Efficient SVM based Object Classification and Detection

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science (by Research)
in
Computer Science

by

Sreekanth Vempati
200402044
v_sreekanth@research.iiit.ac.in

Center for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2010

Copyright © Sreekanth Vempati, 2010
All Rights Reserved

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Efficient SVM based Object classification and detection" by Sreekanth Vempati, has been carried out under our supervision and is not submitted elsewhere for a degree.

Date	Adviser: Dr. C. V. Jawahar
Date	Adviser: Dr. Andrew Zisserman

To my parents and sister

Acknowledgments

Firstly, I am very thankful for my advisors Dr. C. V. Jawahar and Dr. Andrew Zisserman for the guidance and encouragement they have provided for the work in this thesis. I am indebted to Dr. Andrea Vedaldi for all his comments and ideas which helped me in writing this thesis.

Special mentions to Mihir Jain, Marcin, James Philbin, and Omkar for their discussions and cooperation during TRECVID work. I am also grateful to Rakesh, Chetan, Rasagna, Pradeep, Pradhee Tandon and Chandrika for their useful technical discussions.

Most importantly, I would like to thank my parents and sister for their support, without which this thesis wouldn't have been possible.

I am also thankful to Dr. P. J. Narayanan for providing such a good infrastructure at CVIT, IIIT-Hyderabad. I would also thank all the people in CVIT lab for accompanying me and endowing a friendly environment.

Also, I would like to acknowledge R.Satyanarayana for his help in administrative issues.

Abstract

Over the last few years, the amount of image and video data present over the internet, and in the personal collections has been increasing rapidly. Therefore, the need for organizing and searching these vast collections of data efficiently has also increased. This has led to the research in the areas of content based retrieval and recognition of scenes/objects in visual data. There has been lot of research in these areas over the last few years and are still not yet at a deployable stage for real-world usage. For these technologies to be deployable, these solutions should not only be accurate, but also efficient and scalable. For all these related problems of visual recognition, there are two major phases, namely feature extraction stage and learning stage. The feature extraction stage deals with building a representation for the image/video data and the learning stage deals with learning how a function which can distinguish the classes. In this thesis, we focus on building efficient methods for visual content recognition and detection in images and videos. We mainly propose new ideas for the learning stage. For this purpose we start from using the state-of-the-art techniques and then show how our proposed ideas influence the computational time and performance.

Firstly, we show the utility of state of the art image representations and classification methods for the purpose of large scale semantic video retrieval. We demonstrate this on TRECVID 2008 and TRECVID 2009 datasets containing videos for the retrieval of various scenes, objects and actions. We use Support Vector Machines(SVMs) as classifiers, which have been the popular choice for classification tasks in many fields. They have become popular mainly because of their good generalization capability. For obtaining non-linear decision boundaries, SVMs use a kernel function. This kernel function helps in finding a linear classifier in some high dimensional feature space, without actually computing the higher dimensional vectors. In many situations, we need to use computationally expensive non-linear functions as kernels. On the other hand, Linear kernel is computationally inexpensive, however it gives poorer results in most of the cases.

Another contribution of this thesis is a method for improving the performance of computationally inexpensive classifiers like linear SVM. For this purpose, we explore the utility of sub-categories, which are the sub groupings present in the feature space of each semantic class of data. We model these subcategories by using Structural SVM framework. Also, we analyze how the choice of the groupings effect the results and present a method to learn the optimal groupings. We investigate our methods on various synthetic two dimensional datasets and real world datasets namely, VOC 2007 and TRECVID 2008.

Non-linear kernel methods yield state-of-the-art performance for image classification and object detection. However, large scale problems require machine learning techniques of at most linear complexity and these are usually limited to linear kernels. This unfortunately rules out gold-standard kernels such as the generalized RBF kernels (e.g. exponential-χ^2). All the No-linear kernels help in computing the inner product in a high dimensional space different from the input space. This helps in overcoming the explicit computation of the high dimensional vectors. The function which can be used to compute this high dimensional feature vector is called the feature map. But this feature map is hard to compute and is very high dimensional. In the literature, explicit feature finite dimensional feature maps have been proposed to approximate the *additive kernels* (intersection, χ^2) by linear ones, thus enabling the use of fast machine learning technique in a non-linear context. Also, an analogous technique was proposed for the *translation invariant RBF kernels*. As a part of this thesis, we complete the construction and combine the two techniques to obtain explicit feature maps for the generalized RBF kernels. Furthermore, we investigate a learning method using l^1 regularization to encourage sparsity in the final vector representation, and thus reduce its dimension. We evaluate this technique on the VOC 2007 detection challenge, showing when it can improve on fast additive kernels, and the trade-offs in complexity and accuracy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last few years, there has been an exponential increase in the amount of visual data (like images and videos) present over the web servers and personal computing devices. For example, according to official statistics of Youtube(a video sharing website), amount of video content uploaded per minute on their website has increased from 6 hours in 2007 to 20 hours in 2009, which shows the rate of increase in the video content over the web. This ubiquitous amount of multimedia content is mainly caused by various advancements and availability of cheaper image acquisition devices(cameras), data storage, internet bandwidth, etc., This tremendous increase in visual data has triggered the need for methods to *automatically organize* and *perform a real-time semantic search* for the content of interest. These methods are needed to be automatic as manual labelling of the information present in the images/videos is a cumbersome task and requires lot of human labour.

This thesis deals with the design of large-scale techniques mainly for the problems of *Image Categorization*, *Object Detection* and *Concept-Based Image/Video retrieval*. The task of Image Categorization is to assign a category label for a given image based on the visual content present in the given image. This is also referred to as Image Classification in the literature. One needs to note that in image categorization, what we are dealing with is an "image category recognition" problem and not a "specific sample recognition" problem. In the case of specific sample recognition, the task is to find whether a specific example object/scene is present in the given image. Tasks of Scene/Object classification can also be considered as Image Categorization problems. In the case of Object Detection, object category label along with the exact position (bounding box) of the object has to be found. The aim of Concept-Based Retrieval is to rank the given set of images/videos based on confidence of presence of the given concept. A concept is nothing but a category which can be either object, scene or even an action. Examples of concepts are *Cityscape, Aeroplane, Person playing a musical instrument, etc.* Note that the main difference between image categorization and concept retrieval is that ***i)*** in the case of image categorization, the category label assigned to a given image is important and ***ii)*** in the case of image retrieval, the ranking of the images is important. Most of the image categorization techniques form the base for both Object Detection and Concept-Based Retrieval techniques[73, 62]. Solutions for all these problems will help in automatic understanding and analysis of the visual data by machines.

1.1 Challenges

Categorization/Retrieval of the visual data is very challenging due to the wide range of variations possible for images belonging to a category of interest. Some of these possible variations are explained as follows:

- *Transformations and varying view angle:* Images of the same scene can go through different affine transformations like rotation, scale and shear. These kind of transformations can be induced in the process of data acquisition (ex: changes in the camera angle) or by the post-processing of videos.
- *Illumination:* Lighting conditions of the environment will have a profound influence on the appearance of a scene/object. Images taken of the same location can have different illumination, which mainly depends on the time and the other objects in and around the location. There can also be shadows depending on the position of light sources, which changes the appearance of the scene/object. These differences in appearance of the scenes/objects due to change in lighting conditions makes the task of visual categorization/retrieval very much difficult. Images of the "cityscape" category taken at different lighting conditions can be seen in the **Figure** 1.2

Figure 1.1 Intra-class variations present in the images belonging to the category "Boat/Ship". You can see that there are a lot of variations possible visually, even though they all have the same semantic meaning according to the humans.

Figure 1.2 Effect of illumination on the images of "Cityscape" category. Because of the change in lighting conditions, the task of image categorization becomes more challenging.

Figure 1.3 Images of "Car" category having different poses. This is another challenging aspect of image classification as can have variations in both, model and pose of the car.





Figure 1.4 Effect of Occlusions and Truncations: Here you can find some of the objects with some of their parts either occluded by other objects or not covered in the image acquisition process. Image classification needs to be robust to handle this problem.

- *Semantic Variations:* There can be lot of variations in the appearance of objects/scenes belonging to the same category (intra-class variations). For example, we can see the diversity of the images belonging to the category "cars" in the **Figure** 1.3. Similar example can be seen for the "boat/ship" category in the **Figure** 1.1. Also there can be lot of similarities between the images of objects/scenes belonging to different categories(inter-class variations). These similarities can be in any of their features like color, shape, texture etc.,
- *Occlusions:* Truncations and occlusions result in the invisibility of important parts of the object in the images/videos, which makes it difficult to recognize them. This problem can be seen from examples shown in **Figure** 1.4.

Other variations include complex backgrounds (sometimes background can be similar to the object of interest), articulations etc.,

1.2 Applications

Some of the applications where classification/retrieval of visual data can be very useful are given below.

1. *Broadcast Video Search:* With the various developments in storage devices, large volumes of broadcast data can be recorded and archived. During many points of time, people may have to manually browse all these video archives in order to find their content of interest. Automatic labelling of this data with appropriate tags can be useful for faster and easier retrieval of "content of interest" from this large scale data.
2. *Web data Indexing:* There is a large volume of images and videos present over the web. With the advent of many multimedia sharing websites like Flickr[24], Youtube[80], Picasa[54] etc., the need for good search engines has become necessary. Most of the present multimedia search engines depend on the meta information like filename and text around the images. This information may not describe the content properly most of the times, resulting in the "incorrect" results for a query in the search engines.
3. *Security:* Because of the availability of advanced CCTV cameras for surveillance, there is a lot of visual data available. This data needs to be continuously monitored for security purposes. Object

detection techniques will be very much helpful for detecting objects of interest in these videos. Also, object classification/detection techniques can be helpful in the design of intelligent parking systems.

4. *Robotic Systems:* Object/scene classification can be very useful for the design of autonomous vehicles, which can take required decisions based on the category detected. These kind of robotic systems can be very useful in order to give warning messages prior to any dangers.
5. *Human Computer Interaction:* Object detection and retrieval techniques can be very much useful in providing advanced input methods. It presents a new way of providing automatic inputs for the computers. For example, one can show the object to the webcam attached to the camera and search for the details of the object using various object detection and retrieval techniques.
6. *Others:* Some of the other applications of object classification/detection/retrieval include automation in manufacturing industries, intelligent traffic monitoring and analysis, clinical diagnosis, satellite image processing etc.,

1.3 Contributions

In this thesis, we start by showing the utility of the state-of-the-art image classification techniques for indexing large scale image and video data. This is demonstrated on TRECVID 2008 and TRECVID 2009 video collections for retrieval of various scene, object and action categories.

One of the ideas that we propose in thesis is the use of sub-categories. The objective of this investigation is to obtain better classification performance with computationally inexpensive kernels. The basis for our idea is that there exist natural groups or sub-categories for a given category. Our idea is to find classifiers for each of these groups separately and combine the outputs of the classifiers for each of these subcategories. We have used structured-SVM [69] framework for doing this. We show experimentally how the choice of these subcategories can effect the performance. We also present a method to automatically find these groupings [81].

Another idea that we propose as a part of this thesis is the generalized RBF feature maps for speeding up the training and testing time of SVM. Though SVM classifiers with non-linear kernels have shown a good performance, they are computationally expensive. Most of the CPU cycles during training and testing are spent in the computation of the non-linear kernel function between the pairs of samples. There have been methods to speedup the training and testing time for RBF kernels[55] and homogeneous family of kernels[75]. In this thesis, we propose a method to speedup the training and testing for the generalized RBF-kernels by combining these two ideas.

Our contributions through this thesis can be summarized as follows:

- Demonstrated the utility of efficient classification methods for indexing large scale image and video data. We validate the results on TRECVID 2008 and 2009 video collections.
- Showed how modeling of subcategories can be useful for improving the performance of the linear classifiers with experiments on large datasets.

- Proposed generalized RBF-feature maps for efficient object detection and classification.

1.4 Organization

The remaining part of this thesis is organized as follows. In the Chapter 2, we shall present the technical background about features and support vector machines. Then, we discuss each of our contributions in three separate chapters. In Chapter 3, we shall present the utility of efficient classification methods for large scale video retrieval purposes on TRECVID video collections. Investigations about the usage of subcategories for performance improvements is presented in Chapter 4. We propose generalized RBF-feature maps and their utility for efficient detection in the Chapter 5. Finally, we draw conclusions from this thesis in Chapter 5.

Chapter 2

Background

This chapter helps in gaining technical background for understanding next chapters. Image categorization mainly involves the use of Computer Vision and Machine Learning techniques. Many of the architectures for image classification in literature consists of three major phases, ***i)*** *Feature Extraction* and ***ii)*** *Training* and ***iii)*** *Testing*. Feature Extraction phase involves the computation of "feature representations" which characterize a given image. A simple example of feature representation is *Color Histogram*, which captures the distribution of different color quantization levels in the given image. There can be lot of unimportant data in images which cannot be used for discriminating it from other scenes/objects. Good features are the ones which can describe discriminative properties of a category. Many feature representations proposed in literature are designed to handle one or more of the above mentioned variations. Also, for each of the categories, different set of features can be useful. For example, a shape feature can be useful for classification between bananas and apples, but it is not a good feature for classification between apples and peaches. Some of the successful features for object/scene classification tasks can be divided into two types, *Global features* and *Local features*.

Global features are computed over the entire image, where as local features are computed over the local regions of an image. Global features are more sensitive to image variations, such as occlusion and viewpoint variations. Examples for global representations are color histograms [64], eigen-spaces [48] and GIST[67]. Examples for local representations are Bag of Words(BoW) representation [13], Histogram of Oriented Gradients(HoG) [12]. Extraction of local feature representation involves two steps, finding interesting local regions in an image, and computing descriptors for these regions. A well known interest point detector and descriptor is the Scale Invariant Feature Transform (SIFT) [39]. Traditionally, local features are computed around interest points like corners or edge points. Good detectors are the ones which can detect an interest point independent of the imaging conditions. Other alternatives for detection of interesting regions is to use a densely sampled grid of regions in the image. This method is shown to be successful for image classification tasks[73, 13]. One of the disadvantages with dense sampling is that it requires generation of many descriptors. Local feature representations are found to be more robust for visual classification/retrieval tasks. Especially SIFT-based visual word representa-

tions are shown to be one of the most successful for image classification, detection and retrieval tasks [60, 13, 73]. Details about feature extraction methods used in this thesis are explained in Section 2.1.

Image Classification methods can be broadly divided into two types i) *Generative approaches* ii) *Discriminative approaches*

Examples for generative classifiers include Probabilistic Latent Semantic Analysis(pLSA) [29], Latent Dirichlet Allocation(LDA) [14, 20], etc., Examples for discriminative approaches include Support Vector Machines(SVM) [11, 13], Adaboost [25, 65], etc. We have used SVMs as classifiers in this thesis. SVM classifiers have shown a great success for image classification tasks. They are designed with the intuition of maximizing the *margin*, which is the maximum possible distance between the samples near to the boundary of the classifier. SVMs have become successful mainly due to their ability of separating non-linear input spaces. This is made possible with the use of a mapping function which transforms the feature space to a large-dimensional space, where the samples can be easily divided. SVM optimization functions mainly involve the dot product of this feature transform. The problem of computing this dot product even in the infinite dimensions is made easier by the functions called as *kernel functions* which help in overcoming the computation of the feature transform. These kernel functions help in finding a discriminative classifier in a higher dimensional space.

In training phase of the classification, a function(model) which can differentiate between the given two classes is learnt using the labelled training examples. Note that, we are referring to the case of binary classification, where we need to classify any given image into only one of the two categories. The other kind of classification, where a given sample is to be assigned to one of the n-classes($n \geq 2$) is called as multi-class classification. More details about multi-class classification is discussed in the Chapter 2. In this thesis we focus on the training methods which use batch learning way, i.e., if there is a new sample, training process starts from the beginning on the set augmented with the new sample. In testing phase of the classification, the learnt function is evaluated on the given test sample to find its label.

The other problem which we deal with in this thesis is that of object detection, which aims to find a bounding box along with the label of the object. It is a tougher task compared to Image Classification. Many of the methods for object detection use sliding-window based classifiers. More specifically, these classifiers search over windows of all possible scales and aspect ratios over the complete image. In order to speedup this process, there have been methods to cut down the search space of the possible windows in the image [33]. Vedaldi *et al.* [73] have proposed a cascade of classifiers, in which the computational complexity of classifier increases as we move along the layers of the cascade. This helps in filtering most of the samples in the first layer using a low-computational complexity classifier. We have used this cascade-based framework for object detection in this thesis.

Literature of works designed for improving the performance of image classification comprises of two categories, a) those which focus on designing better feature representations and b) those which focus on designing better classifiers.Note that the methods designed to improve performance of image classification are applicable to object detection. Gemert *et al.* [26] have proposed a soft-histogram representations for image classification. Some of the other works which focused on better feature representation for im-

age classification include [36, 35, 51, 32]. Varma and Ray [71] presented a method to combine multiple sources of features in a Multiple Kernel Learning framework. It was shown that it results in a good performance in [73, 8] by using multiple sources of information. Other research has been to design techniques to speedup the computation of local image descriptors. Some of these include SURF [4], GLOH [46] and DAISY [16]. Faster methods of computing robust feature representations is very much important to perform the tasks of image classification and retrieval in real-time.

Bosch *et al.* [8] were successful in improving the performance by using a hybrid approach, which combines both discriminative and generative based classifiers. Other direction in order to improve the image classification and detection approaches includes the use of context [49].

Now, we describe in detail various image representations that we use in our experiments for image classification. Later on, we discuss in detail about Support Vector Machines (SVM) classifiers and Kernel Methods which form the backbone of this thesis.

2.1 Image Representation

Raw pixel information of images alone may not be enough for image classification tasks. For example, a pixel-wise comparison between two images that humans would consider very similar can be treated as very different when measured by some distance measures (e.g. Euclidean norm). The reason for this is that the raw pixel representation lacks many invariances. For example, translations, small rotations, small changes in size, blur, brightness, pose and contrast are the factors that humans consider as irrelevant while judging if two images are same. Therefore, the input image data has to be transformed into a meaningful set of *features*. This process of extracting a meaningful set of features from a given data is called in general as *feature extraction*. A good feature representation of images for classification tasks should be

1. *invariant for the images of same "semantic" category* and
2. *discriminant between the images of two different categories*.

In literature, many features are proposed for the purpose of image classification. We explain some of these features that we used in our experiments in later chapters.

2.1.1 GIST

GIST descriptor was first proposed by Oliva & Torralba [50] for scene classification. It is a global scene descriptor, which is based on a set of perceptual dimensions: naturalness (vs. man-made), openness (presence of a horizon line), roughness (fractal complexity), expansion (perspective in man-made scenes), ruggedness (deviation from the horizon in natural scenes) that represent the dominant spatial structure of a scene. These perceptual qualities are together referred as the Spatial Envelope. It has been shown by [50] that spatial envelope properties can be reliably estimated using spectral and coarsely localized information.

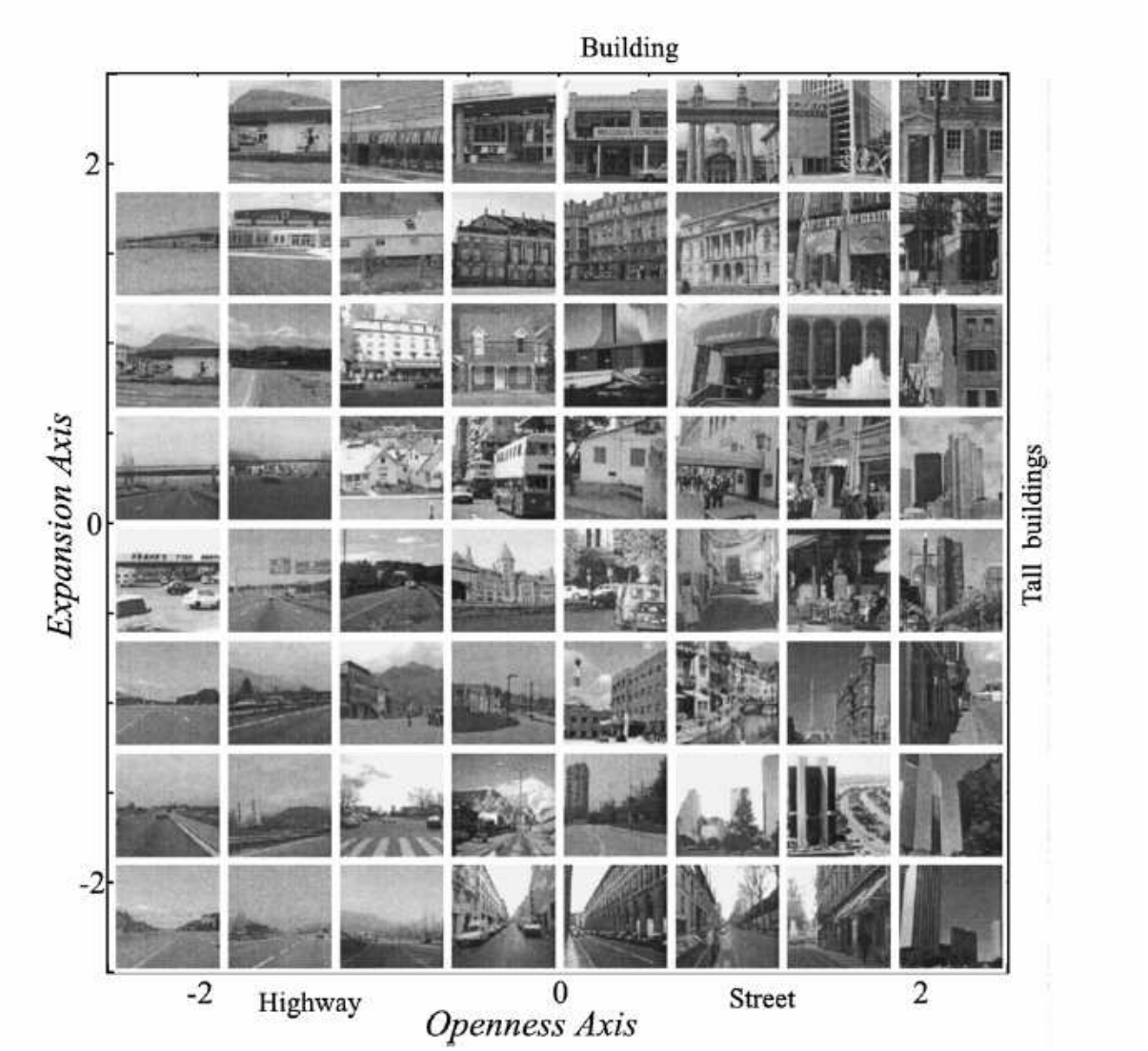

Figure 2.1 Images of Man-made environments when plotted along semantic degrees of openness and expansion (from[50])

Computation: At first, a given image is initially divided into a $m \times m$ (usually $m = 4$) grid of non-overlapping windows. Then for each cell in the grid, a set of filters (of different orientations at different scales) are applied. Each dimension of the final GIST descriptor for a cell corresponds to the average of the output of filter response on that cell. GIST descriptor for the complete image is then obtained by concatenating the GIST descriptors of all the cells. In summary, GIST feature representation for an image is a vector g, where each individual dimension g_k is computed as

$$g_k = \sum_{x,y} w_k(x,y) \times \|I(x,y) \otimes h_k(x,y)\|^2 \tag{2.1}$$

where $\otimes$ denotes image convolution and $\times$ is a pixel-wise multiplication. $I(x,y)$ is the intensity value of the input image at point (x,y), and $h_k(x,y)$ is a filter from a bank of multi-scale oriented Gabor filters (say N_o orientations and N_s scales) and $w_k(x,y)$ is a spatial window that will compute the average output energy of each filter at different image locations. The windows $w_k(x,y)$ divide the image in a grid of $m \times m$ non-overlapping windows. The dimension of the resultant descriptor will be $(m \times m \times N_o \times N_s)$.

2.1.2 Appearance Based Representations (BoW & PHOWGray)

Bag of Words representation (also known as bag-of-features) was first proposed for solving problems related to Natural Language Processing(NLP) and Information Retrieval. In NLP domain, a text document using this model is represented as an unordered collection of words, disregarding the grammar rules and order of words. For example, "this is a thesis" and "thesis this a is" are considered to be same under this model. Similar approach was first used in the computer-vision domain for the problem of texture recognition[38, 72]. The goal of texture recognition is to recognize textures captured from different camera viewpoints, and under varying illumination. At first, Leungand and Malik[38] quantized responses of a filter bank applied densely over an entire image. These quantizations of appearance descriptors are called "textons" and textures are represented by distributions of textons. Varma and Zisserman [72] modified this approach by quantizing small image patches rather than filter responses. BoW representation was later used for the tasks of content-based image retrieval [60]. In the recent past, many works using bag-of-words have shown impressive levels of performance for object/scene classification tasks[22, 13, 20, 73].

