}$

Normally, AdaBoost is sensitive to noisy data and outliers. However, it can be less susceptible to the over-fitting problem than most learning algorithms.

Algorithm 1 Adaboost for the binary classification.

Input:

$\left\{\left(x^{(i)}, y^{(i)}\right) | i = 1, \ldots, m\right\}$, $x^{(i)} \in \mathcal{X}$, $y^{(i)} \in \{-1, 1\}$;

Number of iterations T;

Output:

The strong classifier, $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$;

1: Initialise weights $D_1(i) = 1/m, i = 1, \ldots, m$. $D_j(i)$ is the weight of sample $\left(x^{(i)}, y^{(i)}\right)$ at the j-th round.

2: **for** $t = 1, \ldots, T$ **do**

3: Find a classifier $h_t : \mathcal{X} \to \{-1, 1\}$, $h_t \in \mathcal{H}$ that maximises the absolute value of the difference of the corresponding weighted error rate ϵ_t from 0.5 with respect to the distribution D_t:

$$h_t = \underset{h_t \in \mathcal{H}}{\operatorname{argmax}} \ |0.5 - \epsilon_t| ,$$

where $\mathcal{H}$ is the family of weak classifiers, and $\epsilon_t = \sum_{i=1}^{m} D_t(i) I(y_i \neq h_t(x_i))$, I is the indicator function.

4: **if** $|0.5 - \epsilon_t| \leq \beta$, where β is a previously chosen threshold **then**

5: Stop.

6: **else**

7: Choose $\alpha_t \in \mathbb{R}$, typically $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$.

8: Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t},$$

where Z_t is the normalisation factor: $\sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i))$ which ensures that D_{t+1} will be a probability distribution.

9: **end if**

10: **end for**

11: **return** the final classifier:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

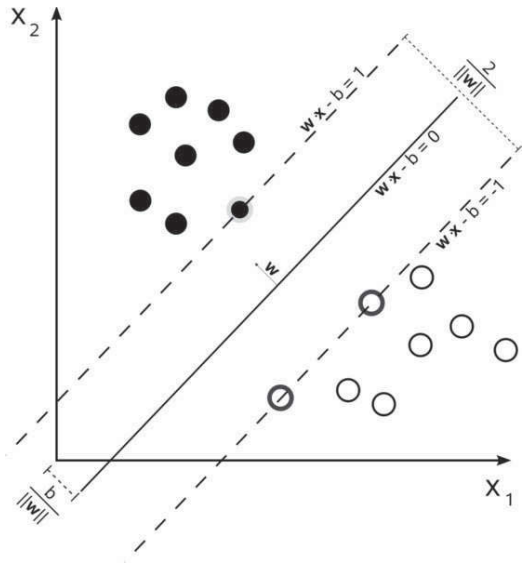


Figure 2.8: Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

2.2.4 Support vector machines

A (binary) support vector machine constructs a hyperplane in a high- or infinite-dimensional feature space. This hyperplane is a classifier to separate positive and negative data. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalisation error of the classifier. Figure 2.8 [1] shows the maximum-margin hyperplane obtained by an SVM. Samples on the margin are called the support vectors.

In real applications, the positive and negative training data may be not linearly separable in the feature space. In that case, the "kernel" has been introduced to map the original finite-dimensional space into a much higher-dimensional space, presumably making the separation easier in that space. Mathematically, the SVM can be formalised as an optimisation problem as follows.

Given some training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, where $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$.

[1] From wikipedia: http://en.wikipedia.org/wiki/Support_vector_machine

The y_i is either 1 or -1, indicating the class to which the point $\mathbf{x}_i$ is positive or negative. Each $\mathbf{x}_i$ is a d-dimensional real vector. SVM trys to find the maximum-margin hyperplane that divides the points having $y_i = 1$ from those points having $y_i = -1$. Such hyperplane can be written as the set of points $\mathbf{x}$ satisfying

$$\mathbf{w} \cdot \mathbf{x} - b = 0,$$

where the $\mathbf{w}$ and b are optimised by :

$$\min_{\mathbf{w},\xi,b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i \right\} \tag{2.17}$$

subject to (for any $i = 1, \ldots n$)

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \tag{2.18}$$

where C is a pre-defined constant, and ξ_i are introduced slack variables which measure the degree of misclassification of the data $\mathbf{x_i}$ if there exists no hyperplane that can "clearly" split the positive and negative training examples. By introducing Lagrange multipliers $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, the above constrained problem can be expressed as:

$$\min_{\mathbf{w},\xi,b} \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i[y_i(\mathbf{w} \cdot \mathbf{x_i} - b) - 1 + \xi_i] - \sum_{i=1}^{n} \beta_i \xi_i \right\} \tag{2.19}$$

with $\alpha_i, \beta_i \geq 0$. This leads to a dual form of SVM optimisation:
Maximise (with respect to α_i)

$$\tilde{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \tag{2.20}$$

subject to (for any $i = 1, \ldots, n$)

$$0 \leq \alpha_i \leq C,$$

and

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

$k(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel, and some common kernels include:

- Polynomial (homogeneous): $k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j})^d$.
- Polynomial (inhomogeneous): $k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$.
- Gaussian radial basis function: $k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2)$, for $\gamma > 0$.
- Hyperbolic tangent: $k(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\kappa \mathbf{x_i} \cdot \mathbf{x_j} + c)$, for some (not every) $\kappa > 0$ and $c < 0$.

2.2.5 Unsupervised and supervised topic models

In this section, we describe a hierarchical Bayesian model, Latent Dirichlet allocation (LDA) [81], that projects a text document or an image into a latent low dimensional space (i.e., topic space). This low-dimensional representation can be later used for regression and classification. LDA is originally an unsupervised model as the side information, e.g., the class labels or responses of documents, has not been utilised. In order to take into account side information for discovering more predictive representations, LDA has been extended to a few supervised manners, e.g., Supervised topic models (sLDA) [82], DiscLDA [83], Labeled LDA [84] and maximum entropy discrimination latent Dirichlet allocation (MedLDA) [85]. Specifically, we will utilise MedLDA for image classification in our later work as presented in the Chapters 5 and 6.

2.2.5.1 Latent Dirichlet allocation (LDA)

In LDA, the given data are a collection of documents, i.e., corpus D, and how many topics, say K, to which these documents belong. Each document consisting of a sequence of words is treated as a latent mixture of K topics, and each topic is a multinomial distribution over M words in a given vocabulary. LDA is a generative model that maps a document into a K-dimensional topic space in which

the regression or classification algorithms can be applied for given documents. Conventionally, the following notations are used in LDA model:

- A corpus is a collection of L documents denoted by $D = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_L\}$, and the i-th document $\mathbf{w}_i$ is a sequence of N_i words denoted by $\mathbf{w}_i = (w_{i1}, w_{i2}, \ldots, w_{iN_i})$, where $w_{i,j}$ is the j-th word in document $\mathbf{w}_i$.
- α is the parameter of the Dirichlet prior on the per-document topic distributions.
- $\beta = [\beta_1, \ldots, \beta_K]$ is the $M \times K$ matrix of topic distribution parameters, of which each β_k parameterises the topic-specific multinomial word distribution for topic k.
- θ_i is the topic distribution for document $\mathbf{w}_i$.
- $\mathbf{z}_{i,j}$ is the topic for the word $w_{i,j}$.

Only the words $w_{i,j}$ are observable variables, and the other variables are latent variables. Suppose all the latent variables have been known, the LDA process that generates all the documents of the corpus D is described as belows.

1. Draw a topic mixing proportion vector θ_i according to a K-dimensional Dirichlet prior: $\theta_i \sim \text{Dir}(\alpha)$, $\text{Dir}(\alpha)$ is the Dirichlet distribution for parameter α.
2. For each of the words $w_{i,j}$ in document $\mathbf{w}_i$, where $j \in \{1, \ldots, N_i\}$,
   (a) Draw a topic assignment $\mathbf{z}_{i,j} \sim \text{Multinomial}(\theta_i)$.
   (b) Draw a word instance $w_{i,j} \sim \text{Multinomial}(\beta_{z_{i,j}})$.

It is worth noting that there is a little abuse of notations on the topic assignment $\mathbf{z}_{i,j}$ which is a K-dimensional indicator vector (only one element is 1; all others are 0), and $\beta_{\mathbf{z}_{i,j}}$ denotes the topic that is selected by the non-zero element of $\mathbf{z}_{i,j}$. In the latter supervised extensions of LDA, usually the average of $\bar{Z} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{z}_n$ is evaluated. This average is actually a mean vector.

The LDA model is represented as a directed graphical model in Figure 2.9. To use LDA model, there are two major problems need to be solved:

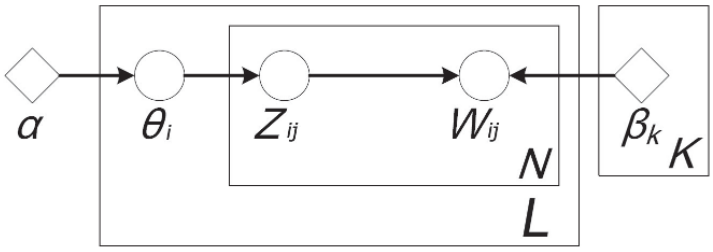


Figure 2.9: Probabilistic graphical model for LDA. The boxes are "plates" representing replicates. The outer plate indicates L documents, while the inner plate represents the repeated selection of topics and words within a document.

1. Inference problem of computing the posterior distribution of the hidden variables θ and $\mathbf{z}$ (both are the low-dimensional representation of the document) given a document $\mathbf{w}$:

$$p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta)}{p(\mathbf{w}|\alpha, \beta)}. \tag{2.21}$$

2. Parameter estimation of α and β given a corpus of documents $D = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_L\}$:

$$(\alpha^*, \beta^*) = \arg\max_{(\alpha,\beta)} \ell(\alpha, \beta) = \arg\max_{\alpha,\beta} \sum_{i=1}^{L} \log p(\mathbf{w}_i|\alpha, \beta). \tag{2.22}$$

Unfortunately, neither inference problem (2.21) nor parameter estimation (2.22) is tractable due to the intractability of the likelihood $p(\mathbf{w}|\alpha, \beta)$[1]. To address the issue, approximate inference algorithms based on variational [81] or Markov Chain Monte Carlo (MCMC) [86] approaches have been widely used for parameter estimation and posterior inference for LDA. Here, we briefly describe the variational inference as an example: let $q(\theta, \mathbf{z}|\gamma, \phi)$ be a proposed variational distribution that approximates the true model posterior $p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta)$ (2.21), where $\{\gamma, \phi\}$ are introduced parameters related to $\{\mathbf{w}, \alpha, \beta\}$. $q(\theta, \mathbf{z}|\gamma, \phi)$ is designed to be a tractable family of lower bounds of (2.21), and the tightest possible

[1]The explanation of intractability of $p(\mathbf{w}|\alpha, \beta)$ is beyond the scope of this thesis, interested readers can refer to the LDA paper [81].

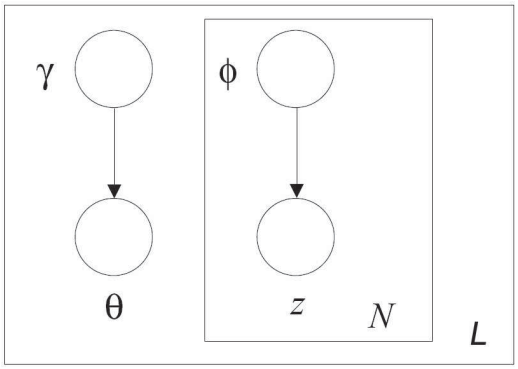


Figure 2.10: Graphical model representation of the variational distribution q.

lower bound can be obtained by:

$$(\gamma^*, \phi^*) = \arg\min_{(\gamma,\phi)} KL\left(q\left(\theta, \mathbf{z}|\gamma, \phi\right) \;||\; p\left(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta\right)\right), \tag{2.23}$$

where $KL\left(\right)$ is the Kullback-Leibler (KL) divergence between the variational distribution and the true posterior. Therefore, $q\left(\theta, \mathbf{z}|\gamma^*, \phi^*\right)$ can be viewed as an approximation to the posterior distribution $p\left(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta\right)$. By making some independence assumption (e.g., mean field) about $q\left(\theta, \mathbf{z}|\gamma, \phi\right)$ as shown in (2.24) whose graphical model is indicated in Figure 2.10, the problem (2.23) can be efficiently optimised via an iterative fixed-point method [81].

$$q\left(\theta, \mathbf{z}|\gamma, \phi\right) = q\left(\theta|\gamma\right) \prod_{i=1}^{N} q\left(z_i|\phi_i\right). \tag{2.24}$$

We would like to note that aside from textual documents, LDA can also be applied to image data. In that case, an image is able to be treated as a document by breaking down an image into small pieces. Thus, the whole image is a document and each piece of such image is a word.

2.2.5.2 Supervised latent Dirichlet allocation (sLDA)

In the unsupervised topic model (i.e., LDA described in Section 2.2.5.1), the side information (e.g., class label or rating of a document) has not been utilised

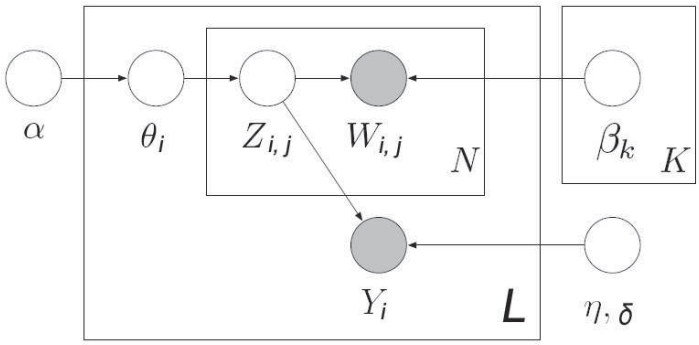


Figure 2.11: Supervised topic models (sLDA).

for learning topics and inferring topic vectors θ. To use side information appropriately for discovering more predictive representations, supervised topic models (sLDA) [82] introduce a response variable Y into LDA for each document, and the graphical model for sLDA is shown in Figure 2.11. Under the sLDA model, each document $\mathbf{w}_i$ and response y_i arises from the following generative process:

1. Draw a topic mixing proportion vector θ_i according to a K-dimensional Dirichlet prior: $\theta_i \sim \text{Dir}(\alpha)$.

2. For each of the words $w_{i,j}$ in document $\mathbf{w}_i$, where $j \in \{1, \ldots, N_i\}$,

   (a) Draw a topic assignment $\mathbf{z}_{i,j}|\theta_i \sim \text{Multinomial}(\theta_i)$.

   (b) Draw a word instance $w_{i,j}|\mathbf{z}_{i,j}, \beta \sim \text{Multinomial}(\beta_{\mathbf{z}_{i,j}})$.

3. Draw response variable $y_i|\mathbf{z}_{i,1}, \ldots, \mathbf{z}_{i,N_i}, \eta, \delta \sim \mathbf{GLM}(\bar{\mathbf{z}}_i, \eta, \delta)$, where $\bar{\mathbf{z}}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} \mathbf{z}_{i,j}$ corresponds to document $\mathbf{w}_i$.

The distribution of the response is a generalised linear model (GLM) [87],

$$p\left(y|\mathbf{z}_1, \ldots, \mathbf{z}_N, \eta, \delta\right) = h\left(y, \delta\right) \exp\left\{\frac{\left(\eta^T \bar{\mathbf{z}}\right) y - A\left(\eta^T \bar{\mathbf{z}}\right)}{\delta}\right\}. \tag{2.25}$$

For each fixed δ, equation (2.25) is an exponential family, with base measure $h\left(y, \delta\right)$, sufficient statistic y, and log-normaliser $A\left(\eta^T \bar{\mathbf{z}}\right)$. For example, the normal distribution corresponds to $h\left(y, \delta\right) = \left(1/\sqrt{2\pi\delta}\right) \exp\left\{-y^2/2\right\}$ and $A\left(\eta^T \bar{\mathbf{z}}\right) =$

$\left(\eta^T \bar{\mathbf{z}}\right)^2/2$. In this case, the usual Gaussian parameters, mean μ and variance σ^2, are equal to $\eta^T \bar{\mathbf{z}}$ and δ, respectively.

Similar to LDA, sLDA also utilises the variational approximation (by proposing a tractable variational distribution q as (2.24)) to infer the posterior, i.e., $p(\theta, \mathbf{z}|\mathbf{w}, y, \alpha, \beta, \eta, \delta)$, and estimate parameters by maximizing the joint likelihood $p(\mathbf{y}, \mathbf{W}|\alpha, \beta, \eta, \delta)$ where $\mathbf{y}$ is the vector of response variables in a corpus and $\mathbf{W}$ are all the words.[1] A more important purpose of applying sLDA is to predict a document response. Given a new document $\mathbf{w}$ and a fitted model $\{\alpha, \beta, \eta, \delta\}$, prediction computes the expected response value,

$$\mathbb{E}[Y|\mathbf{w}, \alpha, \beta, \eta, \delta] = \mathbb{E}\left[\mu\left(\eta^T \bar{Z}\right)|\mathbf{w}, \alpha, \beta, \eta, \delta\right], \tag{2.26}$$

where $\mu(\cdot) = \mathbb{E}_{GLM}[Y|\cdot]$, $\bar{Z} = \frac{1}{N}\sum_{n=1}^{N} Z_n$ ($\bar{\mathbf{z}}$ is an instance of $\bar{Z}$) and $\mathbb{E}\left[\bar{Z}\right] = \bar{\phi} = \frac{1}{N}\sum_{n=1}^{N}\phi_n$. A specific case is that $\mathbf{GLM}\left(\bar{Z}, \eta, \delta\right)$ is a normal distribution parametrised by $\mu = \eta^T \bar{Z}$ and $\delta = \sigma^2$, i.e., $\mathbf{GLM}\left(\bar{Z}, \eta, \delta\right) = N\left(\eta^T \bar{Z}, \sigma^2\right)$, and $Y \in \mathbb{R}$, then (2.26) becomes:

$$\mathbb{E}\left[Y|\mathbf{w}, \alpha, \beta, \eta, \sigma^2\right] = \eta^T \mathbb{E}\left[\bar{Z}|\mathbf{w}, \alpha, \beta, \eta, \sigma^2\right], \tag{2.27}$$

By approximating the posterior mean of $\bar{Z}$ using variational inference, (2.26) can be estimated with

$$\mathbb{E}[Y|\mathbf{w}, \alpha, \beta, \eta, \delta] \approx \mathbb{E}_q\left[\mu\left(\eta^T \bar{Z}\right)\right]. \tag{2.28}$$

2.2.5.3 Maximum entropy discrimination latent Dirichlet allocation (MedLDA)

Most supervised topic models (e.g., sLDA) are built on a likelihood-driven probabilistic inference paradigm. To explore the max-margin based techniques (sometimes arguably more powerful) that are widely used in learning discriminative models to learn supervised topic models, Zhu et al. proposed the Maximum entropy discrimination latent Dirichlet allocation (MedLDA) [85] [88], which integrates maximum likelihood estimation and maximum margin estimation under

[1] The variational methods for inference and estimation of sLDA are not the focus of this thesis, please refer to [82] for details

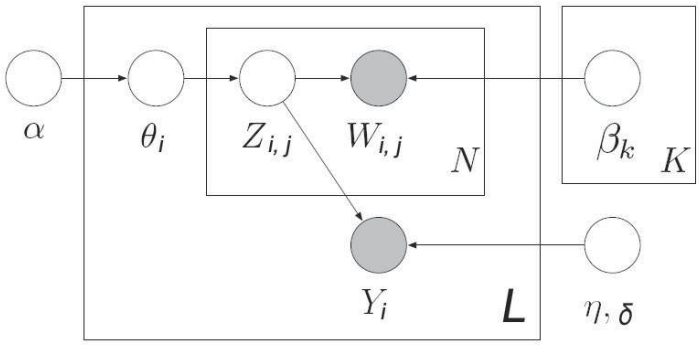


Figure 2.12: Graphical model of MedLDA.

a unified constrained optimisation framework.

Although the topic models that handle side information, e.g., sLDA, DiscLDA and MedLDA, share the same goal which is to uncover the low-dimensional topic representations for documents while retaining predictive power for regression and classification, they differ in the training procedures. sLDA learns the model parameters by maximizing the joint likelihood of data and response variables. DiscLDA is trained by maximizing the conditional likelihood of response variables[1]. MedLDA is a combination of sLDA and support vector regression (SVR). In it, the model parameters are learned in a max-margin manner; and the discovery of latent topics is coupled with the max-margin estimation of the model parameters. This interplay yields latent topic representations that are more suitable for supervised tasks such as regression and classification.

By adopting the same notations as LDA and sLDA, the graphical model of MedLDA is illustrated in Figure 2.12. Observant readers may find that the graphical model of MedLDA is the same[2] as the sLDA as shown in Figure 2.11. However, the learning and prediction process of MedLDA are quite different. Following [85] and [88], we describe MedLDA for regression and classification separately.

[1]The content of DiscLDA is beyond the scope of this thesis, interested readers can refer to [83] for details.

[2]Rigorously, there is a slightly difference of the graphical models between sLDA and MedLDA: sLDA learns a point estimate of the parameter η, whereas MedLDA learns a distribution $q(\eta)$ instead.

MedLDA for regression: For regression, the document response variables y are continuous, i.e., $y \in \mathbb{R}$. Instead of learning a point estimate of regression coefficient η as in sLDA, regressional MedLDA learns a distribution $q(\eta)$ in a max-margin manner, and the prediction rule for regression of MedLDA is:

$$\hat{y} \triangleq \mathbb{E}\left[Y|\mathbf{w}, \alpha, \beta, \sigma^2\right] = \mathbb{E}_{q(\eta)}\left[\eta^T \bar{Z}|\mathbf{w}, \alpha, \beta, \sigma^2\right], \tag{2.29}$$

where $\bar{Z} = \frac{1}{N}\sum_{n=1}^{N} Z_n$ ($\bar{\mathbf{z}}$ is an instance of $\bar{Z}$).

Let $q(\theta, \mathbf{z}, \eta|\gamma, \phi)$ is a variational distribution to approximate the posterior $p(\theta, \mathbf{z}, \eta|\mathbf{W}, \mathbf{y}, \alpha, \beta, \sigma^2)$, and the response variable y is drawn from a linear Gaussian: $y|\mathbf{z}_{1:N}, \eta, \sigma^2 \sim N\left(\eta^T \bar{\mathbf{z}}, \sigma^2\right)$ as shown in the generative process of sLDA model, the learning problem is defined as:

$$\begin{aligned} \text{P1(MedLDA}^r\text{):} \quad & \min_{q,\alpha,\beta,\sigma^2,\xi,\xi^*} \mathcal{L}(q) + C\sum_{d=1}^{L}(\xi_d + \xi_d^*) \\ s.t. \quad & \forall d: \begin{cases} y_d - \mathbb{E}\left[\eta^T \bar{Z}_d\right] \leq \epsilon + \xi_d \\ -y_d + \mathbb{E}\left[\eta^T \bar{Z}_d\right] \leq \epsilon + \xi_d^* \\ \xi_d, \xi_d^* \geq 0, \end{cases} \end{aligned} \tag{2.30}$$

where

$$\mathcal{L}(q) = -\mathbb{E}\left[\log p\left(\theta, \mathbf{z}, \eta, \mathbf{W}, \mathbf{y}|\alpha, \beta, \sigma^2\right)\right] - \mathcal{H}\left(q\left(\theta, \mathbf{z}, \eta\right)\right) \tag{2.31}$$

is an upper bound of negative log (marginal) likelihood: $-\log p(\mathbf{W}, \mathbf{y}|\alpha, \beta, \eta, \sigma^2)$. As the marginal likelihood is intractable (as in sLDA [82]), MedLDA optimises an upper bound $\mathcal{L}(q)$ instead; $\mathcal{H}(q) \triangleq -\mathbb{E}_q[\log q]$ is the entropy of q; ξ, ξ^* are slack variables absorbing errors in training data; and ϵ is a precision parameter as in support vector regression (SVR).

P1(MedLDA^r) is solved by using a variational EM-algorithm [1].

MedLDA for classification: For classification, the document response variables y are discrete, i.e., $y \in \{1, 2, \ldots, M\}$). The classification model of MedLDA also is to learn a distribution $q(\eta)$. Then, the prediction rule for multi-class

[1]We skip the description of variational methods that solve the optimisation problem P1(MedLDA^r) as they are not the focus of this thesis. Please refer to [85] or [88] for the optimizing procedures.

classification is

$$\hat{y} = \arg\max_{y} \mathbb{E}\left[\eta^T \mathbf{f}\left(y, \bar{Z}\right) | \alpha, \beta, \mathbf{w}\right], \tag{2.32}$$

where η is an MK-dimensional parameter vector whose components from $(y-1)K+1$ to yK are associated with the class y, and $\mathbf{f}\left(y, \bar{Z}\right)$ is a feature vector whose components from $(y-1)K+1$ to yK are those of the vector $\bar{Z}$ and all the others are 0.

Similar to the regression model, the integrated latent topic discovery and multi-class classification model is defined as follows:

$$\begin{aligned} \text{P2(MedLDA}^c\text{):} \quad & \min_{q, q(\eta), \alpha, \beta, \xi} \mathcal{L}(q) + KL\left(q(\eta) || p_0(\eta)\right) + C \sum_{d=1}^{L} \xi_d \\ & s.t. \;\; \forall d, y \neq y_d : \begin{cases} \mathbb{E}\left[\eta^T \Delta \mathbf{f}_d(y)\right] \geq 1 - \xi_d \\ \xi_d \geq 0, \end{cases} \end{aligned} \tag{2.33}$$

where $q(\theta, \mathbf{z}|\gamma, \phi)$ is a variational distribution; $\mathcal{L}(q) = -\mathbb{E}\left[\log p(\theta, \mathbf{z}, \mathbf{W}, |\alpha, \beta)\right] - \mathcal{H}\left(q(\theta, \mathbf{z})\right)$ is a variational upper bound of $-\log p(\mathbf{W}|\alpha, \beta)$; $KL()$ is the Kullback-Leibler (KL) divergence; $p_0(\eta)$ is a prior from which the parameter η is sampled; $\Delta \mathbf{f}_d(y) = \mathbf{f}\left(y_d, \bar{Z}_d\right) - \mathbf{f}\left(y, \bar{Z}_d\right)$, and ξ are slack variables. $\mathbb{E}\left[\eta^T \Delta \mathbf{f}_d(y)\right]$ is the "expected margin" by which the true label y_d is favored over a prediction y.

Finally, a variational EM-algorithm is used to solve the optimisation of P2(MedLDA^c)[1].

2.3 Summary

In this chapter, we have briefly introduced some existing algorithms in feature extraction, machine learning and pattern recognition that closely relate to the following chapters. In the rest of thesis, the algorithms described above will be utilised directly without repeated introductions. For example, in Chapters 3 and 4, Haar-like features and Adaboost approach are used for detecting facial areas. Then, eigenfaces and SVM are used for facial expression recognition. In Chapters 5 and 6, SIFT, SURF etc. will be our feature presentations and KDE

[1]We also skip the description of variational methods that solve the optimisation problem P2(MedLDA^c) as they are not the focus of this thesis. Please refer to [85] or [88] for the optimizing procedures.

and MedLDA etc. will be used as classifiers for object recognition. Although some of these approaches, e.g., MedLDA, are only employed and not improved in the later work, we also put their introductions in this chapter for readers' references.

Chapter 3

Histogram Variances Faces for expression recognition

In human's expression recognition, the representation of expression features is essential for the recognition accuracy. In this work we propose a novel approach for extracting expression dynamic features from facial expression videos. Rather than utilising temporal statistical models, e.g., Hidden Markov Model (HMM), our approach integrates expression dynamic features into a static image, the *Histogram Variances Face (HVF)*, by fusing histogram variances among the frames in a video. The HVFs can be automatically obtained from videos with different frame rates and immune to illumination interference. In our experiments, for the videos picturing the same facial expression, e.g., surprise, happy and sadness etc., their corresponding HVFs are similar, even though the performers and frame rates are different. Therefore the static facial recognition approaches can be utilised for the dynamic expression recognition. We have applied this approach on the well-known Cohn-Kanade AU-Coded Facial Expression database then classified HVFs using PCA and Support Vector Machine (SVMs), and found the accuracy of HVFs classification is very encouraging.

3.1 Histogram Variances Faces

Our goal is to obtain distinct features depicting a certain expression from a face video. Under an assumption that a video (i.e., a sequence of face images) has been properly segmented [1] and aligned [2], an expression is usually featured by the motions at specific facial positions. For instance, "happy" is more likely to have a distinct smile motion around the mouth and "sad" is mostly represented by a frown occurring at the upper portion of a face. Therefore, compared with "sad", "happy" will have greater variance around the mouth and less variance around forehead in the temporal direction. By breaking down the face area with a grid and recording variance of each cell in temporal direction, the dynamic features of an expression can be stored into a single static image, which is called a Histogram Variances Face (HVF) in this work.

In general, the procedures of generating a HVF from a video can be summarised as follows, and the related techniques will be described in later subsections:

1. Automatically align faces in temporal direction by detecting fiducial points (the eyes) per frame.
2. Preprocess and texturise face images.
3. Break down each texturised image into $M \times N$ blocks and compute the histogram variance for each block in temporal direction.
4. Create a new $M \times N$ 8-bit grayscale image, i.e. a *Histogram Variances Face (HVF)*. Each pixel value corresponds to a block histogram variance.

Figure 3.1 illustrates how to construct a *Histogram Variances Face (HVF)*.

3.1.1 Faces Alignment

For different expression videos, normally the scales and locations of human faces in frames are various. To make all the HVFs have the same scale and location,

[1] Only the duration of facial expression, i.e., from neutral to apex, has been included. The process of video segmentation is beyond the scope of this thesis.

[2] Facial areas have been cropped out and aligned in temporal direction.

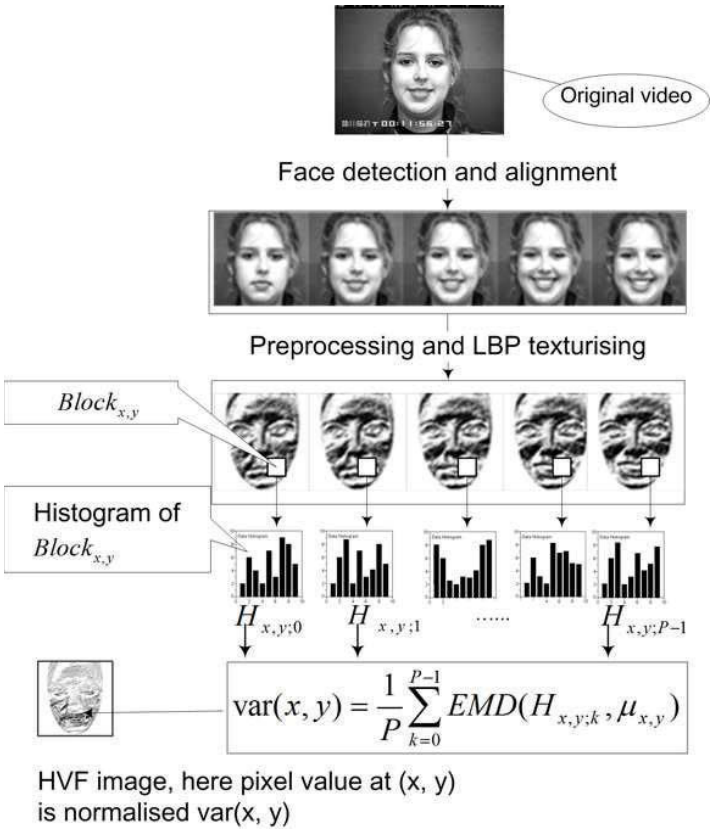


Figure 3.1: Procedures of generating an HVF image, where $H_{x,y;k}$ is the histogram of $Block_{x,y}$ in the *k-th* texture image, P is the number of images, $\mu_{x,y}$ is the mean histogram of $H_{x,y;k}$, $EMD(*)$ is the Earth Mover's Distance

.

it is necessary to detect the face fiducial points and cut the faces out in terms of fiducial points. Meanwhile, bilinear interpolation is used to make sure all the face images have the same size. To detect the fiducial points, we make use of a real-time face detection scheme based on Haar-Like feature classifier cascade and AdaBoost learning [89], called Viola-Jones face detector. It consists of a cascade of classifiers trained by the AdaBoost algorithm. Each classifier uses integral image filters, bases on Haar Basis functions and can be computed very fast at any location and scale. For each stage in the cascade, a subset of features is chosen using a feature selection procedure based on the AdaBoost. This face detection scheme detects and locates the positions of eyes on the Cohn-Kanade expression database [35] precisely and fast. In our system, the eyes are the fiducial points used to cut and align the faces in the frontal face image sequences. The positions of human's eyes determine the face position accurately. The faces in videos are cut out according to eyes' positions and are normalised to a fixed size in proportion to the distance between eyes.

3.1.2 Preprocessing and LBP texturising

After getting aligned faces using Viola-Jones face detector [89], we mask the areas outside an ellipse around each face and leave only the face area as the region of interest (ROI). Histogram equalisation in ROI is also applied to reinforce the gradient. Furthermore, the illumination variety in a video is another issue that could interfere with the histogram variance in the temporal direction. To overcome this, we employ the LBP operator shown in [22][40] to extract the texture of the masked faces, and hence eliminate the illumination interference.

Local Binary Pattern (LBP) describes the surroundings of a pixel by generating a bit-code from the binary derivatives of a pixel. The operator is usually applied to grayscale images and the derivative of the intensities. A typical form of the LBP operator takes the 3×3 surrounding of a pixel and generates a binary 1 if the neighbor of the centre pixel has larger value than the centre pixel. The operator generates a binary 0 if the neighbor is less than the centre. The eight neighbours of the centre can then be represented with an 8-bit unsigned

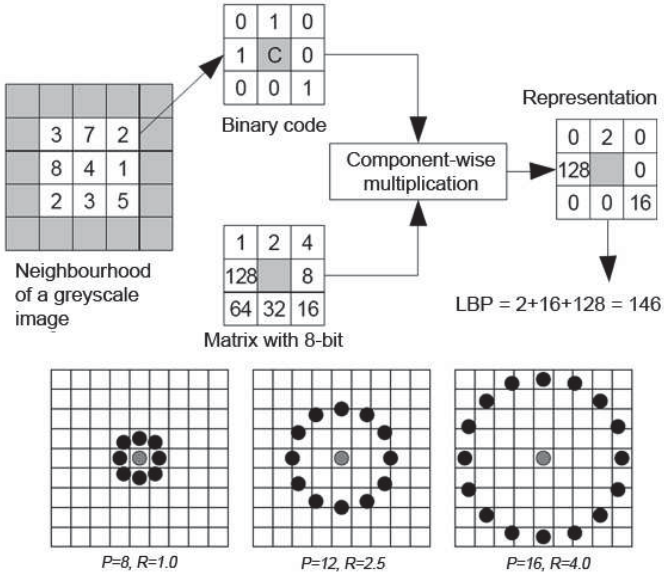


Figure 3.2: An example of computing LBP in a 3×3 neighborhood

integer. The LBP value is calculated using Equation 3.1 [22][40]:

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{3.1}$$

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0, \\ 0 & \text{if } x < 0 \end{cases}$$

where P is the number of neighbors, R is the radius and g_c corresponds to the gray value of the center pixel of a local neighborhood. $g_p(p = 0, \ldots, P-1)$ correspond to the gray values of P equally spaced pixels on a circle of radius R that form a circularly symmetric set of neighbors. Figure 3.2 shows the example of an LBP operator.

3.1.3 Earth Mover's Distance for calculation of histogram variances

There are a number of approaches to compute the similarity between two histograms. Normally these approaches are divided into two categories: bin-to-bin and cross-bin. In our case, although a block may not have texture change during a video, its corresponding histograms in different frames are unlikely to keep the same because of the noise. It is quite often that the block histograms shift slightly according to Gaussian distributions. The bin-to-bin approaches will not work well here because they are sensitive to the slight histogram shifting. Note that the histogram shifts caused by noise are invisible for human vision, so we should ignore these kinds of shifts. Earth Mover's Distance (EMD) is a cross-bin approach and able to address the shift problem caused by noise because slight histogram shifts do not affect the EMD much. EMD is consistent with the human's vision because if two images look more different according to human's vision, generally the histograms of the two images will create a greater EMD value. Another good cross-bin choice can be the *Quadratic Form Distance*. However, it needs a positively definite parameter matrix which must be pre-defined. Our experiments prove that EMD has the best performance in this application.

3.1.3.1 Earth Mover's Distance

Earth Mover's Distance (EMD) is a method to evaluate dissimilarity between two multi-dimensional distributions in some feature space where a distance measure between single features (called the ground distance) is given. It has the excellent capability of matching human's vision on histogram distribution differences. Basically, EMD was formalised as the following linear programming problem. Let

$$P = \{(p_1, w_{p1}), \ldots, (p_m, w_{pm})\} \tag{3.2}$$

be the first signature with m clusters, where p_i is the cluster representative and w_{pi} is the weight of the cluster; and let

$$Q = \{(q_1, w_{q1}), \ldots, (q_n, w_{qn})\} \tag{3.3}$$

the second signature with n clusters; and $D = [d_{ij}]$ is the ground distance matrix where d_{ij} is the ground distance between clusters p_i and q_j. EMD is to find a flow $F = [f_{ij}]$, where f_{ij} is the flow between p_i and q_j, that minimises the overall cost [41]

$$WORK(P, Q, F) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij} \tag{3.4}$$

subject to the following constraints [41]:

1. $f_{ij} \geq 0$; $i \in [1, m]\,, j \in [1, n]$,
2. $\sum_{j=1}^{n} f_{ij} \leq w_{pi}$; $i \in [1, m]$,
3. $\sum_{i=1}^{m} f_{ij} \leq w_{qj}$; $j \in [1, n]$ and
4. $\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = min\left(\sum_{i=1}^{m} w_{pi}, \sum_{j=1}^{n} w_{qj}\right)$

Constraint *1* allows moving "supplies" from P to Q and not vice versa. Constraint *2* limits the amount of supplies that can be sent by the clusters in P to their weights. Constraint *3* limits the clusters in Q to receive no more supplies than their weights; and constraint *4* forces to move the maximum amount of supplies possible. This amount is called the total flow. Once the transportation problem is solved, and the optimal flow F has been found, the EMD is defined as the resulting work normalised by the total flow [41]:

$$EMD(P, Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} f_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}}. \tag{3.5}$$

The normalisation factor is the total weight of the smaller signature because of constraint *4*. This factor is needed when the two signatures have different total weights, in order to avoid favoring smaller signatures. In general, the ground distance d_{ij} can be any distance and will be chosen according to the problem in question.

We employ EMD to measure the distance between two histograms when calculating histogram variances in the temporal direction. In our case, p_i and q_j are the grayscale pixel values, which are in $[0, 255]$. w_{pi} and w_{qj} are the pixel

distributions at p_i and q_j respectively. The ground distance d_{ij} that we choose is the square of euclidean distance between p_i and q_j, i.e., $d_{ij} = (p_i - q_j)^2$.

3.1.3.2 Procedures of calculating histogram variances

1. Suppose a sequence consists of P face texture images. We firstly break down each image evenly into $M \times N$ blocks, denoted by $B_{x,y;k}$, where x is row index, y is column index and k is corresponding to the *k-th* frame in the sequence. Then, calculate every gray-value histogram of $B_{x,y;k}$, denoted by $H_{x,y;k}$, where $x = 0, 1, \ldots, M-1; y = 0, 1, \ldots, N-1; k = 0, 1, \ldots, P-1$.

2. Calculate the histogram variance $var(x, y)$:

$$var(x, y) = \frac{1}{P} \sum_{k=0}^{P-1} EMD(H_{x,y;k}, \mu_{x,y}), \tag{3.6}$$

   where $\mu_{x,y}$ is the mean histogram

$$\mu_{x,y} = \frac{1}{P} \sum_{k=0}^{P-1} H_{x,y;k}, \tag{3.7}$$

   and $EMD(H_{x,y;k}, \mu_{x,y})$ is the Earth Mover's Distance between $H_{x,y;k}$ and $\mu_{x,y}$.

3. Construct an $M \times N$ 8-bit grayscale image as our HVF. Suppose that $hvf(x, y)$ denotes the pixel value at coordinate (x, y) in an HVF image:

$$hvf(x, y) = 255 - \left\lfloor \frac{255 * var(x, y)}{MAX(var(x, y))} \right\rfloor, \tag{3.8}$$

   To make the HVF image more "clean", we set $hvf(x, y) = 255$ when $hvf(x, y) > threshold$, where the *threshold* is predefined, e.g., 200.
   Figure 3.3 shows some HVF examples extracted from happiness, surprise and sadness videos respectively.

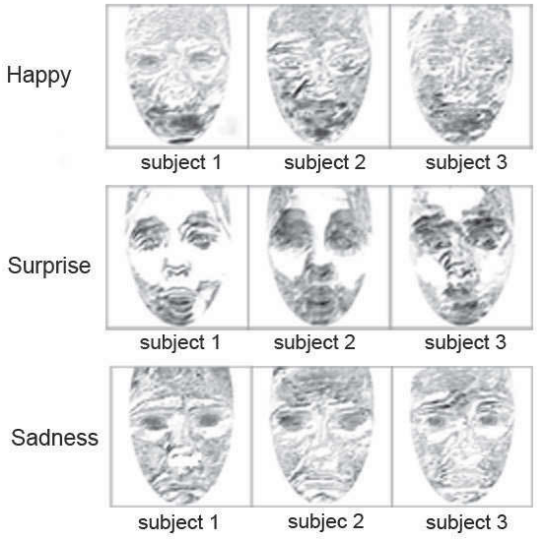


Figure 3.3: Examples of HVF images

3.1.3.3 Computing histograms of various-size blocks

Whether the different block sizes affect our HVFs recognition is one of the questions we are going to answer in this chapter. Hence, we must get the histograms for various-size blocks. To make the histogram computation more efficient, we get the bigger-size histograms by adding small-size ones.

Suppose that $H(\alpha)$ denotes the histogram vector with respect to image area α. Then, we have

$$H(\alpha) + H(\beta) = H(\alpha \cup \beta). \tag{3.9}$$

Therefore, if $H_{x,y;k}(\gamma, \eta)$ denoted the histograms of $\gamma \times \eta$ pixels block at the *x-th* row and *y-th* column in frame k, then

$$H_{x,y;k}(a\gamma, b\eta) = \sum_{i=0}^{a-1} \sum_{j=0}^{b-1} H_{ax+i,by+j;k}(\gamma, \eta), \tag{3.10}$$

where $a, b, \gamma, \eta \in N^+$. We only obtain all histograms with size 3×3 in our experiments. After then, the size 6×6 and 12×12 histograms can be computed fast and easily through Equation 3.10.

3.2 Classifying HVF images using PCA+SVMs

HVF records the dynamic features of the expression. As we can see in Figure 3.3, for the expressions of happiness, surprise and sadness, the homogeneous HVFs look similar and HVFs belonging to different expressions have their own unique features. To verify the performance of HVF image's features, we just utilise the typical facial recognition technologies PCA+SVMs, which have proven to be very well suitable for classification tasks such as facial recognition.

3.2.1 PCA dimensionality reduction

In experiments, all pixel values of an HVF image construct an $n \times 1$ column vector $z_i \in R^n$, and an $n \times l$ matrix $Z = \{z_1, z_2, \ldots, z_l\}$ denotes the training set which consists of l sample HVF images. The PCA algorithm finds a linear transformation orthonormal matrix $W_{n\times r}(n >> r)$, projecting the original high n-dimensional feature space into an r-dimensional feature subspace, where $n >> r$. x_i denotes the new feature vector:

$$x_i = W^T \cdot z_i \quad (i = 1, 2, \ldots, l). \tag{3.11}$$

The columns of matrix W are called eigenfaces [90], which are the r eigenvectors corresponding to the r largest eigenvalues of the scatter matrix S:

$$S = \sum_{i=1}^{l} (z_i - \mu)(z_i - \mu)^T \tag{3.12}$$

where μ is the mean image of all HVF samples and $\mu = \frac{1}{l}\sum_{i=1}^{1} z_i$.

3.2.2 SVMs training and recognition

SVMs [91][92][93] is an effective supervised classification algorithm and its essence is to find a hyperplane that separates the postive and negative feature points with maximum margin in the feature space. Very likely that the real-world problems are not linearly separable. In this case, SVMs map the original input space using

'kernel' functions into a higher dimenional space where the feature points are linearly separable.

Suppose that α denotes the Lagrange parameters that describe the separating hyperplane ω in SVM. Finding the hyperplane that maximises the margin between positive and negative data set involves getting the nonzero solutions α_i of a Lagrangian dual problem, which is a quadratic programming problem and is solvable. Once we find all α_i and given a labeled training set $\langle x, y \rangle$, the decision function can be as follows:

$$f(x) = sgn\left(\sum_{i=1}^{l} \alpha_i y_i K(x, x_i) + b\right), \tag{3.13}$$

where b is the bias of the hyperplane, l is the number of training samples, y_i is the label of train data, x_i is the vector of PCA projection coefficients of HVFs, and $K(x, x_i)$ is the 'kernel mapping'. For linear SVMs,

$$K(x, x_i) = \langle x, x_i \rangle, \tag{3.14}$$

where $\langle x, x_i \rangle$ means the dot product of x and x_i.

Since the SVM is basically a two-class classification algorithm, here we adopt the pairwise classification (one-versus-one) for multi-class. In pairwise classification, there is a two-class SVM for each pair of classes to separate members of one class from members of the other. Specifically, there are maximum $C_6^2 = 15$ two-class SVM classifiers trained for the classification of six sorts of expressions. When recognising a new HVF image, all the 15 two-class classifiers are applied for testing HVF and the winner class is the one that takes the most votes.

3.3 Experiments

3.3.1 Dataset

Our experiments adopt the Cohn-Kanade AU-Coded Facial Expression Database [35]. This database consists of 97 university students ranging in age from 18 to 30 years. 65% are female, 15% are African-American, and 3% are Asian or Latino.

Figure 3.4: Some example sequences in the Cohn-Kanade database for facial expression recognition. Each of the sequences depicting a certain facial expression starts from a neutral face and ends with the expression apex. The frame numbers of sequences are various.

Videos in this database have been recoded using a camera located directly in front of the subject. Subjects have been instructed by an experimenter to perform a series of 23 facial expressions. Subjects begin and end each display with a neutral face. Before performing each display, an experimenter described and modeled the desired display. Image sequences from neutral to expression apex have been digitised into 640 by 480 pixel arrays with 8-bit precision for grayscale values. Figure 3.4 shows some example sequences from the Cohn-Kanade AU-Coded Facial Expression Database.

Although Cohn-Kanade is a benchmark dataset for expression recognition, it only contains the AU-Coded combinations for its objects instead of expression definitions (i.e. surprise, happy, anger etc.). To assess classification accuracy in a supervised way, we need to create "ground truth" by manually labeling each object with a specific expression according to FACS. However, facial expression

labeling is extremely subjective and we found that many of the objects were difficult to be surely labeled. This labeling process prevents us from using all the objects in Cohn-Kanade dataset. Instead, We select 31 subjects that can be labeled with high confidence. On the other hand, because the manual labeling process is subjective, that is why it is not feasible to compare our work with other people's work on Cohn-Kanade dataset.

Each subject has up to 6 expressions (image sequences). The total number of sequences is 169, so 169 HVFs are generated. The image sequences belonging to the same expression have the similar duration but their frame rates are different. For a certain expression, we feed around 80% HVFs to PCA+SVMs training process and the classifiers are later applied to all HVFs.

3.3.2 Parameter selection for HVFs generation

The faces of selected subjects are detected and cut out. Then, these faces are resized to 300×300 pixels and aligned. To eliminate illumination interference, we use a 3×3 neighborhood with radius 1 for the LBP operator. As to the ground distance for EMD, we adopt the square of *Euclidean* distance between two pixel values. The reason for choosing *Euclidean* distance here is because, for human's vision, more difference of pixel value distribution between two image histograms will cause more distinction of the two images. The final data dimensions are reduced to 95 after PCA operation. For classification, we tried Gaussian kernel SVM with various gamma and linear SVMs. We found that linear SVM had better overall performance. Actually, performance of SVM is data-dependent. It seems that linear SVM fits our application better. Our penalty parameter C for the linear SVMs is 8.

Moreover, to check the influence of different block segmentations, we chose the block's sizes as 3×3, 6×6 and 12×12 pixels. So each texture image is broken down into 100×100, 50×50 and 25×25 blocks respectively. Because the histogram variance among blocks in the temporal direction becomes a pixel value in HVF, our HVF sizes are 100×100, 50×50 and 25×25 pixels as well.

	100×100 blocks		50×50 blocks		25×25 blocks	
	Recognition	FPR	Recognition	FPR	Recognition	FPR
HA	96.6%	3.3%	100%	3.3%	100%	3.3%
SU	96.7%	3.4%	96.7%	0.0%	96.7%	0.0%

Table 3.1: Recognition rates of happy and surprise HVFs.

	HA	SU	AN	DI	FE	SA
Recog(%)	97.8	79.3	55.9	60.2	36.8	46.9

Table 3.2: A recent investigation of facial expression recognition by human in [94].

3.3.3 Training and recognition

When labeling the face videos, we were quite confident to recognise original image sequences of happiness and surprise. Therefore, the training data for these two classes can be labeled with high correction. This implies that these two expressions have evidently unique features. Our experimental results (Table 3.1) testifies this point with high HVF recognition rate, where *FPR* is the *false positive rate*.

When we were labeling HVFs of *anger*, *disgust*, *fear* and *sadness*, nearly half of them were very challenging to be attached the convincing classifications, according to neither AU-Coded combinations nor human's perception on original image sequences, especially for *anger* and *sadness*. From the AU-Code of FACS perspective, AU-Coded prototypes in FACS (2002 version) [35] are overlapping for these expressions. From human's vision perspective, one expression of a person may be reflected by several different sequences of images and one sequence of images is also often interpreted as various expressions. An investigation [94] about facial expression recognition by human discloses that compared to the expressions of *happy* and *surprise*, the expressions of *anger*, *fear*, *disgust* and *sadness* are much more difficult to be recognised by people (see Table 3.2).

After trying our best to manually label the expressions under above circumstances, we conduct the following experiments:

1. Feed *happy*, *surprise* and *anger* HVFs into the SVMs. For these three sorts

	100 × 100 blocks		50 × 50 blocks		25 × 25 blocks	
	Recognition	FPR	Recognition	FPR	Recognition	FPR
HA	96.6%	0.0%	100%	0.0%	100%	0.0%
SU	86.7%	3.3%	90.0%	1.7%	90.0%	1.7%
AN	96.8%	6.8%	96.8%	5.1%	96.8%	5.1%
HA	89.7%	0.0%	96.6%	0.0%	96.6%	0.0%
SU	83.3%	7.0%	90.0%	5.3%	90.0%	5.3%
DI	85.7%	13.5%	89.3%	6.8%	89.3%	6.8%
HA	93.1%	1.9%	96.5%	0.0%	96.5%	0.0%
SU	90.0%	7.7%	93.3%	3.8%	90.0%	3.8%
FE	86.9%	10.1%	91.3%	5.1%	86.9%	8.5%
HA	93.1%	1.7%	96.5%	0.0%	96.5%	0.0%
SU	90.0%	3.5%	93.3%	3.5%	93.3%	3.5%
SA	89.2%	8.4%	92.8%	5.1%	89.2%	6.7%

Table 3.3: Recognition rates of happy and surprise versus other sorts of HVFs.

	100 × 100 blocks		50 × 50 blocks		25 × 25 blocks	
	Recognition	FPR	Recognition	FPR	Recognition	FPR
AN	74.1%	12.6%	77.4%	12.6%	70.9%	13.9%
DI	78.6%	12.1%	78.6%	10.9%	75.0%	13.4%
FE	69.5%	8.0%	73.9%	8.0%	69.5%	8.0%
SA	67.8%	3.6%	67.8%	2.4%	67.8%	3.6%

Table 3.4: Recognition rates of anger, disgust, surprise and sadness HVFs.

of expressions, we train $C_3^2 = 3$ two-class classifiers (i.e. happy-surprise, surprise-anger and anger-happy classifiers) and test HVFs using majority voting. Likewise, we keep *surprise* and *anger* unchanged but substitute *anger* with *disgust*, *fear* and *sadness* respectively, and then conduct the same training and testing. The results are displayed in Table 3.3.