In computer vision, BoW representation is based on the analogy of considering "images" as "documents" and "image patches" as "words in the documents". These image patches are specifically represented by using *visual words*, which are formed by vector quantizing the visual features (color, texture, etc.,) like local region descriptors. Success of BoW methods for vision tasks can be mainly attributed to the use of local region descriptors. This is because, local regions are more robust to occlusions and spatial variations. Construction of "Bag-of-Words" representation from the images mainly involves the following steps:

1. Finding Regions of interests

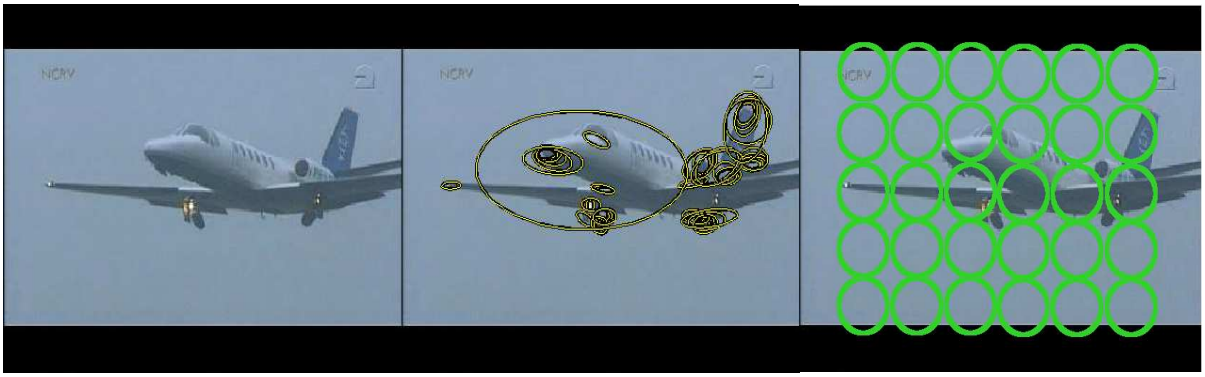

Figure 2.2 a) Original Image b) Original image overlayed with Harris-Affine Regions c) Figure illustrating the computation of dense descriptors

2. Computing local descriptors for regions of interest
3. Vector quantization of the local descriptors into "visual words" to form a *visual vocabulary*
4. Assigning the nearest visual word for each region descriptor

For the tasks of image-classification, bag-of-words are represented with a histogram of the visual words in the given image, which captures the distribution of different visual words in a given image. Histograms help in allowing variations in the positions of the patches in an image and also to have a fixed length representation. Each of these steps required for computing BoW are explained as follows:

Finding Regions of Interest In general, process of finding "regions of interest" is referred to as *Feature detection* and the descriptors computed for these regions are called as *feature descriptors*. Given an image, feature detection involves extraction of "regions of interest" (local patches). Some of the region detectors present in the literature are *i*) Harris Points [28] *ii*) Harris-Laplace regions [44] *iii*) Hessian-Laplace regions [45, 39] *iv*) Harris-Affine [45] *v*) Hessian-Affine [45, 47] *vi*) Maximally Stable Extremal Regions [43] . Many of these region detectors use different image measurements and find regions which are invariant to some properties like scale, rotation and affine transformations. The total number of interesting regions detected in an image depends on the content present in the image. **Figure** 2.2**a** and **Figure** 2.2**b** show the original image containing airplane and the interest points detected using a Harris-Affine detector [45].

Instead of finding these "interesting regions", people have shown that using regions on a regular grid of the image as interesting regions is effective for image classification [20]. In this method, the regions of interest are selected on an evenly sampled grid spaced at $n \times n$ pixels for a given image. The descriptor computed in this manner are referred to as dense descriptors. **Figure** 2.2**c** illustrates the computation of dense descriptors.

Computing local descriptors for regions of interest A descriptor for each of the regions of interest is computed in this step. In this thesis, we use SIFT[39] descriptors as local region descriptors. SIFT

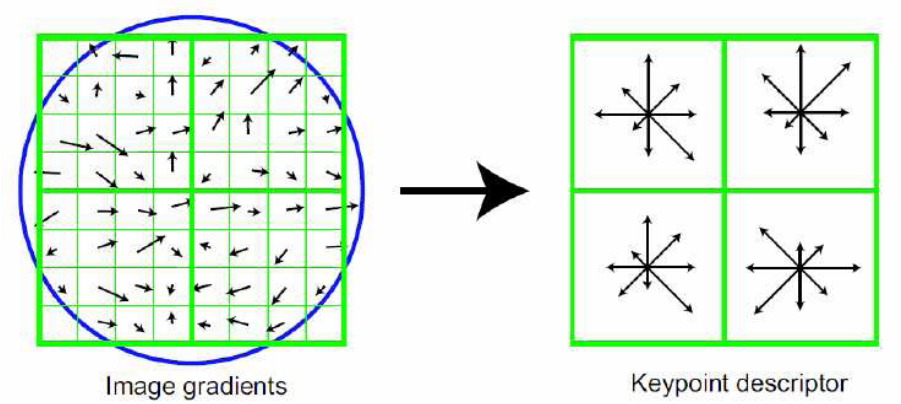

Figure 2.3 The SIFT descriptor of [39]. On the left are the gradients of an image patch. The blue circle indicates the Gaussian center-weighting. These samples are then accumulated into orientation histograms summarizing the contents over 4×4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes along that direction within the region. A 2×2 descriptor array computed from an $8x8$ set of samples is shown here.

descriptors are computed for the normalized image patches. SIFT descriptor is a 3D histogram of gradient location and orientation, where location is quantized into a 4×4 location grid and the gradient angle is quantized into 8 orientations. The resulting descriptor is of dimension 128. Each orientation plane represents the gradient magnitude corresponding to a given orientation. To obtain illumination invariance, the descriptor is normalized by the square root of the sum of squared components. **Figure** 2.3 illustrates the approach of computing SIFT descriptor. Many other variations of SIFT feature descriptors like GLOH [46], PCA-SIFT, SURF [4] are proposed in literature of which, many are mainly designed for faster computations.

Visual Vocabulary: Feature descriptors(local region descriptors) computed for different images are clustered using a clustering algorithm like *k-means* to obtain a set of cluster centers referred as "visual-words". The set of "visual-words" is together referred to as a "visual vocabulary".

Histograms of visual words Once we obtain a vocabulary, we assign the nearest visual word from the vocabulary for each region descriptor in the image. So, now we will have a set of visual words occurring in the image. Histogram of visual words for the given image captures the distribution of different visual words in vocabulary. One can also use a soft assignment in preparation of histograms, which can give better results as reported in [52, 26] **Figure** 2.5 schematically describes the steps involved in the constructing the bag-of-words model for image-classification.

Spatial information can be a valuable source of information for discrimination of many classes(for e.g., sky is usually present on the top region showing a mountain/cityscape category). Unfortunately, bag-of-words representation discards all information about spatial structure present in the image. For this purpose, many methods have been proposed to incorporate this spatial information [57, 37]. Therefore, to augment bag-of-features with global spatial information, Lazebnik et al. proposed a method for

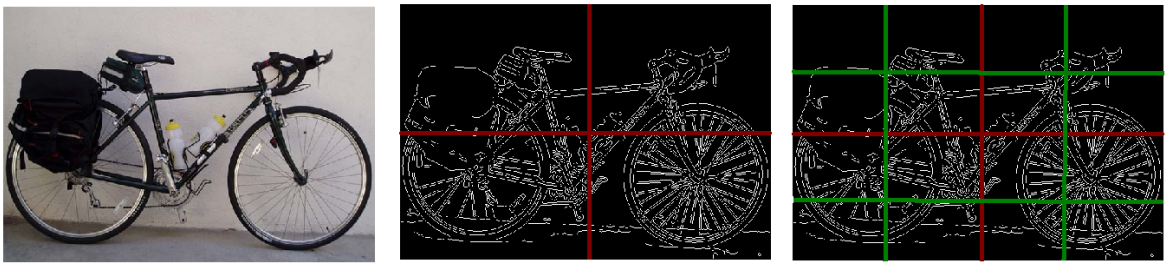

Figure 2.4 Example spatial pyramid division

including spatial information in BoW [37]. In this approach, instead of a representing the whole image with a single global "histogram of visual words", a number of local histograms are formed, typically in a pyramid structure from coarse to fine. This pyramid structure, which is a multi-level recursive image decomposition can be seen in **Figure** 2.4. At *level 0*, the decomposition has only single cell(the whole image). And for each l^{th} level, $l > 0$, each cell of the $(l-1)^{th}$ is divided into four sub-quadrants. Therefore l^{th}-level in the spatial pyramid will have 4^l cells.

To summarize, the representation of Pyramid Histogram of visual Words is a concatenation of the histograms of different levels into a single vector. In this thesis, we have used three levels for the pyramid representation. The distance between the two PHOW descriptors reflects the extent to which the images contain similar appearance and the extent to which the appearances correspond in their spatial layout.

2.1.3 Shape Descriptors (HOG & PHOG)

PHOG descriptor, which is also referred as Shape descriptor was proposed by Bosch *et. al* [7] for the purpose of image classification. It was designed with an objective of describing an image with its local shape and the spatial layout of the shape. It captures local shape by using the distribution of edge orientations within a region. The spatial layout is captured by tiling the image into regions at multiple resolutions. The final descriptor is a concatenation of a histogram of orientation gradients over each subregion of the image at different pyramid levels, hence called as *Pyramid - Histogram of Gradients*. This descriptor is similar to the *Histogram of Gradients(HoG)* which gave promising results for pedestrian detection [12]. The main differences are that HoG is computed in a more dense manner and involves normalization of the histograms grouped by multiple cells.

Computation: An edge detector(e.g. canny detector) is first applied on the given image. PHOG descriptor is the concatenation of "histogram of edge orientations" computed over cells defined by a multi-level recursive image decomposition similar to the PHOW descriptor [37] mentioned in the above section. And for a given cell in the image, its local shape is represented with a histogram of edge orientations quantized into b bins, where each bin of the histogram corresponds to the number of edges that

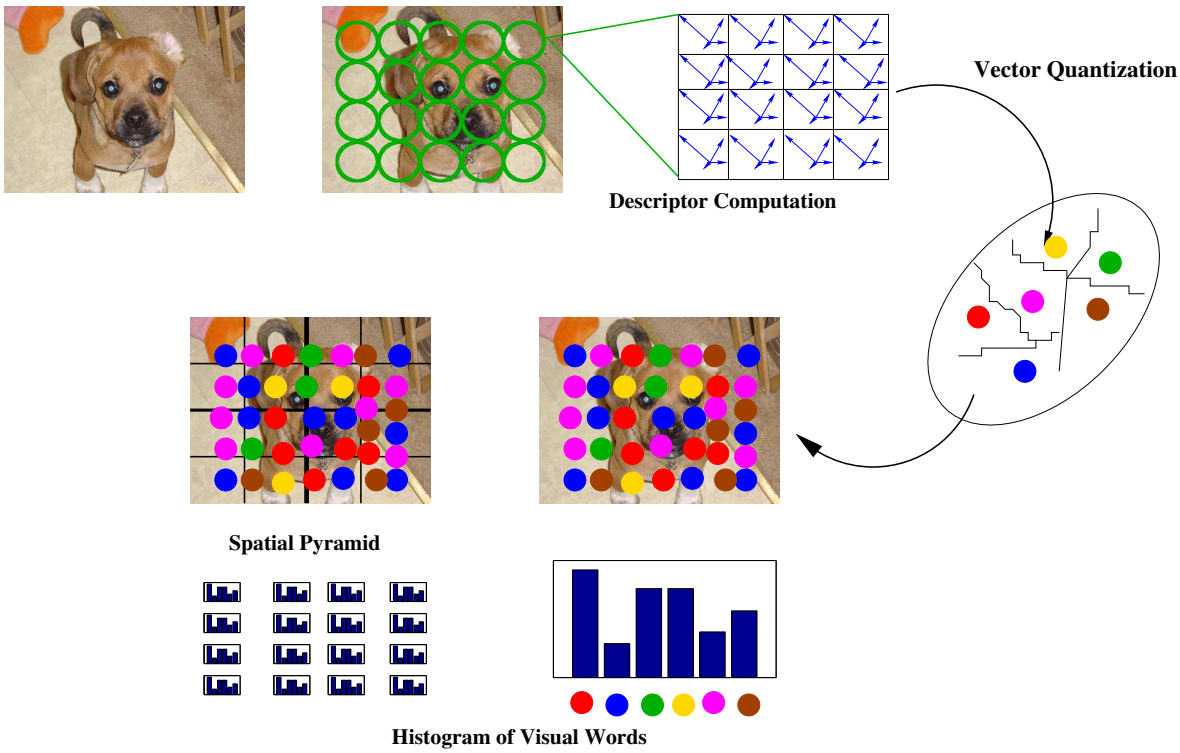

Figure 2.5 Complete pipeline of preparing "Histogram of Visual Words" for a given image

have orientations belonging to a certain range. The contribution of each edge is weighted according to its magnitude, with a soft assignment to neighboring bins. There are two variants for this descriptor:

1. *PHOG 180/*$Shape_{180}$: In this case, orientations in the range [0,180] are considered i.e., the contrast sign of the gradient is ignored.
2. *PHOG 360/*$Shape_{360}$: In this case, orientations in the complete range [0,360] are considered.

2.2 Support Vector Machines (SVM)

Classification of the given data is a common task in machine learning. In a classification problem, the learner approximates a function which can map a given vector data into one of the various class labels. In supervised setting, it is done by looking at a set of input-output examples of the function. The finite input-output example data which is used for learning the classification function is called the *training data*.

Support Vector Machines(SVM) is one of the successful supervised learning methods for this problem. They have strong theoretical foundations and have shown excellent empirical success in various fields. Support Vector Machines are trained so that the decision function would classify the unseen example data accurately. This ability to classify unseen example data accurately is referred to as *generalization*. High generalization capability is one of the main reasons for the success of SVMs. From now

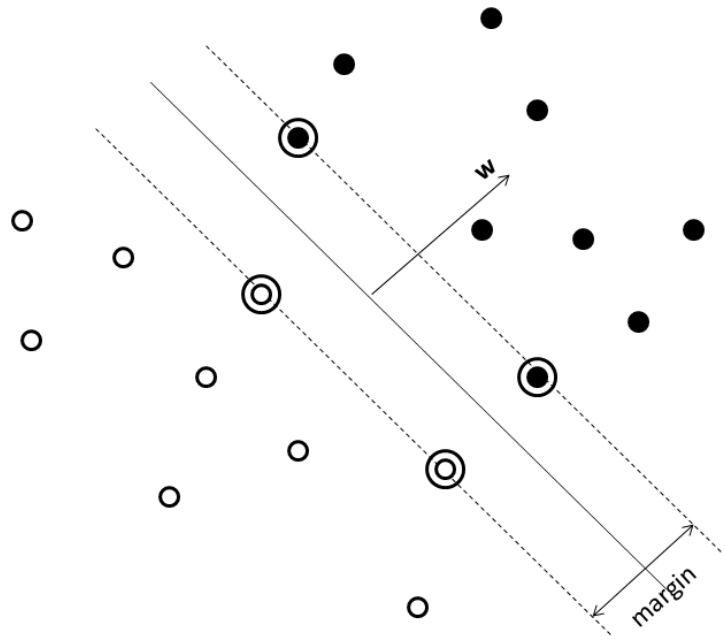

Figure 2.6 Example showing the margin and support vectors in the case of linearly separable data

on, we would be looking at a two-class case (+ve class and -ve class) unless specified. Lets see the basic idea behind the SVMs at first. Given a set of a d-dimensional vectors, a *linear classifier* tries to separate them with a $(d-1)$-dimensional hyperplane. There are many hyperplanes that might classify the data. If we define "margin" as the distance between the nearest samples on both sides of the hyperplane, SVMs are designed to choose the hyperplane that has the largest margin between the two classes. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines is known as a *maximum margin classifier*. We shall review the basic theory of SVMs for different cases in the following sections.

Notations: To describe the task in mathematical terms, we introduce the following notations

- an example data point is denoted by $x \in R^d$,
- class membership for a data point is denoted by $y \in \{-1, +1\}$,
- the set of training examples is denoted by $X = \{x_1, \ldots, x_n\}$,
- class labels for the training set is denoted by $Y = \{y_1, \ldots, y_n\}$

2.2.1 Linear Support Vector Machines: Hard-Margin case

At first, lets see the case when the data can be linearly separable. From the training data, we would like to learn a classification function $F : \mathbb{R}^d \rightarrow \{-1, +1\}$ $i.e.$, a function that decides the class of a given example x.

In SVMs, F is chosen as the sign of a linear function i.e.,

$$F = sign(f(x)) \tag{2.2}$$

where $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is the decision function. This decision function is based on the hyperplane separating the two classes. Any hyperplane separating the two classes will be of the form :

$$w^1x^1 + w^2x^2 + \ldots w^dx^d + b = 0 \tag{2.3}$$

where $x^1, \ldots, x^d$ are the components of x, and $w^1, \ldots, w^d$ are the coefficients of the weight vector w and b is the bias. Therefore, decision function f can be written as

$$f(x) = \langle w, x \rangle + b \text{ , where } \langle ., . \rangle \text{ denotes the dot product} \tag{2.4}$$

The points x_i which lie on the hyperplane satisfy $\langle w, x \rangle + b = 0$, where w is normal to the hyperplane, $\frac{b}{\|w\|}$ is the perpendicular distance from the hyperplane to the origin, and $\|w\|$ is the Euclidean norm of w. Let d^+ and d^- be the shortest distance from the separating hyperplane to the closest positive and negative example respectively. The *margin* of a separating hyperplane is then defined as $d^+ + d^-$. For the linearly separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin. As the data is linearly separable in this case, we can select two hyperplanes in a way that there are no points between them and maximize the distance between them. These hyperplanes can be written as follows

$$\langle w, x \rangle + b = 1$$

$$\langle w, x \rangle + b = -1$$

The margin which is the distance between those hyperplanes will be equal to $\frac{2}{\|w\|}$. As we want to maximize the margin, we want to minimize the term $\|w\|$. It would be difficult to solve min$\|w\|$ because of the square root involved in calculation of $\|w\|$. Therefore, min$\|w\|^2$ is used as the optimization problem to make it easier. Also, as we do not want any data points falling into the margin, we add the following constraints

$$\langle w, x \rangle + b \geq 1 \; \forall x_i \quad \text{with} \quad y_i = 1$$

and

$$\langle w, x \rangle + b \leq -1 \; \forall x_i \quad \text{with} \quad y_i = -1$$

After combining the above inequality constraints into a single constraint, we will have the following primal optimization problem. This is a quadratic programming problem.

$$\min_{w,b} \frac{1}{2}\|w\|^2 \tag{2.5}$$

subject to

$$y_i(\langle w, x_i \rangle + b) \geq 1 \tag{2.6}$$

Note that factor $\frac{1}{2}$ is used for mathematical convenience. The Lagrangian formulation of the above problem which replaces the inequality constraints with equality constraints.

$$L_P = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\langle w, x_i \rangle + b) - 1] \tag{2.7}$$

where $\alpha_i \geq 0$, $\forall i$ are lagrangian multipliers for each of the inequality constraints **Equation** (2.6). The above Lagrangian is maximized with respect to α_i, and minimized with respect to w and b. Consequently, at this saddle point, the derivatives of L with respect to primal variables must vanish,

$$\frac{\partial}{\partial b} L_P(w, b, \alpha) = 0$$

$$\frac{\partial}{\partial w} L_P(w, b, \alpha) = 0$$

$$\alpha_i [y_i(\langle w, x_i \rangle + b) - 1] = 0 \forall i$$

$$\alpha_i \geq 0$$

which leads to

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.8}$$

and

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{2.9}$$

This problem is solved by using standard quadratic programming techniques. It can also be solved by solving its dual problem, which is easier and gives the same solutions as the one obtained by solving the primal version. Due to the convexity of the primal optimization problem, the solution is unique but the coefficients α_i need not be unique. The points for which α_i are non-zero are called as *support vectors*. **Figure** 2.6 summarizes the situation for a 2-dimensional data, plotting support vector points with extra circles around them. Dual form of the optimization problem is written as

$$L_D = \max_{\alpha \in \mathbb{R}^n} \{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{2.10}$$

$$\text{subject to } \alpha_i \geq 0, i = 1, \ldots, n \tag{2.11}$$

$$\text{and } \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.12}$$

If we substitute the **Equation** (2.9) in the original decision function **Equation** (2.4), then we have

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i x_i^T x + b \tag{2.13}$$

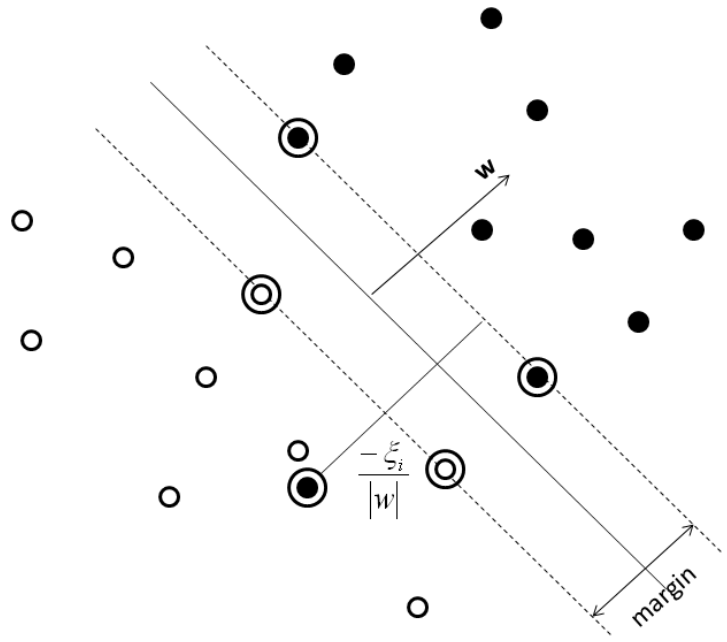

Figure 2.7 Example showing the case where linear-hyperplane cannot be drawn

2.2.2 Linear Support Vector Machines: Soft-Margin case

Till now, we showed how SVMs are designed when that the training data is linearly separable. When the data is not linearly separable, there is no feasible solution and hard-margin SVM is unsolvable. An example of this can be found in the **Figure** 2.7. This section shows how hard SVMs are modified to apply them in inseparable case. In order to assign the penalty on the errors, *non-negative slack variables* $\xi_i \geq 0, i = 1, \ldots, n$ are introduced in the **Equation** (2.6) as follows

$$(y_i \langle w.x_i \rangle + b) \geq 1 - \xi_i \text{ for } i = 1, \ldots, n \tag{2.14}$$

$$\xi_i \geq 0 \forall i \tag{2.15}$$

For any point in the training data x_i (**Figure** 2.7), if $0 < \xi_i < 1$, the data do not have the maximum margin but are still correctly classified. But if $\xi_i \geq 1$, the data are misclassified by the optimal hyperplane. Therefore, we have $\sum_i \xi_i$ as an upper bound on the number of training errors. In order to assign an extra cost for these errors, the objective function to be minimized will be

$$Q(w, b, \xi) = \|w\|^2 + C(\sum_i \xi_i^k) \tag{2.16}$$

$$\text{subject to } y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i \quad \text{for } i = 1, \ldots, n \tag{2.17}$$

In the above equations, C is the margin-parameter to be chosen by the user, which determines the trade-off between the maximization of the margin and minimization of the classification error. This is a convex programming problem for any positive integer k; for $k = 2$ and $k = 1$ it is also a quadratic

programming problem, and the choice $k = 1$ has the further advantage that neither the ξ_i, nor their Lagrange multipliers, appear in the dual optimization problem.

We call the obtained hyperplane the soft-margin hyperplane. When $k = 1$, we call the support vector machine as the $L1$ soft-margin support vector machine and when $k = 2$, the L2 soft-margin support vector machine. First we shall discuss $L1$ soft-margin support vector machines. Similar to the case of separable case, at first lagrangian multipliers are introduced in the optimization function as follows

$$L_P(w, b, \xi, \alpha, \beta) \equiv \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(\langle w, x\rangle + b) - 1 + \xi_i) - \sum_{i=1}^{n}\beta_i\xi_i \tag{2.18}$$

where $\alpha_i, \beta_i, \forall i$ are non-negative Lagrangian multipliers. For optimal solution, the following Karush-Kuhn-Tucker(KKT) conditions should be satisfied

$$\frac{\partial L_P(w, b, \xi, \alpha, \beta)}{\partial w} = 0 \tag{2.19}$$

$$\frac{\partial L_P(w, b, \xi, \alpha, \beta)}{\partial b} = 0 \tag{2.20}$$

$$\frac{\partial L_P(w, b, \xi, \alpha, \beta)}{\partial \xi} = 0 \tag{2.21}$$

$$\alpha_i(y_i(\langle w, x_i\rangle + b) - 1 + \xi_i) = 0 \text{ for } i = 1, \ldots, n \tag{2.22}$$

$$\beta_i\xi_i = 0 \text{ for } i = 1, \ldots, n \tag{2.23}$$

$$\alpha_i \geq 0\ ,\ \beta_i \geq 0\ ,\ \xi_i \geq 0 \text{ for } i = 1, \ldots, n \tag{2.24}$$

Applying Equations (2.19) to (2.21) on **Equation** (2.18), we have the following

$$w = \sum_{i=1}^{n}\alpha_i y_i x_i \tag{2.25}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0\ , \tag{2.26}$$

$$\alpha_i + \beta_i = C \text{ for } i = 1, \ldots, n \tag{2.27}$$

Substituting, the above into **Equation** (2.18), we obtain the following dual problem,

$$L_D(\alpha) \equiv \max \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j\langle x_i, x_j\rangle y_i y_j \tag{2.28}$$

$$\text{subject to the constraints: } 0 \leq \alpha_i \leq C\ , \tag{2.29}$$

$$\sum_i \alpha_i y_i = 0 \text{ for } i = 1, \ldots, n \tag{2.30}$$

Not that the main difference now is that the α_i is now bounded by C. Substituting the **Equation** (2.25) in **Equation** (2.4) our decision function will become

$$f(x) = \sum_{i=0}^{n} \alpha_i y_i \langle x_i, x \rangle + b \tag{2.31}$$

where nV is the number of training samples out of which, the feature vectors for x_i, for which α_i is non-zero are called as support vectors.