2. Put *anger*, *disgust*, *fear* and *sadness* in one group. For these four tough expressions, we train a set of classifiers which has $C_4^2 = 6$ two-class classifiers and test new HVFs using majority voting. Table 3.4 shows our results.

3. Put all of the HVFs together, train $C_6^2 = 15$ two-class classifiers. Use this

	100 × 100 blocks		50 × 50 blocks		25 × 25 blocks	
	Recognition	FPR	Recognition	FPR	Recognition	FPR
HA	93.1%	0.0%	96.5%	0.0%	93.1%	0.0%
SU	90.0%	0.0%	93.3%	0.0%	90.0%	0.0%
AN	80.6%	10.1%	80.6%	8.6%	80.6%	10.1%
DI	75.0%	8.5%	82.1%	7.8%	75.0%	8.5%
FE	78.2%	3.4%	78.2%	2.7%	73.9%	3.4%
SA	75.0%	0.0%	71.4%	0.0%	71.4%	1.4%

Table 3.5: Recognition rates of all sorts of HVFs.

set of classifier to recognise all of the HVFs by voting. We obtain the experimental results as shown in Table 3.5.

3.4 Discussion

1. From Table 3.1, we can see that both happy and surprise HVFs have very high recognition rates. For example, happy HVFs reach amazing 100% recognition rate with only 3.3% false positive rate (FPR). They are also quite distinguishable from the rest HVFs according to Table 3.3. These results coincide with our observations on the original image sequences, as human can also easily identify the original happy and surprise sequences from the Cohn-Kanade database. This fact confirms that HVFs preserve the dynamic features well.

2. From Table 3.4, the recognition rates for anger, fear, disgust and sadness HVFs are a lot lower. This reflects the challenges that we have encountered when labeling the training data (noting that nearly half of the training data in these four expressions are not convincing for us to label a class because of expression features entanglement). An investigation of facial expression recognition by human [94] also indicates that human is not sensitive to recognise the anger, fear, disgust and sadness expressions. This fact is exactly embodied in our HVF recognition results.

3. Table 3.5 shows the recognition results when all six expressions are fed to

SVMs for training. We can see that happy and surprise HVFs still stand out and the rest ones are hampered by the entanglement of features. Taking into account our difficulties for labeling the training data, the recognition rates in Table 3.5 make sense.

4. The frame rate of videos and the faces location in frames do not affect our experimental results evidently, but the durations of the expressions have to be similar, e.g., from neutral to apex. Moreover, the size of block is not critical to our results, but generally, the 50×50 block segmentation has the best performance in our experiments.

3.5 Conclusion

According to Figure 3.1, an HVF is a set of variances of local cells (blocks) in temporal direction. HVF uses LBP to extract texture and EMD to calculate the variance, so it can resist illumination and noise influence. In addition, HVF only takes aligned face area as the region of interest, which makes the cell at a certain position represent the same part of a human face. Our experiments demonstrate HVF is an effective representation of the dynamic and internal features of a face video or image sequence. HVF is able to integrate well the dynamic features of a certain duration of expression into a static image through which the static facial recognition approaches can be utlised to recognise the dynamic expressions.

Chapter 4

Hexagonal Histogram Variances Faces for expression recognition

In our earlier work, we have proposed an HVF (Histogram Variance Face) approach and proved its effectiveness for facial expression recognition. In this chapter, we extend the HVF approach and present a novel approach for facial expression. We take into account the human perspective and understanding of facial expressions. For the first time, we propose to use the Local Binary Pattern (LBP) defined on the hexagonal structure to extract local, dynamic facial features from facial expression images. The dynamic LBP features are used to construct a static image, namely *Hexagonal Histogram Variance Face (HHVF)*, for the video representing a facial expression. We show that the HHVFs representing the same facial expression (e.g., surprise, happy and sadness etc.) are similar no matter if the performers and frame rates are different. Therefore, the proposed facial recognition approach can be utilised for the dynamic expression recognition. We have tested our approach on the well-known Cohn-Kanade AU-Coded Facial Expression database. We have found the improved accuracy of HHVF-based classification compared with the HVF-based approach.

4.1 Hexagonal Histogram Variances Faces

The HVF image is a representation of the dynamic features in a face video [95]. An extension of HVF represented on the hexagonal structure, namely HHVF, is performed in this section.

4.1.1 Fiducial point detection and face alignment

For different expression videos, normally the scales and locations of human faces in frames are various. To make all the HHVFs have the same scale and location, it is critical to detect the face fiducial points. Bilinear interpolation is used to scale the face images to the same size. To detect the fiducial points, we apply Viola-Jones face detector [89], a real-time face detection scheme based on Haar-like features and AdaBoost learning. We detect and locate the positions of eyes for the images on the Cohn-Kanade expression database [35]. Each face image is cut and scaled according to eyes' positions and the distance between the eyes.

4.1.2 Conversion from square structure to hexagonal structure

We follow the work shown in [96] and also in the previous chapter to represent images on the hexagonal structure. As shown in Figure 4.1, the hexagonal pixels appear only on the columns where the square pixels are located. As illustrated in Figure 4.1, for a given hexagonal pixel (denoted by X but not shown in the figure), there exist two square pixels (denoted by A and B but again not shown in the figure), lying on two consecutive rows and the same column of X, such that point X falls between A and B. Therefore, we can use the linear interpolation algorithm to obtain the light intensity value of X from the intensities of A and B. When we display the image on the hexagonal structure, every two hexagonal rows as shown in Figure 4.1 is combined into one single square row with their columns unchanged.

Although there are some additional computations for conversion between square to hexagonal structure, these computations are light-weighted and can be ignored.

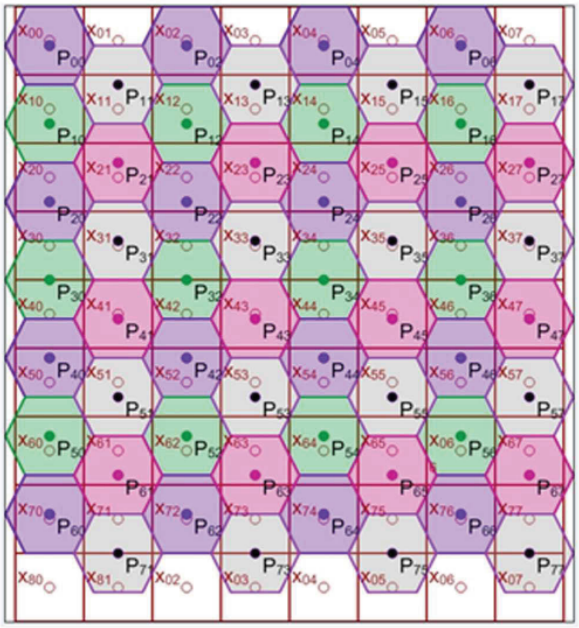


Figure 4.1: A 9x8 square structure and a constructed 7x8 hexagonal structure [96]

4.1.3 Preprocessing and LBP texturising

After each input face image is aligned, cut, rescaled and normalised, we replace the values of the pixels outside the ellipse area around the face by 255, and keep the pixel values unchanged in the elliptic face area. To eliminate the illumination interference, we employ an LBP operator [97][22][40] to extract the texture (i.e., LBP) values in each masked face.

LBP was originally introduced by Ojala et al. in [98] as texture description and defined on the traditional square image structure. The basic form of an LBP operator labels the pixels of an image by thresholding the 3×3 neighborhood of each pixel by the grey value of the pixel (the centre). An illustration of the basic LBP operator is shown in Figure 4.2. Similar to the construction of basic LBP on the square structure, the basic LBP on the hexagonal structure, called Hexagonal LBP (HLBP) is constructed as shown in Figure 4.3 [97] on a cluster of 7 hexagonal pixels. By defining the HLBPs on the hexagonal structure, the number of different patterns has been reduced from $2^8 = 256$ on the square

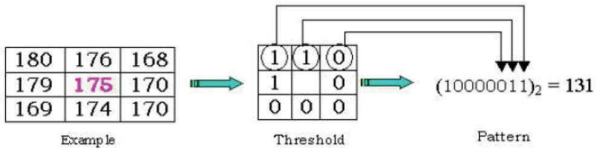


Figure 4.2: An example of computing LBP in a 3×3 neighborhood on square structure [97]

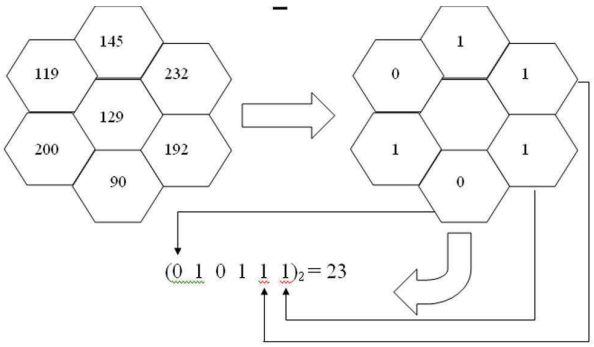


Figure 4.3: An example of computing HLBP in a 7-pixel hexagonal cluster [97].

structure to $2^6 = 64$ on the hexagonal structure. More importantly, because all neighboring pixels of a reference pixel have the same distance to it, the grey values of the neighboring pixels have the same contributions to the reference pixel on the hexagonal structure.

4.1.4 Earth Mover's Distance (EMD)

We employ EMD as shown in Section 3.1.3.1 to measure the distance between two histograms when calculating histogram variances in the temporal direction. In Equations 3.2 and 3.3, p_i and q_j are the grayscale pixel values, which are in $[0, 63]$. w_{pi} and w_{qj} are the pixel distributions at p_i and q_j respectively. The ground distance d_{ij} that we choose is the square of Euclidean distance between p_i and q_j, i.e., $d_{ij} = (p_i - q_j)^2$.

4.1.5 Histogram variances

The steps to compute the histogram variance on hexagonal structure are similar to the ones shown in Section 3.1.

1. Break down each image evenly into $M \times N$ blocks, denoted by $B_{x,y;k}$, where x is row index, y is column index and k is the k-th frame in the sequence. Here, the block size, rows and columns are corresponding to those images displayed on the square structure as described in Subsection 4.1.2. We then calculate every gray-value histogram of $B_{x,y;k}$, denoted by $H_{x,y;k}$.

2. Calculate the histogram variance $var(x, y)$ using Equation 3.6 of Section 3.1.3.2.

3. Construct an $M \times N$ 8-bit grayscale image as our HHVF. Similar to former HVF, Suppose that $hhvf(x, y)$ denotes the pixel value at coordinate (x, y) in an HHVF image. Then, $hhvf(x, y)$ is also computed by Equation 3.8. Figure 4.4 shows some HHVF examples extracted from happiness, surprise and sadness videos respectively.

To consider if the different block sizes may affect the recognition, we obtain HHVFs with size 3×3 in our experiments, then with the sizes of 6×6 and 12×12. How a row and a column are defined has been described in Section 4.1.2 and illustrated in Figure 4.1 showing the difference from that in the square structure.

4.2 Classification

HHVF records the dynamic features of the expression. As we can see in Figure 4.4, for the expressions of happiness, surprise and sadness, the homogeneous HHVFs look similar and HHVFs belonging to different expressions have very distinct features. To verify the performance of HHVF image's features, we utilise the typical facial recognition technologies PCA+SVMs as shown in Section 3.2.

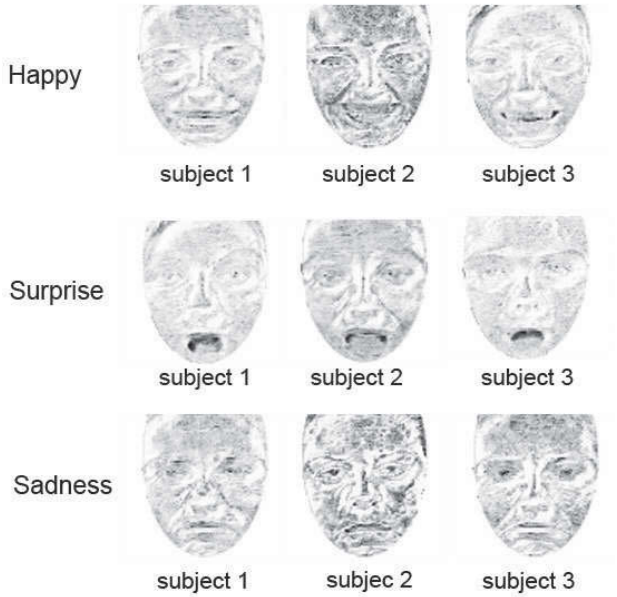


Figure 4.4: Examples of HHVF images

4.3 Experiments

4.3.1 Dataset

The same as Chapter 3, our experiments also use the Cohn-Kanade AU-Coded Facial Expression Database [35]. Because of the restriction described in Section 3.3, we select 49 subjects from the database instead using all the objects. Each subject has up to 6 expressions. The total number of expression sequences is 241. The image sequences belonging to the same expression have the similar duration but their frame rates are various and are from 15-30 frames per second. For each expression, we use about 80% HHVFs for PCA+SVMs training and use all HHVFs for classification (i.e., testing).

4.3.2 HHVFs generation

The faces of selected subjects are detected and cut. Then these faces are scaled to size of 300×300 and aligned. These cut and rescaled images are then converted to the images on the hexagonal structure. The size of the new images are of 259×300.

	3×3 blocks		6×6 blocks		12×12 blocks	
	Recog rate	FPR	Recog rate	FPR	Recog rate	FPR
Happy	100%	2.17%	97.78%	2.17%	97.78%	2.17%
Surprise	97.82%	0.00%	97.82%	2.22%	97.82%	2.22%

Table 4.1: Recognition rates of happy and surprise HHVFs.

HLBP operator is then applied to the images on the hexagonal structure. The final data dimension is reduced to 220 after the PCA operation.

4.3.3 Training and recognition

As a supervised learning, the training data (HHVFs) are labeled according to their classes. Since the Cohn-Kanade database contains only the AU-Coded combinations for image sequences instead of expression definitions (i.e. surprise, happy, anger etc.), we need to label each HHVF with an expression definition manually according to FACS before feeding it to SVMs. In terms of human perception, we are quite confident to recognise original image sequences of happiness and surprise. Therefore, the training data for these two classes can be labeled with high accuracy. This implies that these two expressions have evidently unique features. Our experimental results (Table 4.1) testify this point with high recognition rate when we train and test only these two expressions. In Table 4.1, *FPR* stands for *false positive rate*.

When we manually label HHVFs of *anger*, *disgust*, *fear* and *sadness*, nearly half of them are very challenging to be correctly classified. Neither AU-Coded combinations nor human's perception can satisfactorily classify these facial expression sequences, especially the *anger* and *sadness* sequences. Actually, AU-Coded prototypes in FACS (2002 version) [35] have created large overlaps for these expressions. From human's vision perspective, one expression appearance of a person may reflect several different expressions, so the image sequence of a facial expression can often be interpreted as for different expressions. An investigation [94] about facial expression recognition by human indicates that compared to the expressions of *happy* and *surprise*, the expressions of *anger*, *fear*, *disgust* and *sadness* are much more difficult to be recognised by people. (see Table 4.2).

	Happy	Surprise	Angry	Disgust	Fear	Sadness
Recognition rate(%)	97.8	79.3	55.9	60.2	36.8	46.9

Table 4.2: A recent investigation of facial expression recognition by human in [94].

	3×3 blocks		6×6 blocks		12×12 blocks	
	Recog rate	FPR	Recog rate	FPR	Recog rate	FPR
Angry	74.36%	13.51%	76.92%	13.51%	74.36%	14.41%
Disgust	73.68%	10.71%	73.68%	9.82%	71.05%	10.71%
Fear	68.57%	8.69%	71.43%	7.83%	68.57%	8.69%
Sadness	68.42%	5.36%	68.42%	5.36%	68.42%	5.35%

Table 4.3: Recognition rates of anger, disgust, surprise and sadness HHVFs.

Despite the above-mentioned difficulties, we have tried our best to manually label the expressions. We have also conducted the following experiments:

1. We test for only *anger*, *disgust*, *fear* and *sadness* classes. For these four tough expressions, we train a set of classifiers having $C_4^2 = 6$ two-class classifiers and test the HHVFs based on majority voting. Table 4.3 shows our results.

2. We consider all HHVFs, and train $C_6^2 = 15$ two-class classifiers. We use this set of classifiers to recognise all HHVFs based on voting. We obtain the experimental results as shown in Table 4.4.

3. To compare performances between HHVFs and HVFs, we replace HHVFs with HVFs and conduct the same experiments of Table 4.4 on the same dataset. The average recognition rates between HVFs and HHVFs are shown in Table 4.5.

4.4 Discussion

1. From Table 4.1, we can see that both happy and surprise HHVFs have produced very high recognition rates. For example, the happy HHVFs

	3 × 3 blocks		6 × 6 blocks		12 × 12 blocks	
	Recog rate	FPR	Recog rate	FPR	Recog rate	FPR
Happy	93.33%	0.0%	95.56%	0.0%	95.56%	0.0%
Surprise	91.3%	0.51%	93.48%	0.51%	93.48%	0.51%
Angry	82.05%	7.43%	82.05%	5.94%	76.92%	5.94%
Disgust	78.95%	4.93%	81.58%	4.43%	81.58%	4.93%
Fear	77.14%	5.34%	80.00%	5.34%	77.14%	5.34%
Sadness	78.95%	0.49%	78.95%	0.49%	78.95%	1.48%

Table 4.4: Recognition rates of all sorts of HHVFs

	3 × 3 Recog rate		6 × 6 Recog rate		12 × 12 Recog rate	
	HHVFs	HVFs	HHVFs	HVFs	HHVFs	HVFs
Happy	93.33%	91.10%	95.56%	93.33%	95.56%	93.33%
Surprise	91.30%	91.30%	93.48%	93.48%	93.48%	91.30%
Angry	82.05%	79.49%	82.05%	79.49%	76.92%	76.92%
Disgust	78.95%	78.95%	81.58%	81.58%	81.58%	81.58%
Fear	77.14%	77.14%	80.00%	77.14%	77.14%	77.14%
Sadness	78.95%	78.95%	78.95%	78.95%	78.95%	75.60%
Average rate	**83.60%**	**82.82%**	**85.27%**	**83.99%**	**83.94%**	**82.65%**

Table 4.5: Recognition rates between HVFs and HHVFs. The last row is the average recognition rates of six categories. In our experiments, HHVFs slightly outperforms HVFs.

reach amazing 100% recognition rate with only 2.17% false positive rate (FPR). These results coincide with our observation on the original image sequences, as human can also easily identify the original happy and surprise sequences from the Cohn-Kanade database.

2. From Table 4.3, the recognition rates for anger, fear, disgust and sadness HHVFs are much lower. This reflects the challenges that we have encountered when labeling the training data (that nearly half of the training data for these four expressions are not convincing us to accurately label their classes because of the expression feature entanglement).

3. Table 4.4 shows the recognition results when all six expressions were fed to SVMs for training. We can see that the happy and surprise HHVFs still stand out and the rest are hampered by the entanglement of features. Taking into account our difficulties for labeling the training data, the recognition rates in Table 4.4 make sense.

4. The size of blocks is not critical to our results, although the segmentation into 6×6 blocks has the best performance in our experiments as shown in Table 4.4.

4.5 Conclusion

Our experiments demonstrate that HHVF is an effective representation of the dynamic and internal features of a face video or an image sequence. Comparing to HVFs, HHVFs can achieve better recognition rates for expression recognition in accordance with Table 4.5. HHVF is able to integrate well the dynamic features of a certain duration of expression into a static image through which the static facial recognition approaches can be utlised to recognise the dynamic expressions. The application of HHVFs fills the gap between the expression recognition and facial recognition.

Chapter 5

MIL-SKDE: Multiple-instance learning with supervised kernel density estimation

In previous chapters, our work is focused on the expression recognition which is a categorisation task using single-instance learning because labeling such training data is quite convenient. In this chapter, we switch our job to another application of object categorisation which involves determining whether or not an image contains some specific categories of objects. As mentioned in Section 1.3, the task of object categorisation can be naturally cast as an multiple-instance learning (MIL) problem because individually labeling and normalizing objects for training is expensive. Therefore, we propose an innovative multiple-instance learning algorithm, Multiple-instance Learning with Supervised Kernel Density Estimation (MIL-SKDE), to tackle the object categorisation. Actually, aside from object categorisation, our approach is able to be applied for any other MIL applications, e.g., drug activity detection and region-based image classification.

5.1 MIL-SKDE algorithm

Multiple-instance learning (MIL) is a variation on supervised learning. Instead of receiving a set of labeled instances, the learner receives a set of bags that are

labeled. Each bag contains many instances. The aim of MIL is to classify new bags or instances. Here, we propose a novel algorithm, MIL-SKDE (multiple-instance learning with supervised kernel density estimation), which addresses MIL problem through an extended framework of "KDE (kernel density estimation) + mean shift". Since the KDE + mean shift framework is an unsupervised learning method, we extend KDE to its supervised version, called supervised KDE (SKDE), by considering class labels of samples. To seek the modes (local maxima) of SKDE, we also extend mean shift to a supervised version (called supervised mean shift) by taking into account sample labels. SKDE is an alternative of the well-known diverse density estimation (DDE) whose modes are called concepts. Comparing to DDE, SKDE is more convenient to learn multi-modal concepts and robust to labeling noise (mistakenly-labeled bags). Finally, each bag is mapped into a concept space where the multi-class SVM classifiers are learned. We show the properties of supervised mean shift on synthetic data and test MIL-SKDE in region-based image classification and object categorisation. The results demonstrate that our approach outperforms the state-of-the-art MIL approaches.

5.1.1 Conventional kernel density estimation and mean shift

As described in Section 2.2.2, kernel density estimation (KDE) [76][74] is a well-known nonparametric estimator of univariate or multivariate densities based on a finite amount of data samples. For simplicity, here we utilise KDE with fixed bandwidth, i.e., $h_1 = h_2 = \ldots = h_n$. Supposing that $\{\mathbf{x}_i\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d$ are i.i.d. data drawn from an unknown density $f(\mathbf{x})$, then the density $f(\mathbf{x})$ can be estimated by the KDE with fixed bandwidth:

$$f(\mathbf{x}) \approx \hat{f}_{kde}(\mathbf{x}) = \frac{c}{nh^d} \sum_{i=1}^{n} k\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right), \tag{5.1}$$

where c is the normalisation constant to make $\hat{f}_{kde}$ be integrated to one, h is a fixed bandwidth (variable h is shown in Section 2.2.2), and function $k(x)$ is one

of the kernel profiles. For example, the Gaussian kernel profile

$$k\left(x\right)=\exp\left(-\frac{1}{2}x\right),x\geq 0, \tag{5.2}$$

or the Epanechnikov kernel profile

$$k\left(x\right)=\begin{cases}1-x & 0\leq x\leq 1\\ 0 & x>1.\end{cases} \tag{5.3}$$

To locate the modes of the density function (5.1), we take the gradient of (5.1) and set the gradient to zero, resulting in,

$$\begin{aligned}\nabla\hat{f}_{kde}\left(\mathbf{x}\right)&=\frac{2c}{nh^{d+2}}\sum_{i=1}^{n}\left(\mathbf{x}_i-\mathbf{x}\right)g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\\&=\frac{2c}{nh^{d+2}}\left[\sum_{i=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right]\left[\underbrace{\frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}}_{\text{mean vector}}-\mathbf{x}\right]\\&=0,\end{aligned} \tag{5.4}$$

where

$$g(x)=-k'(x). \tag{5.5}$$

Mean shift [78][79] is an iterative procedure to locate the KDE local maxima $\mathbf{x}'s$ that satisfy (5.4). Let $\mathbf{x}$ be an initial point. Then, $\mathbf{x}$ will gradually converge to a maximum of (5.1) via iteratively setting $\mathbf{x}=\bar{\mathbf{x}}$, where $\bar{\mathbf{x}}$ is the mean vector calculated by

$$\bar{\mathbf{x}}=\frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n}g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}. \tag{5.6}$$

Comaniciu et al. [79] and Li et al. [99] showed that the convergence was guaranteed when $\mathbf{x}$ starting from any data point. Fashing et al. [100] showed that the mean shift was equivalent to Newton's method in the case of piecewise constant kernels and proved that for all kernels mean shift procedure was a quadratic

bound maximisation. Carreira-perpiñán et al. [101] showed that, when the kernel was Gaussian, mean shift was an expectation-maximisation (EM) algorithm, and when the kernel was non-Gaussian, it was a generalised EM algorithm. KDE and mean shift have been widely employed in applications of the feature space analysis, such as image segmentation [79] and object tracking [102].

Our intention is to tackle the MIL problem using the kernel density analysis. By observing (5.1) and (5.6) above, one can see that there is not class label associated with any data point $\mathbf{x}_i$ in both KDE and mean shift. However, the labels of instances (i.e., data points in a feature space) are necessary for all MIL approaches. Since the kernel density analysis cannot be applied in MIL, in the following sections, we propose the supervised versions of KDE and mean shift by introducing class labels into (5.1) and (5.6) in our approach.

We would like to note that there have been existing approaches with titles similar to the title of this chapter, e.g., "semi-supervised kernel density estimation (SSKDE)" [103] and "weakly supervised mean shift" [104]. However, these existing approaches have little relation to our approach. [103] targets on how to assign labels to extra unlabeled data based on a fraction of already labeled data in order to obtain more accurate density estimator. Therefore, the goal of [103] is to compute a posterior $\hat{P}(C_k|\mathbf{x}_j)$, where C_k is the class label to be assigned to $\mathbf{x}_j$ whose label is unknown. However, our approach investigates how the modes of KDE will be influenced if the positive and negative labels are considered in the same time. In [104], the mean shift itself has not been changed but data points are mapped to a new feature space where the constraints that the pairs of points should be clustered together are enforced. In this paper, we focus on developing a novel mean shift process which adopts to the situation that data points are divided into positive and negative.