2.2.3 Non-Linear Support Vector Machines

Till now, we have seen how to compute a large-margin hyperplane and that it is good due to its generalization ability. But there is still a major drawback, as whatever we have done so far is linear in the data. In order to use much more general decision surfaces, the input data $\{x_1, \ldots, x_n\} \in X$ is transformed into a high-dimensional feature space, using a non-linear map $\psi : \mathbb{R}^d \mapsto \mathcal{F};$. A linear classifier is then found in the transformed high-dimensional space. The only requirement of $\mathcal{F}$ is that dot product can be defined in that space. It can be an infinite-dimensional space and no assumptions are made on the dimensionality of $\mathcal{F}$. For a given training data set, SVM is now constructed in $\mathcal{F}$ instead of $\mathbb{R}^d$ *i.e.*, using the set of examples

$$\{\psi(x_1), y_1\}, \ldots, \{\psi(x_n), y_n\} \in \mathbb{R}^N \text{ x } \{\pm 1\} \tag{2.32}$$

Using this mapped set of examples, we need to estimate the decision function in $\mathcal{F}$. Intuitively, the difficulty of constructing a decision function in input space should grow with the dimension of the patterns, which is called as the *curse of dimensionality*.

So now, if we substitute x with $\psi(x)$ in Equations (2.28) and (2.31) , our optimization function will be

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \psi(x_i), \psi(x_j) \rangle \tag{2.33}$$

and decision function will be

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle \psi(x_i), \psi(x) \rangle + b \tag{2.34}$$

Note that the above equations require only the dot product of the feature vectors in the transformed high-dimensional space. These expensive calculations are reduced significantly by using *"kernel-trick"*. Computation of the feature-map is bypassed by using a kernel function k which results in the dot product of data points in the transformed space.

Instead of making a non-linear transformation of the input vectors followed by dot products with support vectors in the high-dimensional space $\mathcal{F}$, the order of operations is interchanged. A comparison is first done between two vectors in the input space, and then a non-linear transformation (*ex:* by taking their dot product or some distance measure) of the result is made. Training a non-linear SVM which

requires the computation of the dot products $\langle \psi(x_i), \psi(x_j) \rangle$ in the transformed space can be reduced by defining a suitable kernel function k, such that

$$k(x_i, x_j) = \langle \psi(x_i), \psi(x_j) \rangle \tag{2.35}$$

By constructing a matrix Q, such that $(Q)_{ij} = y_i y_j k(x_i, x_j)$, the above optimization problem in **Equation** (2.33) is written as

$$\max_{\alpha} \alpha^T 1 - \frac{1}{2} \alpha^T Q \alpha \tag{2.36}$$

$$\text{subject to } \alpha^T y = 0 \tag{2.37}$$

$$\alpha \geq 0 \tag{2.38}$$

$$C\overline{1} - \alpha \geq 0 \tag{2.39}$$

and the decision function **Equation** (2.34) is re-written as follows

$$f(x) = \sum_{i=1}^{nSV} \alpha_i y_i k(\psi(x_i), \psi(x)) + b \tag{2.40}$$

What is a valid kernel ? Now the question is that which function k corresponds to a dot product in some feature space $\mathcal{F}$. In other words, how can we find a map ψ, such that kernel function k computes the dot product in the space mapped by the function ψ. The answer to this question is given by the Mercer's theorem. Before going to Mercer's theorem, lets see the following definitions

Positive Definite Kernel Function: Let $\mathcal{X}$ be a non-empty set. A function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is called positive definite kernel function, if

- k is symmetric, i.e., $k(x, y) = k(y, x)$ for all $x, y \in \mathcal{X}$.
- For any finite set of points $x_1, \ldots, x_n \in \mathcal{X}$, the *kernel matrix* $K_{ij} = (k(x_i, x_j))_{i,j}$ is positive semi-definite, i.e., for all vectors $t \in \mathbb{R}^n$:

$$\sum_{i,j=1}^{n} t_i K_{i,j} t_j \geq 0 \tag{2.41}$$

Hilbert space A vector space $\mathcal{H}$ is called as Hilbert space if it is equipped with an inner product $\langle ., . \rangle_{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \mapsto \mathbb{R}$ and it is complete under the induced norm $\|v\|_{\mathcal{H}} = \sqrt{\langle v, v \rangle_{\mathcal{H}}}$, $i.e.$ all Cauchy sequences of elements in $\mathcal{H}$ converge to a limit that lies in $\mathcal{H}$.

Mercers Theorem: Let $\mathcal{X}$ be a non-empty set. For any positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, there exists ad Hilbert space $\mathcal{H}$ and a feature map $\psi : \mathcal{X} \mapsto \mathcal{H}$ such that

$$k(x, y) = \langle \psi(x), \psi(y) \rangle_{\mathcal{H}} \tag{2.42}$$

where $\langle ., . \rangle$ denotes the inner product in $\mathcal{H}$. For a kernel function k to be valid, the *completeness* property is of less importance, but the existence of an inner product is crucial for using the kernel function k instead of explicit evaluation of inner product in the Hilbert space. Because of this theorem, positive definite kernel functions are also called as *Mercel kernels*.

2.2.4 More on Kernels

In the above section, we have seen how to check if a kernel function is valid. In this section we shall see how kernel functions can be constructed, some of the famous kernels used for pattern recognition and computer vision problems.

Construction kernel functions Although **Equation** (2.41) looks to be simple, it is in practice difficult to check the criteria for a given function $k : \mathcal{H} \times \mathcal{H} \mapsto \mathbb{R}$. However, it is relatively easy to construct functions k that positive definite kernels, by using the following principals:

- For any $\psi : \mathcal{X} \mapsto \mathbb{R}^n, k(x, y) = \langle \psi(x), \psi(y) \rangle$ is a kernel.
- If $d : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$ is a distance function, i.e.,

  - $d(x, y) \geq 0$ for all $x, y \in \mathcal{X}$,
  - $d(x, y) = 0$ only for $x = y$,
  - $d(x, y) = d(y, x)$ for all $x, y \in \mathcal{X}$,
  - $d(x, y) \leq d(x, z) + d(z, y)$ for all $x, y, z \in \mathcal{X}$,

  then $k(x, y) = e^{-\gamma d(x,y)}$ is a kernel for any $\gamma \in \mathbb{R}^+$.
- We can construct kernels from other kernels:

  - If k is a kernel and $\alpha \in \mathbb{R}^+$, then $k + \alpha$ and αk are also kernels.
  - If k_1, k_2 are kernels, then $k_1 + k_2$ and $k_1.k_2$ are also kernels.

Example Kernels Some of the first kernels used for pattern recognition problems are as follows

1. *Linear Kernel:* If we have a linearly separable data in the original input space, we need not map the input space into a high-dimensional space. We can use linear kernel in such a situation, which is a dot product of the two vectors in the original space.

$$k(x, y) = \langle x, y \rangle \tag{2.43}$$

2. *Polynomial Kernel:* Polynomial kernels of degree p are given by

$$k(x, y) = (\langle x, y \rangle + 1)^p \tag{2.44}$$

3. *Radial Basis Function Kernel(Gaussian Kernel):*

$$k(x, y) = \exp \frac{-\|x-y\|^2}{2\sigma^2} \tag{2.45}$$

where σ is the parameter which controls the radius and $\sigma > 0$

4. *Sigmoid Kernel:*

$$k(x, y) = tanh(m\langle x, y\rangle + c) \text{ , for some(not every) } m > 0 \text{ and } c > 0 \tag{2.46}$$

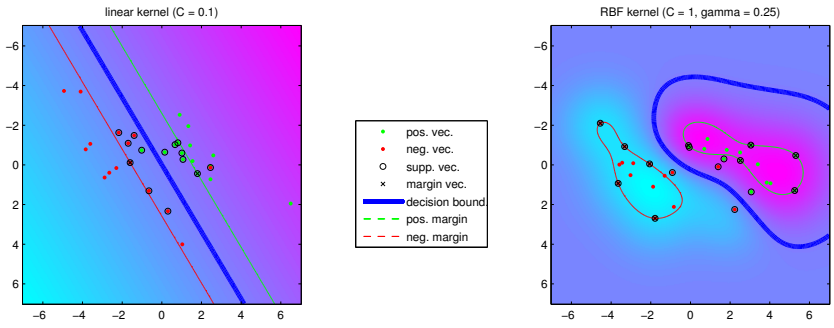

Figure 2.8 Boundaries obtained using SVMs with linear and RBF kernels

2.2.4.1 Kernels in Computer Vision

Kernel methods have been extensively used in computer vision in the past decade. There has been much attention by Computer Vision researchers on finding good data representations and algorithms to tackle problems, such as *Optical Character Recognition(OCR), Object/Scene Classification, Action Recognition, and Content Based Image/Video Retrieval*. Kernel methods proved to be successful for all these problems because of their interpret-ability and flexibility, mainly because: in constructing a kernel function one can integrate knowledge that humans have related to the specific problem. This leads to the improved performance over the methods where this kind of prior knowledge cannot be integrated.

Some of the popular choice of kernels used for computer vision tasks are given below

- *Generalized Intersection Kernel*

$$k(x, y) = \sum_i \min(x_i, y_i)^\gamma \tag{2.47}$$

  when $\gamma = 1$ it is called as Intersection Kernel

- *Exponential-χ^2 Kernel*

$$k(x, y) = e^{-\gamma \frac{1}{2} \sum_i \frac{(x_i-y_i)^2}{(x_i+y_i)}} \tag{2.48}$$

- *Chi2 Kernel*

$$k(x, y) = 2 \sum_i \frac{x_i y_i}{(x_i + y_i)} \tag{2.49}$$

We shall now look at some of the kernels designed for specific representations.

Spatial Pyramid Kernel Spatial Pyramid matching works by placing a sequence of increasingly coarser grids over the feature space (not over the image) and taking a weighted sum of the number of matches that occur at each level of resolution (L). At any fixed resolution, two points are said to match if they fall into the same bin of the grid; matches found at finer resolutions are weighted more highly than matches found at coarser resolutions w_1 represents the weight at level l. The bag of visual words representation discards all information about spatial structure from the image. However, spatial information can be a valuable source of information, *e.g.* in image segmentation, where sky regions tend to occur much more frequently at the top of the image than at the bottom. Consequently, the idea of local histograms has proved useful in this setup as well. Instead of one global visual word histogram, a number of local histograms are formed, typically in a pyramid structure from coarse to fine. Each sub-histogram has K bins and counts how many descriptors with center point in the corresponding pyramid cell have a specific code-book vector as nearest neighbor. Subsequently, either all local histograms are concatenated into a single larger histogram, or separate kernel functions are applied for each level and cell, and the resulting kernel values combined into a single spatial pyramid score, *e.g.* by a weighted sum

$$k_S P(x, y) = \sum_{l=1}^{L} \beta_l \sum_{k=1}^{K_l} k(h^x_{(l,k)}, h^y_{(l,k)}) \tag{2.50}$$

where L is the number of levels and β is a per-level weight factor. K_l is the number of cells in the l^{th} level, and $h^x_{(l,k)}$ and $h^y_{(l,k)}$ are the local histograms of h^x and h^y respectively. The base kernel k is typically chosen from the same selection of histogram kernels as above, with or without separate histogram normalization.

2.2.5 Multiple Kernel Learning

Fusing multiple sources of data is important in computer vision. Images can be represented by multiple set of features, each capturing certain specific aspects like color, textures and shapes. The importance of different features changes with the tasks; for instance, color information increases the detection of stop signs while coloring is almost irrelevant to find the cars as they can have different colors. Techniques for combining the relevant features required for the task at hand are therefore important for object categorization problems.

For object classification tasks, each of the descriptors provides good classification accuracies for different image classification tasks. Combining information from such multiple sources has been shown to be more effective. It has been shown in literature on how multiple features can be combined by using kernels corresponding to each feature representation [71]

If k_f denotes the kernel for the feature f, and if there are F features to be combined, then the resultant linear combination of kernels can be written as

$$k(x, y) = \sum_{f=1}^{F} d^f k_f \tag{2.51}$$

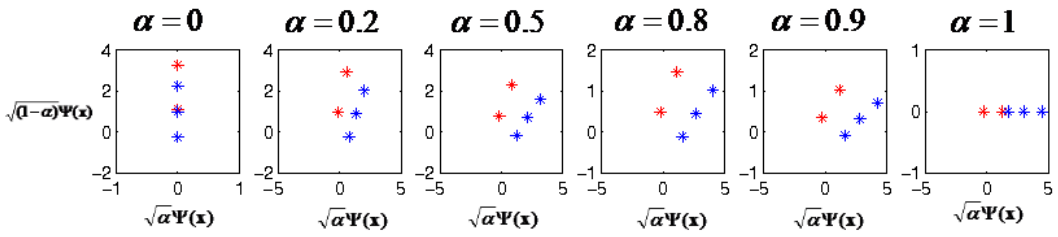

Figure 2.9 To understand the result of linear combination $k = \alpha k1 + (1-\alpha)k2$ for the given two kernels $k1$, $k2$, the induced feature spaces $\sqrt{(}\alpha)\psi(x), \sqrt{(}a-\alpha)\psi(x)$ corresponding to the two kernels are plotted for a various values of α

where k_f is the kernel for the f^th feature. Note that, the above combination of kernels is a mercer kernel according to the sum rule provided that each of the kernels k_f is a mercer kernel. Learning the weights of individual kernels d_f along with the SVM parameters is called as Multiple kernel learning. It was first proposed for small scale problems by [34] and for large scale problems by [63]. Varma et. al [71] proposed a gradient descent method for MKL and showed impressive results on varied object classification tasks. Giving weights to different kernels allows us to give more weightage(importance) to the more discriminative features for a given class. If $d_f = 0$,then it implies that the corresponding feature is not discriminative and it is not useful for the classification task.

Chapter 3

Large scale video retrieval based on Semantic Concepts

Traditionally many video retrieval engines have been designed by using the meta information of videos like *filename, tags*, and the text surrounding the image/video. These retrieval methods result in a poor retrieval performance when the information is not explicitly mentioned or if the information mentioned is not semantically apt. In order to have correct information, one can think of annotating these videos manually. But annotation of these videos with correct information requires huge amount of manual labor and time. This leads to the requirement of automatic semantic video retrieval methods. The main problem for any semantic video retrieval approach is the *semantic gap* between the data interpretation in the computers(pixel level) and their conceptual interpretation by humans(like description of content in video) [62]. In order to limit this semantic gap, video retrieval approaches have focused on semantic methods. Key-word based approach is one of them, where specific semantic concepts (categories) with a small intra-class variability are detected. These methods are based on image-classification techniques. One of the works which use image classification for content based indexing is proposed in [70]. In order to have a good video retrieval engine, we need these concepts detectors for thousands and thousands of concepts. We refer this methodology as "video retrieval based on semantic concepts".

In this chapter, we explain the generic framework for this task, followed by our framework and various experiments that we have performed for this purpose. In our methods, we use only visual information at individual frames and do not use any textual, audio and temporal information. These techniques are developed and used for our submissions in the TRECVID (see Section 3.1), high-level-feature extraction competition [68, 61] in the years 2008 and 2009. Note that, the task of video retrieval based on semantic concepts is referred as "high-level feature extraction" (HLF task) by TRECVID.

3.1 TRECVID Benchmark

The TREC Video Retrieval Evaluation (TRECVID) [68] is an international benchmarking activity conducted annually by NIST, to encourage research in content based video information retrieval by providing a large test collection, uniform scoring procedures, and a forum for organizations interested in comparing their results. This involves evaluation of a set of approaches by different teams from all

over the world for versatile of tasks related to video retrieval. Tasks that are evaluated in the years 2008 and 2009 include i) Surveillance Event Detection, ii) High-Level Feature Extraction, iii) Search (Interactive, Manually-assisted, and/or fully automatic), iv) Video summarization and v) Content-Based Copy Detection.

3.1.1 High-level Feature Extraction

The task of "high-level feature extraction" can be formally explained as follows: given a collection of videos shots, for a given concept category, (e.g. Mountain, people-dancing) return the list of the top video shots from the collection, ranked according to the highest probability of the shot containing the concept. Every year, this competition is conducted for retrieving a predefined set of concepts. These set of concepts range over objects, scenes, people, and events with varying degrees of complexity. Example images of categories evaluated in the years 2008 and 2009 can be seen in the **Figure** 3.1.

Data: All the participating teams in the competition are provided with two sets of MPEG-1 video data (of resolution 352×288), i) Development dataset (DEVEL) ii) Test dataset (TEST). This video dataset contains broadcast TV videos covering wide range of programs like science news, news reports, documentaries, educational programming, and archival video almost entirely in Dutch. Complete statistics of the development and test data of 2008 and 2009 is summarized in the Table 3.2. Note that 2008 and 2009 have the same set of data which in turn is the combination of 2007 development data and 2007 test data. TRECVID dataset is more challenging and of large size compared to the other image datasets, in terms of variability of images present in the dataset and number of test images. Table 3.1 compares the statistics of images present in different publicly available datasets for classification.

Dataset	**Type**	**No. of Images**	**No. of categories**
PASCAL VOC 2009 [17]	Images	14,743	20
CALTECH 256 [27]	Images	30,607	256
Scene-15 [37]	Images	4485	15

Table 3.1 Statistics of various image datasets available with the class level information of scenes or objects present in them. TRECVID dataset is the only major dataset for video data. There are other huge(in orders of lakhs) image datasets like imagenet [15] and LabelMe [56]. These datasets are not fixed in number as new images along with annotations are added to these datasets. Presently ImageNet has 195,331 annotated images and LabelMe has 43,244 annotated images.

Each video-shot of development data is annotated (as positive or negative) manually by users of participating teams through a collaborative annotation [1]. A team can acquire the complete annotations for the development data only if it annotates 3% of the total number of annotations. Our team [53, 76] has participated in collaborative annotations in 2008 and 2009 to get the annotations for the development data. Statistics of the positives and negatives for different categories found according to the annotations can be seen in Table 3.3. Teams are expected to develop their algorithms using the positive and negative

Set Name	Total Duration (hours)	No. of shots	Storage Space (for Videos)	No. of key-frames Extracted
2008-DEVEL	100	36,262	61 GB	43,616
2008-TEST	100	39,873	60 GB	81,274
2009-DEVEL	100	36,262	61 GB	73,859
2009-TEST	280	97,149	107GB	2,00,990

Table 3.2 TRECVID 2008 and 2009 Data Statistics

examples provided in the development data. Annotations will not be available for the test data, and is used for a common evaluation of generalization ability of algorithms from various teams. Each participating team submits ranked file-lists of video-shots for each semantic concept to NIST for evaluation. A ranked list obtained by using a certain method is referred to as a *run*, and up to 6 different runs can be submitted for evaluation.

Evaluation Measures: In general, *Average Precision* (AP) [77] is used as the evaluation measure for retrieval problems, which is designed such that importance is given to the correct ranking of the documents (video-shots in our case). It forms the basis of performance measure for TRECVID HLF task and most of the experiments in this chapter. Average Precision is a single-valued measure that is proportional to the area under a precision-recall curve. This value is equal to the average of the precision values at different recall levels, giving a single combined measure of precision and recall. It is defined as

$$AP = \frac{\sum_{r=1}^{N}(P(r) \times R(r))}{N_{rel}} \tag{3.1}$$

where r is the rank, N is the number of retrieved shots, $R(r)$ is a binary function stating the relevance of the shot retrieved with rank r, $P(r)$ is the precision at the rank r, and N_{rel} is the total number of relevant shots in the test set. In TRECVID HLF tasks, N is set to 2000, i.e., only a maximum of 2000 shots for a *run* can be submitted and evaluated.

The *mean average precision* (MAP) which is the mean of the average precision values over all the concepts evaluated, is used to compare the combined performance of all the concept detectors in TRECVID. In order to evaluate the performance, one needs the ground truth for test data. But due to the large size of the test data, it was not possible for NIST-TRECVID organizing team to prepare a concept-wise ground truth of the complete test data. For this purpose, they used the following pooling technique. First, a pool of possibly relevant shots is obtained by gathering the sets of shots returned by the participating teams. These sets are then merged, duplicate shots are removed, and only the remaining shots are annotated manually. These annotations are then used for evaluation of all the submitted runs. It should be observed that this technique can result in the underestimation of the performance of new algorithms and, new runs which were not part of the official evaluation, as all unique relevant shots retrieved by them will be missing from the ground truth collected.

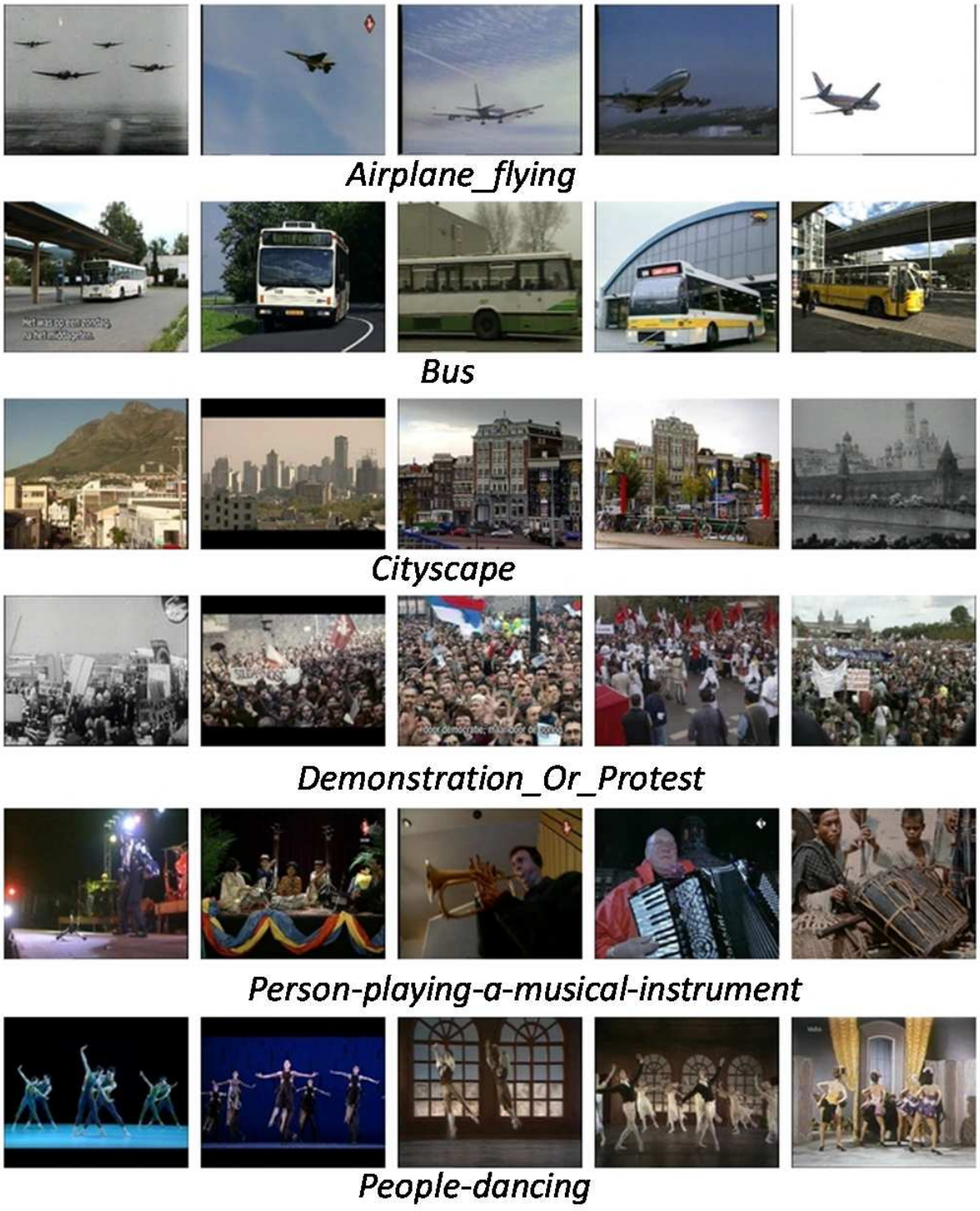

Figure 3.1 Example images for different classes from TRECVID 2009 dataset

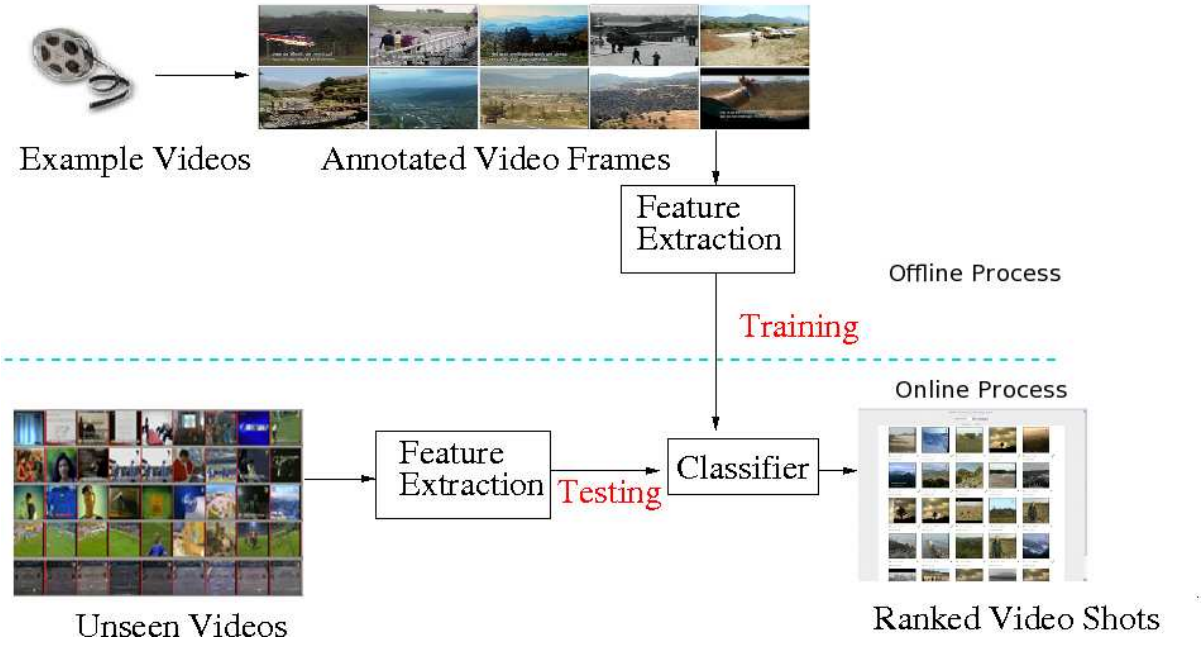

Figure 3.2 Pipeline for semantic concept based video retrieval

The approximate AP calculated in this manner is referred to as *Inferred Average Precision* (IAP) [79], and is used for comparing the performance of various detectors on the test data. And the mean of this performance measure over all concept detectors is referred to as the *mean inferred average precision* (mIAP).

3.2 Complete Pipeline for key-word based semantic video retrieval

In order to retrieve videos of semantic concepts at accurate points of the time in video, video sequences are first divided into basic segments (called as *video-shots*). These video-shots consist of sequence of frames that represent a continuous action in space and time. There are many automatic video segmentation algorithms in literature, most of which depend on comparison of successive frames in the video at pixel or frame level (e.g. a change in the camera view point). Each of these video-shots are represented by a single frame which is called as a *key-frame*. One simple choice of a key-frame can be the middle frame of the video-shot. We can also choose to have multiple key-frames for a given video-shot.

Semantic Concept based Video retrieval is solved by treating it as a supervised machine-learning problem of image-classification, where a classifier's confidence score for the key-frame is used for ranking the corresponding video-shot. If there are multiple key-frames in a video-shot, then the maximum of the confidences for all the key-frames in the shot is considered for ranking that shot. Note that

Category	TRAIN		VAL	
	No. of pos.	**No. of neg.**	**No. of pos.**	**No. of neg.**
Two_people	1964	19723	2543	19140
Hand	1108	20437	916	20297
Street	949	20852	972	20746
Cityscape	626	21275	442	20990
Flower	335	21400	391	21353
Boat_Ship	256	21282	266	21610
Nighttime	254	21271	275	21583
Kitchen	253	21347	67	21746
Singing	143	21409	427	21266
Classroom	141	21417	153	21312
Telephone	132	21416	88	21905
Mountain	132	21398	147	21430
Driver	117	21398	192	21413
Bridge	113	21433	93	21690
Emergency_Vehicle	86	21498	75	21900
Dog	79	21466	78	21722
Bus	78	21463	31	22034
Demonstration_Or_Protest	70	21397	89	21655
Harbor	66	21446	163	21723
Airplane_flying	11	21517	68	21770

Table 3.3 Statistics of example key-frames in DEVEL set for each category of TRECVID-2008 HLF task

the main difference between image classification and semantic concept video retrieval is that ranking is important in the problem of retrieval irrespective of the label assigned by the classifier. This HLF task is mainly challenging due to the wide variety of variations present in the dataset and also because of the huge size of the dataset. TRECVID dataset is very huge compared to many other publicly available scene and object category datasets like PASCAL VOC [18, 17], 15-Scene Category dataset [37], Caltech-256 dataset [27].

Given a d-dimensional feature vector x_i corresponding to a image i, the aim is to obtain a classifier/model that can provide a confidence measure indicating whether semantic concept C_j is present in the given image i. This computation of feature vectors x_i for each image i is called as the *Feature extraction*. As explained in Chapter 1, the process of finding the confidence measure is a standard machine-learning problem involving two phases i) *Training* and ii) *Testing* .

Figure 3.2 summarizes the complete pipeline of semantic concept based video retrieval. We have used vision only approach in our framework, focusing mainly on the scene-like categories(where localization of any object is not required). Feature representations that we used in our experiments include GIST, PHOW(Pyramid Histogram of visual Words) and PHOG (Pyramid Histogram of Oriented Gradients). Our approaches use Support Vector Machines (SVMs) as the classifiers. More details about

Category	TRAIN		VAL	
	No. of pos	No. of neg	No. of pos	No. of neg
Hand	1058	35508	824	36329
Female-human-close-up	924	35696	1030	36209
Cityscape	542	36038	445	36718
Doorway	378	36155	204	36977
Nighttime	341	36267	237	37002
Person-playing -a-musical-instr.	221	36397	591	36638
People-dancing	207	36344	270	36743
Demonstration_Or_Protest	196	36417	277	36962
Classroom	151	36447	135	37096
Boat_Ship	127	36493	60	37179
Telephone	116	36504	127	37112
Chair	109	36511	196	37043
Person-eating	104	36502	71	37164
People-singing	87	36533	380	36859
Traffic_Intersection	86	36496	49	37165
Bicycle	74	36538	189	37030
Person-playing-soccer	71	36549	74	37165
Bus	46	36574	32	37207
Infant	37	36504	30	37149
Airplane_Flying	19	36601	78	37161

Table 3.4 Statistics of example key-frames in DEVEL set for each category of TRECVID-2009 HLF task

the feature representations we used and the SVM classifiers is provided in Chapter 2. We discuss our experimental setup and various experiments that we have performed for HLF-task(TRECVID 2008 and 2009) in the next section.

3.3 Experiments

Experimental Setup In order to evaluate the performance of the learned model, we have divided the TRECVID DEVEL set into two sets, i) training set which we refer as TRAIN and ii) validation set which we refer as VAL. TRAIN set corresponds to TRECVID 2007-DEVEL set and VAL set corresponds to TRECVID 2007-TEST set (refer Table 3.3). Parameters for various feature representations that we used in our experiments are given below:

- *GIST:* We have computed GIST descriptors using the publicly available MATLAB code [50]. We used a grid size of $m = 4$ (4×4 blocks) for $N_o = 8$ orientation filters at $N_s = 4$ different scales, resulting in a 512 dimensional($4 \times 4 \times 8 \times 4$) descriptor for a given image.
- *Pyramid Histogram of Gradients (PHOG):* We have used PHOG-180 version with 8 bins at 3-levels of spatial pyramid, i.e., histograms for each cell at each level are computed by assigning

pixels based on the gradient angle to one of the 8 equal divisions of the range $[0 - 180]$ ignoring the sign of the gradient.

- *Pyramid Histogram of Visual Words (PHOW):* We use $\text{PHOW}_{\text{gray}}$ version in which the descriptors are computed using the gray image. These features are computed using dense SIFT[13] descriptors on a regular grid with spacing of $M = 5$ pixels. At each grid point of the gray images, the descriptors are computed over circular support patches at 4 different scales: $5, 10, 15, 20$. To deal with the empty patches, we zero all SIFT descriptors with $L2$ norm below a threshold of 200. These dense SIFT descriptors are vector quantized into visual words using *k-means* clustering with $k = 300$ to obtain a *visual vocabulary*. As usage of all the descriptors is infeasible due to memory constraints, we randomly subsample the descriptors during clustering. Then for any image, each pixel is assigned to the nearest visual word using $L2$ distance. Finally, histograms of visual words for $l = 3$ levels are computed to obtain the $\text{PHOW}_{\text{gray}}$ features for a given image.

3.3.1 Performance Variation with different features

First, we show the performance of different features for some of the scene classes in 2009 HLF task. For this experiment, we used a SVM classifier with exp-χ^2 kernel, where $\chi^2(x, y), x, y \in R^d$ is defined as

$$\chi^2(x, y) = \frac{1}{2} \sum_{i=1}^{d} \frac{(x_i - y_i)^2}{(x_i + y_i)} \tag{3.2}$$

Statistics of the number of positive and negative samples used for training and testing for each class can be seen in the table Table 3.4. Also, we present the performance obtained by combination of features. We combine multiple features by using Multiple Kernel Learning framework [71], where each kernel corresponds to a feature. Recall that the final kernel which is the weighted linear combination of individual kernels is given by

$$K_{comb} = \sum_{f=1}^{F} d_i K_f \tag{3.3}$$

The weights for these kernels are learnt using the gradient descent based MKL algorithm by [71]. We found that the combining of features gives only a slight improvement in the performance. We can

Class Name	**GIST**	**PHOG180**	**$\text{PHOW}_{\text{gray}}$**	**MKL($\text{PHOW}_{\text{gray}}$+PHOG+GIST)**
Cityscape	0.33	0.26	0.56	0.57
Demonstration _Or_Protest	0.25	0.14	0.50	0.49
Traffic_ Intersection	0.18	0.14	0.21	0.24
Classroom	0.01	0.02	0.09	0.10
Nighttime	0.07	0.05	0.38	0.39

Table 3.5 Performance variation with different feature representations for key-frames. We can see that using $\text{PHOW}_{\text{gray}}$ gives the best performance out of all the cases

ClassName	Linear		RBF		Intersection		exp-χ^2	
Normalization	L1	L2	L1	L2	L1	L2	L1	L2
1. Cityscape	0.32	0.33	0.39	0.41	0.53	0.48	**0.56**	**0.54**
2. Demonstration _Or_Protest	0.35	0.33	0.34	0.36	0.46	0.42	**0.50**	**0.49**
3. Traffic_ Intersection	0.12	0.15	0.14	0.19	0.20	0.17	**0.21**	**0.21**
4. Classroom	0.03	0.02	0.02	0.02	0.08	0.05	**0.09**	**0.08**
5. Nighttime	0.22	0.13	0.30	0.19	0.35	0.36	**0.38**	**0.35**

Table 3.6 Performance Variation with different choice of kernels using L1-normalized and L2-normalized $\text{PHOW}_{\text{gray}}$ features

notice that $\text{PHOW}_{\text{gray}}$ outperforms all the other features for almost all the classes. We therefore have concentrated mainly on using $\text{PHOW}_{\text{gray}}$ features for the most of our experiments in this chapter.

3.3.2 SVM parameter selection

We now show the effect of different SVM parameters on the performance of HLF task. These experiments are performed using only 3 levels of $\text{PHOW}_{\text{gray}}$ features. SVM parameters that can effect the performance are i) the penalty parameter C. ii) the choice of Kernel function $k(x, y)$ and We found that the value of $C = 1$ is the optimal choice in our experiments.

Choice of kernel function: There are also parameters which are specific to some of the kernels. For example, γ is the parameter that is needed to be tuned for exponential family of kernels. However using the value as $\gamma = \frac{1}{mean(D)}$ where D is the distance matrix has shown to be a good heuristic in literature [71]. Table 3.6 shows the performance using *Linear*, *RBF*, *Intersection* and *exp-*χ^2 kernels. We show these results using both $L1$-normalized and $L2$-normalized $\text{PHOW}_{\text{gray}}$ features.

We can observe that exp-χ^2 with $L1$-normalization out-performs all the other kernels. Also, we can see that results with $L2$-normalization are better than $L1$-normalization for linear and RBF kernels. And in the case of Intersection and exp-χ^2 kernels, we can see that using $L1$-normalization is better than $L2$-normalization. Results for some of the classes of 2009 using the intersection kernel are given in the Table 3.7. In this table, the performance measure used on VAL data is AP, where as the performance measure used on TEST data is infAP. We can see that there is a significant difference between the AP and infAP for most of the classes.

3.3.3 Speedup with Fast Intersection Kernel

In the previous section, we have seen that exp-χ^2 kernel and intersection kernel gives better performance than linear kernels. Unfortunately these results come at a great computational expense compared to the linear kernels, because non-linear kernels require memory and computation, linearly proportional to the number of support vectors during testing. But, the good thing about intersection kernel is that there is a speed-up technique proposed recently for its computation[42]. In this section, we show that use of fast intersection kernel is a good choice for large scale categorization tasks and especially for

Category	AP	inferred AP
Training Set **Testing Set**	**TRAIN** **VAL**	**TRAIN+VAL** **TEST**
Cityscape	0.54	0.28
Demonstration_ Or_Protest	0.45	0.03
Doorway	0.41	0.21
Nighttime	0.40	0.24
Hand	0.39	0.20
Boat_Ship	0.23	0.17
Female-human-face-closeup	0.20	0.19
Traffic_Intersection	0.17	0.16
Person-playing-soccer	0.11	0.31

Table 3.7 The table shows performance obtained for some of the classes of 2009. It reports the average precision obtained by using PHOW + fast intersection kernel SVM when trained on TRAIN and evaluated on VAL, and TRECVID inferred AP when trained on TRAIN+VAL. To compute average precision on TRAIN+VAL the complete and cleaned annotations were used. In several cases the difference in AP and infAP is remarkable.

HLF-task. At first, we shall see how the speedup with intersection kernel is achieved. Recall that intersection kernel $k(x, y)$ for the given d-dimensional histograms x and y ($x \in R^d$, $y \in R^d$) is defined as

$$K(x, y) = \sum_{i=1}^{d} min(x_i, y_i) \tag{3.4}$$

and the decision function f for classification using Support Vector Machines is defined as

$$f(x) = \sum_{j=1}^{nSV} \alpha_j c_j k(x_i, x) + b \tag{3.5}$$

$$= \sum_{j=1}^{nSV} \alpha_j c_j \left(\sum_{i=1}^{d} min(x_j(i), x(i)) \right) + b \tag{3.6}$$

where α_j and y_j denote the support vector coefficient and class label for the j sample correspondingly, b denotes the bias and nSV denotes number of support vectors.

Time complexity for evaluating the function $f(x)$ takes $O(nSV.d)$. Maji *et. al* [42] proposed a method to achieve this result in $O(d. \log nSV)$. They also extend it further to find an approximate classifier in $O(d)$. The trick for speeding up starts with the exchanging of the summations in the above equation, to obtain

$$f(x) = \sum_{j=1}^{nSV} \alpha_j c_j \left(\sum_{i=1}^{d} min(x_j(i), x(i)) \right) + b \tag{3.7}$$

$$= \sum_{i=1}^{d} \left(\sum_{j=1}^{nSV} \alpha_j c_j min(x_j(i), x(i)) \right) + b \tag{3.8}$$

$$= \sum_{i=1}^{d} f_i(x(i)) + b \tag{3.9}$$

Now we have the decision function as the summation of individual functions f_i, each of them defined as

$$f_i(s) = \sum_{j=1}^{nSV} min(x_i(j), s) \tag{3.10}$$

Each of the above functions f_i can be computed in $O(\log m)$ time as follows. Let us consider the function $f_i(s)$ for a fixed value of i. If $\bar{x}_j(i)$ denotes the sorted values of $x_j(i)$in increasing order with corresponding α's and labels as α_j and c_j. If r denotes the largest integer such that $\bar{x}_r(i) \leq s$, then we have,

$$f_i(s) = \sum_{j=1}^{nSV} \bar{\alpha}_j \bar{c}_j min(\bar{x}_j(i), s) \tag{3.11}$$

$$= \sum_{1 \leq j \leq r} \bar{\alpha}_j \bar{c}_j \bar{x}_j(i) + s \sum_{r \leq l \leq nSV} \bar{\alpha}_j \bar{c}_j \tag{3.12}$$

denote the first term in the above equation by $A_i(r)$ and second term by $B_i(r)$.

As the above function is piecewise linear, and the functions $A_i(r)$ and $B_i(r)$ depend only on the support vector parameters α and y. By precomputing the d functions f_i, then $f_i(s)$ can be computed in two steps. i) first finding r, the position of $s = x(i)$ in the sorted list $\bar{x}(i)$ using binary search and ii) linearly interpolating between $f_i(\bar{x}_r)$ and $f_i(\bar{x}_{r+1})$. This enables the reduction of complexity for computing the decision function $f(x)$ from $O(d * nSV)$ to $O(d \log nSV)$.

In order to compute $f(x)$ approximately in $O(d)$,[42] compute $f_i(s)$ with a table look-up in the piecewise linear/constant approximation. This is made possible by using the fact that $f_i(s)$ can be represented as a piecewise linear segments.

In order to show the trade-off between performance and the testing time, we show the following experiment comparing the AP (Average Precision) and Testing Time for some of the scene categories in TRECVID 2009 using exp-χ^2 kernel, intersection and fast-intersection kernel. We use the MATLAB implementation[41] provided by the authors of [42] for results of fast-intersection kernel. For the fast-intersection kernel, we have used a piecewise-linear approximation. In the Table 3.8, we can see that a speedup of nearly $10X$ is achieved by using fast-intersection kernel for all the categories over using

exp-χ^2 with a slight change in the performance.

ClassName	Training Size		Validation Size		exp-χ^2		Intersection		Fast-Intersection	
	Pos	Neg	Pos	Neg	AP	Testing Time (secs)	AP	Testing Time (secs)	AP	Testing Time (secs)
Cityscape	542	7208	445	36781	0.56	1926	0.53	1440	0.53	170
Demonstration Or Protest	196	7284	277	36962	0.50	1132	0.46	840	0.46	112
Traffic Intersection	86	7300	49	37165	0.21	845	0.17	670	0.17	106
Classroom	151	7290	135	37096	0.07	1408	0.08	980	0.08	109
Nighttime	351	7254	237	37002	0.38	987	0.35	792	0.35	114

Table 3.8 We can see the speedup achieved using fast-intersection Kernel with only a slight effect on performance

3.3.4 Effect of training data size

In this experiment, we show the effect of number of training positives and training negatives on the performance of the classifier. We performed this experiment for Cityscape and Demonstration_Or_Protest categories, using the fast-intersection kernel[42] and $L1$-normalized PHOW$_{\text{gray}}$ features. We have fixed the value of C as one for this experiment.

Graphs in the **Figure** 3.3 and **Figure** 3.4 show the variation of AP(z-axis) with change in the percentage of total available positives and negative samples used. The value of AP for each point in the graphs is averaged over AP's obtained by training with 3 different random sets of positive and negative training samples. We can see that the performance mainly increases by increasing the number of positive training samples. Also, we can observe that there is not much effect by the change of the number of negative training samples. From the graphs, we can see that 10% of total negative samples can give the same performance as the one given by using all the negative samples(which is nearly equal to 0.55 for Cityscape and 0.45 for Demonstration_Or_Protest).

3.3.5 Effect of noise in the training data

We found that the collaborative annotations for the TRECVID high level features to be quite noisy: some shots are wrongly annotated, and others are labeled as skip when they are, in fact, unambiguously positive or negative for the feature. To remove this noise in the annotation, we used a weak classifier trained on the noisy data for each high level feature as follows

1. Train a classifier using all the +ves and a subset of -ves in TRAIN and VAL sets according to the Collaborative Annotation.

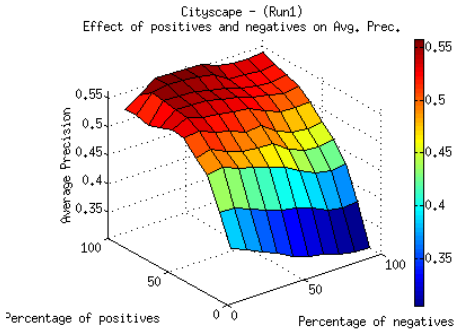

Figure 3.3 Variation of Average Precision with number of positive samples and negative samples for "Cityscape" category

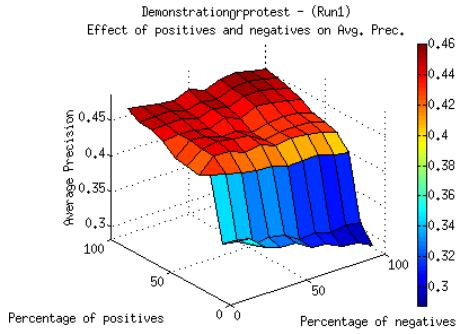

Figure 3.4 Variation of Average Precision with number of positive samples and negative samples for "Demonstration_Or_Protest" category

2. Re-rank all the images in the TRAIN+VAL set based on the classifier output.

3. Refine the annotations of the top 5000 ranked images.

In this manner, we could find many of the wrong annotations with minimal manual effort. This refinement was found to be very effective. For example, for the *Doorway* category the AP performance increased from 0.16 using noisy annotations to 0.41 using cleaned annotations. Note that the performance on both of these results is assessed on the cleaned annotated VAL set.

As there are very less number of positive samples in the training data, we have downloaded images with tags related to each of the categories using Google Image Search and photo-sharing websites like Flickr. As these image search engines are not content-based, we got lot of images which do not have the query scene in them. For this purpose, we have filtered these images manually and then used them for training. We refer this set of extra images downloaded from internet as WEB set

In order to evaluate the effect of adding these extra images, we have trained classifiers using TRAIN + WEB sets and evaluated it on the VAL set. The performance comparison with and without extra images for the 2 scene categories "Cityscape" and "Classroom" is shown in the Table 3.9. We can see that there is only a slight improvement in the performance by adding extra images.

ClassName	Trained with TRAIN set	Trained with WEB set	Trained With TRAIN+WEB set
Cityscape Trecvid size: 543 WEB size: 3652	0.44	0.33	0.45
Classroom(543 + 3652) Trecvid size:160 WEB size:3183	0.12	0.04	0.06

Table 3.9 Comparison of performance(AP) on VAL set obtained by training using TRAIN set, WEB set and TRAIN+WEB set

ClassName	SIFT based $PHOW_{gray}$	DAISY based $PHOW_{gray}$
Cityscape	0.36	0.21
Two_People	0.12	0.11
Mountain	0.18	0.10
Hand	0.11	0.09

Table 3.10 Comparison between the performance obtained using SIFT based $PHOW_{gray}$ and DAISY based $PHOW_{gray}$. We can see that the performance of SIFT is slightly higher in almost all the categories. But, DAISY is faster to compute compared to SIFT descriptor.

3.3.6 Speeding up the feature extraction

We have seen that $PHOW_{gray}$ feature representation is very much useful for HLF task. Majority of the time taken for computing these $PHOW_{gray}$ features is spent in the computation of dense SIFT descriptors. In order to speedup this process, we have experimented by replacing SIFT descriptors with its variant called as DAISY descriptors [66]. Computation of DAISY descriptor for an image of 352×288 took only 1 second as opposed to the 30 seconds. Comparison of the results using SIFT based PHOW and DAISY based PHOW representation is presented in the Table 3.10. This experiment is done by training on some of the TRECVID 2008 categories by training on TRECVID 2008 DEVEL set and testing on TRECVID 2008 VAL set. We have used SVM with intersection kernel as classifier. We can see that the results obtained using DAISY based $PHOW_{gray}$ are almost equal to those obtained using SIFT based $PHOW_{gray}$.

3.3.7 Use of Subcategories

Images belonging to a category can be further divided into groups of images (or *sub-categories*) based on their visual appearance(color, pose, appearance, etc.,). For example, cityscape category can be divided into the following categories: *Cityscape-topview, Cityscape-Far, Cityscape-near, Cityscape-veryfar and Cityscape-verynear,etc.,*

Most of the image categorization techniques in literature[13, 7, 20] learn the classifier for the given category using a single training set of images covering all these sub-categories. This can result in an inaccurate model for the given category, as it is a tough task for the classifier to learn all the variations present in the training set.

For this purpose, we propose to build a classifier for each of the sub-categories SC_i of category C_i separately, and then combine the outputs of these classifiers for testing the presence of category C_i in a given image. We conduct this experiment for two categories of TRECVID 2009 HLF task namely *Cityscape* and *Demonstration_Or_Protest*. For this purpose, we have annotated the key-frames of TRAIN and VAL sets with the subcategory it belongs to. Example images belonging to each of the subcategories are shown in the figure Figures 3.5 and 4.1. We used $PHOW_{gray}$ features and SVM with fast intersection kernel as classifier for this experiment. Though quantitatively the AP obtained by using subcategories was not greater than the AP without using subcategories, we found that a different

ranking of images is obtained by using this method. Also we found that some of the positive images which have low scores get high scores when subcategories are used. More experiments using this idea of sub-categories are presented in the next chapter.

3.4 Submitted Results

We have submitted runs for all the categories in 2008 and 2009. For 2008 submission, we have used exp-χ^2 kernel, without refining the data. For 2009 submission, we have used the fast intersection kernel with refined data. Key-frames of the top ranked shots in TEST data 2009 submitted for HLF task for categories "Hand" and "Female-human-face-closeup" can be seen in the Figures 3.7 and 3.8. **Figure** 3.6 shows the inferredAP according to TRECVID 2008. According to this performance measure, we performed very well and are in the third position for the Cityscape category.

3.5 Generalization: Results on a different video dataset

Classifiers trained using TRECVID data were used to retrieve some of the semantic categories from video data provided by BBC TV channel. This data is completely different from the TRECVID DEVEL and TEST datasets, is a collection of 428 very old videos(1970's) of different TV programmes. The total duration of the videos is 220 hours and in total there are 137921 key-frames. These videos are divided into shots by using a automatic video segmentation. We then retrieved the video-shots in the same way as we did for TRECVID test data. Top results for the categories Person-playing-a-musical-instrument, Boat ship and Cityscape categories are shown in the Figures 3.9 and 3.10. These results show that the classifiers have a very good generalization power as these classifiers are trained on a completely different data.

3.6 Discussions

In this chapter, we analyzed the performance of various features and SVM-based classification methods for the task of large scale concept retrieval. We presented our attempts to improve the performance beyond the best performing state-of-the-art methods. Augmented with various experiments, we have shown the success of SVM with combination of PHOW features for many scene categories in TRECVID 2009 High Level Feature extraction competitions. We also presented how the choice of fast SVM's is highly suitable for the task of large-scale semantic concept retrieval.