5.1.2 Supervised kernel density estimation

In this section, we derive a new density to estimate instance distribution given the bag samples. We desire that the modes of the new density are able to represent the concepts to be learned. To formulate such density, the kernel density estimation (KDE) seems to be a good option as seeking modes in KDE has been

well investigated in the existing work. However, a gap between MIL and KDE is that MIL is a generalised supervised learning whereas KDE is an unsupervised learning. To bridge this gap, we derive an supervised version of KDE in which the class labels are considered.

Firstly, a preprocessing on the original data set is needed. Suppose data set $\mathcal{D} = \{\langle B_i, y_i\rangle\}_{i=1}^{m}$, where $B_i = \{\mathbf{x}_{i,j}\}_{j=1}^{|B_i|}$. To reduce problem complexity, we cluster the instances $\{\mathbf{x}_{i,j}\}_{j=1}^{|B_i|}$ of each B_i separately using mean shift, and replace the instances of each bag B_i with the cluster centers, then resulted concise bag corresponding to B_i is denoted by B_i' ($|B_i'| \leq |B_i|$). This clustering operation merges the similar instances together. It does not change the nature of MIL at all but saves much computation later on by reducing bag sizes from $|B_i|$ to $|B_i'|$. Besides, the condition that the instances in any bag are distinct from each other will reduce the disturbance of our concept learning. Basically, any clustering methods, e.g., K-means or DBSCAN, can be applied in the preprocessing step. However, the reason to choose mean shift in the preprocessing step is that, after preprocessing, we need to apply the supervised mean shift for MIL concept learning. The mean shift in the preprocessing step and the supervised mean shift can share the same bandwidth value, to make our job more consistent. After preprocessing, the new data set is denoted by $\mathcal{D}' = \{\langle B_i', y_i\rangle\}_{i=1}^{m}$.

Since the labels are on bag level instead of instance level, to learn concepts from $\mathcal{D}'$, we firstly pass the bag labels to the instances, i.e., the instances within a positive (negative) bag are labeled positive (negative). Then, by reindexing instances, the data set is now $\mathcal{D}' = \{\langle \mathbf{x}_i, y_i\rangle\}_{i=1}^{n}$, where each $\mathbf{x}_i \in \mathbb{R}^d$ is an instance, $y_i \in \{0, 1\}$ is a label and $n = \sum_{i=1}^{m} |B_i'|$. Although passing bag labels to instances and reindexing instances release the relationship between bags and instances, there exist a couple of clues that help us to induce the concepts from the new data set $\mathcal{D}'$. These labeled instances are the data points in the feature space. Note that each point in the feature space now has a class label. Therefore, conventional KDE methods cannot be applied directly to estimate instance distribution. To tackle this issue, we firstly investigate two properties of concepts in MIL.

Property 1: Any potential concept must stay far away from the high density of the negative instances in the feature space. The reason is that any concept

will never frequently occur across the negative bags.

Property 2: Any potential concept must stay in the high density of the positive instances in the feature space. The reason is that a concept tends to frequently occur across positive bags. Therefore, a concept only comes from a high density of the positive instances.

Figure 5.1 illustrates these two properties of MIL in a feature space.

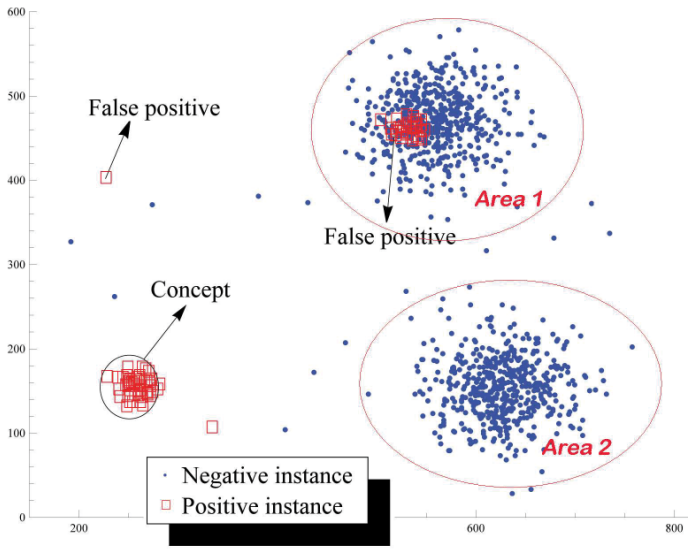


Figure 5.1: The properties of concepts. As shown above, the sporadic positive instances are actually the false positive instances, and the positive instances that form a cluster but stay close to negative clusters are also false positive instances (e.g., Area 1). Only those positive instances form a cluster, as well as staying far away from negative clusters are true positive instances. The centroid of a cluster of the true positive instances is a concept to be learned. Note that the instances that form a cluster must come from different bags because a clustering operation has been applied on each bag beforehand.

Properties 1 and 2 provide us the clues to build a new density function which has high values at compact areas of positive points and low values at compact areas of negative points. Then, the modes of such density function would be the concepts to be sought. Suppose that the sample set $\mathcal{D}' = \mathbf{z} = \{\mathbf{z}_i\}_{i=1}^n = \{\langle \mathbf{x}_i, y_i \rangle\}_{i=1}^n$ are i.i.d. samples, and let $\mathbf{x} \in \mathbb{R}^d$ denote a concept, then the density

$Pr(\mathbf{x})$ is:

$$\begin{aligned} Pr(\mathbf{x}) &= \int_{\mathbf{z}} Pr(\mathbf{x}|\mathbf{z}) Pr(\mathbf{z}) d\mathbf{z} \\ &= \mathbb{E}_{\mathbf{z}}[Pr(\mathbf{x}|\mathbf{z})] \\ &\approx \frac{1}{n} \sum\nolimits_{i=1}^{n} Pr\left(\mathbf{x}|\mathbf{z}_i\right). \end{aligned} \tag{5.7}$$

For clarity, we separate notations of positive and negative samples, then the sample set $\mathcal{D}' = \{\mathbf{z}_i\}_{i=1}^{n} = \left\{\mathbf{x}_i^+\right\}_{i=1}^{n^+} \bigcup \left\{\mathbf{x}_i^-\right\}_{i=1}^{n^-}$, where $\mathbf{x}_i^+$ and $\mathbf{x}_i^-$ denote the positive (having $y_i = 1$) and negative (having $y_i = 0$) points respectively, and $n = n^+ + n^-$. Finally,

$$Pr(\mathbf{x}) \approx \hat{f}(\mathbf{x}) = \frac{1}{n} \left(\sum_{i=1}^{n^+} Pr\left(\mathbf{x}|\mathbf{x}_i^+\right) + \sum_{i=1}^{n^-} Pr\left(\mathbf{x}|\mathbf{x}_i^-\right) \right). \tag{5.8}$$

Now, we define $Pr\left(\mathbf{x}|\mathbf{x}_i^+\right)$ and $Pr\left(\mathbf{x}|\mathbf{x}_i^-\right)$ in (5.8) as follows.

$$Pr\left(\mathbf{x}|\mathbf{x}_i^+\right) = \frac{c}{h^d} k\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i^+}{h} \right\|^2 \right), \tag{5.9}$$

$$Pr\left(\mathbf{x}|\mathbf{x}_i^-\right) = 1 - \frac{c}{h^d} k\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i^-}{h} \right\|^2 \right), \tag{5.10}$$

where c is a normalisation constant to ensure $Pr\left(\mathbf{x}|\mathbf{x}_i^+\right) \in [0, 1]$, h is the bandwidth constant, d is the feature space dimensionality, and $k(\cdot)$ is one of the kernel profiles such as the Gaussian profile as shown in (5.2). Putting (5.9) and (5.10) into (5.8) we get:

$$\hat{f}\left(\mathbf{x}\right) = \frac{c}{nh^d} \sum_{i=1}^{n^+} k\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i^+}{h} \right\|^2 \right) - \frac{c}{nh^d} \sum_{i=1}^{n^-} k\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i^-}{h} \right\|^2 \right) + \frac{n^-}{n}. \tag{5.11}$$

Rewrite (5.11) into a compact form:

$$\hat{f}\left(\mathbf{x}\right) = \frac{c}{nh^d} \sum_{i=1}^{n} (-1)^{1-y_i} k\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right) - \frac{1}{n} \sum_{i=1}^{n} y_i + 1. \tag{5.12}$$

Note that (5.12) is an extension of KDE (5.1) by embedding labels y_i's into KDE. If all $\mathbf{x}_i$'s are positive , i.e., all $y_i = 1$, then (5.12) will revert to a standard KDE (5.1). To distinguish it from standard KDE, we call (5.12) the supervised KDE (SKDE).

5.1.2.1 SKDE versus DDE

Supervised Kernel Density Estimation (SKDE) (5.12) is an alternative of the widely adopted Diverse Density Estimation (DDE) (1.1) which has been the base of many other algorithms, e.g., GEM-DD [58] and DD-SVM [59]). The DDE defines a "concept" that is an area where there is both high density of positive instances and low density of negative instances. The problem of trying to find a concept changes to a problem of finding the mode of DDE (1.1). Likewise, the concepts can also be found by seeking modes of SKDE (5.12). For example, if a point $\mathbf{x}$ is in the area of high density of positive instances like the concept area in Figure 5.1, that means $\mathbf{x}$ has many near positive neighbors $\mathbf{x}_i$ and leads the value of (5.12) to be high. On the contrary, if $\mathbf{x}$ is in the area of high density of negative instances like Area 1 and Area 2 in Figure 5.1, the value of (5.12) will be low. This justifies that the modes of (5.12) stand for the concepts to be learned.

However, SKDE has advantages over DDE in terms of robustness to labeling noise and mode seeking. For DDE (1.1), if $\mathbf{x} = \mathbf{t} \in \mathbb{R}^d$ is a concept, with all bags correctly labeled, then

$$\forall\ B_i : Pr\left(\mathbf{t}|B_i\right) \to 1 \xrightarrow{\text{via (1.1)}} \hat{f}_{dde}\left(\mathbf{t}\right) \to 1. \tag{5.13}$$

However, if a bag, say B_i^+ (or B_i^-), is mislabeled, then it means

$$\begin{aligned} &\forall\ \mathbf{x}_{i,j} \in B_i^+ : Pr\left(\mathbf{t}|\mathbf{x}_{i,j}\right) \to 0 \xrightarrow{\text{via (1.2)}} Pr\left(\mathbf{t}|B_i^+\right) \to 0 \\ &\left(\text{or } \exists\ \mathbf{x}_{i,j} \in B_i^- : Pr\left(\mathbf{t}|\mathbf{x}_{i,j}\right) \to 1 \xrightarrow{\text{via (1.3)}} Pr\left(\mathbf{t}|B_i^-\right) \to 0\right) \\ &\xrightarrow{\text{via (1.1)}} \hat{f}_{dde}\left(\mathbf{t}\right) \to 0. \end{aligned} \tag{5.14}$$

From (5.14), we can see that even a single mislabeled bag (labeling noise) will significantly change DDE. In contrast, for SKDE (5.12), if $\mathbf{x} = \mathbf{t} \in \mathbb{R}^d$ is a concept,

having a fraction of labels y_i's negated, the $\hat{f}(\mathbf{t})$ will not changed greatly because (5.12) is the summation instead of the product in DDE (1.1).

Additionally, in a high-dimensional space $\mathbb{R}^d$, swiftly finding the modes of DDE (1.1) is hard. One of the reasons is that computing the Hessian in a high-dimensional space is costly. However, SKDE represented in (5.12) is not an arbitrary function. It is basically a kernel density and its modes can be found by modifying the existing techniques of kernel density analysis such as mean shift, which starts from an initial point and quickly converges to the mode without computing the Hessian. In Section 5.1.3 below, we propose a supervised version of mean shift to tackle the mode seeking of SKDE.

5.1.3 Supervised mean shift

In the mean shift algorithm for mode seeking of KDE, a starting point $\mathbf{x}$ will quickly converge to a mode (local maximum) of KDE by iteratively shifting $\mathbf{x}$ to a mean vector $\bar{\mathbf{x}}$ calculated by (5.6). The convergence is guaranteed as the mean shift vector $\bar{\mathbf{x}} - \mathbf{x}$ always points toward the direction of maximum increase in the KDE, i.e., the gradient of (5.1). The mean shift is steepest ascent with a varying step size which is the magnitude of the gradient. Mean shift does not compute the Hessian matrix which is very expensive in a high-dimensional space during optimisation.

By embedding the class label into mean shift iteration, we extend mean shift to seek modes of SKDE and the extended mean shift is named supervised mean shift. Following previous notations, let

$$\mathcal{D}' = \{\langle \mathbf{x}_i, y_i \rangle\}_{i=1}^{n} = \left\{\mathbf{x}_i^+\right\}_{i=1}^{n^+} \bigcup \left\{\mathbf{x}_i^-\right\}_{i=1}^{n^-}, \tag{5.15}$$

where data point $\mathbf{x}_i \in \mathbb{R}^d$, label $y_i \in \{0, 1\}$ and $n = n^+ + n^-$. By taking derivative of (5.12), we have that

$$\nabla \hat{f}(\mathbf{x}) = \frac{2c}{nh^{d+2}} \sum_{i=1}^{n} (-1)^{1-y_i} (\mathbf{x}_i - \mathbf{x})\, g\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2\right). \tag{5.16}$$

For clarity, we separate positive and negative points explicitly and denote

$$\tau_{\mathbf{x}}^{\mathbf{x}_i,h} = g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right). \tag{5.17}$$

Then,

$$\begin{aligned}
\nabla \hat{f}(\mathbf{x}) &= \frac{2c}{nh^{d+2}} \sum_{i=1}^{n} (-1)^{1-y_i} (\mathbf{x}_i - \mathbf{x}) \tau_{\mathbf{x}}^{\mathbf{x}_i,h} \\
&= \frac{2c}{nh^{d+2}} \left[\sum_{i=1}^{n^+} \left(\mathbf{x}_i^+ - \mathbf{x}\right) \tau_{\mathbf{x}}^{\mathbf{x}_i^+,h} - \sum_{i=1}^{n^-} \left(\mathbf{x}_i^- - \mathbf{x}\right) \tau_{\mathbf{x}}^{\mathbf{x}_i^-,h}\right] \\
&= \frac{2c}{nh^{d+2}} \left[\sum_{i=1}^{n} \tau_{\mathbf{x}}^{\mathbf{x}_i,h}\right] \left[\frac{\sum_{i=1}^{n^+} \left(\mathbf{x}_i^+ - \mathbf{x}\right) \tau_{\mathbf{x}}^{\mathbf{x}_i^+,h} - \sum_{i=1}^{n^-} \left(\mathbf{x}_i^- - \mathbf{x}\right) \tau_{\mathbf{x}}^{\mathbf{x}_i^-,h}}{\sum_{i=1}^{n} \tau_{\mathbf{x}}^{\mathbf{x}_i,h}}\right] \\
&= \frac{2c}{nh^{d+2}} \left[\sum_{i=1}^{n} \tau_{\mathbf{x}}^{\mathbf{x}_i,h}\right] \left[\frac{\sum_{i=1}^{n^+} \mathbf{x}_i^+ \tau_{\mathbf{x}}^{\mathbf{x}_i^+,h} + \sum_{i=1}^{n^-} \left(2\mathbf{x} - \mathbf{x}_i^-\right) \tau_{\mathbf{x}}^{\mathbf{x}_i^-,h}}{\sum_{i=1}^{n} \tau_{\mathbf{x}}^{\mathbf{x}_i,h}} - \mathbf{x}\right].
\end{aligned} \tag{5.18}$$

In (5.18), the first term $\sum_{i=1}^{n} \tau_{\mathbf{x}}^{\mathbf{x}_i,h}$ is a positive number. The second term is the supervised mean shift vector

$$m(\mathbf{x}) = \frac{\sum_{i=1}^{n^+} \mathbf{x}_i^+ \tau_{\mathbf{x}}^{\mathbf{x}_i^+,h} + \sum_{i=1}^{n^-} \left(2\mathbf{x} - \mathbf{x}_i^-\right) \tau_{\mathbf{x}}^{\mathbf{x}_i^-,h}}{\sum_{i=1}^{n} \tau_{\mathbf{x}}^{\mathbf{x}_i,h}} - \mathbf{x}. \tag{5.19}$$

Rewrite (5.19) into a compact form by embedding the label y_i:

$$m(\mathbf{x}) = \underbrace{\frac{\sum_{i=1}^{n} \left[\mathbf{x}_i^{y_i} (2\mathbf{x} - \mathbf{x}_i)^{1-y_i} \tau_{\mathbf{x}}^{\mathbf{x}_i,h}\right]}{\sum_{i=1}^{n} \tau_{\mathbf{x}}^{\mathbf{x}_i,h}}}_{\text{mean vector } \bar{\mathbf{x}}} - \mathbf{x}. \tag{5.20}$$

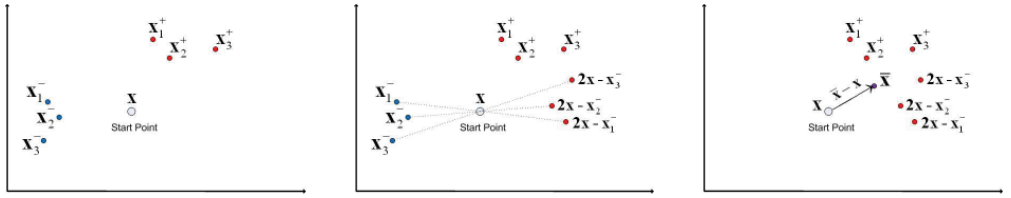
Therefore, the mean vector $\bar{\mathbf{x}}$ in our supervised mean shift is:

$$\bar{\mathbf{x}} = \frac{\sum_{i=1}^{n} \left[\mathbf{x}_i^{y_i} (2\mathbf{x} - \mathbf{x}_i)^{1-y_i} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right]}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}. \tag{5.21}$$

In our proposal, a starting point $\mathbf{x}$ will converge to a mode (local maximum)

of SKDE by iteratively shifting $\mathbf{x}$ to a mean vector $\bar{\mathbf{x}}$ calculated using Equation (5.21).

Differing from mean vector computation in mean shift (5.6), the mean vector of supervised mean shift (5.21) is computed by taking into account class labels of data points. If all the data points are positive, (5.21) will reduce to (5.6). The iteration of supervised mean shift can be interpreted as follows. Suppose $\mathbf{x}$ is the initial point, before shifting $\mathbf{x}$ to mean vector $\bar{\mathbf{x}}$, every negative $\mathbf{x}_i$ will be flipped to the other side of $\mathbf{x}$, i.e., $(2\mathbf{x} - \mathbf{x}_i)$. We call point $(2\mathbf{x} - \mathbf{x}_i)$ the *shadow point* of the negative point $\mathbf{x}_i$. Then, calculate the mean vector $\bar{\mathbf{x}}$ among positive points and shadow points using standard mean shift. Finally, shift $\mathbf{x}$ to $\bar{\mathbf{x}}$. Figure 5.2 shows one of the iterations in our supervised mean shift procedure. The procedure will finally converge to a mode of SKDE.



(a) Original feature space with positive ($\mathbf{x}_i^+$) and negative ($\mathbf{x}_i^-$) points. $\mathbf{x}$ is the starting position for an iteration.

(b) Flip every negative point $\mathbf{x}_i^-$ to the other side of $\mathbf{x}$, which is $(2\mathbf{x} - \mathbf{x}_i^-)$ and it is called the shadow point of $\mathbf{x}_i^-$.

(c) Compute the mean vector $\bar{\mathbf{x}}$ of all the positive points and shadow points, then shift current $\mathbf{x}$ to $\bar{\mathbf{x}}$.

Figure 5.2: One of the iterations of the supervised mean shift. Starting with a proper initial point in the feature space, this process goes on until convergence and the convergence point is a mode of SKDE.

5.1.3.1 Selecting starting points

For seeking multiple modes of KDE using mean shift, the starting points are normally all the sample points and the mean shift process is applied to each starting point until convergence. However, for seeking the modes of SKDE using supervised mean shift, the starting points do not need to be the all sample instances. Actually, the negative instances do not contribute to the modes of density and they can be ignored as initial points. Hence, the starting points are only chosen

from the positive instances. Furthermore, those positive instances that stay in the high-density negative areas (like those positive instances in Area 1 of Figure 5.1) can also be ignored. As the training set is usually dominated by negative instances, the selection of starting points will greatly decrease the complexity of the mode seeking. Here, we advance a criterion for selecting initial points. Given a point and its label $(\mathbf{x}, y) \in \mathcal{D}' = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, $\mathbf{x}$ will be regarded as a starting point if it satisfies:

$$\psi(\mathbf{x}, y) = \sum_{i=1}^{n} (-1)^{1-y_i} \, k\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right) \mathbf{1}(y = 1) > 1, \tag{5.22}$$

where indicator function $\mathbf{1}(y = 1) = \begin{cases} 1 & \text{if } y = 1, \\ 0 & \text{otherwise .} \end{cases}$

The selection criterion (5.22) above is a variation of (5.12) and it ensures that any starting point is a positive point whose neighbors are mostly positive. If $\mathbf{x}$ is a false positive point, i.e., a positive point whose most of neighbors are negative ($y_i = 0$), then most of the terms of $(-1)^{1-y_i} \, k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)$ that have large absolute values are negative and the criterion (5.22) is unlikely to hold. It indicates that $\mathbf{x}$ is not qualified to be a starting point. Also, all negative points cannot be the starting points because they do not satisfy the criterion. Only those positive points in the dense areas of positive points will have high $\psi(\mathbf{x}, y)$ values as shown in (5.22) and will be the starting points for supervised mean shift. Algorithm 2 describes how to learn concepts and corresponding weights using the proposed supervised mean shift.

5.1.3.2 Bandwidth estimation for supervised kernel density estimation

Bandwidth estimation is important to ensure KDE/SKDE to properly estimate the underlying density. Inappropriate bandwidth will cause that KDE/SKDE cannot truly reflect the underlying distribution of the sample points. To illustrate effect of bandwidth selection on kernel density estimation, we take a simulated random sample from the standard normal distribution as shown in Figure 5.3

Algorithm 2 Concept learning using supervised mean shift.

Input:

Feature points, $\mathcal{D}' = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{0, 1\}$;
Predefined bandwidth, h;

Output:

$\mathbb{M} = \{(\boldsymbol{\phi}_1, w_1), \ldots, (\boldsymbol{\phi}_\nu, w_\nu)\}$, where $\boldsymbol{\phi}_i$'s are distinct convergence points and w_i's are weights;

1: Initialise $\mathbb{M} = \emptyset$, and $\nu = 0$.
2: Select starting points $\{\mathbf{x}\}$ from $\mathcal{D}'$ using (5.22).
3: For each $\mathbf{x} \in \{\mathbf{x}\}$:
    Do{
        $\boldsymbol{\phi}_{\nu+1} := \mathbf{x}$;
        Calculate the mean $\bar{\mathbf{x}}$ with respect to $\mathbf{x}$ using (5.21);
        $\mathbf{x} := \bar{\mathbf{x}}$;
        }While ($\mathbf{x} \neq \boldsymbol{\phi}_{\nu+1}$)
    If $\boldsymbol{\phi}_{\nu+1} \notin \mathbb{M}$ Then
        $w_{\nu+1} := \hat{f}\left(\boldsymbol{\phi}_{\nu+1}\right)$ using (5.12);
        Append $(\boldsymbol{\phi}_{\nu+1}, w_{\nu+1})$ to $\mathbb{M}$;
        $\nu := \nu + 1$;
4: **return** $\mathbb{M} = \{(\boldsymbol{\phi}_1, w_1), \ldots, (\boldsymbol{\phi}_\nu, w_\nu)\}$;

[1](plotted as the blue spikes on the horizontal axis). The grey curve represents the true density $X \sim \mathcal{N}(0, 1)$. In comparison, the red curve is under-smoothed as it adopts a too small bandwidth. The green curve is over-smoothed since using a too large bandwidth. Only the black curve with a proper width is considered to be optimally smoothed since its density estimate is close to the true density.

Here, we apply a bandwidth h estimation method based on the nearest neighbors as shown in Algorithm 3.

In Algorithm 3, we assign $k =$ the number of positive bags, and $n = 100$. In our experiments, a bag is an image in the training data, and the instances in a bag are the feature representations of local regions of an image. Therefore, the k is equal to the number of images in positive set. These settings work pretty well for estimating bandwidth h on the training sets. In practice, the k does not have to strictly be equivalent to positive image number. Actually, a minor change of

[1]Image is originally from Wikipedia http://upload.wikimedia.org/wikipedia/en/e/e5/Comparison_of_1D_bandwidth_selectors.png.

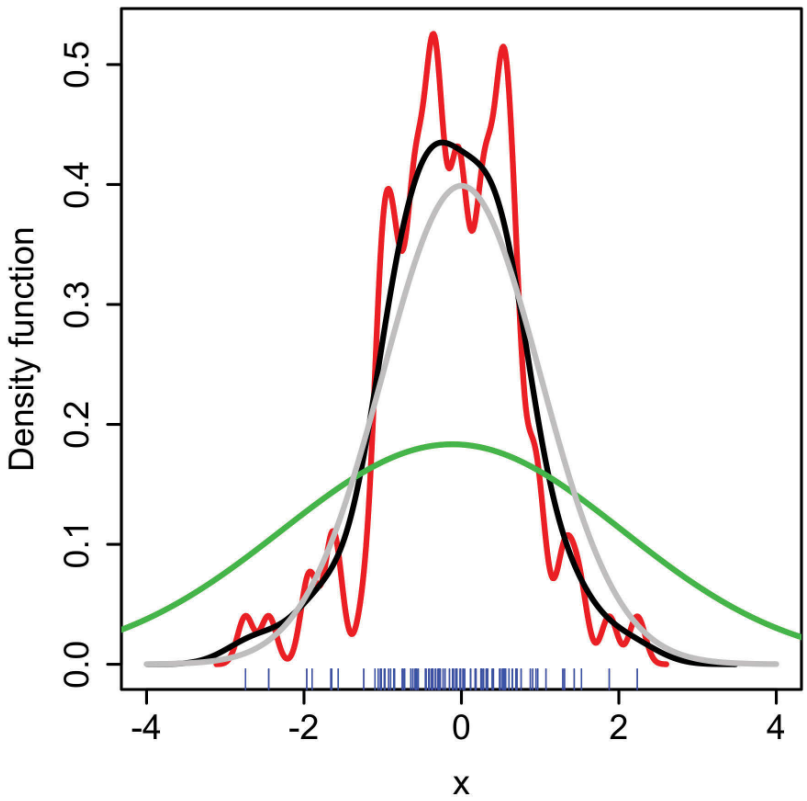


Figure 5.3: Kernel density estimate with different bandwidths for a standard normal distribution. Grey curve represents the true density (standard normal). Red curve is an estimated density with too small bandwidth. Green curve is an estimated density with too large bandwidth. Black curve shows the optimally estimated density with a proper bandwidth.

the k value (e.g., by up to 5% change of positive image number) does not change the bandwidth significantly. Figure 5.12 presented in Section 5.2.3.1 illustrates the relationship between the k and the bandwidth h in our experiments of object categorisation.