Cityscape_topView

Cityscape_far

Cityscape_near

Cityscape_nighttime

Cityscape_veryNear

Figure 3.5 Example images for different sub-classes for ”Cityscape” from TRECVID 2009 DEVEL dataset

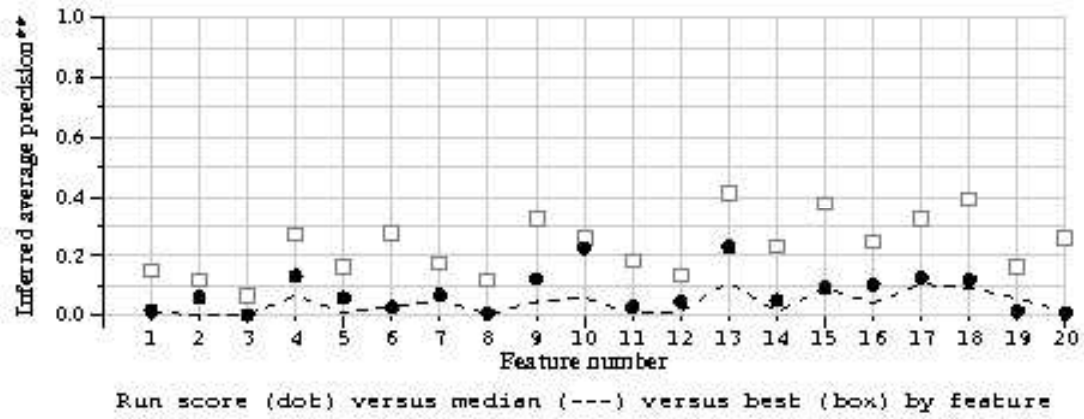

Figure 3.6 Inferred AP for different categories for our TRECVID08 submission

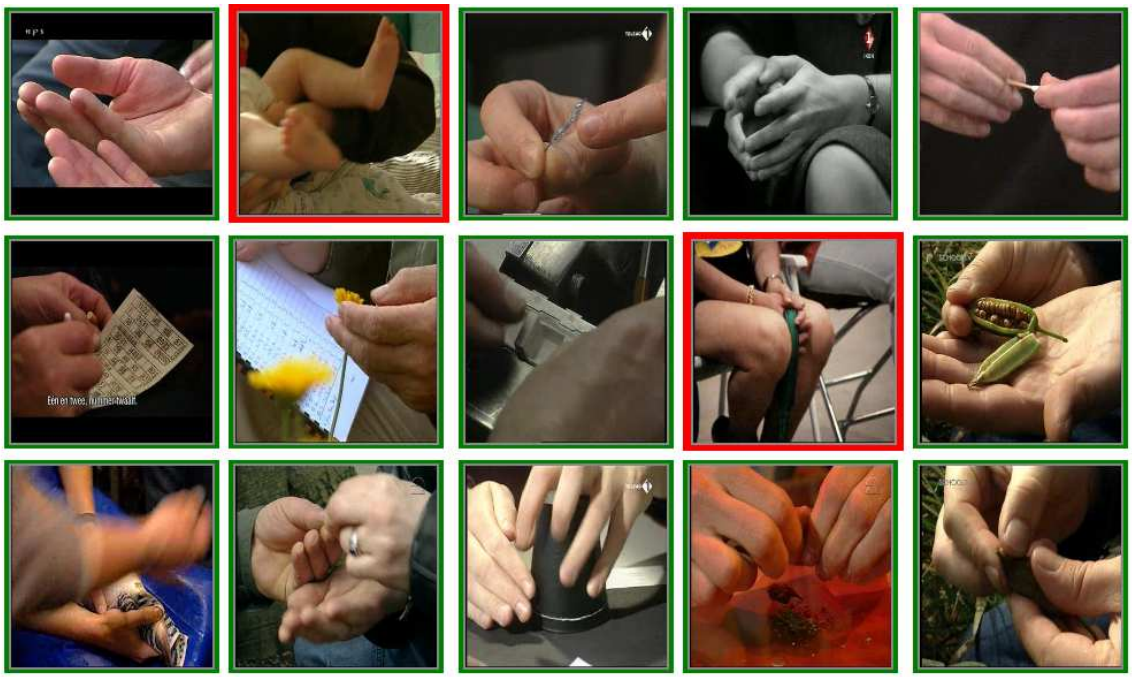

Figure 3.7 Results for category "Hand" on TRECVID09 Test Data

Figure 3.8 Results for category “Female-human-face-closeup” on TRECVID09 Test Data

Figure 3.9 Results for category “Cityscape” in BBC data using the classifiers trained on TRECVID data

Figure 3.10 Results for category "Person-playing-a-musical-instrument" in BBC data using the classifiers trained on TRECVID data

Chapter 4

Modelling Sub-Categories

4.1 Introduction

As discussed previously in the section 3.3.7, set of images belonging to a semantic category can be further sub-divided into sub-groups, such that images belonging to a sub-category share a common aspect(for example "pose"). In general, training a classifier uses only the class label and the feature vectors of examples. In this chapter, we show how sub-categories(sub-classes) information can be augmented with class labels, to improve the performance of SVMs. We mainly show how this helps in boosting the performance of computationally inexpensive kernels like "linear kernel". We present our attempts using sub-category information to improve the performance of SVMs with "linear kernels" such that it can perform as good as computationally expensive kernels like "quasi-linear(ex: intersection)" and "non-linear" kernels (ex: RBF).

To train a classifier for a category of interest, standard SVM formulation uses pairs of $(\mathbf{x}_i, y_i), i \in \{1, \ldots, n\}$. Here, $\mathbf{x}_i$ denotes the feature vector and y_i denotes the label for the given sample i. If y_i^s denotes the sub-category label for a given sample, we want to use tuples of the form $(\mathbf{x}_i, y_i, y_i^s)$ in the training process. We use structural SVM to show how this sub-categories information can be included in the training process [69]. Structured SVMs is a generalized SVM classifier designed for learning a function which can predict a complex label like trees, sequences or sets (e.g. prediction of a parse tree for a given sentence). Also, we investigate the use of different kinds of sub-category groupings for improving the performance. For this purpose, we analyze the performance for problems of classification and detection obtained using different ways of groupings. Example way of grouping the set of images is to cluster them in their features space. We present an iterative algorithm similar to the one proposed in [81] for learning sub-category labels for the training set that can lead to optimal performance. We also analyze the performance variation based on different initializations. The structure of this chapter is as follows. At first, we provide the background for structured SVMs [69] and latent structural SVMs [81]. Later we present our approach of using sub-category information using structured SVMs and latent structural SVMs. We show the performance comparison for classification/detection

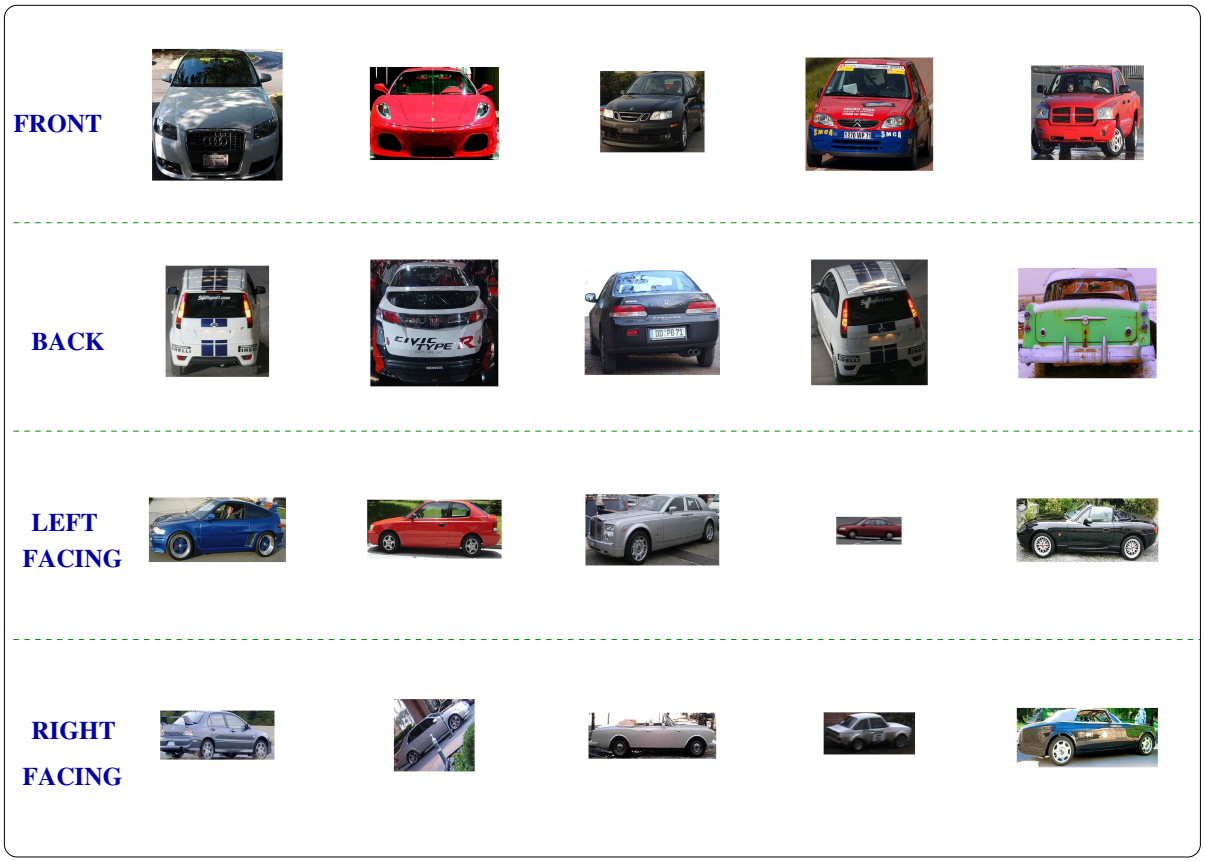

Figure 4.1 Example images for different subclasses present in the category "CAR" based on "pose" information (from VOC 2007 dataset). The annotations of subclasses are already included in the dataset.

using sub-categories with the approach that does not use sub-categories on 2-D synthetic dataset and also for some of the categories in PASCAL VOC 2007 [18] and TRECVID 2009 datasets.

4.2 Structured SVMs

Background: Suppose we are given a training set S of input and output pairs $(\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_n, y_n) \in (\mathcal{X} \times \mathcal{Y})$. Structural SVMs learn a prediction rule of the form

$$f_w(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \langle \mathbf{w}, \psi(\mathbf{x}, y) \rangle \tag{4.1}$$

where ψ is a joint feature map which describes the relationship between input vector $\mathbf{x}$ and the structured output label y. The design of this join feature map function ψ mainly depends on the application. Here $\mathbf{w}$ is the parameter vector to be determined in the process of training using the training set S.

In training stage, $\mathbf{w}$ is determined by minimizing the regularized risk on the training set S. This risk is measured by using a loss function $\Delta(y, y')$, which measures the risk of predicting the output of

a sample as y', whose ground truth is y. Training process in Structural SVMs optimizes the objective function whose final form after including the Lagrangian multipliers can be written as

$$\min_{\mathbf{w}} \frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{n} [\max_{\hat{y} \in \mathcal{Y}} [\Delta(y_i, \hat{y}) + \langle \mathbf{w}, \psi(\mathbf{x}_i, \hat{y}) \rangle] - \langle \mathbf{w}, \psi(\mathbf{x}_i, y_i) \rangle] \tag{4.2}$$

This is a convex optimization problem and is solved efficiently by using cutting-plane or stochastic gradient methods [69, 82, 31, 23]. For our problem, we consider the class label y augmented with the subclass label y_i^s as the output label for a given sample. If we have C classes and S_i subclasses, $i \in \{1, 2 \ldots, C\}$ for each of the class, then we will have a total of $\sum_{i=1}^{C} S_i$ possible tuples of (y, y_i^s).

In order to introduce non-linearity, a *joint kernel function* corresponding to the *joint feature map* is used in structured SVMs.

Our formulation: In our case, the joint feature map and the joint kernel function are defined as follows

$$\psi(\mathbf{x}, (y, y^s)) = [(v \otimes 1_d) \odot \psi(\mathbf{x})] \tag{4.3}$$

$$K(\mathbf{x}_i, y_i, y_i^s, \mathbf{x}_j, y_j, y_j^s) = K(\mathbf{x}_i, \mathbf{x}_j) \quad if \quad y_i == y_j \quad and \quad y_i^s == y_j^s \tag{4.4}$$

where v's are a set of indicator variables. Number of such variables is equal to the total number of possible y^s and 1_d denotes a d-dimensional unit vector. Here, $v_j \in \{0, 1\}$ and $v_j = 1$ means that the image belongs to that combination of class label and subcategory label. And, $\otimes$ denotes the Kronecker product and $\odot$ denotes the Hamard product. The expression for the joint feature map ψ can be understood as a stacking of vectors, placing the feature representation $\psi(\mathbf{x})$ at the place where v is equal to 1.

We use the following loss function which penalizes only the misclassification's at the class level,

$$\Delta(y_i, y_j, y_i^s, y_j^s) = 1 \quad if \quad y_i == y_j \tag{4.5}$$

In our experiments, we consider the classifier "without using sub-class information" as the baseline classifier. If we use structured SVM for this 2-class problem (no subclasses), we will $y_s \in \{1, 2\}$ and $y \in \{-1, 1\}$

Structured SVM formulation without sub-class information is slightly different from the standard SVM formulation due to a missing bias term. In the next section, we show how structured SVM formulation without sub-class can be equalized to the standard SVM formulation.

Adding Bias Given a set of example inputs $\{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n\}$ with labels $\{y_1, y_2, ...y_n\} \in \{+1, -1\}$. Our aim is to prepare a prediction function f which maps from $\mathbf{x}$ to y. Recall that, in standard SVM formulation, our objective function which tries to minimize the margin w is

$$\min_{\mathbf{w},b} \frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i$$
$$\xi \geq (1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)) \quad \forall i$$
$$\xi \geq 0$$

where ξ is the slack variable which allows mistakes and C is the regularization constant.

In training step, structured SVM's find the parameter w which minimizes a risk function on the training data. The objective function minimized during training is

$$\min_{\mathbf{w}} \frac{1}{2} ||\mathbf{w}||^2 + C \sum_{i=1}^{n} \xi_i$$
$$\xi_i \geq (1 + \langle \mathbf{w}, \psi(\mathbf{x}_i, \hat{y}) \rangle - \langle \mathbf{w}, \psi(\mathbf{x}_i, y) \rangle) \triangle(y_i, \hat{y}_i) \quad \forall i$$

In order to make structured SVM formulation equivalent to standard SVM formulation without any subcategories, we end up with the optimization function of the following form after substituting the corresponding loss function and feature map function. This formulation involves a additional variable B. It is found that setting B to a high value (around 100) gives acceptable results.

$$\min_{\mathbf{w},b} \frac{1}{2} ||\mathbf{w}||^2 + \frac{1}{2} (\frac{b}{B})^2 + \sum_{i} \xi_i$$
$$\xi_i \geq 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$$
$$\xi_i \geq 0 \tag{4.6}$$

4.3 Learning the optimal sub-categories

Till now, we have seen how sub-categories can be included in a structured SVM framework for improving the performance of classification. Now we consider the problem of finding the groupings which can give best possible performance. To solve this problem, we use the work of *learning structural SVMs with latent variables* [81]. This work proposes the use of latent variables in the training data and shows how structural SVMs can be learnt using latent variables. This is proposed with an intuition that input-output relationships are not only characterized by $(\mathbf{x}, y) \in (\mathcal{X} \times \mathcal{Y})$ pairs in the training set alone, but also on a set of unobserved latent variables $h \in \mathcal{H}$. [81] present an algorithm using which latent variables and structural SVMs are optimized alternatively in an iterative manner. In our case, we consider sub-category labels as the latent variables in the formulation. Use of latent variables in structural SVM formulation was also introduced in a different manner in [21].

Structural SVM with latent variables: If $(\mathbf{x}_i, y_i), (i = 1, \ldots, n)$ are the given set of training samples and h_i be the latent variable for each sample i, then structural SVM formulation is extended to include h by adding an extra argument in the joint feature map $\psi(\mathbf{x}, y)$ making it $\psi(\mathbf{x}, y, h)$. This joint feature map describes the relation among input x, output label y and latent variable h. A joint kernel function can also be defined in similar manner.

The prediction rule in the case of latent structured SVM id defined as

$$f_w(\mathbf{x}) = \underset{(y,h)\in\mathcal{Y}\times\mathcal{H}}{\arg\max} [\langle \mathbf{w}, \psi(\mathbf{x}, y, h)\rangle] \tag{4.7}$$

The objective function optimized in this case is of the form

$$\min_{\mathbf{w}}[\frac{1}{2}\,||\mathbf{w}||^2 + C\sum_{i=1}^{n} \max_{(\hat{y},\hat{h})\in(\mathcal{Y}\times\mathcal{H})} [w.\psi(\mathbf{x}_i, \hat{y}, \hat{h}) + \Delta(y, \hat{y}_i, \hat{h})] - [C\sum_{i=1}^{n} \max_{h\in\mathcal{H}} \langle \mathbf{w}, \psi(\mathbf{x}_i, y_i, h)\rangle]] \tag{4.8}$$

Algorithm 1 Algorithm for Latent Variable Optimization.

1: **for** $t = 1$ to $nIterations$ **do**
2: Train the Struct. SVM classifier to find the parameter $\mathbf{w}(t)$ using $h(t-1)$
3: Use Struct. SVM parameters to find the new subcategory labels

$$h_i(t) = \underset{\hat{h}\in\mathcal{H}}{\arg\max}[\langle \mathbf{w}, \phi(\mathbf{x}_i, y_i, \hat{h})\rangle] \quad \forall i \in 1, \ldots, n$$

4: $t \leftarrow t + 1$
5: **end for**

The complete algorithm for learning the optimal sub-category labels starting from any set of sub-category labels is presented in the **Algorithm. 1**. This algorithm alternates between i) finding the latent h_i^* variables that best explain the training pair$(\mathbf{x}_i, y_i)$ and ii) finding the structural SVM parameters. It is similar to the iterative optimization process of Expectation-Maximization algorithm.

4.4 Experimental Results

For our experiments, we used a MATLAB implementation of structural SVM which is based on the publicly available C-implementation [9]. At first, we show how sub-categories can be useful for improving the classification performance on a synthetic 2-dimensional data. For this purpose, we have generated different kinds of 2-D data. These different 2-dimensional data are plotted in the **Figure** 4.2. Each of them differ in the arrangement of positive and negative examples. We compare the performance obtained using "linear kernel" and "intersection kernel", with subcategories and without subcategories. We denote different cases as follows

- Linear Kernel without subcategories by LIN

- Linear Kernel with subcategories by LIN^{sub}
- Intersection Kernel without subcategories by INT
- Intersection Kernel with subcategories by INT^{sub}

For each of these 2-D data, Figures 4.4 and 4.5 show the boundary and performance comparison for LIN, LIN^{sub} and INT. Red colored points are the positive samples and the green points are the negative samples. In order to show how the space is classified, we have plotted the space with a color-map based on the scores obtained at different points in the space. We can clearly see how sub-category information is very much useful in the case of linear kernel. By using sub-category information the region marked with red color is being classified properly. Also, it is observed that LIN^{sub} over-performs INT when the data is highly correlated. We can see an example case in the **Figure** 4.5, where LIN^{sub} over-performs the INT. Effect of the parameter C for all the three cases is shown in the **Figure** 4.3. **Figure** 4.6 shows how the category and sub-category labels are changed with each iteration (We show this for some iterations to make it easily understandable). Initially, we have started with an idea of using subclasses only for the positive class, but it is found to be beneficial using subclasses also for the negative class. This can be especially seen in the **Figure** 4.7, which shows the variation of boundaries and performance with change in the number of subclasses used for positive and negative classes. Also, we show that similar performance can be obtained by using different types of initialization. We have considered two kinds of sub category initializations: 1. RANDOM subclass labels and 2. Subclass labels obtained by using any Clustering algorithm (ex: *k-means*).

4.4.1 Real Data

We study the effect of using subcategories by performing experiments on real data for TRECVID 2009 High Level Feature Extraction task for videos [61] and PASCAL Visual Object Category 2007 object detection task for images [18].

VOC 2007 For object detection task on VOC 2007 dataset [18], we have used the pipeline of [73]. This pipeline contains a cascade layer of classifiers, varying from weak to strong classifiers with increase in layer. Also, the computational complexity of the classifier increases with each layer. Specifically, the first layer uses a classifier based on linear kernel, second layer uses a χ^2 kernel and the third stage uses exp-χ^2 kernel. Training is done using bounding boxes of the object category examples and negative examples, each of them having different scales and aspect ratios. We work on the output of the second layer of this cascade. We have used only level 2 of PHOW features for these experiments. We performed the experiments, same as those performed on the 2-dimensional synthetic dataset. Figures 4.8 and 4.9 show the variation of AP on training and validation data with change in the number of subcategories in positive and negative data for linear kernel. Figures 4.10 and 4.11 shows the improvement of performance for linear kernel with each iteration of optimizing subcategory labels using latent SVM. **Figure** 4.12 shows the similar performance variation using intersection kernel. The summary of the results is that we were able to get better performance by using linear kernel with subcategories compared

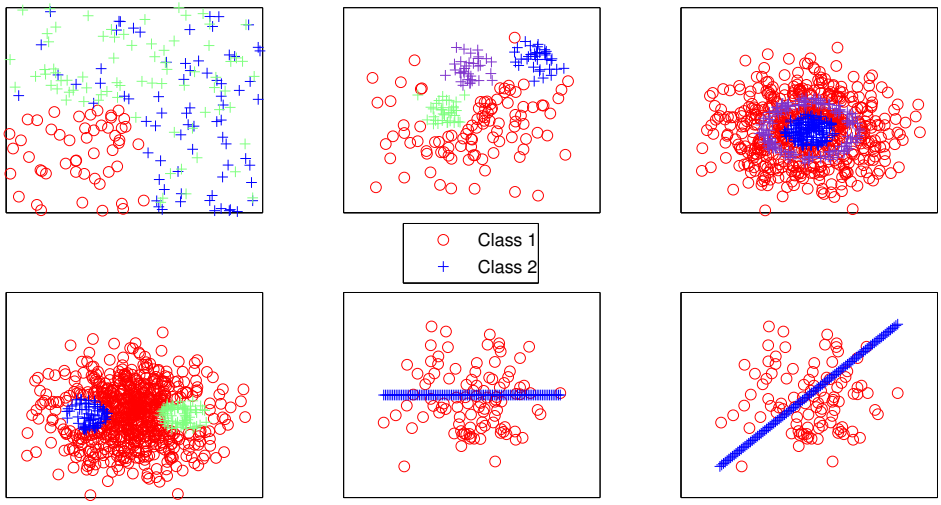

Figure 4.2 Example synthetic 2-D datasets that we used for our experiments to show how "SVM with linear kernel using subcategories" can perform as good as "SVM with intersection kernel without using subcategories". In each dataset, we can see that the positive and negative data are arranged in different manner. In each of these figures, we have two classes (one class marked with '+' and other with 'o') and for some of the classes, we show how subclasses can exist in one of the classes. Each of these subclasses are marked with a different colour.

to the linear kernel without subcategories. But "linear kernel with subcategories" performed poorly compared to the "intersection kernel without subcategories". And similarly the performance(Average Precision) obtained using "intersection kernel with subcategories" is less than the one obtained using "exp-χ^2 kernel without subcategories".

TRECVID: We have performed experiments using two feature representations, *PHOW* and *PHOG*. Table Table 4.1 summarizes the results of different experiments for linear kernel. We can see that there is a slight improvement in the Average Precision for all the cases of "linear kernel + subcategories" compared to the case of "linear kernel without subcategories". Similarly Table 4.2 shows the performance comparison for the case of intersection kernel. Here, "intersection kernel without and with subcategories" are compared with the "exp-χ^2 kernel without subcategories". We can see that in most of the cases that sub-categories helps in improving the performance. But it can be observed that, "linear+subcategories" still performs poorly compared to "intersection kernel+without subcategories". Similarly "intersection kernel + subcategories" does not perform as good as "exp-χ^2 kernel + without subcategories". Results using PHOG features can be seen in the table Table 4.3. We can observe that same pattern of performance comparison both for PHOW and PHOG features between different kernels with and without using subcategories.

ClassName	Linear Without Subclass		Linear - With SubClass-random		Linear - With SubClass-kmeans		Intersection Without Subclass	
SetName	Train	Val	Train	Val	Train	Val	Train	Val
1. Cityscape	0.89	0.27	0.97	0.29	0.97	0.29	**0.98**	**0.46**
2. Doorway	0.99	0.22	0.99	0.21	0.99	0.22	**1.00**	**0.32**
3. Nighttime	0.89	0.09	0.82	0.12	0.90	0.12	**0.97**	**0.25**

Table 4.1 Performance Variation with and without subcategories using linear kernel and level2 of $\text{PHOW}_{\text{gray}}$ features. We consider "linear kernel without subcategories" as the baseline and aim to reach the performance of "intersection kernel" by using "linear kernel with subcategories"

ClassName	Intersection Without Subclass		Intersection - With SubClass-random		Intersection - With SubClass-kmeans		exp-χ^2 Without Subclass	
SetName	Train	Val	Train	Val	Train	Val	Train	Val
1. Cityscape	0.98	0.46	0.99	0.45	0.99	0.42	**1.00**	**0.49**
2. Doorway	1.00	0.32	1.00	0.33	1.00	0.29	**1.00**	**0.33**
3. Nighttime	0.97	0.25	0.99	0.25	0.99	0.27	**0.98**	**0.27**

Table 4.2 Performance Variation with and without subcategories using intersection kernel and level2 of $\text{PHOW}_{\text{gray}}$ features. We consider "intersection without subcategories" as the baseline and aim to reach the performance of exp-χ^2 by using "intersection kernel with subcategories"

4.5 Summary

We have presented a method for using the subcategory information to improve the performance of the low complexity kernels like linear kernel. We showed how the subcategory information can be learnt automatically with an iterative optimization method. Performance of the proposed method is showcased on various synthetic 2-dimensional datasets, and also on real world datasets for video retrieval and object detection problems.

ClassName	Linear Without Subclass		Linear - With SubClass-random		Intersection Without Subclass	
SetName	Train	Val	Train	Val	Train	Val
1. Cityscape	0.65	0.21	0.78	0.24	0.86	0.25
2. Doorway	0.76	0.12	0.87	0.12	0.77	0.14

Table 4.3 Performance Variation with and without subcategories using linear kernel and level2 of PHOG-180 features. We consider "linear kernel without subcategories" as the baseline and aim to reach the performance of "intersection kernel" by using "linear kernel with subcategories"

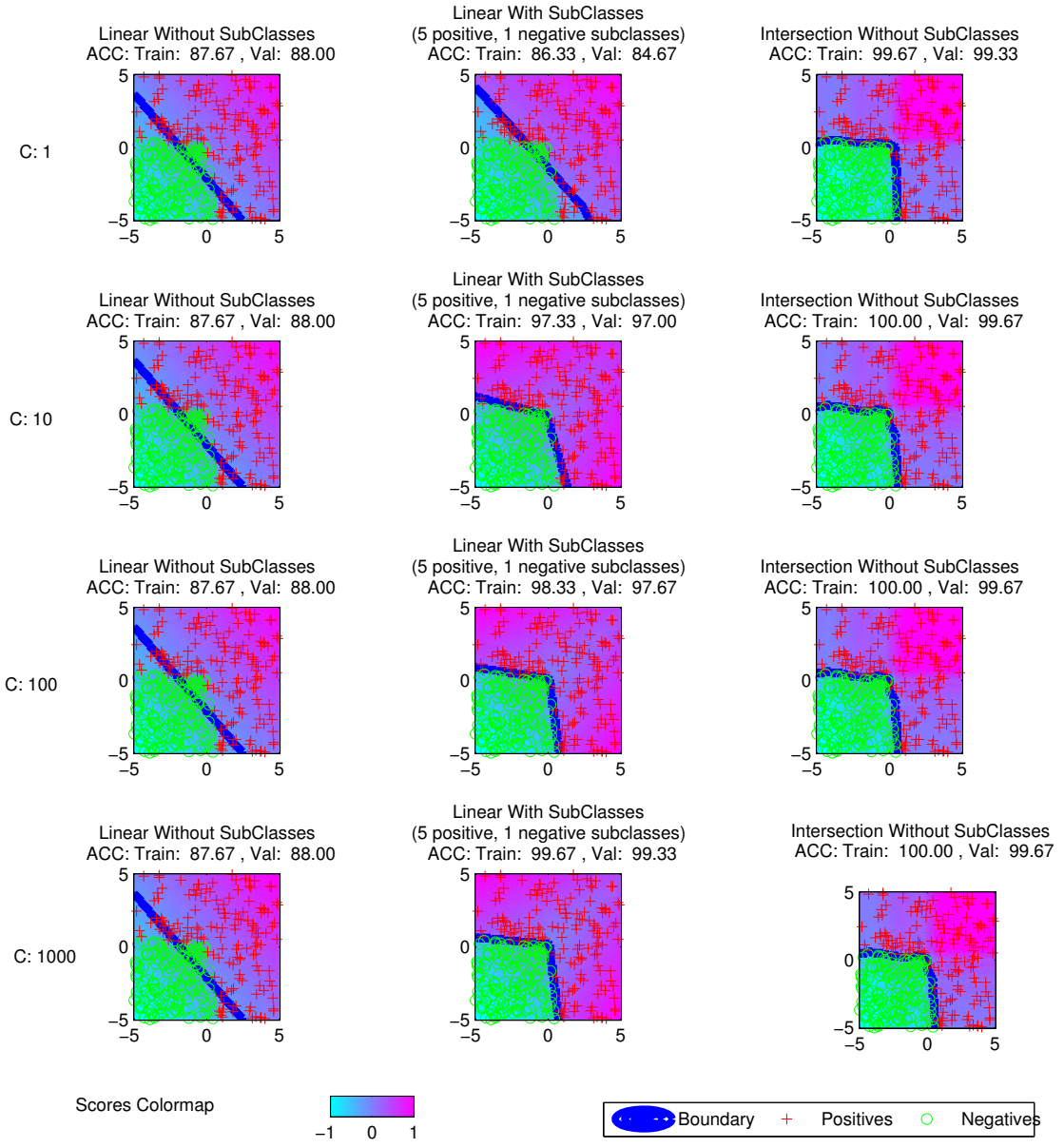

Figure 4.3 Comparison of boundaries obtained with and without subcategories using linear kernel for "DataType 1" with different values of SVM parameter C. We can also see the boundary obtained using intersection kernel. Each row corresponds to a different value of C. We can see that linear kernel with sub-categories is better than the one without using sub-categories.

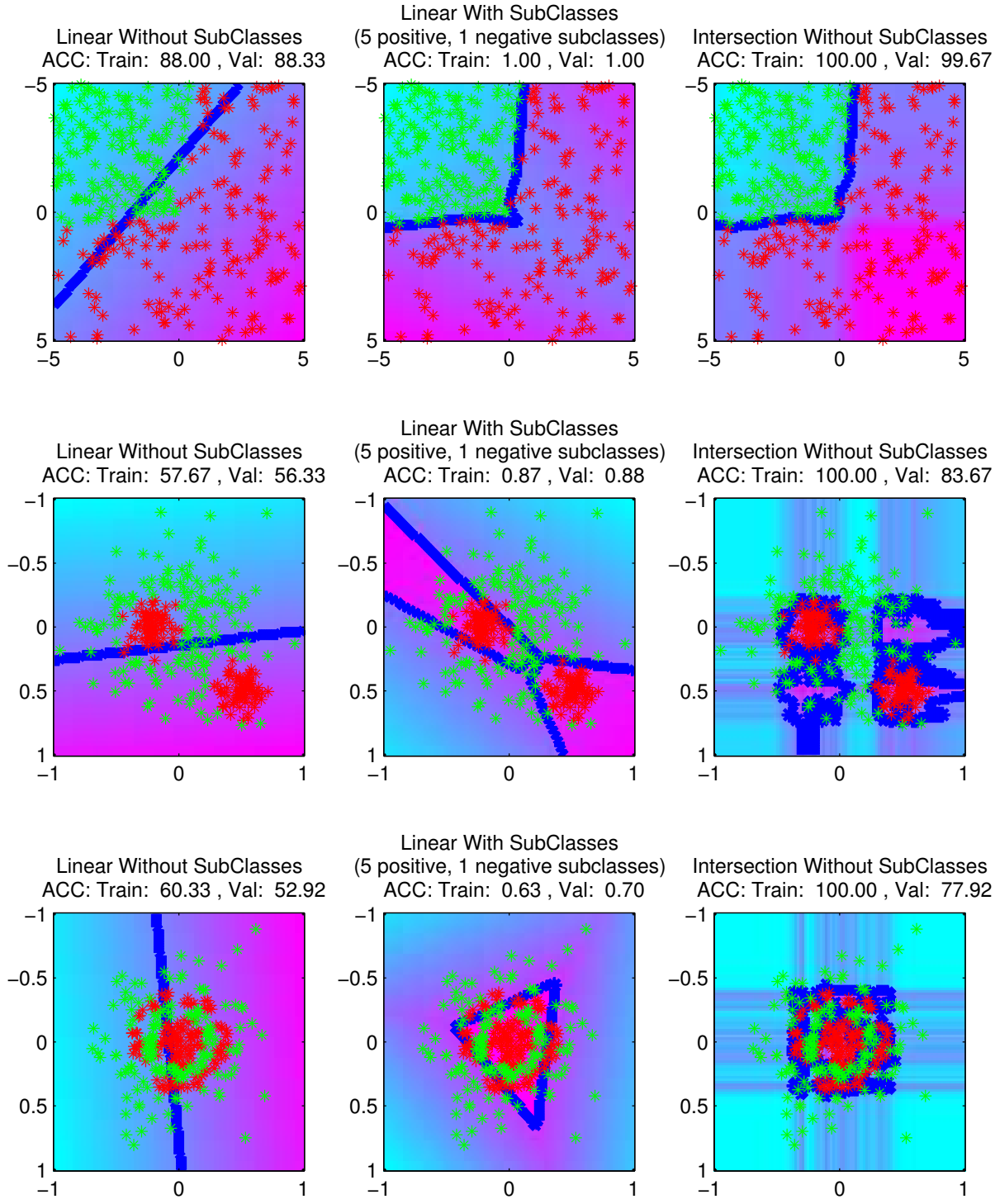

Figure 4.4 For datasets 1,2,3: Comparison of boundaries obtained and performance obtained using a) linear kernel with subcategories, b) linear kernel without subcategories and c) intersection kernel without subcategories. We can see that linear kernel with subcategories performs as good as intersection kernel.

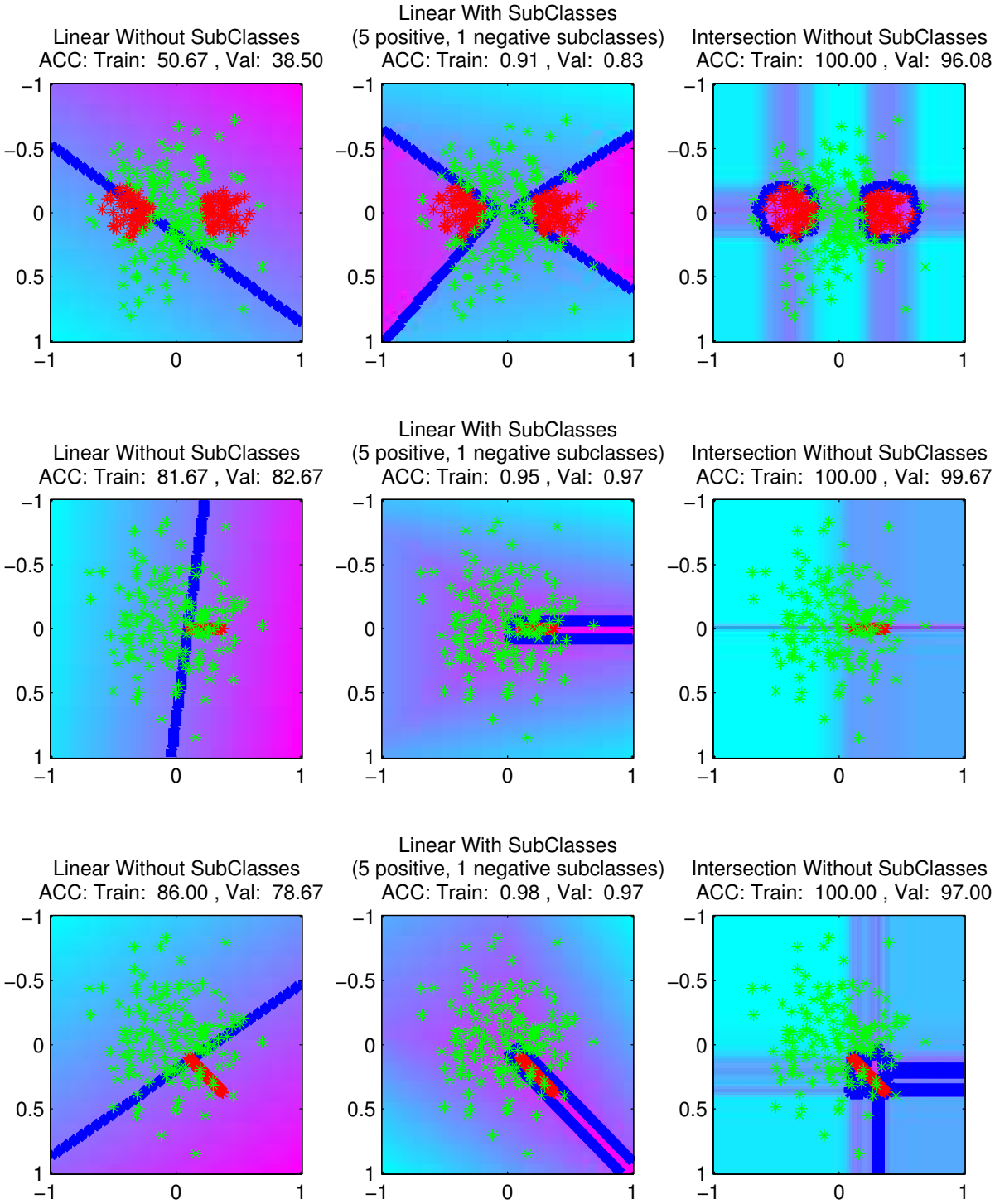

Figure 4.5 For datasets 4,5,6: Comparison of boundaries obtained and performance obtained using a) linear kernel with subcategories, b) linear kernel without subcategories and c) intersection kernel without subcategories. We can see that linear kernel with subcategories performs as good as intersection kernel.

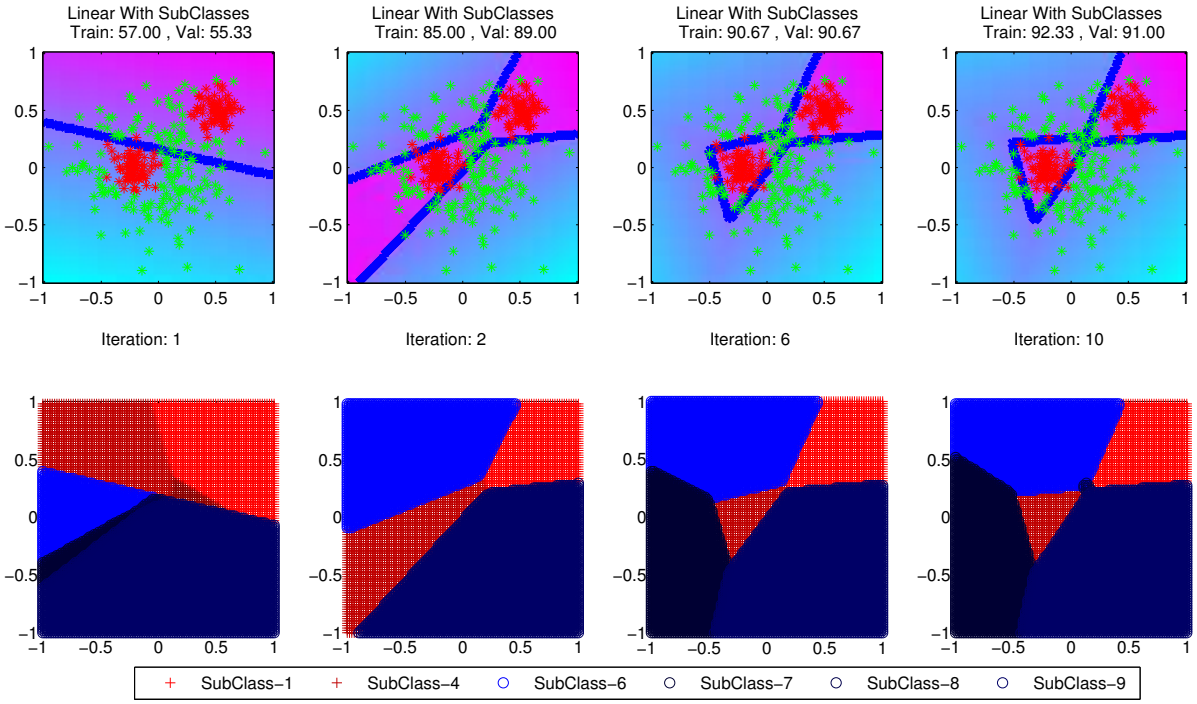

Figure 4.6 This figure shows how the classification accuracy and boundaries improve with each iteration of latent optimization using different initialization for "Datatype 2". Here we show the boundaries for iterations 1,2,6 and 10. The performance on validation data has changed from 55.3% after 1st iteration to 91.0% after the 10^{th} iteration.

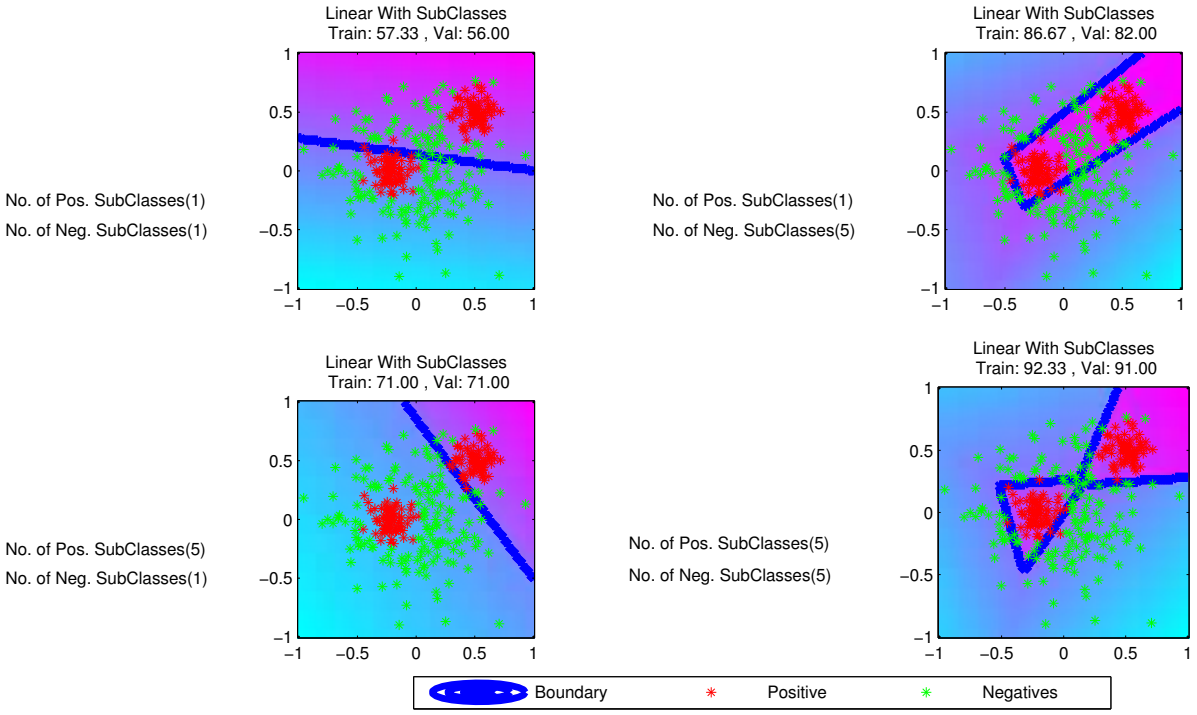

Figure 4.7 In the above figure, we can compare the performance and boundaries obtained using different number of positive and negative clusters for "Datatype 2". We can see that including subcategories for both positive and negative classes results in a better boundary and performance

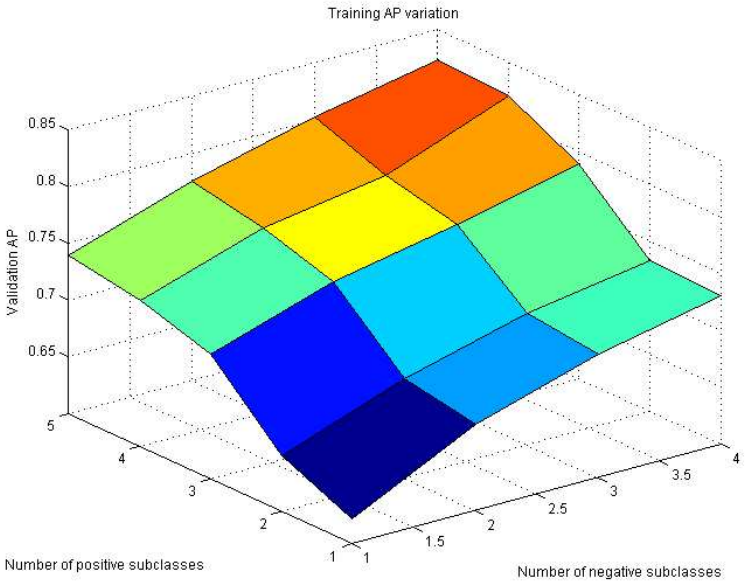

Figure 4.8 This result is on VOC 2007 data for "car" category. Variation of Average Precision on training set by varying the number of subclasses in positive and negative data. Here we train on training set and test on training set only. We can see that the performance increases by increasing both the number of positive and negative clusters.

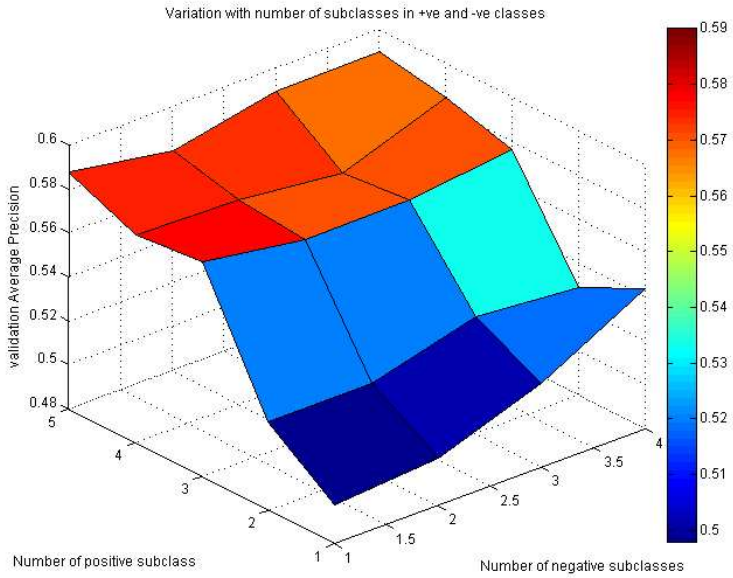

Figure 4.9 This result is on VOC 2007 data for "car" category. Variation of AP on validation set by varying the number of subclasses in positive and negative data. Here we train on training set and test on validation set. Here we can see that the performance increases with increase in the number of positve subclasses, but there is only slight variation with change in the number of negative subclasses

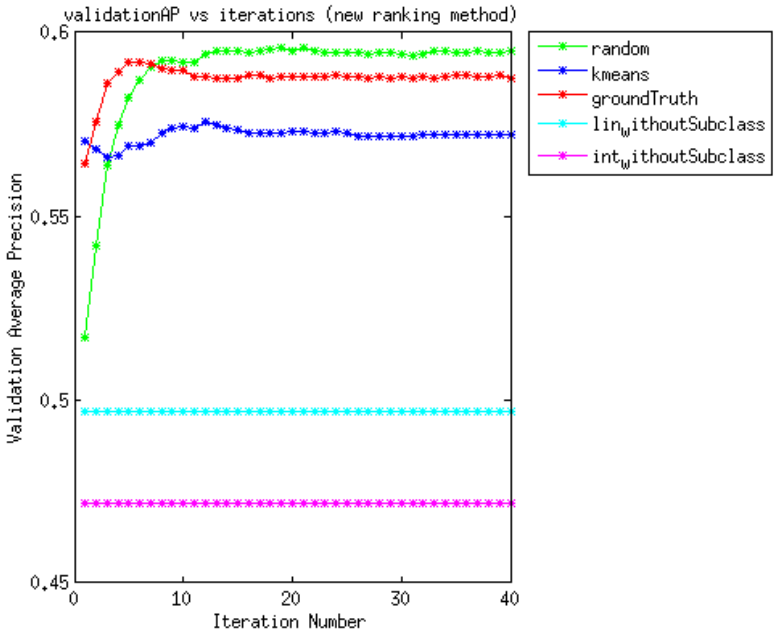

Figure 4.10 This result is on VOC 2007 data for "car" category. Variation of AP with Linear kernel using different initializations: Here we use linear kernel along with the subcategories. The two horizontal lines at the bottom correspond to the linear and intersection kernel without using subcategories. We can see that Average Precision increases with iterations of latent variable optimization for all types of initializations.

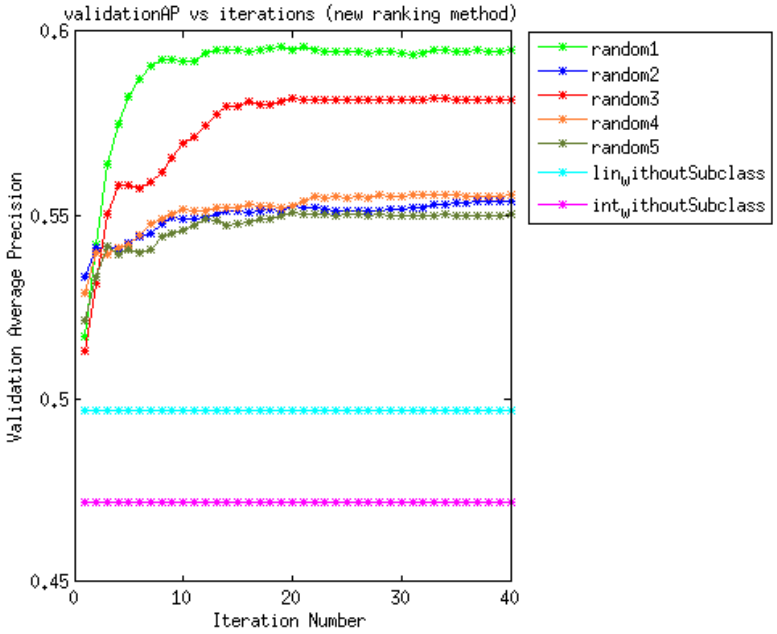

Figure 4.11 This result is on VOC 2007 data for "car" category. Variation of AP with Linear kernel using different seeds of random initializations: Here we observe how the random initializations effect the optimization. We can see that there is only a little difference in the final performance obtained using different random initializations.