5.1.4 Algorithm summary

Here, we summarise the MIL-SKDE algorithm when it is used to learn multiple concepts for a certain class.

Algorithm 3 Bandwidth estimation.

Input:

Positive data set, $D^+ = \left\{\mathbf{x}_i^+\right\}_{i=1}^{n^+}$, $\mathbf{x}_i^+ \in \mathbb{R}^d$;
k nearest neighbors, k;
Loop rounds, n;

Output:

Value of bandwidth, h;

Initially set $h = +\infty$;
for $i = 1$; $i < n$; $i++$ **do**
  Randomly pick out an $\mathbf{x}_i^+$ from D^+;
  Find k nearest neighbors of $\mathbf{x}_i^+$ from $\left\{\mathbf{x}_i^+\right\}_{i=1}^{n^+}$, denoted as $\mathbf{X} = \left\{\mathbf{x}_{i1}^+, \mathbf{x}_{i2}^+, \ldots, \mathbf{x}_{ik}^+\right\}$;
  $h^* = \mathbb{E}\left\|\mathbf{X} - \mathbb{E}\mathbf{X}\right\|$, where $\|\cdot\|$ is L_1 or L_2 norm;
  if $h > h^*$ **then**
    $h = h^*$;
  end if
end for
return h;

Given a set of data $\mathcal{D} = \{\langle B_i, y_i\rangle\}_{i=1}^m$, where bags $B_i = \{\mathbf{x}_{i,j}\}_{j=1}^{|B_i|}$ and class labels $y_i \in \{0, 1\}$, The procedures of learning concepts and corresponding weights in MIL-SKDE can be summed up as follows.

1. Apply a mean shift clustering on each bag B_i and replace the instances of B_i with cluster centers. This clustering ensures that the instances in B_i are distinct (note that, at this step, the bandwidth of mean shift is not critical, and it can be decided by the level of distinction of instances).

2. Pass the bag labels to the instances and reindex instances across all bags to generate the training set $\mathcal{D}' = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d$ and $n = \sum_{i=1}^m |B_i|$.

3. Estimate a bandwidth h using Algorithm 3 on the dataset $\mathcal{D}'$, where the value of nearest neighbors $k = m$.

4. Use the supervised mean shift (Algorithm 2) with bandwidth h to obtain the concepts and weights on $\mathcal{D}'$, denoted as $\mathbb{M} = \{(\boldsymbol{\phi}_i, w_i)\}_{i=1}^\nu$, where $\boldsymbol{\phi}_i \in \mathbb{R}^d$ and $w_1 \in [0, 1]$.

5. Remove those $(\boldsymbol{\phi}_i, w_i)$ from $\mathbb{M}$ whose w_i's are below average, i.e., those $w_i < \mathbb{E}[w] = \frac{1}{s}\sum_{i=1}^{\nu} w_i$. Therefore, only important concepts are kept. We call the final set $\mathbb{M} = \{(\boldsymbol{\phi}_i, w_i)|\}_{i=1}^{t}$, where $w_i \geq \mathbb{E}[w]$ and $t < \nu$, the *pattern* of a class which is later used to map bags for classification.

5.1.5 Classification

Suppose that there are patterns $\{\mathbb{M}_i\}_{i=1}^{\gamma}$ from γ categories. Each pattern is denoted as $\mathbb{M}_i = \left\{(\boldsymbol{\phi}_{ij}, w_{ij})\right\}_{j=1}^{t_i}$, where $\boldsymbol{\phi}_{ij} \in \mathbb{R}^d$ is the j-th concept in pattern i, $w_{ij} \in [0, 1]$ is the weight and $t_i \in \mathbb{N}$ is the concept number for set i. To train the classifiers, we firstly map each image (or called a bag) $B_i = \{\mathbf{x}_{ij}\}_{j=1}^{|B_i|}$ into a concept space whose dimension is $\mathrm{V} = \sum_{i=1}^{\gamma} t_i$. Specifically, by concatenating all the patterns and reseting indexes, we get $\mathbb{M}' = \mathbb{M}_1 \cup \mathbb{M}_2 \cup \ldots \cup \mathbb{M}_\gamma = \{(\boldsymbol{\phi}_i, w_i)\}_{i=1}^{\mathrm{V}}$. For each concept $\boldsymbol{\phi}_i$ in $\mathbb{M}'$, we define the distance between $\boldsymbol{\phi}_i$ and an image B_k as

$$d_{ki} = \min_{\mathbf{x} \in B_k} \|\boldsymbol{\phi}_i - \mathbf{x}\|, \tag{5.23}$$

where $\|\cdot\|$ is 1 or 2-norm. Hence, any image B_i can be represented as a feature vector

$$\mathbf{z}_i = \left(\frac{d_{i1}}{w_1 + \epsilon}, \frac{d_{i2}}{w_2 + \epsilon}, \ldots, \frac{d_{iV}}{w_V + \epsilon}\right)^T, \tag{5.24}$$

where ϵ is a smoothing constant which is set to the minimum floating-point number. For multiple classification, we adopt a Multi-Class Support Vector Machine (SVM) described in [105]. For a training set $\{(\mathbf{z}_1, y_1)...(\mathbf{z}_n, y_n)\}$ with labels $y_i \in [1..\gamma]$, Multi-Class SVM finds the solution of the following optimisation problem during training.

$$\begin{aligned}
&\min_{\mathbf{w}_i, \zeta_i} f(\mathbf{w}_1, \ldots, \mathbf{w}_\gamma) = \sum_{i=1}^{\gamma} \|\mathbf{w}_i\|^2 + \frac{C}{n}\sum_{i=1}^{n} \zeta_i \\
&\text{s.t. } \forall y \in [1..\gamma] : \mathbf{w}_{y_i}^T \mathbf{z}_1 \geq \mathbf{w}_y^T \mathbf{z}_1 + 100 \cdot \Delta(y_1, y) - \zeta_1 \\
&\qquad \vdots \\
&\qquad \forall y \in [1..\gamma] : \mathbf{w}_{y_n}^T \mathbf{z}_n \geq \mathbf{w}_y^T \mathbf{z}_n + 100 \cdot \Delta(y_n, y) - \zeta_n.
\end{aligned} \tag{5.25}$$

Here, C is the usual regularisation parameter that trades off margin size and training error. $\Delta(y_i, y)$ is the loss function that returns 0 if $y_i = y$, and 1 otherwise. Here, there are γ linear classifiers being learned in parallel. Given the testing example $\mathbf{z}_i$, the decision value of classifier k is calculated by $\mathbf{w}_k^T \mathbf{z}_i$, and $\mathbf{z}_i$ is assigned label k if the decision value returned by the $k-th$ classifier is the largest among those returned decision values of all the classifiers.

5.2 Experiments

Firstly, we qualitatively analyse the supervised mean shift, which is a key component of MIL-SKDE, on synthetic data. Secondly, to evaluate the proposed MIL-SKDE, we apply it to two typical but challenging MIL applications, region-based image categorisation and object categorisation, based on the publicly available benchmark datasets. The results are compared with other state-of-the-art approaches. The source codes of some previous MIL approaches are kindly provided online by researchers in this area. The downloaded packages include mi/MI SVM [60][1], DD-SVM [59][2] and MILES [46][3]. We implement MILIS [62] and the proposed MIL-SKDE in Visual C++ with OpenCV library[4] and MOSEK optimisation package[5].

For the sake of clarity, we list some MIL terms and describe their counterparts in the experiments as follows.

1. A *bag* is a single image in the data sets.
2. An *instance* is a small patch in an image.
3. A *concept* is the representation of a region (patch) of interest that really represents the features of a category.
4. A *pattern* is a set of pairs. Each pair contains one concept and its weight. All the concepts within a pattern together depict the essence of a category and are used to map images into a new feature space where the classifiers are trained using multi-class SVM (as described in Section 5.1.5) or MedLDA (as described

[1]Downloaded from http://www.cs.cmu.edu/~juny/MILL/index.html
[2]Downloaded from http://www.cs.olemiss.edu/~ychen/ddsvm.html
[3]Downloaded from http://cs.olemiss.edu/~ychen/MILES.html
[4]OpenCV library http://opencv.willowgarage.com/wiki/
[5]For more information visit http://www.mosek.com/index.php

in Section 2.2.5.3).

5.2.1 Experiments on synthetic data

The experiments on synthetic data qualitatively analyse the supervised mean shift algorithm by showing how the negative points affect the local modes. Furthermore, we will show that multiple concepts can be found using our approach without the prior knowledge of concept number, and this can hardly be achieved by the typical Diverse Density methods.

The negative points represent the instances in negative bags, so the local modes should be kept far away from negative points but close to the high-density of positive areas. Firstly, we apply a simple scenario by generating two multivariate normal distributions in a feature space. $X^+ \sim \mathcal{N}(\mu^+, \Sigma^+)$ and $X^- \sim \mathcal{N}(\mu^-, \Sigma^-)$ are positive and negative points with distributions respectively. The covariance matrices are $\Sigma^+ = \sigma_+^2 \boldsymbol{I}$ and $\Sigma^- = \sigma_-^2 \boldsymbol{I}$.

For each normal distribution, we generate 1000 points. This simulation can be seen as an example with only one concept in MIL. If we remove all the negative points, then the situation becomes a typical mean shift clustering.

5.2.1.1 Mixture of positive and negative points

Firstly, we study the influence on confidences (weights) of local modes with respect to the mixing degree of positive and negative points. We mix positive points $X^+ \sim \mathcal{N}(\mu^+, \Sigma^+)$ and negative points $X^- \sim \mathcal{N}(\mu^-, \Sigma^-)$ gradually by tuning the values of $\|\mu^+ - \mu^-\|$ and σ_-, then observe the concept confidences $\hat{f}_{dde}(\mathbf{x})$ by using (5.12). We also evaluate the final local mode displacement by using norm $\|\mathbf{x} - \mu^+\|$. Figure 5.4 and Figure 5.5 illustrate our results.

5.2.1.2 Unbalance of positive and negative points

Here, we investigate how the data unbalance between positive and negative points affects the local mode. Suppose two normal distributions with the same mean, $X_{N^+}^+, X_{N^-}^- \sim \mathcal{N}(\mu, \Sigma)$. N^+ and N^- denote the positive and negative point numbers respectively. By tuning N^+ and N^-, we get the $\hat{f}(\mathbf{x})$ (using (5.12)) illustrated in Figure 5.6.

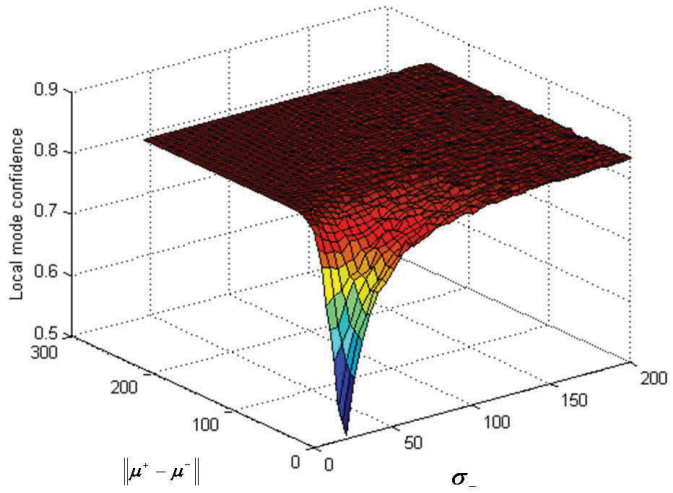


Figure 5.4: Local mode confidence with respect to the mixing degree of positive and negative points. When $\|\mu^+ - \mu^-\|$ increases, the negative points are gradually going away from the positive points, and the confidence of local mode increases. When σ_- increases, the negative points become sparser, and the confidence of local mode increases.

5.2.1.3 Multiple concepts learning

Finally, we consider the situation of seeking multiple concepts. As a simulation example, we generate three positive normal distributions: $X_i^+ \sim \mathcal{N}(\mu_i, \Sigma^+), i = 1, 2, 3$, where every μ_i is far away from the others, so that these concepts are quite different. We also generate the negative distribution $X^- \sim \mathcal{N}(\mu_3, \Sigma^-)$ with the same mean as X_3^+. Starting with the points selected by (5.22) and after convergence, we get the top three local modes and their corresponding confidences $\hat{f}(\mathbf{x}) \in [0, 1]$ as illustrated in Table 5.1.

	Concept 1	Concept 2	Concept 3
Confidence	0.799	0.801	0.498

Table 5.1: Learning for multiple concepts. The confidence of concept 3 is lower than the others because there are many negative points close to it.

From Table 5.1, we can see that the surrounding negative points of a concept (i.e., concept 3) significantly affect the confidence of the concept.

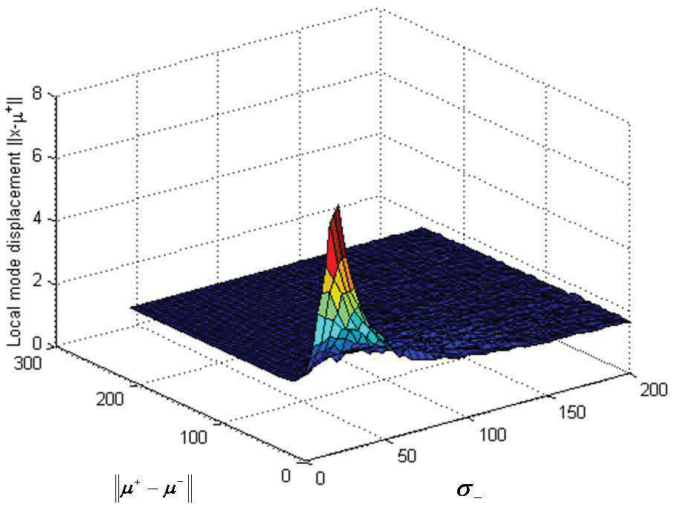


Figure 5.5: Local mode displacement. If there are no negative points, the local mode of a normal distribution should be the mean ($\mathbf{x} \approx \mu^+$) using a proper bandwidth. With a bigger mixing degree (smaller $\|\mu^+ - \mu^-\|$), the local mode $\mathbf{x}$ will be "pushed" more away from μ^+ (i.e., $\|\mathbf{x} - \mu^+\|$ is bigger). Also, the higher density of negative points (i.e, a smaller σ_-) will displace the $\mathbf{x}$ more.

5.2.2 Region-based image categorisation

The purpose of image categorisation is to assign images into predefined categories. Because the background noise and intra-class variability, the imagery features of a category could only be defined by using a part of local regions. This region-based image categorisation has been naturally formulated as a MIL problem.

5.2.2.1 Experiment setup

We test our method on the COREL dataset and compare the performance with other MIL approaches. COREL is a benchmark dataset for region-based image categorization, e.g., it has been used in [46] and [62]. Using such dataset can provide convenience to compare our method with other state-of-the-art approaches as the same dataset is used. The dataset contains 20 different categories with 100 images in each category. Each image is segmented into several local regions and features are extracted from each region. The dataset and extracted features

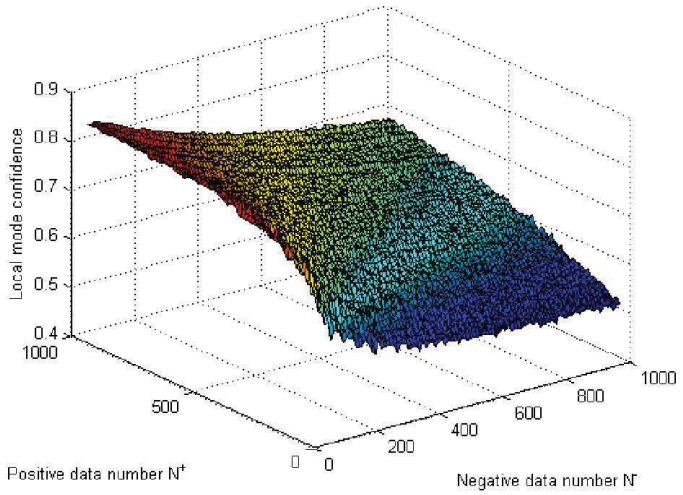


Figure 5.6: Unbalance of data points. In a local mixture area, with increasement of negative points, the local mode confidence will decrease. On the contrary, more positive points will make the local mode become more confident.

are from *http://www.cs.olemiss.edu/∼ychen/ddsvm.html*. For detailed information of image segmentation and feature extraction, interested readers are referred to [59] and [46]. Figure 5.7 shows some image samples from the COREL image database for region-based image categorisation.

Our tests are for the 10-category and the 20-category categorisations. Firstly, the first 10 categories are selected for training and testing, then all 20 categories are used. For each category, we randomly select 50% of images of the category as positive training data and the remaining 50% of images as testing data. The negative training data for the category pattern learning are all the other categories. Training and testing are repeated for five random partitions and the average results are reported. After the supervised mean shift to get the class patterns, the multi-class SVM classifiers are trained according to the description in Section 5.1.5.

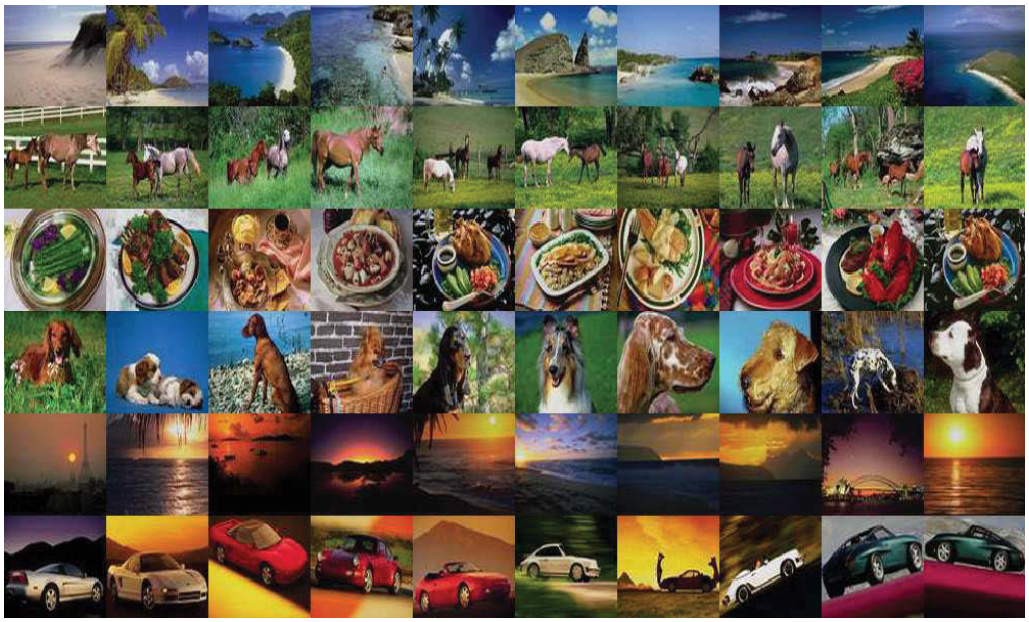


Figure 5.7: Image samples from the COREL image database for region-based image categorisation.

5.2.2.2 Image categorisation results

The average classification accuracy rates over five random partitions with the corresponding 95% confidence intervals are listed in Table 5.2, with the comparison with results in MILES [46] and MILIS [62], under the same experimental enviornments.

Table 5.2 indicates that the proposed MIL-SKDE is very competitive for this categorisation task compared with other approaches. MIL-SKDE outperforms the other methods in both 10-category and 20-category categorisation. Unlike other MIL approaches, the classification of our method is based on not only the learned concepts but also the weights of concepts. By taking into account the weights of concepts, the class features can be acquired more distinctively which helps to achieve better classification performance.

Algorithms	10-category set	20-category set
MIL-SKDE	**84.2:[83.3,85.1]**	**73.5:[72.1, 74.9]**
MILIS [62]	83.8:[82.5,85.1]	70.1:[68.5,71.8]
MILIS_{L1} [62]	82.5:[80.8,84.2]	67.4:[65.3,69.4]
MILES [46]	82.6:[81.4,83.7]	68.7:[67.3,70.1]
DD-SVM [59]	81.5:[78.5,84.5]	67.5:[66.1,68.9]
MI-SVM [60]	74.7:[74.1,75.3]	54.6:[53.1,56.1]
mi-SVM [60]	76.4:[75.3,775]	53.7:[52.2,55.2]

Table 5.2: Comparison of image categorisation accuracy rates (in %) for MIL-SKDE and other methods. The values in each pair of brackets are the 95 percent confidence interval of the categorisation accuracy on the experiments for five different random partitions. The confidence intervals are computed on the assumption that the experimental results for five random partitions follow a Gaussian distribution. The middle values of the confidence intervals are treated as the average accuracy rates.

5.2.2.3 Sensitivity to labeling noise

Now, we test and compare our method with other alternatives in terms of the sensitivity to labeling noise. Labeling noise can be seen as the likelihood that an image is mislabeled. The robustness to labeling noise is very important to classifiers because obtaining "pure" training data is often difficult because manual label process is often subjective.

We perform the tests using binary classification. Category 2 (Historical buildings) and Category 7 (Horses) are selected and treated as positive and negative training sets respectively. Both categories are distinctive and well classified in our image categorisation, and they are good datasets for testing influences of labeling noise. The training and testing data are randomly split by 50% and 50% in each category. To generate the "labeling noise", we firstly select x% (up to 20%) of training images from each of these two categories, then negate the original labels of the selected images, i.e., changing positive (negative) to negative (positive), and deeming those 'mislabeled' images as labeling noise. On each level of labeling noise (0-20% with step size 2%), 5 rounds of classification are conducted. The average classification accuracies using MIL-SKDE and other methods are presented

in Figure 5.8.

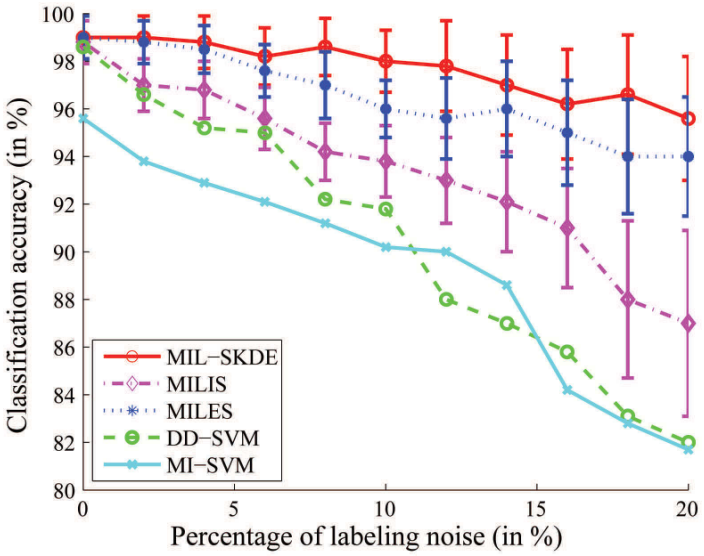


Figure 5.8: Comparison of sensitivity to labeling noise among MIL-SKDE, MILIS [62], MILES [46], DD-SVM [59] and MI-SVM [60] in terms of binary classification ("Historical buildings" and "Horses"). The average classification accuracies are computed over five randomly generated datasets. For the best three approaches, i.e., MIL-SKDE, MILES and MILIS, the corresponding 95% confidence intervals are also given.

Figure 5.8 shows that the MIL-SKDE is robust to labeling noise and it beats all the other methods in the experiments. MIL-SKDE has the highest classification accuracy too. MILES is the second best. MILIS' performance drops faster compared with MILES because it has much less instance prototypes and the labeling noise affects the prototypes selection significantly. DD-SVM decreases very quickly because it obtains concepts by diverse density which is very sensitive to mislabeled instances. Finally, MI-SVM does not perform well because the labeling noise causes large penalty in the optimisation of MI-SVM.

5.2.3 Object categorisation

The object categorisation (or object category recognition as called in some literatures) is to determine whether or not an image contains a certain class of objects.

Specially, we treat object categorisation as a task that classifies images according to which objects these images contain. Due to the high intra-class variability, the global appearance of the objects within the same category may be quite different. It demands that the category patterns should be representative to the objects that belong to the same category and be flexible enough to accommodate intra-class variability. An intuitive way is that one category is modeled as collection of salient parts (i.e., representative regions) and these salient parts are used for classifier training. The MIL is a natural framework to learn category patterns without explicitly labeling the local parts in training samples and its applications on category recognition has been widely investigated as shown in [44][62][45][52][46][106].

5.2.3.1 Experimental setup

We test MIL-SKDE on two well-known benchmark datasets, Caltech-4 [107] and SIVAL (obtained from *http://www.cse.wustl.edu/accio/*). The two datasets are popular for object categorization. Hence, the performance of our methods can be compared fairly with other methods. Caltech-4 contains 5 categories, i.e., "face", "motorbike", "airplane", "car" and "background", with image numbers ranging from around 450 to 1050 in each category. Figure 5.9 shows some image samples from the Caltech-4 dataset for object categorisation. The other group of training data is SIVAL dataset which includes 25 categories with 60 images in each category and Figure 5.10 illustrates some image samples from the SIVAL dataset. The objects of interest within the images of both datasets have not been cropped out and aligned, and backgrounds are cluttered. All categories in Caltech-4 and five categories in SIVAL are chosen for experiments. The categories chosen from SIVAL are "ajax detergent", "apple", "coke can", "data mining book" and "running shoes". We repeat our experiments for five rounds. At each round, 50% images in each category are randomly selected as positive samples, and the rest 50% in the category are used for testing. The negative data are all the other categories in the same dataset as training data. The average results of the five rounds are reported.