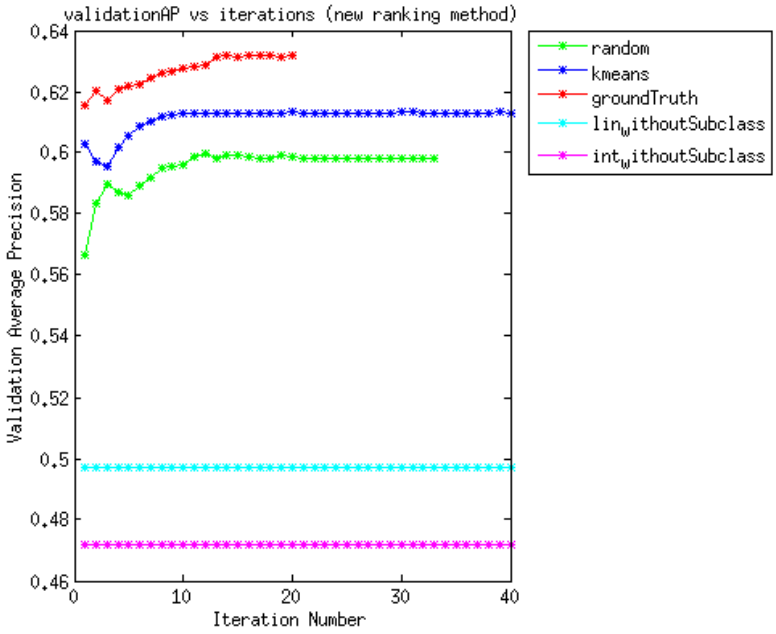

Figure 4.12 This result is on VOC 2007 data for "car" category. The above figure shows the variation of AP with intersection kernel using different initializations. We can see that the average precision obtained using different initializations vary by around 0.02-0.03. However, we can observe that intersection kernel with subcategories is better than that of the one without using subcategories for all types of initializations.

Chapter 5

Efficient Detection and Classification

5.1 Introduction

In computer vision applications such as object category classification and detection, the gold-standard kernels are the so called *generalized radial-basis function (RBF) kernels* [83, 74]. A typical example is the exponential-χ^2 kernel

$$K(\mathbf{x},\mathbf{y}) = e^{-\frac{1}{2\sigma^2}\chi^2(\mathbf{x},\mathbf{y})}, \quad \chi^2(\mathbf{x},\mathbf{y}) = \frac{1}{2}\sum_{l=1}^{d}\frac{(x_l - y_l)^2}{x_l + y_l}.$$

These kernels combine the benefits of two other important classes of kernels: the homogeneous additive kernels (e.g. the χ^2 kernel) and the RBF kernels (e.g. the exponential kernel). The homogeneous additive kernels are designed to compare histograms and are particularly useful in computer vision since most descriptors are, in fact, histograms (e.g. SIFT [39], GIST [67], HOG [12], bag-of-words [13], spatial pyramids [37]). The RBF kernels, on the other hand, can represent local templates which is useful for highly variable visual aspects.

A major problem in the use of the generalized RBF kernels is their computational cost. Training a non-linear SVM with such kernels is typically $O(dN^2)$ or $O(dN^3)$, where N is the number of training examples and d the data dimensionality. Testing the learned SVM is also very expensive, usually $O(dN)$, from the need to compare each novel datum to the support vectors determined during training (and these are usually of order N). In contrast, there exist methods that can train a linear SVM in time $O(dN)$ only (e.g. SVM-perf [30], PEGASOS [59], LIBLINEAR [19]) and testing is only of order $O(d)$ for a linear kernel (since it only involves a scalar product between the learnt weight vector $\mathbf{w}$ and the feature vector of the test image $\mathbf{x}$).

The fact that kernels can be seen as inner product in an appropriate vector space suggests that it may be possible to train and test efficiently SVMs even in the non-linear case. In symbols, for every positive-definite (PD) kernel $K(\mathbf{x},\mathbf{y})$ there exists a *feature map* $\Psi(\mathbf{x})$ such that

$$K(\mathbf{x},\mathbf{y}) = \langle \Psi(\mathbf{x}), \Psi(\mathbf{y}) \rangle \tag{5.1}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product in feature space. Typically $\Psi(\mathbf{x})$ is infinite dimensional and therefore not suitable for numerical computations. It may however be possible to find an *approximate* feature map $\hat{\Psi}(\mathbf{x})$ that (i) is finite dimensional and that (ii) generates a kernel $\langle \Psi(\mathbf{x}), \Psi(\mathbf{y}) \rangle$ that is a close approximation of (5.1), i.e.

$$\hat{K}(\mathbf{x}, \mathbf{y}) = \langle \hat{\Psi}(\mathbf{x}), \hat{\Psi}(\mathbf{y}) \rangle, \quad \hat{K}(\mathbf{x}, \mathbf{y}) \approx K(\mathbf{x}, \mathbf{y}). \tag{5.2}$$

So far feature maps have been proposed for the homogeneous additive kernels [40, 75] and for the RBF kernels [55]. In this paper we complete the story and give *efficient approximated feature maps for the generalized RBF kernels.* Specifically, Sect. 5.2.1 and Sect. 5.2.2 review the homogeneous additive and RBF kernels and their feature maps. Sect. 5.2.3 then reviews the generalized RBF kernels and derives feature maps for them, summarizing results on the approximation error and the computational cost, and Sect. 5.3 describes learning methods using l^1 regularization to encourage sparsity and improve testing speed. Finally, Sect. 5.5 compares empirically the various kernels and their approximations for the case of object detectors on the VOC 2007 dataset [18]. We show that the approximate feature map can improve performance significantly over that of the original additive kernels. An alternative to the method we propose is to employ Nyström-type approximations [78, 2, 5] to obtain linear feature maps, but these require a data dependent training step that we avoid here.

5.2 Kernels and feature maps

This section reviews the additive homogeneous kernels (Sect. 5.2.1), the RBF kernels (Sect. 5.2.2), and their feature maps [75, 55]. It then introduces the generalized RBF kernels and gives a finite dimensional approximate feature map for them (Sect. 5.2.3).

5.2.1 Additive homogeneous kernels

Common additive homogeneous kernels [75], such as the χ^2, intersection, Jensen-Shannon (JS), Hellinger's, are designed to compare probability distributions. An additive kernel is defined as

$$K(\mathbf{x}, \mathbf{y}) = \sum_{l=1}^{d} k(\mathbf{x}_l, \mathbf{y}_l) \tag{5.3}$$

where d is the dimension of the input histograms $\mathbf{x}, \mathbf{y}$, l is the component (bin) index, and $k(x, y)$ is an homogeneous PD kernel on the non-negative reals $\mathbb{R}_0^+$ (the kernel k is *homogeneous* if $k(cx, cy) = ck(x, y)$ for any $c > 0$). For instance, setting $k(x, y) = \min(x, y)$ in (5.3) yields the intersection kernel, and setting $k(x, y) = 2xy/(x + y)$ yields the χ^2 kernel.

[75] proposes closed-form approximated feature maps for all common homogeneous kernels. The construction starts by factorizing $k(x, y)$ as

$$k(x, y) = \sqrt{xy}\mathcal{K}(\lambda), \quad \lambda = \log \frac{y}{x}. \tag{5.4}$$

where $\mathcal{K}(\lambda)$ is called the kernel *signature*, and is a scalar function that fully describes the kernel k. It is then noted that $\mathcal{K}(\lambda)$ is the Fourier transform of a symmetric non-negative measure $\kappa(\omega)\, d\omega$

$$\mathcal{K}(\lambda) = \int_{-\infty}^{\infty} e^{-i\omega\lambda} \kappa(\omega)\, d\omega. \tag{5.5}$$

This in turn can be used to define a feature map $k(x, y) = \langle \Psi(x), \Psi(y) \rangle$ by

$$\begin{aligned} k(x,y) = \sqrt{xy}\mathcal{K}\left(\log\frac{y}{x}\right) &= \int_{-\infty}^{\infty} \left(\sqrt{x\kappa(\omega)}e^{-i\omega\log x}\right)^* \left(\sqrt{y\kappa(\omega)}e^{-i\omega\log y}\right) d\omega \\ &= \int_{-\infty}^{\infty} [\Psi(x)]_\omega^* [\Psi(y)]_\omega \, d\omega, \quad \boxed{[\Psi(x)]_\omega = \sqrt{x\kappa(\omega)}e^{-i\omega\log x}} \end{aligned} \tag{5.6}$$

An approximate finite feature map $\hat{\Psi}(\mathbf{x})$ can be constructed by sampling and truncating (5.6):

$$[\hat{\Psi}(x)]_j = \sqrt{L}[\Psi(x)]_{jL}, \quad j = -n, \ldots, n. \tag{5.7}$$

Due to symmetries of the feature map $\Psi(x)$, (5.7) reduces to a real vector of dimension $2n + 1$ [75]. The explicit form of the feature map in the case of χ^2 is given in Fig. 5.1.

Since for the intersection, χ^2, Jensen-Shannon, and Hellinger's kernels the density $\kappa(\omega)$ can be computed in closed form, the approximated feature map $\hat{\Psi}(x)$ can also be computed from (5.7) in closed form. A corresponding approximate feature map for an additive homogeneous kernel $K(\mathbf{x}, \mathbf{y}) \approx \langle \hat{\Psi}(\mathbf{x}), \hat{\Psi}(\mathbf{y}) \rangle$, is obtained by stacking $\hat{\Psi}(\mathbf{x}_l)$ for each of the d components of $\mathbf{x}$, i.e. $\hat{\Psi}(\mathbf{x}) = \mathrm{stack}_{l=1,\ldots,d} \hat{\Psi}(\mathbf{x}_l)$.

5.2.2 RBF kernels

The *radial-basis function (RBF) kernels* such as the exponential (Gaussian) kernel are function of the Euclidean distance between vectors $\mathbf{x}, \mathbf{y}$:

$$K_{\text{RB}}(\mathbf{x}, \mathbf{y}) = k(\|\mathbf{x} - \mathbf{y}\|_2^2)$$

where $k : \mathbb{R}_0^+ \to \mathbb{R}$ is the *kernel profile* [10]. An important example is the exponential kernel

$$K_{\text{RB}}(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{y}\|_2^2\right).$$

Such kernels are a particular cases of the *translation invariant kernels* that can be written as

$$K_{\text{RB}}(\mathbf{x}, \mathbf{y}) = \mathcal{K}_{\text{RB}}(\boldsymbol{\lambda}), \quad \boldsymbol{\lambda} = \mathbf{y} - \mathbf{x}.$$

We may call $\mathcal{K}_{\text{RB}}$ the kernel signature in analogy with the homogeneous case, but here the signature has d-dimensional input $\boldsymbol{\lambda}$. Moreover $\boldsymbol{\lambda} = \mathbf{y} - \mathbf{x}$ has a linear rather than logarithmic dependency on $\mathbf{x}$ and $\mathbf{y}$. As noticed in [55] and analogously to (5.6), the signature is the Fourier transform of a non-negative symmetric density $\kappa_{\text{RB}}(\boldsymbol{\omega})$ (Bochner theorem) and $\kappa_{\text{RB}}(\boldsymbol{\omega})$ can be used to define a feature map $\Psi_{\text{RB}}(\mathbf{x})$ for the RBF kernel:

$$\mathcal{K}_{\text{RB}}(\boldsymbol{\lambda}) = \int_{\mathbb{R}^d} e^{-i\boldsymbol{\omega}^\top\boldsymbol{\lambda}} \kappa_{\text{RB}}(\boldsymbol{\omega})\, d\boldsymbol{\omega}, \quad \boxed{[\Psi_{\text{RB}}(\mathbf{x})]_{\boldsymbol{\omega}} = \sqrt{\kappa_{\text{RB}}(\boldsymbol{\omega})}e^{-i\boldsymbol{\omega}^\top\mathbf{x}}.} \tag{5.8}$$

Differently from (5.7), it is not practical to regularly sample $[\Psi_{\text{RB}}(\mathbf{x})]_{\boldsymbol{\omega}}$ to obtain an approximate feature map due to the high dimensionality of $\boldsymbol{\omega} \in \mathbb{R}^d$. Instead, [55] proposes to use random sampling. Without loss of generality, assume that the kernel is properly normalized, i.e. that $K_{\text{RB}}(\mathbf{x}, \mathbf{x}) = 1$. Then $1 = K_{\text{RB}}(\mathbf{x}, \mathbf{x}) = \mathcal{K}_{\text{RB}}(0) = \int \kappa_{\text{RB}}(\boldsymbol{\omega})\, d\boldsymbol{\omega}$ implies that $\kappa_{\text{RB}}(\boldsymbol{\omega})$ is non-negative and sums to one. Hence $\kappa_{\text{RB}}(\boldsymbol{\omega})$ can be thought of as a probability density and (5.8) approximated by an empirical average:

$$\mathcal{K}_{\text{RB}}(\boldsymbol{\lambda}) = E[e^{-i\boldsymbol{\omega}^\top \boldsymbol{\lambda}}] \approx \frac{1}{m} \sum_{j=1}^{m} e^{-i\boldsymbol{\omega}_j^\top \boldsymbol{\lambda}}, \quad \boldsymbol{\omega}, \boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m \sim \kappa_{\text{RB}}(\boldsymbol{\omega}).$$

The summation can be expressed as $\langle \hat{\Psi}_{\text{RB}}(\mathbf{x}), \hat{\Psi}_{\text{RB}}(\mathbf{y}) \rangle$, where the approximated feature map $\hat{\Psi}_{\text{RB}}(\mathbf{x})$ is given by

$$[\hat{\Psi}_{\text{RB}}(\mathbf{x})]_j = \frac{1}{\sqrt{m}} e^{-i\boldsymbol{\omega}_j^\top \mathbf{x}}, \quad j = 1, \ldots, m.$$

Note that the vectors $\boldsymbol{\omega}_j \in \mathbb{R}^d$ are randomly sampled from $\kappa_{\text{RB}}(\boldsymbol{\omega})$ and act on the data $\mathbf{x}$ as random projections. Note also that the complex vector $\hat{\Psi}_{\text{RB}}(\mathbf{x})$ can be equivalently written as a $2m$ dimensional real vector in terms of sine and cosine functions [55]. Its form for the exponential kernel is given in Fig. 5.1.

5.2.3 Generalized RBF kernels

The *generalized RBF kernels* extend the RBF kernels to use a metric not necessarily Euclidean. For our purposes, this is best seen in terms of kernels. Recall that, for any PD kernel $K(\mathbf{x}, \mathbf{y})$, the equation

$$D^2(\mathbf{x}, \mathbf{y}) = K(\mathbf{x}, \mathbf{x}) + K(\mathbf{y}, \mathbf{y}) - 2K(\mathbf{x}, \mathbf{y}) \tag{5.11}$$

defines a corresponding squared metric [58]. For instance, from the intersection, χ^2, JS, and Hellinger's kernel one obtains the l^1 distance, and the squared χ^2, JS, and Hellinger's distances respectively. Given an RBF kernel $K_{\text{RB}}(\mathbf{x}, \mathbf{y}) = k(\|\mathbf{x} - \mathbf{y}\|_2^2)$, one can then obtain a corresponding generalized variant

$$K_{\text{RB}D^2}(\mathbf{x}, \mathbf{y}) = k(D^2(\mathbf{x}, \mathbf{y})). \tag{5.12}$$

Constructing an approximate feature map for (5.12) involves two steps. First, the feature map (5.7) can be used to approximate $D^2(\mathbf{x}, \mathbf{y})$ as *Euclidean distance* in feature space:

$$\begin{aligned} D^2(\mathbf{x}, \mathbf{y}) &\approx \hat{K}(\mathbf{x}, \mathbf{x}) + \hat{K}(\mathbf{y}, \mathbf{y}) - 2\hat{K}(\mathbf{x}, \mathbf{y}) \\ &= \langle \hat{\Psi}(\mathbf{x}), \hat{\Psi}(\mathbf{x}) \rangle + \langle \hat{\Psi}(\mathbf{y}), \hat{\Psi}(\mathbf{y}) \rangle - 2\langle \hat{\Psi}(\mathbf{x}), \hat{\Psi}(\mathbf{y}) \rangle \\ &= \|\hat{\Psi}(\mathbf{x}) - \hat{\Psi}(\mathbf{y})\|_2^2. \end{aligned} \tag{5.13}$$

Hence the generalized RBF kernel $K_{\text{RB}D^2}$ can be approximated by the RBF kernel

$$K_{\text{RB}D^2}(\mathbf{x}, \mathbf{y}) \approx K_{\text{RB}}(\hat{\Psi}(\mathbf{x}), \hat{\Psi}(\mathbf{y})) = k(\|\hat{\Psi}(\mathbf{x}) - \hat{\Psi}(\mathbf{y})\|_2^2).$$

Compute a $2m$ dimensional approximate finite feature map for the exponential-χ^2 kernel $K(\mathbf{x}, \mathbf{y}) = \exp(-\frac{1}{2\sigma^2}\chi^2(\mathbf{x}, \mathbf{y}))$.

Preprocessing: Draw m random vectors $\boldsymbol{\omega}$, sampled from a $(2n+1)d$ isotropic Gaussian of variance $1/\sigma^2$.

Given: A vector $\mathbf{x} \in \mathbb{R}^d$.
Compute: The approximate feature map $\hat{\Psi}_{\text{RB}\chi^2}(\mathbf{x})$

1: Construct the $2n+1$ dimensional vector $\hat{\Psi}(\mathbf{x})$ by setting for $j = 0, \ldots, 2n$

$$[\hat{\Psi}(\mathbf{x})]_j = \begin{cases} \sqrt{xL\,\mathrm{sech}(0)}, & j = 0, \\ \sqrt{2xL\,\mathrm{sech}(\pi\frac{j+1}{2}L)}\cos\left(\frac{j+1}{2}L\log x\right) & j > 0 \text{ odd}, \\ \sqrt{2xL\,\mathrm{sech}(\pi\frac{j}{2}L)}\sin\left(\frac{j}{2}L\log x\right) & j > 0 \text{ even}, \end{cases} \tag{5.9}$$

2: Construct the $2m$ dimensional vector $\hat{\Psi}_{\text{RB}\chi^2}(\mathbf{x})$ by setting for $j = 1, \ldots, 2m$

$$[\hat{\Psi}_{\text{RB}\chi^2}(\mathbf{x})]_j = \begin{cases} \frac{1}{\sqrt{m}}\cos\left(\boldsymbol{\omega}_{\frac{j+1}{2}}^\top \hat{\Psi}(\mathbf{x})\right), & j \text{ odd}, \\ \frac{1}{\sqrt{m}}\sin\left(\boldsymbol{\omega}_{\frac{j}{2}}^\top \hat{\Psi}(\mathbf{x})\right), & j \text{ even}. \end{cases} \tag{5.10}$$

Figure 5.1 Feature map for the exponential-χ^2 kernel. The resulting vector is $2m$ dimensional. Here n controls the χ^2 approximation, and is typically chosen as a small number, e.g. $n = 1$ (and in this case $L \approx 0.8$, see [75] for details on how to choose this parameter). The algorithm requires only two modifications for any other RBF-D^2 kernel. First, (5.9) should be adjusted according to (5.6) to match the metric D^2 (closed forms are given in [75]). Second, the projections $\boldsymbol{\omega}_j$ should be sampled from from the density $\kappa_{\text{RB}}(\boldsymbol{\omega})$ corresponding to the desired RBF profile as given by (5.8).

Second, the random Fourier features (Sect. 5.2.2) can be used to approximate K_{RB}. Composing the two approximations yields an approximated feature map for the generalized RBF kernel:

$$\boxed{[\hat{\Psi}_{\text{RB}D^2}(\mathbf{x})]_j = e^{-i\boldsymbol{\omega}_j^\top \hat{\Psi}(\mathbf{x})}, \quad j = -n, \ldots, n.} \tag{5.14}$$

The complete procedure (from measured feature vector $\mathbf{x}$ to approximating feature vector $\hat{\Psi}_{\text{RB}\chi^2}(\mathbf{x})$ for the RBF-χ^2 kernel is given in Fig. 5.1.

Errors and computational cost. The cost of computing the feature map (5.14) is $O(mnd)$, where m is the number of random Fourier features, n the dimensionality of the feature map (5.7) of the additive kernel, and d is the dimensionality of the data $\mathbf{x}$. As shown in [75], usually small values of n (e.g. $n = 1, 2$) are sufficient to yield good accuracy. In particular, it can be shown (the proof is omitted for brevity) that the error decreases exponentially fast with n for the smooth kernels such as χ^2. Based on (5.11) and (5.12), the error in approximating the additive kernel propagates to the RBF-D^2 kernel multiplied by the Lipschiz constant of the RBF kernel profile k, which is usually small. Overall, it can be shown that the error is dominated by approximating the RBF kernel by the m random Fourier

features. In particular, based on the error analysis of [55], $m = \Omega(1/\epsilon^2)$ random projections are needed to achieve a uniform approximation error ϵ with any fixed (large) probability.

5.3 Classification with a standard Linear SVM

A limitation of the random Fourier features is the relatively large number of projections required to obtain good accuracy (Sect. 5.5). As seen in Sect. 5.2.3, the accuracy improves only as $O(1/\sqrt{m})$ with the number of projections, and the cost of evaluating the feature map $\hat{\Psi}_{\text{RB}D^2}(\mathbf{x})$ is $O(mdn)$. The parameter n can usually be small (e.g. $n = 1$), but m is often in the order of several thousands. Note, however, that the random Fourier features $\boldsymbol{\omega}_j$ can be "optimized" in several ways. For instance, it is possible to select (e.g. learn) from a large set of such features only the ones that are useful for a *specific* problem.

5.4 Speedup by learning useful projections using sparse SVM

The simplest way to select useful random Fourier features is to use an appropriate regularizer for SVM training. Recall that in training a linear SVM one solves the problem

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{i=1}^{N} l(\mathbf{x}_i, y_i; \mathbf{w}), \quad l(\mathbf{x}_i, y_i; \mathbf{w}) = \max\{0, 1 - y_i\langle \mathbf{w}, \hat{\Psi}_{\text{RB}D^2}(\mathbf{x}_i)\rangle\} \tag{5.15}$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_N$ are training vectors, $y_1, \ldots, y_N \in \{-1, +1\}$ their labels, and $l(\mathbf{x}_i, y_i; \mathbf{w})$ the hinge loss. We consider two other formulations where l^1 regularization is used in order to encourage sparsity in the $\mathbf{w}$ vector, and for which efficient implementations exist in LIBLINEAR [19]:

$$\min_{\mathbf{w}} \tfrac{1}{2}\|\mathbf{w}\|_1 + \tfrac{C}{N}\textstyle\sum_{i=1}^{N} l(\mathbf{x}_i, y_i; \mathbf{w})^2, \tag{5.16}$$

$$\min_{\mathbf{w}} \tfrac{1}{2}\|\mathbf{w}\|_1 + \tfrac{C}{N}\textstyle\sum_{i=1}^{N} \log(1 + \exp(-y_i\langle \Psi(\mathbf{x}_i), \mathbf{w}\rangle)). \tag{5.17}$$

these are known respectively as l^1l^2-SVM and l^1-logistic regression. We will refer to these implementations as SVM$^{\text{sparse}}$ and LR$^{\text{sparse}}$ respectively, and to the standard SVM of (5.15) as SVM$^{\text{dense}}$.

From (5.10) we see that successive components of $\hat{\Psi}_{2j-1}(\mathbf{x})$, $\hat{\Psi}_{2j}(\mathbf{x})$ of the feature map share the same projection $\boldsymbol{\omega}_j$. Thus if $\mathbf{w}_{2j-1} = \mathbf{w}_{2j} = 0$ the projection $\boldsymbol{\omega}_j$ can be discarded in the evaluation of the SVM $\langle \mathbf{w}, \hat{\Psi}_{\text{RB}D^2}(\mathbf{x})\rangle$. Setting such projections to zero can be encouraged by considering the problem

$$\min_{\mathbf{w}} \frac{1}{2}\sum_{j=1}^{m} \sqrt{\mathbf{w}_{2j-1}^2 + \mathbf{w}_{2j}^2} + \frac{C}{N}\sum_{i=1}^{N} l(\mathbf{x}_i, y_i; \mathbf{w}), \tag{5.18}$$

This problem is known as Multiple Kernel Learning SVM (MKL-SVM) [3, 71] where we defined a base kernel for each projection $\boldsymbol{\omega}_j$.

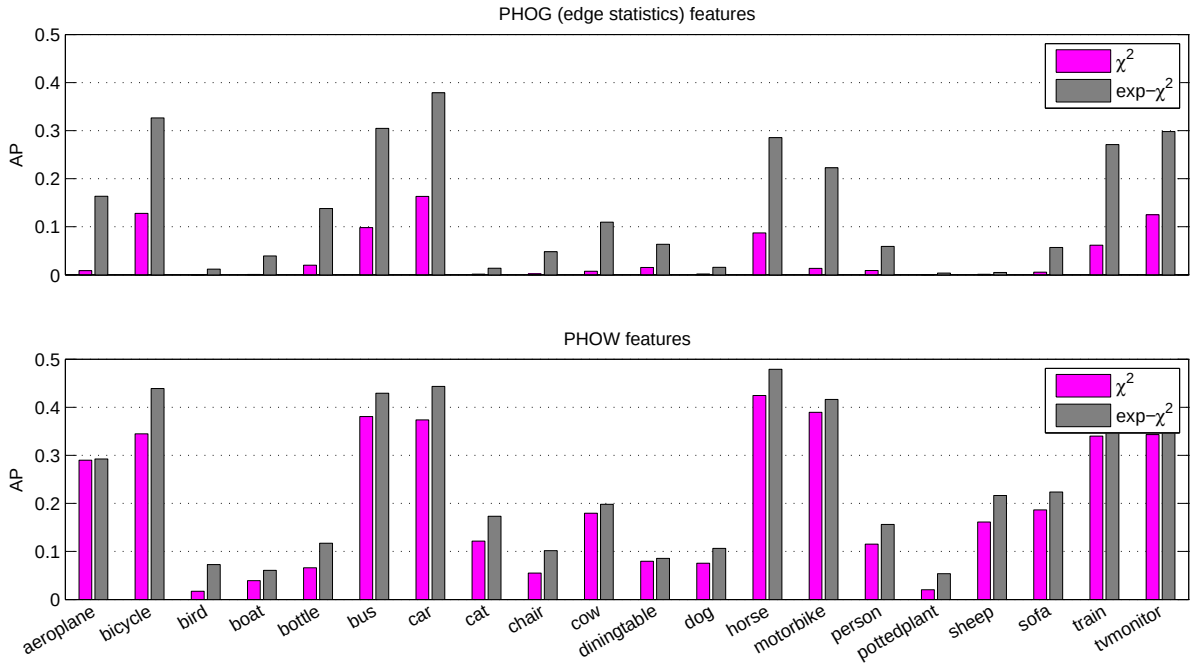

Figure 5.2 Effect of RBF kernels on different feature types. AP of the sliding window object detector [75] for PHOW and PHOG features and χ^2 and exp-χ^2. The PHOG features benefit substantially from the use of the exponential kernel, while the PHOW features are much less sensitive.

Notice that, compared to the MKL formulation (5.18), the SVM$^{\text{sparse}}$ and LR$^{\text{sparse}}$ formulations encourage any coefficient, but not specifically *pairs* of related coefficients, to be zero. In practice, thus, we can remove a projection $\boldsymbol{\omega}_j$ only if both $\mathbf{w}_{2j-1}$ and $\mathbf{w}_{2j}$ are set to zero. This is usually sufficient to discard a very large number of projections, but we would expect an implementation of the MKL formulation to results in an even larger number of discarded projections. We don't investigate MKL any further here though.

5.5 Experiments

We evaluate the proposed feature maps as part of the construction of an object detector on the PASCAL VOC 2007 [18] data. The VOC detection challenge involves predicting the bounding box and label of each instance of the target class in several thousand test images. We work on top of the state-of-the-art multiple-stage detector proposed in [73], using source code provided by the authors. The multiple stages are a cascade of a linear, χ^2 and exponential-χ^2 (exp-χ^2) detector. Thus the feature map (5.14) can be used to speed-up the third stage of the cascade (exp-χ^2), which is also noted to be the bottleneck in [73].

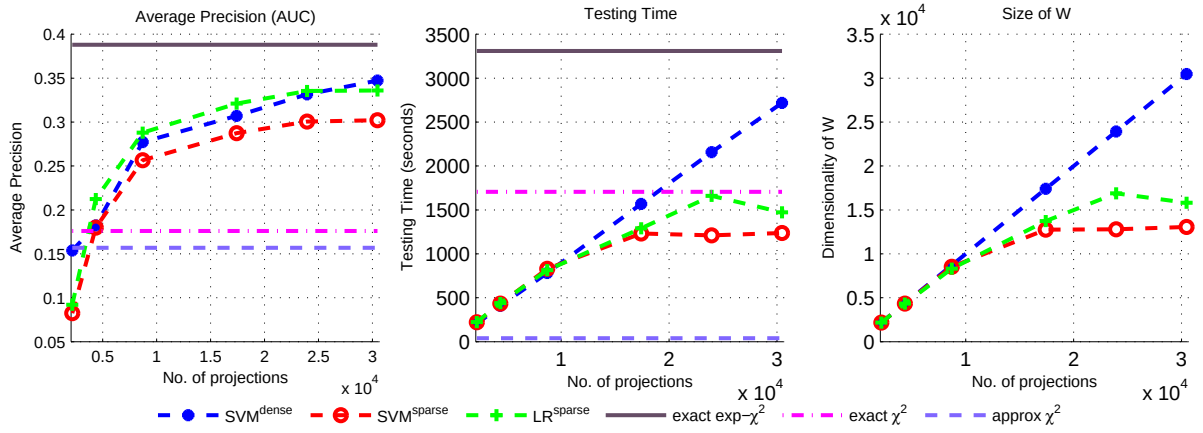

Figure 5.3 Additive, exponential, and approximated kernels. Performance of a car detector. The baselines are the exact exp-χ^2 kernel (corresponding to the third stage of the cascade [73]) and the exact additive χ^2 kernel (second stage). The exponential variant is better by 20% AP. The approximated exp-χ^2 kernel is shown for an increasing number of random Fourier features m (results are averaged over five sets of random projections). The approximation trades-off speed and accuracy. For instance, at 30% AP the approximation is twice as fast as the exact exp-χ^2 kernel and twice as accurate as the exact additive χ^2 kernel. The sparse solutions found by SVM$^{\text{sparse}}$ and LR$^{\text{sparse}}$ are faster still with only a small impact on performance (and in some case with improved performance).

5.5.1 Experimental Setup

While [73] uses up to six image descriptors, here we focus only on PHOG and PHOW since these illustrate the main features of the proposed approach. PHOW are visual words obtained from rotationally invariant SIFT descriptors extracted on a regular grid with five pixels spacing, at four multiple scales (10, 15, 20, 25 pixel radii), zeroing the low contrast ones. Descriptors are then quantized in 300 visual words. PHOG [6] is a sparse multi-scale version of HOG [12]. The Canny edge detector is used to compute an edge map and the underlying image gradient is used to assign an orientation and a weight to each edge pixel. The orientation angle is then quantized in sixteen bins with soft linear assignment and an histogram is computed. Both PHOW and PHOG features are then converted into spatial histograms [37] with 1×1, 2×2, and 4×4 subdivisions in order to characterize each candidate object bounding box B. Overall each region is described by a 6,300 dimensional vector $\mathbf{x}$ for the PHOW features and 336 dimensional for the PHOG features.

Fig. 5.2 compares the performance of the exact kernels with PHOW and PHOG features. A first important observation is that, while the PHOG features benefit substantially from the exponential kernel, the improvements with the PHOW features is much more limited. Our interpretation is as follows: The PHOG features (edglets) are not very discriminative in isolation, but their *combination*, capturing the shape of the object, is. This makes the local exponential kernel particularly important (for its template matching ability). The PHOW features, on the other hand, are quite discriminative in isolation since they match the semi-local SIFT 'footprint', and can be used with an additive kernel that, by operating

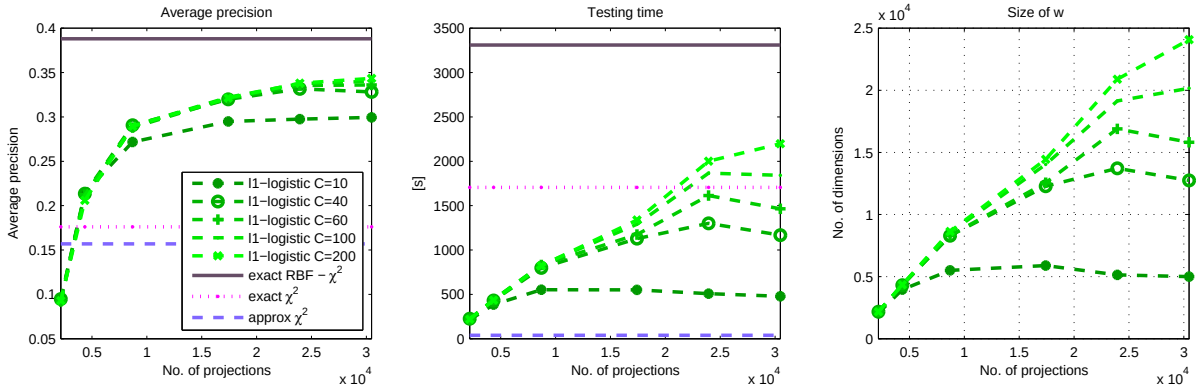

Figure 5.4 Sparsity vs performance. In order to make the approximated random Fourier features more competitive, we perform a random selection of the useful projections. Based on the formulations of Sect. 5.3, the only parameter that controls sparsity is C, which also controls overfitting. As C is increased from 10 to 200 the AP matches the one of the dense SVM, but also the testing time. For low value of C it is however possible to obtain fairly competitive AP (about 30%, still much better than the exact χ^2 kernel) with a seven-fold increase of speed compared to the exact kernel.

independently on each component, is more similar to a voting process than to template matching. In this case, just the *presence* of certain visual words is important, not so much their specific combination. Thus less is gained by the exp-χ^2 kernel over the χ^2 kernel. Therefore, in the rest of the experiments we focus only on the PHOG features.

5.5.2 Approximated kernels.

Fig. 5.3 compares the exact and approximated exp-χ^2 generalized RBF kernels and the χ^2 additive homogeneous kernel on the object class *car*. The exact exp-χ^2 kernel performs much better than χ^2 for the PHOG features. The dense approximated version, SVM$^{\text{dense}}$, converges to the exact exp-χ^2 performance as more random projections are added. With around 10^4 projections the approximated exp-χ^2 kernel is already much better than the χ^2 kernel, and it is about seven times faster in testing than the exact exp-χ^2. The sparse SVM, and especially the sparse logistic regression, can further discard up to half of the projections as redundant, without impacting accuracy significantly. In this example the approximation does not improve training time compared to th exact kernel due to the limited amount of training data; however, the training complexity is just linear, compared to quadratic of the exact kernel, so that the approximate representation would be better for large enough data sets. The exact RBF kernel, which could be approximated by using directly the technique from [55], was also tested but resulted in extremely poor performance (below 11% AP). This is due to the fact that the l^2 metric is a particularly poor match for the PHOG features. Fig. 5.4 and 5.5 illustrate in more detail the effect of C on the sparsity and speed and their trade-off. It can be seen that an AP performance far superior to that of an exact χ^2 kernel can be achieved at a lower test cost. For example for $C = 10$ and 2×10^4 projections the AP is

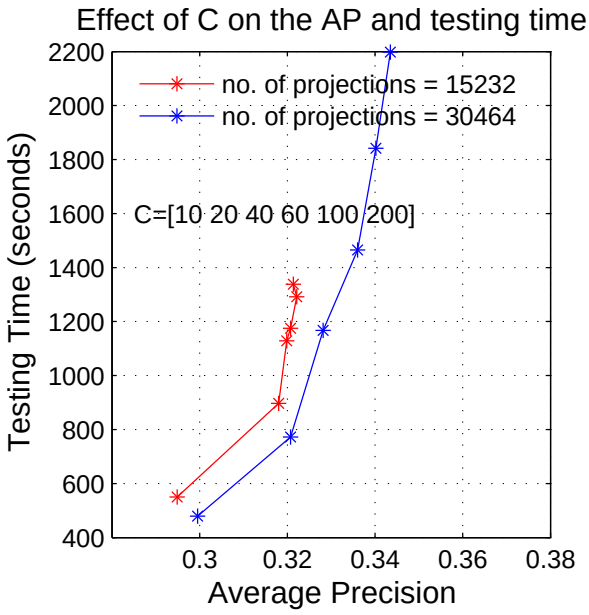

Figure 5.5 Effect of C on sparsity. Both SVM$^{\text{sparse}}$ and the LR$^{\text{sparse}}$ control sparsity through the regularization parameter C. The figure illustrates for LR$^{\text{sparse}}$ the variation of testing time (inversely proportional to the sparsity of the learned weight vector $\mathbf{w}$) and average precision as C is varied. The experiment is averaged over five sets of random projections, and repeated for sets of 15×10^3 and 30×10^3 projections. Notice how searching among more projections finds smaller sets of projections for a given level of accuracy.

about twice that of exact χ^2 and it is about three times faster. Fig. 5.6 shows that similar effects hold for all the 20 VOC classes.

5.6 Summary

We have introduced a method to construct a finite dimensional approximate feature map for the generalized RBF kernels. In general, the approximation is independent of the number of support vectors, yields linear training complexity, and may easily be included into an on-line training framework. We have shown that the finite feature map can be used to speedup testing significantly in a detection task while still yielding an accuracy far superior to that of the additive kernels for certain visual features.

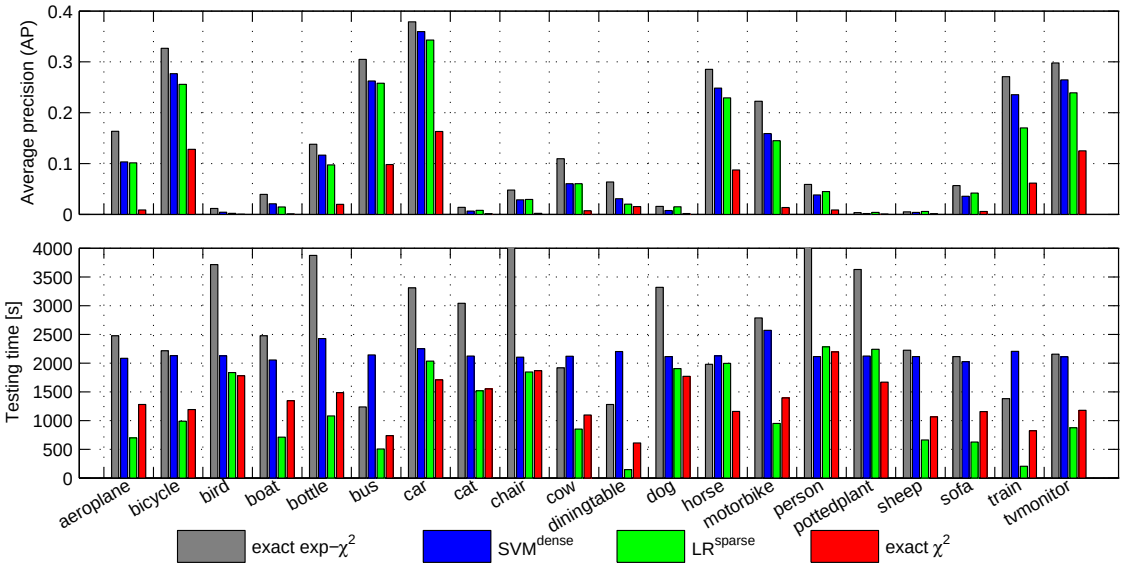

Figure 5.6 Effect on all VOC classes Exact exponential and additive χ^2 kernels for the twenty classes of the VOC 2007 detection challenge, along with their approximations. Top: both the dense and sparse approximations perform nearly as well as the exact kernel. Bottom: the testing time of the approximated sparse SVM is from two to three times better than the dense SVM.

Chapter 6

Conclusions and Future work

To conclude this thesis, we have proposed methods for efficient detection and classification of visual data. Our aim has been to achieve speedup with good classification performance. Each of the contributions of this thesis can be summarized as follows.

We have started working using SVMs which have proven to be a very good choice of classifiers for a wide range of applications in literature. But the problem with these has been that their computational complexity. We have mainly focused on working on large real world datasets.

To this end, we have experimented using various state-of-the-art image representations (like bag of words, histogram of orient gradients) and SVM classifiers for the task of semantic video retrieval. We have shown how the choices of representations, classifier parameters, noise in the annotations effect the performance of the classifiers. We have demonstrated this on the datasets of TRECVID High level feature extraction task for the years 2008 and 2009 for various categories of scenes, objects and actions. We have also shown that fast intersection kernel can be a good choice for this task of semantic video retrieval. Possible future directions in this work are more experiments on feature parameters and replacing the local feature descriptors namely SIFT with other descriptors which can be computed in a faster manner. Also we have used a generic approach, that is we have used a common set of features and classification methodologies for all the classes. We can try to find the optimal set of features and classifiers for each of the categories separately. Also, we can apply some post-processing classifiers which are designed based on the heuristics specific to the category of interest. This can boost the results further.

Then, we have shown how the sub-categories within each class can be used to improve the performance of classification/detection. We have shown that this can be particularly useful for the case of computationally inexpensive kernels like linear kernels. We have also proposed a method for learning these sub-groupings, that help in achieving the optimal performance. We have performed experiments on synthetic 2-dimensional datasets and real life datasets(VOC 2007 and TRECVID). We have investigated the use of subcategories using different sets of features, different SVM parameters, number of subclasses and choice of classes for different kernels using subcategories. Possible directions in this

work would be the use of learning better combination of features for each subclass by usina multiple kernel learning frameworkr for each subclass.

Finally, we have introduced a method to construct a finite dimensional approximate feature map for the generalized RBF kernels. In general, the approximation is independent of the number of support vectors, yields linear training complexity, and may easily be included into an on-line training framework. We have shown that the finite feature map can be used to speedup testing significantly in a detection task while still yielding an accuracy far superior to that of the additive kernels for certain visual features. As a future work, the feature maps for generalized RBF kernels can be used in conjunction with the feature maps of intersection kernel to obtain a better performance in a fast manner. In order to speedup further, we should find a better mechanism of finding/learning the good projections.

Related Publications

- *Sreekanth Vempati*, Andrea Vedaldi, Andrew Zisserman, C. V. Jawahar,
  "Generalized RBF feature maps for efficient Detection",
  in *Proceedings of the* 21^{st} *British Machine Vision Conference*(BMVC) 2010 (Oral Presentation),
  Aberystwyth, UK.
- *Sreekanth Vempati*, Mihir Jain, Omkar M. Parkhi, C. V. Jawahar,
  Andrea Vedaldi, Marcin Marszalek, Andrew Zisserman
  "Oxford-IIIT Notebook Paper",
  in *Proceedings of the NIST-TRECVID Workshop*, 2009 Gaithersburg, Md., USA.
- James Philbin, Manuel Marin-Jimenez, Siddharth Srinivasan and Andrew Zisserman,
  Mihir Jain, *Sreekanth Vempati*, Pramod Sankar and C. V. Jawahar
  "Oxford-IIIT Notebook Paper",
  in *Proceedings of the NIST-TRECVID Workshop*, 2008 Gaithersburg, Md., USA.

Bibliography

[1] S. Ayache and G. Quenot. Video corpus annotation using active learning. In *ECIR*, 2008. pages 27

[2] F. R. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *ICML*, 2005. pages 63

[3] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML*, 2004. pages 67

[4] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. *Computer Vision and Image Understanding*, 110, 2008. pages 8, 12

[5] L. Bo and C. Sminchisescu. Efficient match kernels between sets of features for visual recognition. In *nips*, 2009. pages 63

[6] A. Bosch, A. Zisserman, and X. Munoz. Scene classification via pLSA. In *ECCV*, 2006. pages 69

[7] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *ICCV*, 2007. pages 13, 39

[8] A. Bosch, A. Zisserman, and X. Munoz. Scene classification using a hybrid generative/discriminative approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30, 2008. pages 8

[9] S. S. Code. `http://www.cs.cornell.edu/People/tj/svm_light/svm_struct.html`. pages 50

[10] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 2002. pages 64

[11] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 1995. pages 7

[12] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. pages 6, 13, 62, 69

[13] C. Dance, J. Willamowski, G. Csurka, and C. Bray. Categorizing nine visual classes with bags of keypoints, 2004. pages 6, 7, 10, 33, 39, 62

[14] M. I. J. David M. Blei, Andrew Y. Ng. Latent dirichlet allocation. *JMLR*, 2003. pages 7

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. pages xiv, 27

[16] D. A. efficient dense descriptor applied to wide baseline stereo. Engin tola and vincent lepetit and pascal fua. *PAMI*, 5, 2010. pages 8

[17] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html, 2009. pages 27, 31

[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, 2007. pages 31, 47, 51, 63, 68

[19] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9, 2008. pages 62, 67

[20] L. F. Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005. pages 7, 10, 11, 39

[21] P. Felzenswalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008. pages 49

[22] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003. pages 10

[23] T. Finley and T. Joachims. Training structural svms when exact inference is intractable. In *ICML*, 2008. pages 48

[24] Flickr. http://www.flickr.com. pages 3

[25] Y. Freund and R. E. Schapire. Experimetns with a new boosting algorithm. In *ICML*, 1996. pages 7

[26] J. V. Gemert, J.M.Geusebroek, C.J.Veenman, and A.W.M.Smeulders. Kernel codebooks for scene recognition. In *ECCV*, 2008. pages 7, 12

[27] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007. pages 27, 31

[28] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988. pages 11

[29] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, 1999. pages 7

[30] T. Joachims. Training linear SVMSs in linear time. In *KDD.*, 2006. pages 62

[31] T. Joachims and C.-N. J. Yu. Sparse kernel svms via cutting-plane training. *Machine Learning*, 76(2-3):179–193, 2009. pages 48

[32] J.Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *ICCV*, 2005. pages 8

[33] C. Lampert, M. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization with efficient subwindow search. In *CVPR*, 2008. pages 7

[34] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72, 2004. pages 25

[35] D. Larlus and F. Jurie. Latent mixture vocabularies for object categorization. In *BMVC*, 2006. pages 8

[36] S. Lazebnik and M. Raginsky. Supervised learning of quantizer codebooks by infor-mation loss minimization. *PAMI*, 2008. pages 8

[37] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bag of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR.*, 2006. pages 12, 13, 27, 31, 62, 69

[38] T. Leungand and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 43, 2001. pages 10

[39] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. pages x, 6, 11, 12, 62

[40] S. Maji and A. C. Berg. Max-margin additive classifiers for detection. In *ICCV*, 2009. pages 63

[41] S. Maji, A. C. Berg, and J. Malik. Fast-intersection kernel implementation. pages 36

[42] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, 2008. pages 34, 35, 36, 37

[43] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*, 2002. pages 11

[44] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *ICCV*, 2001. pages 11

[45] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 2004. pages 11

[46] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10), 2005. pages 8, 12

[47] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *IJCV*, 2005. pages 11

[48] H. Murase and S. Nayar. Visual learning and recognition of 3d objects from appearance. *IJCV*, 1995. pages 6

[49] K. Murphy, A. Torralba, and W. T. Freeman. Using the forest to see the trees: A graphical model relating features, objects and scenes. In *NIPS*, 2003. pages 8

[50] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42, 2001. pages x, 8, 9, 32

[51] F. perronnin. Universal and adapted vocabularies for generic visual categorization. *PAMI*, 2008. pages 8

[52] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008. pages 12

[53] J. Philbin, M. Marin-Jimenez, S. Srinivasan, A. Zisserman, M. Jain, S. Vempati, P. Sankar, C. Jawahar, and S. Srinivasan. Oxford-IIIT Notebook Paper. In *TRECVID*, 2008. pages 27

[54] Picasa. `http://picasaweb.google.com`. pages 3

[55] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007. pages 4, 63, 64, 65, 67, 70

[56] B. Russell, A. Torralba, K. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 2007. pages xiv, 27

[57] S. Savarese, J. Winn, and A. Criminissi. Discriminative object class models of appearance and shape by correlatons. In *CVPR*, 2006. pages 12

[58] B. Scölkpf. The kernel trick for distances. In *NIPS.*, 2001. pages 65

[59] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient SOlver for SVM. In *ICML*, 2007. pages 62

[60] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. pages 7, 10

[61] A. F. Smeaton, P. Over, and W. Kraaij. High-Level Feature Detection from Video in TRECVid: a 5-Year Retrospective of Achievements. In A. Divakaran, editor, *Multimedia Content Analysis, Theory and Applications*, pages 151–174. Springer Verlag, Berlin, 2009. pages 26, 51

[62] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *PAMI*, 22(12), 2000. pages 1, 26

[63] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *JMLR*, 7, 2006. pages 25

[64] M. J. Swain and D. H. Ballard. Color indexing. *Internation Journal of Computer Vision*, 1991. pages 6

[65] L. W. T. Serre and T. Poggio. Object recognition with features inspired by visual cortex. In *CVPR*, 2005. pages 7

[66] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *CVPR*, 2008. pages 39

[67] A. Torralba and A. Oliva. Statistics of natural image categories. *Network: Computation in Neural Systems.*, 2003. pages 6, 62

[68] TRECVID. `http://trecvid.nist.gov/`. pages 26

[69] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, and Y. Singer. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6, 2005. pages 4, 46, 48

[70] A. Vailaya, M. A. T. Figueiredo, A. K. Jain, and H.-J. Zhang. Image classification for content based indexing. *IEEE Transactions on Image Processing*, 2001. pages 26

[71] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *ICCV.*, 2007. pages 8, 24, 25, 33, 34, 67

[72] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *CVPR*, 2003. pages 10

[73] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009. pages xii, 1, 6, 7, 8, 10, 51, 68, 69

[74] A. Vedaldi and A. Zisserman. Structured output regression for detection with partial occlusion. In *NIPS*, 2009. pages 62

[75] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *CVPR*, 2010. pages xii, 4, 63, 64, 66, 68

[76] S. Vempati, M. Jain, O. Parkhi, C. V. Jawahar, A. Vedaldi, M. Marszalek, and A. Zisserman. Oxford-IIIT Notebook Paper. In *TRECVID*, 2009. pages 27

[77] E. M. Voorhees and D. K. Harman. *TREC: Experiment and Evaluation in Information Retrieval*. Cambridge, USA: The MIT Press, 2005. pages 28

[78] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *nips*, 2001. pages 63

[79] E. Yilmaz and J. A. Aslam. Estimating average precision with incomplete and imperfect judgements. In *CIKM*, 2006. pages 30

[80] Youtube. `http://www.youtube.com`. pages 3

[81] C. N. Yu and T. Joachims. Learning structural svms with latent variables. In *ICML*, 2009. pages 4, 46, 49

[82] C.-N. J. Yu and T. Joachims. Training structural svms with kernels using sampled cuts. In *KDD*, pages 794–802, 2008. pages 48

[83] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *IJCV.*, 2007. pages 62