In our experiments, each image is treated as a MIL bag, and SIFT feature

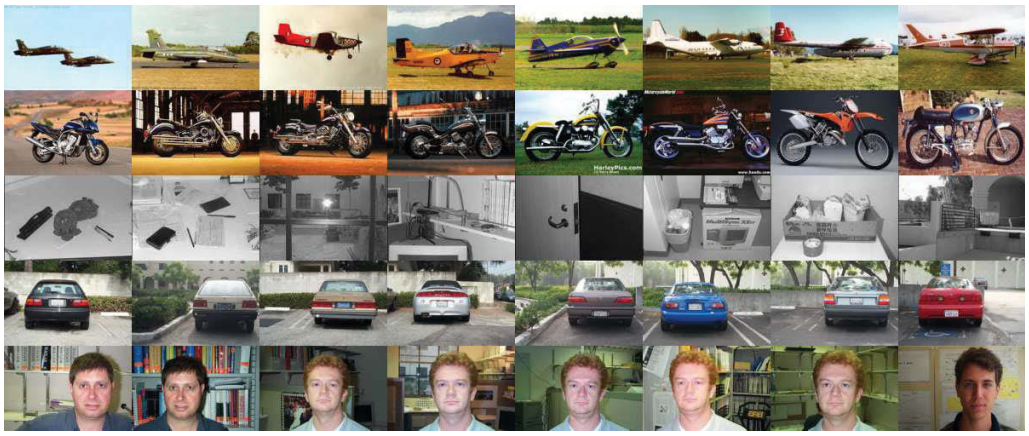
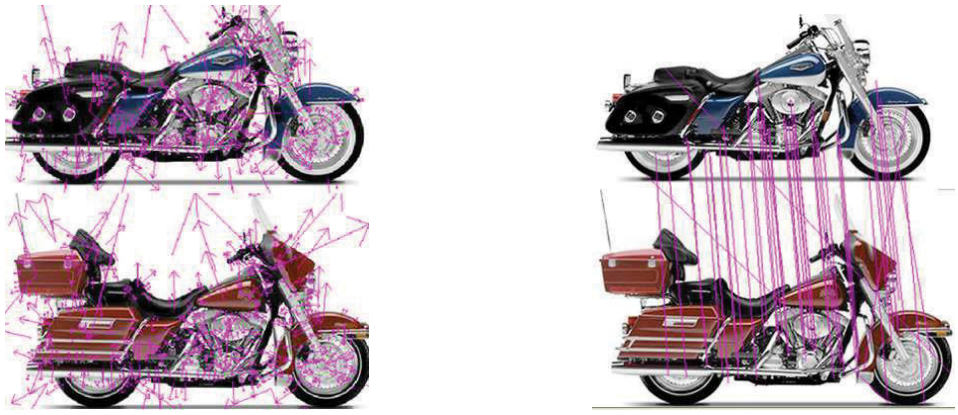


Figure 5.9: Some image samples from the Caltech-4 dataset for object categorisation.

Figure 5.10: Some image samples from the SIVAL dataset.

points [55] detected from the image are the instances inside the bag. The SIFT feature points are represented as 128-dimensional descriptors. Those SIFT descriptors are robust to scale and rotation changes and are good salient part candidates. Our task is to find those true salient parts to represent the object category. Figure 5.11 is an example to match similar regions between two objects using Rob Hess's SIFT implementation with RANSAC[1] [108]. Such phenomenon suggests that the feature vectors of similar instances will be close to each other and the clusters will be formed in the feature space using SIFT. To avoid that too many SIFT points are detected, we resize the widths of all images to 225 pixels and keep the original ratios between widths and heights. SKDE adopts the fixed bandwidth and the bandwidth h is estimated by Algorithm 3 as well. Figure 5.12 illustrates the bandwidth calculation on SIVAL dataset using Algorithm 3.



(a) SIFT points detection (b) Matches of SIFT points

Figure 5.11: An example of instances detection and matching using SIFT on Caltech-4 [107]. (a) shows detected feature points (instances) using SIFT, and (b) shows the robustness of SIFT features to match similar regions between two different objects belonging to the same category.

5.2.3.2 Recognition results

The performance of our method is compared with DD-SVM [59], MILES [46] and MILIS [62], these three methods have been reported to have outperformed

[1]Codes are downloaded from http://blogs.oregonstate.edu/hess/code/sift/

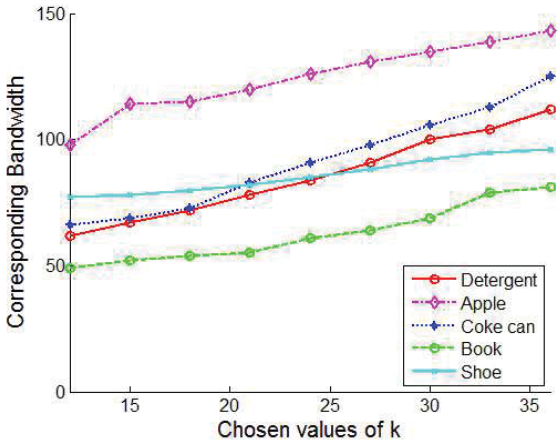


Figure 5.12: A figure shows bandwidth calculation on SIVAL dataset using Algorithm 3. The horizontal axis represents the differently chosen values of k and the vertical axis represents the corresponding bandwidths. There are five categories, "ajax detergent", "apple", "coke can", "data mining book" and "running shoes", included. Here, the optimal value of k is 30 which equals to the number of positive samples.

other existing MIL methods for object categorisation. The parameter selections of DD-SVM, MILES and MILIS have been followed by the original publications respectively. We repeat our experiments for five rounds. At each round, 50% images in each category are randomly selected as positive samples, and the rest 50% in the category are used for testing. The negative training data are the rest categories in the same dataset. The recognition is a multiclass classification and a confusion matrix is constructed for each round. Table 5.3 shows the Confusion Matrices with average values over five rounds on Caltech-4 and SIVAL data set respectively using MIL-SKDE.

For comparison, we report the averages of the diagonals in the confusion matrices and the corresponding 95% confidence intervals computed over five rounds of experiments for the proposed method, DD-SVM, MILES and MILIS in Table 5.4.

Table 5.4 shows that the performance of our method is comparable with the-state-of-the-art approaches in both challenging datasets. Overall, our method has the best performances in experiments. The increase of recognition accuracy

	Face	Motor.	Air.	Car
Face	**83.9**	2.3	4.7	9.1
Motorbike	2.6	**81.7**	10.6	5.1
Airplane	6.4	9.9	**76.1**	7.6
Car	2.8	12.9	11.2	**73.1**

(a) Confusion matrix on Caltech-4 data set (in %).

	Detergent	Apple	Coke C.	R. shoe	Book
Detergent	**82.7**	4.9	4.9	5.0	2.5
Apple	4.1	**79.6**	2.1	8.2	6.0
Coke can	3.7	3.7	**90.7**	1.9	0
R. shoe	6.0	8.1	2.0	**81.9**	2.0
Book	6.2	4.1	0	2.1	**87.6**

(b) Confusion matrix on SIVAL data set (in %).

Table 5.3: Confusion matrices of object categorisation on Caltech-4 and SIVAL dataset respectively using MIL-SKDE. Each row of a table lists the average percentages (over five rounds) of test images in one category classified to different categories. The diagonal shows the recognition accuracies. The optimum would be 100% only on the diagonal and zero elsewhere.

benefits from the better concept learning of MIL-SKDE which is actually a feature selection processing. Each of the concepts represents a truly distinctive salient part of a category and the weight of a concept indicates the confidence of a salient part.

Finally, we would like to note that in this thesis, the issues of visual categorisation are tackled under an assumption that each image from a dataset only contains one object. Such assumption is a common setting for many popular benchmark datasets, e.g., Corel, Caltech-4 and SIVAL. Therefore, it is handy to compare our work with other state-of-the-art approaches. Recognising image containing multiple objects is beyond the scope of our work.

5.2.3.3 Feature selection

We utilise the SIFT descriptor to build the instances of each bag (image). Basically, there are a number of instances in a bag but only a tiny fraction of instances (called concepts) are the representative features (salient parts) of target object

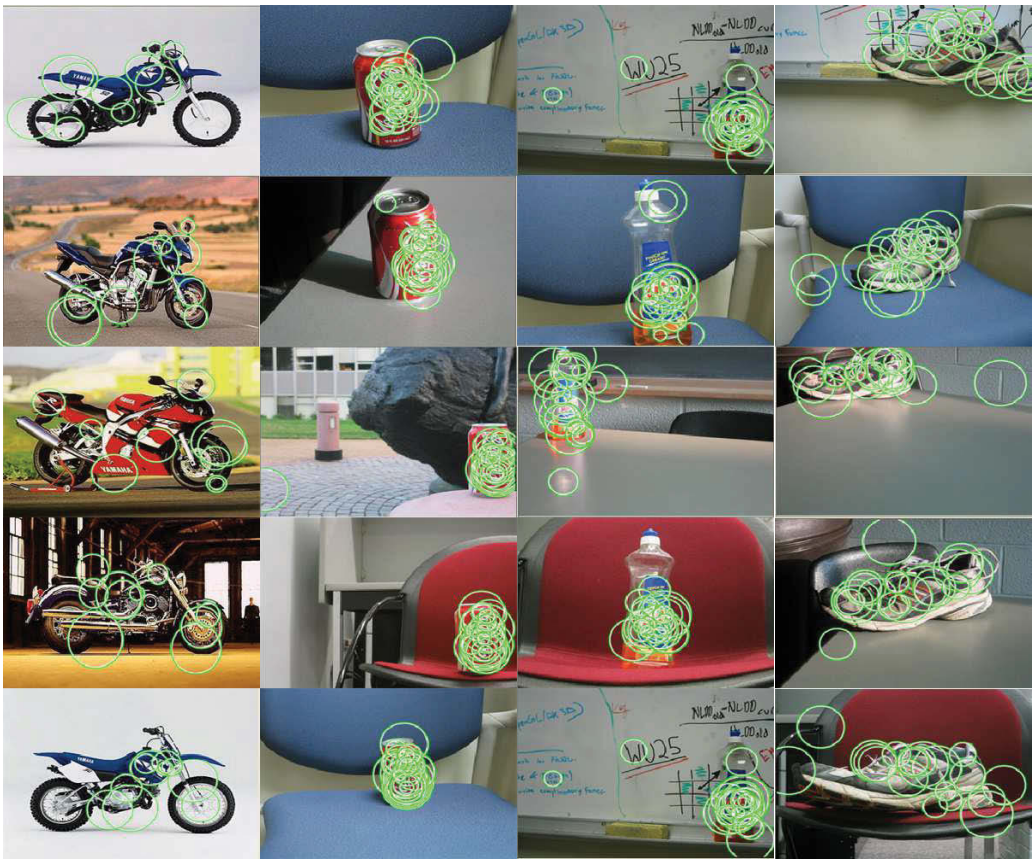
Algorithms	Caltech-4 (in %)	SIVAL (in %)
MIL-SKDE	**78.7 : [77.6, 79.8]**	**84.5 : [82.7, 86.3]**
DD-SVM [59]	72.3 : [71.0, 73.6]	77.5 : [76.1, 78.9]
MILIS [62]	74.2 : [73.0, 75.4]	81.2 : [79.1, 83.3]
MILES [46]	75.1 : [73.9, 76.3]	82.1 : [80.6, 83.6]

Table 5.4: The recognition rates for MIL-SKDE, DD-SVM, MILIS and MILES. The middle values of the 95% confidence intervals (in brackets) over five rounds of experiments are treated as the recognition rates. During each round, the means of the diagonal values of confusion matrices are computed. To compute the confidence intervals, we assume that those means of diagonal values from different rounds follow a Gaussian distribution.

category. Figure 5.13 shows that the stable features (salient parts) of corresponding object categories are correctly selected via obtaining the *patterns*. We choose "Motorbike" from Caltech-4, and "Coke can" and "Detergent" from SIVAL as the examples to show the salient parts. Given a *pattern* $\mathbb{M} = \{(\boldsymbol{\phi}_i, w_i)\}_{i=1}^{t}$, we circle out the regions that correspond to the nearest neighbors of $\boldsymbol{\phi}_i$'s in some correctly recognised images. The radius of each circle indicates the weight w_i. Those circle areas illustrate the salient parts of the categories. We can see that the majority of salient parts concentrate on the objects of interest. There are also a small fraction of circles that are outside the target objects as some background regions are also similar to the 'salient parts' of target category. However, those "off-target" parts will not affect the recognition results much because as we can see that both the quantity and the weights of the missing circles are small.

5.3 Discussion

We have presented a novel MIL algorithm, multiple-instance learning with supervised kernel density estimation (MIL-SKDE), and showed its nice performances in the experiments. However, our approach is based on an extension of mean shift called supervised mean shift, and the mean shift is often expensive for a large-scale data set. Here, we give a brief discussion about the computational complexity about the supervised mean shift. Given the data set $\mathcal{D} = \{\langle B_i, y_i \rangle\}_{i=1}^{m}$, where m



(a) Motorbike (b) Coke can (c) Detergent (d) Running shoe

Figure 5.13: Some sample images that are recognised as containing a target object. The circles in images represent the nearest neighbors of $\boldsymbol{\phi}_i$ in the corresponding pattern $\mathbb{M} = \{(\boldsymbol{\phi}_i, w_i)\}_{i=1}^{t}$. The radius of a circle indicates the weight w_i. Since the majority of circles concentrate on the objects of interest, the interference of background clutters has been suppressed.

is the number of MIL bags, the bag $B_i = \{\mathbf{x}_{i,j}\}_{j=1}^{|B_i|}$, $|B_i|$ is the size of B_i and the label $y_i \in \{0, 1\}$. The complexity of our algorithm is $O(RST)$, where R is the number of starting points for supervised mean shift, $S = \sum_{i=1}^{m} |B_i|$ is the total number of instances and T is the number of iterations until convergence. One can see that the supervised mean shift has the same complexity as the conventional mean shift. So the use of such an approach may be hampered in data set with large-scale training samples. In our work, we tackle this issue as the following ways:

- Our supervised mean shift algorithm has superlinear convergence, which is fast for convergence. Therefore, T is not a major concern here. Only R (the number of starting instances) and S (the total number of instances) are related to the number of training samples.

- To reduce R, we develop a criterion to select starting points described in Section 4.1. In terms of the nature of our application, the selecting criterion can greatly reduce the number of starting points. For instance, without the selecting criterion, $R = S$. However, after starting point selection, $R \approx S/10$ in our experiments.

- To reduce S, we perform a preprocessing operation on the data set described in Section 3 (the second paragraph). In the preprocessing operation, we cluster the instances $\{\mathbf{x}_{i,j}\}_{j=1}^{|B_i|}$ of each B_i separately using mean shift, and replace the instances of each bag B_i with the cluster centers, then resulted concise bag corresponding to B_i is denoted by B_i' ($|B_i'| \leq |B_i|$). This clustering operation merges the similar instances together. It does not change the nature of MIL at all but saves much computation later on by reducing bag sizes from $|B_i|$ to $|B_i'|$. After preprocessing, $S = \sum_{i=1}^{m} |B_i'| \leq \sum_{i=1}^{m} |B_i|$. In our experiments, S is reduced by 50%.

So far, we have addressed the complexity issue by reducing the scale of final training data for our algorithm. However, conducting experiments on the whole SIVAL dataset using MIL-SKDE is still expensive. To get results within a reasonable period of time, we only chose part of the SIVAL dataset (5 out of all the 25 categories) for experiments. In the next chapter, we will further tackle the

complexity issue and propose a speed-up version of MIL-SKDE which uses an approximation method for mode seeking.

5.4 Conclusion

In this chapter, we have presented a novel MIL algorithm based on proposed SKDE and supervised mean shift (MIL-SKDE). Our method has extended the feature space clustering in which the dataset is divided into positive and negative points. The experiments on real-world datasets demonstrated that our approach has superior performances over state-of-the-art alternatives.

Chapter 6

MIL-SS: Multiple-instance learning with a speed-up supervised kernel density estimation

In previous chapter, we have presented a novel multiple-instance learning (MIL) algorithm, multiple-instance learning with supervised kernel density estimation (MIL-SKDE), and showed its excellent performances in experiments of visual categorisation. The MIL-SKDE utilises a supervised mean shift clustering (described in Section 5.1.3 of Chapter 5) to learn the visual concepts and those concepts are used for later classification. Quickly and precisely learning those visual concepts are critical for our MIL algorithm. However, facing a large-scale dataset, i.e., a huge number of data points in the feature space, the supervised mean shift clustering is often time-consuming for convergence, especially when the training data are dominated by negative samples. In MIL-SKDE algorithm, we have preliminarily addressed the complexity issue. In this chapter, we further tackle reducing the computational complexity in concept learning and propose the Multiple-instance Learning with a Speed-up SKDE (MIL-SS) which learns concepts by approximately locating the multiple modes of SKDE function. The new MIL approach can significantly speed up the training process with little

accuracy sacrifice.

6.1 MIL-SS algorithm

6.1.1 Revisit supervised kernel density estimation and mode seeking

As described in Section 5.1.2, given the preprocessed dataset $\mathcal{D}' = \{\langle \mathbf{x}_i, y_i \rangle\}_{i=1}^{n}$, where each $\mathbf{x}_i \in \mathbb{R}^d$ is an instance, i.e., $\mathbf{x}_i$ represents a point in the feature space, $y_i \in \{0, 1\}$ is a label and n is the total number of training instances, the supervised kernel density estimator (SKDE) can be formulated as:

$$\hat{f}(\mathbf{x}) = \frac{c}{nh^d} \sum_{i=1}^{n} (-1)^{1-y_i} \, k\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right) - \frac{1}{n} \sum_{i=1}^{n} y_i + 1, \tag{6.1}$$

where h is a pre-defined bandwidth and c is a pre-defined constant (e.g., $c = 1$). Since we are interested in those modes (local maxima) of the above density function (6.1), a supervised mean shift process has been proposed to locate those modes from some selected initial points $\mathbf{x}$'s by iteratively setting $\mathbf{x} = \bar{\mathbf{x}}$ until converge, where $\bar{\mathbf{x}}$ is calculated by

$$\bar{\mathbf{x}} = \frac{\sum_{i=1}^{n} \left[\mathbf{x}_i^{y_i} \, (2\mathbf{x} - \mathbf{x}_i)^{1-y_i} \, g\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right) \right]}{\sum_{i=1}^{n} g\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right)}. \tag{6.2}$$

Supervised mean shift iteration (6.2) can precisely locate modes of SKDE function (6.1). However, precisely locating SKDE modes is often time-consuming as even a medium dataset could lead to a huge instance space, i.e., the value of n is large in (6.2), which causes each iteration of supervised mean shift is expensive. Our intention is to replace n of Equation 6.1 with a new n^* and get a new density estimator:

$$\hat{f}^*(\mathbf{x}) = \frac{c}{n^* h^d} \sum_{i=1}^{n^*} (-1)^{1-y_i} \, k\left(\left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2 \right) - \frac{1}{n^*} \sum_{i=1}^{n^*} y_i + 1, \tag{6.3}$$

such that $\hat{f}^*(\mathbf{x})$ has the similar modes with original $\hat{f}(\mathbf{x})$ but with a lot fewer data points, i.e., $n^* << n$, so the supervised mean shift iteration has much less complexity, i.e.,

$$\bar{\mathbf{x}} = \frac{\sum_{i=1}^{n^*} \left[\mathbf{x}_i^{y_i} \left(2\mathbf{x} - \mathbf{x}_i\right)^{1-y_i} g\left(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{h} \right\|^2 \right) \right]}{\sum_{i=1}^{n^*} g\left(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{h} \right\|^2 \right)}. \tag{6.4}$$

To reduce the data scale without significantly affecting the important modes, we need to observe where those modes occur in the feature space. Here, we revisit Figure 5.1 in Chapter 5 for an analysis of mode positions in the feature space. Note that in the feature space, the data points are divided into positive and negative sets.

Figure 5.1 is a simulation of mode positions in a feature space. Differing from conventional feature space analysis, the data points in the feature space of our case are divided into positive and negative sets. In the above image, the red squares and solid blue dots represent positive and negative data respectively. One can realise that in the areas like Area 1 and Area 2 (in red circles), the value of $\hat{f}(\mathbf{x})$ (6.1) will be small as negative points dominate those areas. This is easy to be testified by (6.1) as the most terms $k\left(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{h} \right\|^2 \right) \geq 0$ with high values have -1 coefficients. In contrast, the local maxima (modes) of SKDE (6.1) will only occur in the high-dense positive areas like concept areas, e.g., the area in the dark circle.

In practical MIL applications, the most of the instances are actually negative which form minus terms of (6.1), and the mode (local maxima) are not likely to occur in the area packed with negative instance points. This phenomenon can be illustrated using Figure 5.1. The mode does not occur in Areas 1 and 2 because there are too many negative points there. According to (6.1), any point $\mathbf{x}$ in Areas 1 and 2 will lead to a low probability value. Since we are looking for the modes (maxima) of (6.1), removing all the points (both negative and positive) located in such areas will not significantly affect the modes. Since most instances of MIL are negative or false positive (e.g., positive points in Area 1 of Figure 5.1), most of data points are actually unrelated for mode seeking. Here, we devise an *Instance Selection Process* to remove unrelated instances.

6.1.2 Instance selection process

In our MIL algorithm, the complexity of supervised mean shift for seeking modes of SKDE largely depends on the total number of instances. To improve the efficiency, here we come up with a criterion to remove false positive instances as follows:

$$f_{fp}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}(-1)^{1-y_i}\, k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{\alpha^{1-y_i}h}\right\|^2\right) < \beta, \tag{6.5}$$

where α is a parameter controlling how much the negative points affect the instance selection and is usually equal to 1, and β is a predefined threshold and is usually set to 0. By explicitly denoting positive and negative instances, i.e., $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n} = \left\{\left(\mathbf{x}_i^+, y_i = 1\right)\right\}_{i=1}^{n^+} \bigcup \left\{\left(\mathbf{x}_i^-, y_i = 0\right)\right\}_{i=1}^{n^-}, n = n^+ + n^-$. The instance selection process contains two steps:

1. Check all the positive instances $\mathbf{x}_i^+$ using criterion (6.5). If $\mathbf{x}_i^+$ suffices (6.5), i.e., if $f_{fp}\left(\mathbf{x}_i^+\right) < \beta$, then such $\mathbf{x}_i^+$ will be removed from the training set.

2. After checking all the positive instances and removing those false positive data, further remove all the negative points $\left\{\mathbf{x}_i^-\right\}_{i=1}^{n^-}$.

Instance selection using criterion (6.5) is able to remove those most likely false positive points because if $\mathbf{x}$ is a false positive point in the feature space, $\mathbf{x}$ will be either in a dense area of negative points or far away from any dense areas of positive points. When $\mathbf{x}$ is in a dense area of negative points, its close neighbors are mostly negative ($y_i = 0$), so that the terms $(-1)^{1-y_i}\, k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{\alpha^{1-y_i}h}\right\|^2\right)$ with high values are negative numbers and $f_{fp}(\mathbf{x})$ should be smaller than a threshold β. It means (6.5) holds and $\mathbf{x}$ will be removed. Likewise, if $\mathbf{x}$ is far away from any dense areas of positive points ($y_i = 1$), then the positive terms $(-1)^{1-y_i}\, k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{\alpha^{1-y_i}h}\right\|^2\right)$ have low values, so $f_{fp}(\mathbf{x})$ should be small and hence be removed. Only those likely true positive points which are in the dense areas of positive points will have big $f_{fp}(\mathbf{x})$ values.

As we know from Section 5.1.2 of Chapter 5, the SKDE formula (6.1) has an equivalent form by explicitly separating positive and negative instances as follows.

$$\hat{f}(\mathbf{x}) = \frac{c}{nh^d}\sum_{i=1}^{n^+} k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i^+}{h}\right\|^2\right) - \frac{c}{nh^d}\sum_{i=1}^{n^-} k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i^-}{h}\right\|^2\right) + \frac{n^-}{n}. \tag{6.6}$$

By applying instance selection, n^+ reduces greatly and $n^- = 0$ in (6.6), then SKDE becomes a standard KDE (5.1) but those important modes of SKDE are maintained. For the simulated feature space shown in Figure 5.1, after applying instance selection process, the training data will be reduced as illustrated in Figure 6.1. Now, since all the data points are positive, the supervised mean shift will be reduced to a conventional mean shift. Therefore, the mean shift can be applied to locate the modes on a much smaller dataset after instance selection.

Instance Selection Process is a key step to decrease runtime by reducing experimental data scales. How much runtime can be saved is largely dependent on datasets. Basically, the higher ratio that negative instances take in a dataset, the more runtime will be saved.

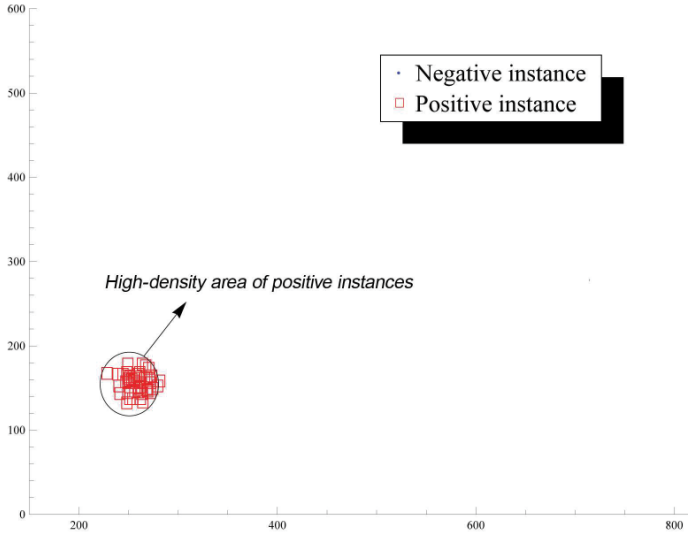


Figure 6.1: Remained training data after instance selection process. Compared to Figure 5.1, only those instances staying in the high-density areas of positive instances are remained, and all the other negative and false positive instances are removed. After instance selection process, the scale of training data is greatly reduced and the important modes areas are remained.

6.1.3 Bandwidth estimation

Bandwidth estimation is important for desirable kernel density estimation. Here, we employ the same bandwidth estimation approach shown in Section 5.1.3.2 of Chapter 5. Note that the bandwidth estimation should be performed before the instance selection process.

6.1.4 MIL-SS algorithm summary

Given the training set $\mathcal{D} = \{\langle B_i, y_i \rangle\}_{i=1}^{m}$, where bag $B_i = \{\mathbf{x}_{ij}\}_{j=1}^{|B_i|}$, with $\mathbf{x}_{ij} \in \mathbb{R}^d$, label $y_i \in \{0, 1\}$ indicates whether or not the bag contains target object. The procedures of learning modes of SKDE can be summed up as follows.

1. Pass the bag labels to the instances and reindex instances across all bags to generate the training set $D' = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $n = \sum_{i=1}^{m} |B_i|$.

2. Estimate a bandwidth h using the same approach as shown in Section 5.1.3.2 of Chapter 5.

3. Perform instance selection process (described in Section 6.1.2) and return the remaining positive points, denoted as $D'' = \{\mathbf{x}_i\}_{i=1}^{n^*}$. Empirically, $|D''| << |D'|$, i.e., $n^* << n$.

4. Apply mean shift on D'' to obtain the modes, denoted as $\boldsymbol{\phi} = \{\boldsymbol{\phi}_i\}_{i=1}^{s}$, where $\boldsymbol{\phi}_i \in \mathbb{R}^d$.

5. For each mode $\boldsymbol{\phi}_i \in \boldsymbol{\phi}$, compute corresponding weight $w_i = \hat{f}(\boldsymbol{\phi}_i)$ using (6.1), where $w_i \in [0, 1]$. Return those modes $\boldsymbol{\phi}_i' s$ whose weights $w_i' s$ are above average, i.e., those $w_i \geq \mathbb{E}[w] = \frac{1}{s} \sum_{i=1}^{s} w_i$, to keep only important modes. We call the final set $\mathbb{M} = \{(\boldsymbol{\phi}_i, w_i)\}_{i=1}^{t}$, where $w_i \geq \mathbb{E}[w]$ and $t < s$, the *pattern* of a category which is later used to map bags for classification.

For distinguishing from previous MIL-SKDE algorithm, the new speed-up version of MIL-SKDE is denoted by MIL-SS.

6.1.5 Classification

For convenient comparisons, here we also use the same classification method, i.e., a Multi-Class Support Vector Machine (SVM) [105], as shown in Section 5.1.5 of Chapter 5.

6.2 Experiments

Similar to MIL-SKDE, we evaluate the MIL-SS algorithm in the following ways. Firstly, we test our method on region-based image categorisation and compare the results against other approaches. Secondly, MIL-SS is applied for object categorisation, and we also compare the performance among MIL algorithms, as well as the efficiency between MIL-SKDE and MIL-SS. Thirdly, the robustnesses to labeling noise between our approach and other MIL methods are evaluated. Finally, we show the regions of interest that are detected correctly using MIL-SS on some categories.

The parameters for *Instance Selection*, see (6.5), are empirically chosen as $\alpha = 1.1$ and $\beta = 0$. The bandwidth h in SKDE (6.1) is chosen according to Section 6.1.3. For the sake of clarity, we also list some MIL terms and their corresponding counterparts as follows.

1. A *bag* is a single image in the datasets.

2. An *instance* is a small patch in an image. Here, we detect the SIFT points as instances and each instance is represented as a 128-d SIFT descriptor.

3. A *concept* is the representation of a region (patch) of interest that really represents the features of a category.

4. A *pattern* is a set of pairs. Each pair contains one concept and its weight. All concepts within a pattern together depict the essence of a category and are used to map images into a new feature space where the classifiers are trained using a multi-class SVM method (as described in Section 6.1.5).

6.2.1 Region-based image categorisation

The purpose of image categorisation is to assign images into predefined categories. Because the background noise and intra-class variability, the global imagery fea-

tures of a category are not reliable for classification. However, many local regions may be stable among a category. Therefore, the region-based image categorisation has been naturally formulated as a MIL problem.

We test our method on the COREL dataset and compare the performance with other MIL approaches. As stated in previous chapter, the dataset contains 20 different categories with 100 images in each category. Each image is segmented into several local regions and features are extracted from each region. The dataset and extracted features are from *http://www.cs.olemiss.edu/∼ychen/ddsvm.html*. The detailed information of image segmentation and feature extraction can be found in [59] and [46].

The same as those in last chapter, our tests are applied on the 10-category and the 20-category categorisations respectively. For each category, we randomly select 50% of images of a category as the positive training data and the remaining 50% of images as the testing data. The negative training data for the category pattern learning are all other categories. Training and testing are repeated for five random partitions and the average results are listed in Table 6.1. MIL-SS is compared with MIL-SKDE, MI/mi-SVM [60] [1], DD-SVM [59] [2], MILES [46] [3] and MILIS [62], that have been reported to have performed well for object categorisation. The parameter selections of those MIL algorithms have been followed by the original publications respectively. Table 6.1 indicates that the proposed MIL-SS is also comparative with other approaches and it just has a slight loss of performance compared with MIL-SKDE.

6.2.2 Object categorisation

Object categorisation is a typical task of computer vision which involves determining whether or not an image contains some specific category of object. Specially, we treat object categorisation as a task that classifies images according to which objects that these images contain. In this section, all of the experiments are conducted on the popular Caltech-4 [107] and SIVAL [4] datasets. Caltech-4 contains

[1]Source code - http://www.cs.cmu.edu//∼juny/MILL/index.html
[2]Source code - http://www.cs.olemiss.edu/∼ychen/ddsvm.html
[3]Source code - http://cs.olemiss.edu/∼ychen/MILES.html
[4]*http://www.cse.wustl.edu/accio/*

Algorithms	10-category set	20-category set
MIL-SS	83.1:[81.4,84.8]	71.3:[69.9, 72.7]
MIL-SKDE	**84.2:[83.3,85.1]**	**73.5:[72.1, 74.9]**
MILIS [62]	83.8:[82.5,85.1]	70.1:[68.5,71.8]
MILIS_{L1} [62]	82.5:[80.8,84.2]	67.4:[65.3,69.4]
MILES [46]	82.6:[81.4,83.7]	68.7:[67.3,70.1]
DD-SVM [59]	81.5:[78.5,84.5]	67.5:[66.1,68.9]
MI-SVM [60]	74.7:[74.1,75.3]	54.6:[53.1,56.1]
mi-SVM [60]	76.4:[75.3,775]	53.7:[52.2,55.2]

Table 6.1: Comparison of image categorisation accuracy rates (in %) for MIL-SKDE and other methods. The values in each pair of brackets are the 95 percent confidence interval of the categorisation accuracy on the experiments for five different random partitions. The middle values of the confidence intervals are treated as the average accuracy rates.

5 categories, i.e., "face", "motorbike", "airplane", "car" and "background", with image numbers ranged from around 450 to 1050 in each category. SIVAL dataset includes 25 categories with 60 images in each category. The objects of interest within the images of both datasets have not been cropped out and aligned and backgrounds are cluttered. All categories in Caltech-4 and four categories in SIVAL are chosen for experiments. The categories chosen from SIVAL are "ajax detergent", "apple", "coke can" and "running shoes". We compare our method with DD-SVM [59], MILES [46] and MILIS [62], that have been reported to have outperformed other existing MIL methods for object categorisation. We repeat our experiments for five rounds. At each round, 50% images in each category are randomly selected as positive samples, and the rest 50% in the category are used for testing. The negative training data are the rest categories in the same dataset. The categorisation is a multiclass classification and a confusion matrix is constructed for each round. The averages of the diagonals in the confusion matrices and the corresponding 95% confidence intervals computed over five rounds of experiments are listed in Table 6.2.

Table 6.2 shows that the overall performances of MIL-SS are also comparable with the-state-of-the-art approaches in both Caltech-4 and SIVAL datasets. The results show that the approximate mode locating of MIL-SS truly grasps the

Algorithms	Caltech-4 (%)	SIVAL (%)
MIL-SS	77.2 : [76.3, 78.1]	83.4 : [82.5, 84.3]
MIL-SKDE	**78.7 : [77.6, 79.8]**	**84.5 : [82.7, 86.3]**
DD-SVM [59]	72.3 : [71.0, 73.6]	77.5 : [76.1, 78.9]
MILIS [62]	74.2 : [73.0, 75.4]	82.2 : [81.1, 83.3]
MILES [46]	75.1 : [73.9, 76.3]	82.1 : [80.6, 83.6]

Table 6.2: The recognition rates for MIL-SS, MIL-SKDE, DD-SVM, MILIS and MILES. The middle values of the 95% confidence intervals (in brackets) over five rounds of experiments are treated as the recognition rates. During each round, the means of the diagonal values of confusion matrices are computed.

underlying characteristics of the visual categories. One can see all the methods perform better in SIVAL dataset. This is because the SIFT features are more stable for object categories in SIVAL dataset.

The major advantage of MIL-SS over MIL-SKDE is the efficiency. As we have stated before, MIL-SS is able to eliminate those unrelated instances for mode seeking from the training data, so the training speed will be significantly increased because the scale of training data, which is the main cause of computational complexity of MIL-SKDE, will be greatly reduced. To verify this, we record the training time of MIL-SS and MIL-SKDE for the above object categorisation experiments [1]. In Table 6.3, we show the training scales [2] for MIL-SS and MIL-SKDE training, as well as the spent time for training the patterns of all the categories of Caltech-4 and SIVAL datasets.

Table 6.3 indicates that MIL-SS significantly reduces the time expense for training the patters compared with MIL-SKDE. For training patters of Caltech-4 dataset, the data scale is reduced from 118,756 instances down to 31,753 instances and MIL-SS takes around 6% of MIL-SKDE training time. For object categorisation on SIVAL dataset, MIL-SS works even better, and it only takes around 2% of MIL-SKDE training time. This is because for object recognition, only the areas of target objects are our regions of interest (ROI), and other areas

[1]The configuration of the computer used for training is: Inter(R) Xeon(R) CPU 2.53GHz, 12.0GB RAM, 64-bit Window server 2008 R2 Enterprise.

[2]Here, the training scale means the aggregation number of training instances over all the categories in a dataset.

	Caltech-4		4 categories from SIVAL	
	Training scale	Training time	Training scale	Training time
MIL-SS	31,753	76 mins	9,753	18 mins
MIL-SKDE	118,756	1,291 mins	63,861	753 mins

Table 6.3: The average training scales and time (over five rounds) of MIL-SS and MIL-SKDE for object categorisation corresponding to Table 6.2. Training scale represents the number of total feature vectors (instances) extracted over all categories in a dataset. Training time represents the total time consumption for training all patterns in a dataset.

like backgrounds are unrelated for recognition. After instance selection process, most of the instances extracted from unrelated areas will be removed. On the other hand, MIL-SS has comparable recognition rates on object categorisation with SKDE as shown in Table 6.2.

6.2.3 Sensitivity to labeling noise

In this section, we compare MIL-SS with other alternatives in terms of the sensitivity to labeling noise. Labeling noise can be seen as the likelihood when an image is mislabeled. The robustness to labeling noise is very important to classifiers because obtaining "pure" training data is often difficult. For instance, a category of images retrieved from search engines by keywords may contain many unrelated images, and manual label process is often subjective.

We perform the tests using binary classification. "ajax detergent" and "coke can" in SIVAL dataset are selected and treated as positive and negative training sets respectively. Both categories are distinctive and well classified in our object categorisation, so they are good sets for testing influences of labeling noise. The training and testing data are randomly split into 50% and 50% in each category. To generate the "labeling noise", we firstly select x% (up to 20%) of training images from each of these two categories, and negate the original labels of the selected images, i.e., change positive (negative) to negative (positive), and deem those 'mislabeled' images as labeling noise. On each level of labeling noise (0-20% with step size 2%), 5 rounds of classification are conducted. The average

classification accuracies using SKDE for MIL and other methods are presented in Figure 6.2. Figure 6.2 shows that, like MIL-SKDE, the MIL-SS is robust to

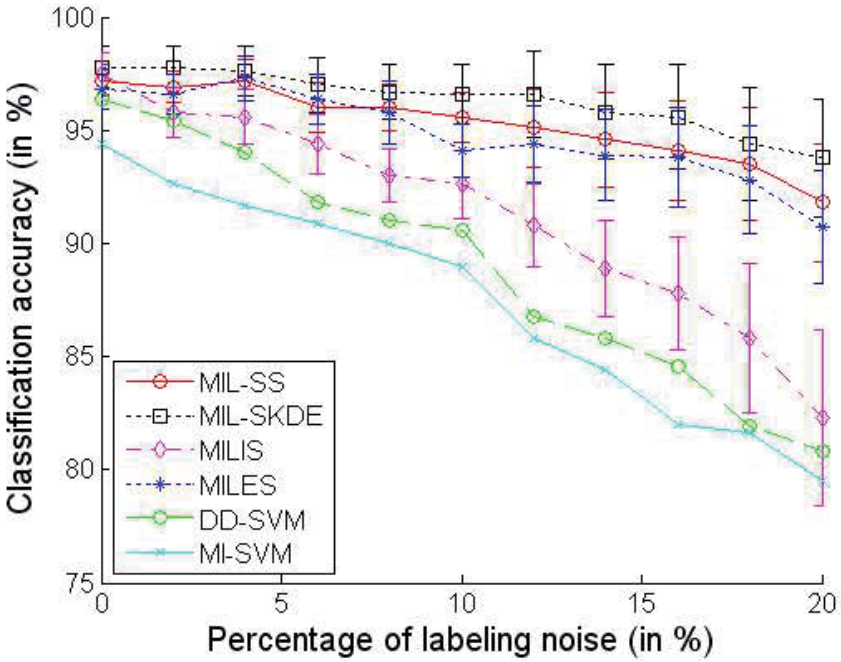


Figure 6.2: Comparisons of sensitivity to labeling noise among the proposed approach, MIL-SKDE, MILIS, MILES, DD-SVM and MI-SVM in terms of binary classification ("ajax detergent" and "coke can"). The average classification accuracies are computed over five randomly generated datasets. For the best three approaches (our method, MILES and MILIS), the corresponding 95% confidence intervals are given.

labeling noise as well and it is only slightly below MIL-SKDE but beats MILIS, MILES, DD-SVM and MI-SVM in the experiments. Meanwhile, MIL-SS has high classification accuracy too. MILES is the third best. MILIS' performance drops faster compared with MILES because it has a lot fewer instance concepts and the labeling noise affects the concepts selection significantly. DD-SVM decreases very quickly because it obtains concepts by diverse density which is very sensitive to mislabeled instances. Finally, MI-SVM[1] does not perform well because the labeling noise causes large penalty in the optimisation of MI-SVM.

[1]Source code - http://www.cs.cmu.edu/~juny/MILL/index.html

6.2.4 Regions of interest detection

Figure 6.3 shows that the obtained *patterns* truly represent the stable features within corresponding object categories. Given a *pattern* $\mathbb{M} = \{(\boldsymbol{\phi}_i, w_i)\}_{i=1}^{t}$, we circle out the regions that correspond to the nearest neighbors of $\boldsymbol{\phi}_i$'s in some correctly recognised images. Those circle areas reflect the regions of interest (ROIs) of the categories. We can see that the majority of ROIs concentrate on the targeting objects themselves and the interference of background clutters has been suppressed by using the SKDE. This is because those false positive instances from positive bags that represent background are mostly mixed with large amount of negative points (i.e., instances in negative bags). After instance selection process, those false positive instances are eliminated. Therefore, the final ROIs mostly locate in the targeting objects. Although there are a small quantity of circles outside the target objects (due to some background regions have the similar feature as those of ROIs), these circles staying outside ROIs hardly affect the overall categorisation decision because both the quantity and the weights of the 'missing' circles are small as we can see in Figure 6.3.

6.3 MIL-SS for bag-of-words model

In the rest of the chapter, we present a software system implemented by C++ for object categorisation. This system utilises the well-known bag-of-words (BoW) model and MIL-SS algorithm is used to generate codebook. In a typical BoW model, usually the whole area of an image is considered as the region of interest (ROI) for visual codebook generation. This will introduce many noises for the final codebook. In our implementation, we learn the visual keywords mainly from the regions of target objects and the unrelated backgrounds will be excluded for generating codebook. This is achieved by using the MIL-SS algorithm. In addition, the MIL-SS has been proved to be very efficient, so the object categorisation task can be easily applied on the whole benchmark SIVAL dataset with 25 categories. In the classification stage, unlike previous work which utilised the multiple-class Support Vector Machines (SVMs), here we utilise a maximum margin supervised topic model, Maximum Entropy Discrimination Latent Dirichlet

(a) Motorbike (b) Coke can (c) Detergent (d) Book

Figure 6.3: The regions of interest (ROIs) (illustrated by circles) detected by our method. The radius of a circle indicates the weight w_i in pattern $\mathbb{M}$. Since the majority of ROIs concentrate on the objects of interest, the interference of background clutters has been suppressed by applying instance selection.

Allocation (MedLDA), for classification. The final performance of our work is quite encouraging.

6.3.1 Bag-of-words model

In the past decade, the bag-of-words (BoW) model, originated from natural language processing and information retrieval, has been well recognised as a state-of-the-art method in various visual classification tasks. It has been adopted and proved to work surprisingly well in various applications, e.g., object categorisation [109], scene classification [110][111], action recognition [112], human pose estimation [113] and visual recognition [114].

In the textual cases, the training data for classification are documents consisting of textual words. Analogically, in computer vision, each image is treated as a document. However, the words within an image are not as straightforward as those texts. A common solution is to break down each image into local regions and extract descriptors such as the Scale Invariant Feature Transform (SIFT) from those local regions. Then, the high-dimensional local descriptors are quantised into discrete visual words (this step is called feature coding) by utilizing a visual codebook. After the quantisation, each image is represented by the frequency histogram of a bag of visual words (this step is called feature pooling) and those histograms are the feature vectors for final classification. The typical procedures of a bag-of-words model are illustrated using Figure 6.4. In this section, we present a C++ implementation of the BoW model which follows the classic steps: feature extraction, codebook generation, feature coding, feature pooling and classification.

In codebook generation step, traditional approach often considers the whole area of an image as the region of interest (ROI) and uses clustering (e.g. K-means) to generate the a set of visual words (i.e., codebook) from training images. This may be all right when the whole image area contains available clues, e.g., in natural scene classification. However, for many other applications, e.g., object categorisation which involves determining whether or not an image contains a specific category of objects, only the regions of target objects are ROIs and the backgrounds are unrelated areas. In this case, traditional clustering approach

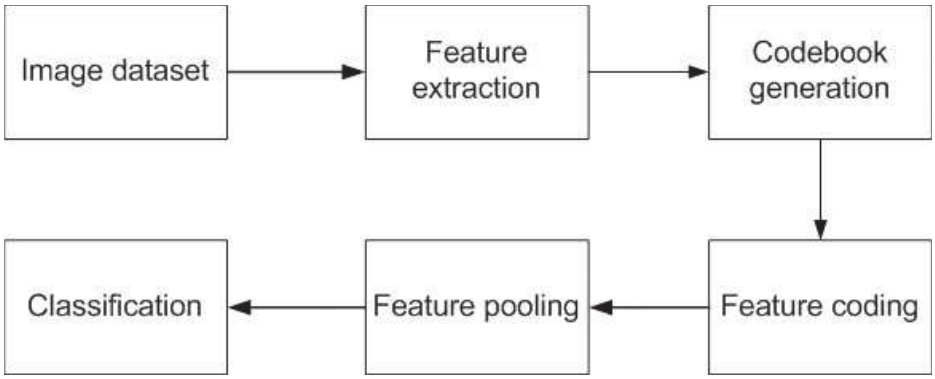


Figure 6.4: General procedures for a bag-of-words model.

will generate many noise words from background to the final codebook. Those noise words will degrade the later classification especially when the backgrounds of images in the training dataset are quite similar. To address this issue, we utilise our MIL-SS algorithm which can generate codebook just from the target object areas in an multiple-instance learning manner.

Another benefit to use the MIL-SS is that learning codebook is quite time-consuming especially for a large-scale dataset. This is because computation of those visual words in a codebook requires clustering of a huge amount of data. In previous experiments, we have proved that MIL-SS is very efficient for training, which ensures that a codebook can be learned in a fast way.

In classification, we utilise the maximum entropy discrimination latent Dirichlet allocation (MedLDA) [88][85] that has been reported being comparable with or outperforming other LDA-based methods [88] and SVMs (with limited or medium scale of training data) [115].

Our system is modularised and each of the steps can be extended easily. In our system, we save the output of each of the steps and such output is the input for next step. Therefore, every step of the BoW model can be checked and verified. A sample experiment of object categorisation is conducted on SIVAL dataset and accuracy is quite encouraging.

6.3.1.1 Feature extraction

Feature extraction is to get local region descriptors from the images. Our system provides sparse and dense feature extractions.

- For spare feature extraction, we use Rob Hess's SIFT [55] implementation[1] to detect key points and get the descriptors of those key points for each image. Loosely speaking, each of the SIFT descriptors can be seen as a visual word.
- For dense feature extraction, we implement the HOG [116] to get descriptors via a scanning window. The users can specify the number of cells per block, the number of pixels per cell, the number of channels per cell histogram, and the step length of scanning window. The default parameters are 3×3 cell blocks of 6×6 pixel cells with 9 histogram channels.

The extracted descriptors within each image form a matrix. Each column of the matrix represents a word and such matrix is stored by means of an XML file.

6.3.1.2 Codebook generation

In this step, we firstly generate codebook for every single category, then combine all the category codebooks into the final codebook. To cluster all the feature descriptors in a category, we treat the feature descriptors extracted from the images containing target object as positive data and feature descriptors extracted from other images as negative data, then utilise MIL-SS, which has been presented in this chapter (a summary of MIL-SS algorithm is in Section 6.1.4), to generate the category codebook.

Suppose that there are γ image categories and their corresponding patterns are $\{\mathbb{M}_i\}_{i=1}^{\gamma}$. In terms of MIL-SS, each pattern can be denoted as $\mathbb{M}_i = \left\{(\boldsymbol{\phi}_j, w_j)\right\}_{j=1}^{t_i}$, where $\boldsymbol{\phi}_j \in \mathbb{R}^d$ is the j-th concept in a pattern, w_j is the weight and t_i is the concept number of pattern $\mathbb{M}_i$. For codebook generation, the concepts for a certain category are the visual words of such category, and the final codebook is generated by combining all the visual words across all categories. Let $\mathbf{x}_j =$

[1]Codes are downloaded from http://blogs.oregonstate.edu/hess/code/sift/

$\boldsymbol{\phi}_j \in \mathbb{R}^d$ be a visual word, then the codebook $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$, where $M = \sum_{i=1}^{\gamma} t_i$. For access convenience, the codebook $\mathbf{X}$ is saved as a matrix (columns are words) through an XML file.

6.3.1.3 Feature coding and pooling

Our goal of this step is to convert an image I into a new feature vector W for training classifiers. Here, $I = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N\}$, and $\mathbf{v}_i$ is an extracted feature descriptor, e.g., the SIFT descriptor. The feature vector $W = \{w_1, w_2, \ldots, w_M\}$ is normally a histogram in a BoW model, where w_i represents the frequency of codebook's i-th word $\mathbf{x}_i$ that occurs in the image I, and the integer M is the size of codebook. Our demo provides two schemes for feature pooling and users can choose either scheme according to the natures of training datasets.

1. Sum scheme:

$$w_i = \sum_{j=1}^{N} K_\sigma \left(D(\mathbf{v}_j, \mathbf{x}_i) \right); \tag{6.7}$$

2. Max scheme:

$$w_i = \max \left(K_\sigma \left(D(\mathbf{v}_1, \mathbf{x}_i) \right), \ldots, K_\sigma \left(D(\mathbf{v}_N, \mathbf{x}_i) \right) \right). \tag{6.8}$$

In the schemes shown in (6.7) and (6.8), $K_\sigma(x)$ is a normalised kernel, e.g., $K_\sigma(x) = \exp\left(-\sigma x^2\right)/C$, where C is a normalisation factor, and $D(\mathbf{v}, \mathbf{x})$ is the distance between $\mathbf{v}$ and $\mathbf{x}$.

6.3.1.4 Classification

In the classification step, we use Maximum Entropy Discrimination Latent Dirichlet Allocation (MedLDA), a topic model recently proposed by Jun et al. [88][85], to train classifiers because it has been reported outperforming other LDA-based methods. To generate the document data, we first calculate the feature vector (a set of frequency of visual word occurrences) W for each image I using previous steps. Secondly, we multiply each W with an integer M (the size of the codebook) and round up all the entries of $M \cdot W$. Then, all document data are fed into the

MedLDA model to train the parameters of this topic model. Finally, the learned parameters of MedLDA model are used for classification. The classification results are stored in a text file showing the classification accuracy of the classified labels against ground truth (if it is available).

6.3.2 Experiments on SIVAL dataset

We test the BoW system on the benchmark SIVAL dataset. As described before, SIVAL dataset includes 25 different image categories with 60 images for each category. The categories consist of images of single objects photographed against diverse backgrounds. The objects may occur anywhere spatially in an image and also may be photographed at a wide-angle or close up. To save the running time, we adjust the image size to 300 pixels and keep the height ratio.

In feature extraction, we detect SIFT key points as the visual words and each word is represented as a 128-d vector. In codebook generation, we use supervised mean shift. We also use K-means to generate the codebook for experimental comparison. In feature coding and pooling, we choose the Max Scheme to generate the features for images. Finally, the MedLDA model is learned for multi-class classification. We repeatedly conduct experiments for 30 runs on the 25-category dataset. The average of the experimental results are compared with other state-of-the-art works on the same dataset. The comparison methods include GMIL [117], RMISSL [118], SIMPLIcity [119], SBN (Single-bolb with Neighbors) [120], aCCIO! [121] and ACCIO!+EM [118]. Table 6.4 indicates the accuracy comparison among different methods over 30 runs on the SIVAL dataset.

6.4 Conclusion

Multiple-instance learning with a speed-up supervised kernel density estimation (MIL-SS) proposed in this chapter is a speed-up version of MIL-SKDE. Compared with MIL-SKDE, MIL-SS greatly increases the training efficiency while obtains comparable performance in image classification and object categorisation. Therefore, in a situation that classification accuracy is a priority and the training time is not critical, MIL-SKDE would be the preference. Otherwise,

	Demo_S[a]	Demo_K[b]	GMIL [117]	RMISSL [118]
Average accuracy	78.6	51.1	82.0	74.8
	ACCIO! [121]	ACCIO!+EM [118]	SIMPLIcity [119]	SBN [120]
Average accuracy	74.6	50.3	57.9	53.9

Table 6.4: Classification accuracy (in %) comparisons among different methods over 30 runs on the SIVAL. Our demo using supervised mean shift outperforms most methods except GMIL. However, GMIL utilised the manual segmentation information and our demo does not need segmentation but detects interest points automatically.

[a]Our demo with Supervised Mean Shift for codebook generation
[b]Our demo with K-means for codebook generation

when the training efficiency is more concerned, MIL-SS is a good alternative. we have also described an C++ implementation of BoW using MIL-SS for visual categorisation. Our implementation is highly modularised and can be easily extended. The experiments have showed that our implementation is comparable with the state-of-the-art methods.

Chapter 7

Conclusions and future work

7.1 Conclusions

In this thesis, we have explored the visual categorisation using single-instance learning and multiple-instance learning. Specifically, we have concentrated on two applications: facial expression recognition using SIL and object categorisation using MIL.

For facial expression recognition, we have proposed a new feature, *Histogram Variances Face (HVF)*, in Chapter 3 to classify the videos of facial expressions. Our experiments have demonstrated that HVF is an effective representation of the dynamic and internal features of a face video or image sequence. HVF is able to integrate well the dynamic features of a certain duration of expression into a static image through which the static facial recognition approaches can be utlised to recognise the dynamic expressions. The application of HVFs fills the gap between the expression recognition and facial recognition.

In Chapter 4, we have extended our work of HVF to a hexagonal structure in which we, for the first time, apply LBPs defined on the hexagonal structure [97] to extract the Hexagonal Histogram Variance Faces (HHVFs). This novel approach not only greatly reduces the computation costs but also improves the accuracy for facial expression recognition.

For object categorisation, we have presented a novel MIL algorithm, MIL-SKDE, in Chapter 5 which utilises the proposed density function, supervised kernel density estimation (SKDE), to model the MIL problem. Compared with

commonly-adopted Diverse Density Estimation (DDE), SKDE is able to learn multiple concepts efficiently through a supervised mean shift and better tolerate the labeling noise. SKDE and supervised mean shift can be seen as the extensions of conventional kernel density estimation and mean shift respectively by considering the class labels of the sample data. The experiments on synthetic data have shown the characteristics of our method. In addition, the experiments conducted on real-world datasets demonstrate that our approach has superior performance over the state-of-the-art approaches in terms of accuracy of content-based image classification and object category recognition, as well as robustness to labeling noise.

To speed up the mode seeking of SKDE function, a fast method has been given in Chapter 6 to obtain the approximation of modes without sacrificing accuracy. The experiments show that the speed-up approach has comparable performance and is more robust to labeling noise than the state-of-the-art alternatives.

Finally, we have presented an C++ implementation of Bag-of-words application in Chapter 6 for visual categorisation. Our implementation is highly modularised and can be easily extended. The experiments have showed that our implementation is comparable with the state-of-the-art methods.

7.2 Future work

There are a few directions for future work about MIL-SKDE and MIL-SS. Here, we list two of those directions:

1. For the supervised kernel density shown in (5.12) and supervised mean shift shown in (5.21), the bandwidth h is set to a fixed value in this thesis. However, adopting variable bandwidth in SKDE and the supervised mean shift may be able to better harmonise the influence between positive and negative data to obtain better performances. This work can be explored in the future.

2. To seek modes of supervised kernel density (5.12) more efficiently, some existing fast mean shift and approximate mode seeking algorithms may

be modified for the supervised mean shift to greatly quicken the pattern acquisition in MIL-SKDE and MIL-SS.

References

[1] Maja Pantic, Student Member, and Leon J. M. Rothkrantz. Automatic analysis of facial expressions: The state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1424–1445, 2000. 3

[2] Beat Fasel and Juergen Luettin. Automatic facial expression analysis: A survey. *PATTERN RECOGNITION*, 36(1):259–275, 2003. 3

[3] Zhihong Zeng, M. Pantic, G.I. Roisman, and T.S. Huang. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(1): 39–58, January 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.52. 3

[4] Michel Valstar, Marcello Mehu, Maja Pantic, and Klaus Scherer. Meta-analysis of the first facial expression recognition and analysis challenge. *IEEE Transactions on Systems, Man and Cybernetics (to appear)*, 2012. 3

[5] P. Ekman and W. Friesen. Facial action coding system. In *Palo Alto, CA: Consulting Psychologists Press*, 1978. 3

[6] Yan Tong, Wenhui Liao, and Qiang Ji. Facial action unit recognition by exploiting their dynamic and semantic relationships. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(10):1683–1699, October 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1094. URL http://dx.doi.org/10.1109/TPAMI.2007.1094. 3, 4

[7] Y.-I. Tian, T. Kanade, and J.F. Cohn. Recognizing action units for facial expression analysis. *Pattern Analysis and Machine Intelligence,*

IEEE Transactions on, 23(2):97–115, feb 2001. ISSN 0162-8828. doi: 10.1109/34.908962. 4, 5

[8] Jenn-Jier James Lien, Takeo Kanade, Jeffrey Cohn, and C. Li. Detection, tracking, and classification of action units in facial expression. *Journal of Robotics and Autonomous Systems*, July 1999. 5

[9] Gianluca Donato, Marian Stewart Bartlett, Joseph C. Hager, Paul Ekman, and Terrence J. Sejnowski. Classifying facial actions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(10):974–989, October 1999. ISSN 0162-8828. doi: 10.1109/34.799905. URL http://dx.doi.org/10.1109/34.799905. 5

[10] Beat Fasel and Jrgen Lttin. Recognition of asymmetric facial action unit activities and intensities. In *Proceedings of the International Conference on Pattern Recognition*, ICPR, pages 1100–1103, Washington, DC, USA, 2000. IEEE Computer Society. URL http://dl.acm.org/citation.cfm?id=876866.877382. 5

[11] Evan Smith, Marian Stewart Bartlett, and Javier Movellan. Computer recognition of facial actions: A study of co-articulation effects. In *Proceedings of the 8th Annual Joint Symposium on Neural Computation*, 2001. 5

[12] J.J. Bazzo and M.V. Lamar. Recognizing facial actions using gabor wavelets with neutral face average difference. In *Proceedings. Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 505–510, May 2004. doi: 10.1109/AFGR.2004.1301583. 5

[13] M.S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan. Recognizing facial expression: machine learning and application to spontaneous behavior. In *Computer Vision and Pattern Recognition*, volume 2, pages 568–573, june 2005. doi: 10.1109/CVPR.2005.297. 5

[14] A. Lanitis, C.J. Taylor, and T.F. Cootes. Automatic interpretation and coding of face images using flexible models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):743–756, July 1997. ISSN 0162-8828. doi: 10.1109/34.598231. 5

[15] Ira Cohen, Nicu Sebe, Ashutosh Garg, Lawrence S. Chen, and Thomas S. Huang. Facial expression recognition from video sequences: temporal and static modeling. *Comput. Vis. Image Underst.*, 91(1-2):160–187, July 2003. ISSN 1077-3142. doi: 10.1016/S1077-3142(03)00081-X. URL http://dx.doi.org/10.1016/S1077-3142(03)00081-X. 5

[16] Jeffrey Cohn, L.I. Reed, Zara Ambadar, Jing Xiao, and Tsuyoshi Moriyama. Automatic analysis and recognition of brow actions and head motion in spontaneous facial behavior. In *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics*, volume 1, pages 610–616, October 2004. 5

[17] Ying-li Tian, Takeo Kanade, and Jeffrey F. Cohn. Evaluation of gabor-wavelet-based facial action unit recognition in image sequences of increasing complexity. In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, FGR '02, pages 218–223, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1602-5. URL http://dl.acm.org/citation.cfm?id=874061.875419. 5

[18] M.F. Valstar, I. Patras, and M. Pantic. Facial action unit detection using probabilistic actively learned support vector machines on tracked facial point data. In *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, page 76, June 2005. doi: 10.1109/CVPR.2005.457. 5

[19] Maja Pantic and Leon J. M. Rothkrantz. Facial action recognition for facial expression analysis from static face images. *IEEE Trans. Systems, Man, and CyberneticsCPart B: Cybernetics*, 34(3):1449–1461, June 2004. 5

[20] M.F. Valstar and M. Pantic. Fully automatic recognition of the temporal phases of facial actions. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42(1):28 –43, feb. 2012. ISSN 1083-4419. doi: 10.1109/TSMCB.2011.2163710. 5

[21] Bihan Jiang, Michel Fran04ois Valstar, and Maja Pantic. Action unit detection using sparse appearance descriptors in space-time video volumes.

In *In Proc. IEEE Conf. Automatic Face and Gesture Recognition*, pages 314–321, 2011. 5

[22] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. Gray scale and rotation invariant texture classification with local binary patterns. In *ECCV*, pages 404–420, 2000. 5, 51, 52, 67

[23] M. Bartlett, G. Littlewort, Tingfan Wu, and J. Movellan. Computer expression recognition toolbox. In *Automatic Face Gesture Recognition, 8th IEEE International Conference on*, pages 1 –2, sept. 2008. doi: 10.1109/AFGR.2008.4813406. 6

[24] Michel Valstar and Maja Pantic. Fully automatic facial action unit detection and temporal analysis. In *Computer Vision and Pattern Recognition Workshop*, volume 17-22 June, page 149, 2006. 6

[25] M.F. Valstar, I. Patras, and M. Pantic. Facial action unit detection using probabilistic actively learned support vector machines on tracked facial point data. In *Computer Vision and Pattern Recognition*, 2005. 6

[26] Danijela Vukadinovic and Maja Pantic. Fully automatic facial feature point detection using gabor feature based boosted classifiers. In *Systems, Man and Cybernetics(ICSMC)*, volume 2, pages 1692–1698, 2005. 6

[27] J. Whitehill and C.W. Omlin. Haar features for facs au recognition. In *Automatic Face and Gesture Recognition, 7th International Conference on*, pages 5 pp. –101, april 2006. doi: 10.1109/FGR.2006.61. 6

[28] S. Koelstra, M. Pantic, and I. Patras. A dynamic texture-based approach to recognition of facial actions and their temporal models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(11):1940–1954, nov. 2010. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.50. 6

[29] K. Scherer and P. Ekman. *Handbook of Methods in Nonverbal Behavior Research*. Cambridge Univ. Press, 1982. 6

[30] Caifeng Shan, Shaogang Gong, and Peter W. McOwan. Facial expression recognition based on local binary patterns: A comprehensive study. *Image and Vision Computing Journal*, 27(6):803–816, may 2009. ISSN 0262-8856. doi: 10.1016/j.imavis.2008.08.005. URL http://dx.doi.org/10.1016/j.imavis.2008.08.005. 6

[31] Guoying Zhao and Matti Pietikainen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):915–928, June 2007. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.1110. URL http://dx.doi.org/10.1109/TPAMI.2007.1110. 6

[32] S. Moore and R. Bowden. Local binary patterns for multi-view facial expression recognition. *Comput. Vis. Image Underst.*, 115(4):541–558, April 2011. ISSN 1077-3142. doi: 10.1016/j.cviu.2010.12.001. URL http://dx.doi.org/10.1016/j.cviu.2010.12.001. 6

[33] A. Dhall, A. Asthana, R. Goecke, and T. Gedeon. Emotion recognition using phog and lpq features. In *Automatic Face Gesture Recognition and Workshops (FG 2011), IEEE International Conference on*, pages 878–883, march 2011. doi: 10.1109/FG.2011.5771366. 6

[34] Stefanos Zafeiriou and Maria Petrou. Nonlinear non-negative component analysis algorithms. *Trans. Img. Proc.*, 19(4):1050–1066, April 2010. ISSN 1057-7149. doi: 10.1109/TIP.2009.203http://bbs.ci123.com/post/101486.html/0ble8816. URL http://dx.doi.org/10.1109/TIP.2009.2038816. 6

[35] T. Kanade, J. F. Cohn, and Y. Tian. Comprehensive database for facial expression analysis. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition, Grenoble, France*, pages 46–53, 2000. 6, 51, 58, 61, 66, 70, 71

[36] Ruicong Zhi, Markus Flierl, Qiuqi Ruan, and W Bastiaan Kleijn. Graph-preserving sparse nonnegative matrix factorization with application to facial expression recognition. *IEEE transactions on systems man and cybernetics Part B Cybernetics a publication of the*

IEEE Systems Man and Cybernetics Society, 41(1):38–52, 2011. URL http://www.ncbi.nlm.nih.gov/pubmed/20403788. 6

[37] A. Asthana, J. Saragih, M. Wagner, and R. Goecke. Evaluating aam fitting methods for facial expression recognition. In *Affective Computing and Intelligent Interaction and Workshops, 2009. ACII 2009. 3rd International Conference on*, pages 1 –8, sept. 2009. doi: 10.1109/ACII.2009.5349489. 7

[38] Jaewon Sung and Daijin Kim. Real-time facial expression recognition using staam and layered gda classifier. *Image Vision Comput.*, 27(9):1313–1325, August 2009. ISSN 0262-8856. doi: 10.1016/j.imavis.2008.11.010. URL http://dx.doi.org/10.1016/j.imavis.2008.11.010. 7

[39] N. Sebe, M. S. Lew, Y. Sun, I. Cohen, T. Gevers, and T. S. Huang. Authentic facial expression analysis. *Image Vision Comput.*, 25(12):1856–1863, Dec. 2007. ISSN 0262-8856. doi: 10.1016/j.imavis.2005.12.021. URL http://dx.doi.org/10.1016/j.imavis.2005.12.021. 7

[40] Timo Ojala, Pietikäinen M, and Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. In *PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, volume 24, pages 971–987, 2002. 7, 51, 52, 67

[41] YOSSI RUBNER, CARLO TOMASI, and LEONIDAS J. GUIBAS. The earth mover's distance as a metric for image retrieval. In *International Journal of Computer Vision*, volume 40(2), pages 99–121, 2000. 7, 54

[42] Li Fei-Fei. The short course of recognizing and learning object categories. *Short Course on ICCV*, 2009. URL http://people.csail.mit.edu/torralba/shortCourseRLOC/index.html. 9

[43] Thomas G. Dietterich, Richard H. Lathrop, Tomas Lozano-Perez, and Arris Pharmaceutical. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997. 9, 11

[44] Zhouyu Fu and Antonio Robles-Kelly. An instance selection approach to multiple instance learning. In *Computer Vision and Pattern Recognition (CVPR)*, pages 911–918, 2009. 10, 12, 99

[45] Sudheendra Vijayanarasimhan and Kristen Grauman. Keywords to visual categories: Multiple-instance learning forweakly supervised object categorization. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008. 10, 99

[46] Yixin Chen, Jinbo Bi, and James Z. Wang. Miles: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1931–1947, 2006. 10, 12, 91, 94, 95, 96, 97, 98, 99, 101, 104, 115, 116, 117

[47] Zheng-Jun Zha, Xian-Sheng Hua, Tao Mei, Jingdong Wang, Guo-Jun Qi, and Zengfu Wang. Joint multi-label multi-instance learning for image classification. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008. 10

[48] Qi Zhang, Sally A. Goldman, Wei Yu, and Jason E. Fritts. Content-based image retrieval using multiple-instance learning. In *International Conference on Machine Learning (ICML)*, pages 682–689. Morgan Kaufmann, 2002. 10

[49] Cheng Yang. Image database retrieval with multiple-instance learning techniques. In *International Conference on Data Engineering (ICDE)*, pages 233–243, 2000. 10

[50] Alexander Vezhnevets and Joachim M. Buhmann. Towards weakly supervised semantic segmentation by means of multiple instance and multitask learning. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3249–3256, 2010. 10

[51] Mu Li, James T. Kwok, and Bao-Liang Lu. Online multiple instance learning with no regret. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1395–1401, 2010. 10

[52] Zhe Lin, Gang Hua, and Larry S. Davis. Multiple instance feature for robust part-based object detection. In *Computer Vision and Pattern Recognition (CVPR)*, page 405, 2009. 10, 99

[53] Vikas C. Raykar, Balaji Krishnapuram, Jinbo Bi, Murat Dundar, and R. Bharat Rao. Bayesian multiple instance learning: automatic feature selection and inductive transfer. In *International Conference on Machine Learning (ICML)*, pages 808–815. ACM, 2008. 10, 12

[54] Dijia Wu, Jinbo Bi, and Kim Boyer. A min-max framework of cascaded classifier with multiple instance learning for computer aided diagnosis. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1359–1366, 2009. 10

[55] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94. 11, 22, 24, 25, 26, 27, 28, 101, 124

[56] Oded Maron and Toms Lozano-Prez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 570–576. MIT Press, 1998. 11, 13

[57] Qi Zhang and Sally A. Goldman. Em-dd: An improved multiple-instance learning technique. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1073–1080. MIT Press, 2001. 12

[58] Rouhollah Rahmani, Sally A. Goldman, Hui Zhang, Sharath R. Cholleti, and Jason E. Fritts. Localized content based image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1902–2002, 2008. 12, 82

[59] Y. Chen and J. Z. Wang. Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5:913–939, 2004. 12, 82, 91, 95, 97, 98, 101, 104, 115, 116, 117

[60] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 561–568. MIT Press, 2003. 12, 91, 97, 98, 115, 116

[61] Razvan C. Bunescu and Raymond J. Mooney. Multiple instance learning for sparse positive bags. In *Proceedings of the 24th Annual International Conference on Machine Learning (ICML)*, pages 105–112, 2007. 12

[62] Zhouyu Fu, Antonio Robles-Kelly, and Jun Zhou. Milis: Multiple instance learning with instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:958–977, May 2011. 12, 91, 94, 96, 97, 98, 99, 101, 104, 115, 116, 117

[63] Jun Wang and Jean-Daniel Zucker. Solving the multiple-instance problem: A lazy learning approach. In *International Conference on Machine Learning (ICML)*, pages 1119–1125. Morgan Kaufmann, 2000. 12

[64] Y. Chevaleyre and J. D. Zucker. Solving multiple-instance and multiple-part learning problems with decision trees and rule sets. application to the mutagenesis problem. *Lecture Notes in Artificial Intelligence*, 2056:204–214, 2001. 12

[65] Paul Viola, John C. Platt, and Cha Zhang. Multiple instance boosting for object detection. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1419–1426. MIT Press, 2006. 12

[66] Christian Leistner, Amir Saffari, and Horst Bischof. Miforests: Multiple-instance learning with randomized trees. In *European Conference on Computer Vision (ECCV)*, volume 6316, pages 29–42, 2010. 12

[67] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1289–1296. MIT Press, 2008. 12

[68] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001. xiv, 17, 18, 19

[69] M.A. Turk and A.P. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 586 –591, jun 1991. doi: 10.1109/CVPR.1991.139758. 20

[70] David Lowe. Object recognition from local scale-invariant features. *ICCV*, pages 1150–1157, 1999. 22, 24

[71] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006. 27

[72] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition*, 1(25):886–893, June 2005. 27, 28

[73] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005. URL http://lear.inrialpes.fr/pubs/2005/MS05. 27

[74] D. W. Scott. Multivariate density estimation: Theory, practice, and visualization. *John Wiley & Sons, Inc., Hoboken, NJ, USA*, 2008. 31, 76

[75] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *Annals of Mathematical Statistics*, 27:832–837, 1956. 31

[76] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33:1065–1076, 1962. 31, 76

[77] D. Comaniciu. Variable bandwidth density-based fusion. *Computer Vision and Pattern Recognition (CVPR)*, 1:56–66, 2003. 32

[78] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:790–799, 1995. 34, 77

[79] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. 34, 77, 78

[80] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT '95, pages 23–37, London, UK, UK, 1995. Springer-Verlag. ISBN 3-540-59119-2. URL http://dl.acm.org/citation.cfm?id=646943.712093. 34

[81] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944937. 38, 40, 41

[82] David Blei and Jon McAuliffe. Supervised topic models. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, pages 121–128, Cambridge, MA, 2008. MIT Press. 38, 42, 43, 45

[83] Simon Lacoste-julien, Fei Sha, and Michael I. Jordan. Disclda: Discriminative learning for dimensionality reduction and classification. In *Advances in Neural Information Processing Systems*, 2008. 38, 44

[84] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. Labeled lda: a supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 248–256, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-59-6. URL http://dl.acm.org/citation.cfm?id=1699510.1699543. 38

[85] Jun Zhu, Amr Ahmed, and Eric P. Xing. Medlda: Maximum margin supervised topic models. *Journal of Machine Learning Research*, page in press, 2012. 38, 43, 44, 45, 46, 123, 125

[86] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, April 2004. 40

[87] Peter McCullagh and John A. Nelder. *Generalized Linear Models*. Chapman and Hall, 1989. 42

[88] Jun Zhu, Amr Ahmed, and Eric P. Xing. Medlda: maximum margin supervised topic models for regression and classification. In *ICML*, pages 1257–1264, 2009. 43, 44, 45, 46, 123, 125

[89] Paul Viola and Michael J. Jones. Robust real-time object detection. In *ICCV*, 2001. 51, 66

[90] M. Turk and A. Pentland. Face recognition using eigenfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991. 57

[91] Boser B., Guyon I., and V. Vapnik. An training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992. 57

[92] Vladimir Vapnik. The nature of statistical learning theory. In *Springer-Verlag*, 1995. 57

[93] V. N Vapnik. Statistical learning theory. wiley interscience. In *Wiley Interscience*, 1998. 57

[94] Tang Jinghai, Ying Zilu, and Zhang Youwei. The contrast analysis of facial expression recognition by human and computer. In *ICSP*, pages 1649–1653, 2006. 61, 63, 71, 72

[95] Ruo Du, Qiang Wu, Xiangjian He, Wenjing Jia, and Daming Wei. Local binary patterns for human detection on hexagonal structure. In *IEEE Workshop on Applications of Computer Vision*, pages 341–347, 2009. 66

[96] Xiangjian He, Daming Wei, Kin-Man Lam, Jianmin Li, Lin Wang, Wenjing Jia, and Qiang Wu. Local binary patterns for human detection on hexagonal

structure. In *Advanced Concepts for Intelligent Vision Systems*, page to appear, 2010. 66, 67

[97] Xiangjian He, Jianmin Li, Yan Chen, Qiang Wu, and Wenjing Jia. Local binary patterns for human detection on hexagonal structure. In *IEEE International Symposium in Multimedia*, pages 65–71, 2007. 67, 68, 128

[98] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29:51–59, 1996. 67

[99] Xiangru Li, Zhanyi Hu, and Fuchao Wu. A note on the convergence of the mean shift. *Journal Pattern Recognition*, 40:1756–1762, 2007. 77

[100] Mark Fashing and Carlo Tomasi. Mean shift is a bound optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:471–474, 2005. 77

[101] Miguel A Carreira-perpiñán. Gaussian mean shift is an em algorithm. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29:2007, 2005. 78

[102] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 564–577, 2003. 78

[103] Meng Wang, Xian-Sheng Hua, Tao Mei, Richang Hong, Guojun Qi, Yan Song, and Li-Rong Dai. Semi-supervised kernel density estimation for video annotation. *Computer Vision and Image Understanding*, 113:384–396, March 2009. 78

[104] Oncel Tuzel, Fatih Porikli, and Peter Meer. Kernel methods for weakly supervised mean shift clustering. In *International Conference on Computer Vision (ICCV)*, pages 48–55, 2009. 78

[105] Koby Crammer, Yoram Singer, Nello Cristianini, John Shawe-taylor, and Bob Williamson. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:2001, 2001. 90, 114

[106] R. Fergus, P. Perona, and A. Zisserman. Weakly supervised scale-invariant learning of models for visual recognition. *International Journal of Computer Vision*, 71:273–303, 2007. 99

[107] D. Liu, Gang Hua, P. Viola, and Tsuhan Chen. Integrated feature selection and higher-order spatial feature extraction for object categorization. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1 –8, june 2008. doi: 10.1109/CVPR.2008.4587403. 99, 101, 115

[108] Rob Hess. An open source sift library. In *ACM Multimedia*, pages 25–29, 2010. 101

[109] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *ECCV*, pages 1–22, 2004. 122

[110] Li Fei-fei and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, pages 524–531, 2005. 122

[111] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pages 2169–2178, 2006. 122

[112] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, Benjamin Rozenfeld, Inria Rennes, Irisa Inria Grenoble, and Lear Ljk. Learning realistic human actions from movies. In *CVPR*, 2008. 122

[113] Huazhong Ning, Wei Xu, Yihong Gong, and T. Huang. Discriminative learning of visual words for 3d human pose estimation. In *CVPR*, pages 1–8, june 2008. 122

[114] Jeff Donahue and Kristen Grauman. Annotator rationales for visual recognition. In *ICCV*, pages 1395–1402, 2011. 122

[115] T. N. Rubin, A. Chambers, P. Smyth, and M. Steyvers. Statistical topic models for multi-label document classification. *Machine Learning, in press*, pages 48–55, 2012. 123

[116] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005. 124

[117] Changhu Wang, Lei Zhang, and Hong-Jiang Zhang. Graph-based multiple-instance learning for object-based image retrieval. In *ACM MIR*, pages 156–163, 2008. 126, 127

[118] Rouhollah Rahmani and Sally A. Goldman. Missl: Multiple-instance semi-supervised learning. In *ICML*, pages 705–712, 2006. 126, 127

[119] James Z. Wang, Jia Li, and Gio Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. *TPAMI*, 23:947–963, 2001. 126, 127

[120] Oded Maron and Aparna Lakshmi Ratan. Multiple-instance learning for natural scene classification. In *ICML*, pages 341–349, 1998. 126, 127

[121] Rouhollah Rahmani, Sally A. Goldman, Hui Zhang, John Krettek, and Jason E. Fritts. Localized content based image retrieval. In *ACM SIGMM international workshop on Multimedia information retrieval*, MIR '05, pages 227–236, 2005. 126, 127