Chapter 5
Application of SVMs to the Bag-of-Features Model: A Kernel Perspective

Lei Wang, Lingqiao Liu, Luping Zhou, and Kap Luk Chan

Abstract The Bag-of-features model has recently achieved great success in image categorisation and become the state of the art. Support vector machines (SVMs) have played an important role in this process. This chapter first introduces the fundamentals of the Bag-of-features model in image categorisation. Following that, it is focused on how the SVM classifiers are applied to this model. In particular, we show the novel kernels developed to compare images based on a variety of representations incurred by this model. Also, how the kernels are implicitly implemented or effectively approximated to work with linear SVMs is discussed. Through this chapter, we will see that the application of SVMs not only demonstrates its elegance and efficiency but also raises new research issues to stimulate the development of SVMs.

5.1 The Bag-of-Features Model

Image categorisation is one of the fundamental tasks in the field of computer vision. It aims to classify an image to a predefined set of classes according to its visual content. By appropriately defining the classes, image categorisation can be used as an effective tool to determine the presence of objects in an image (object recognition), infer the location of an object in an image (object localisation) or in a

L. Wang (✉) • L. Zhou
University of Wollongong, Wollongong, NSW 2522, Australia
e-mail: leiw@uow.edu.au; lupingz@uow.edu.au

L. Liu
Australian National University, Acton, ACT 0200, Australia
e-mail: lingqiao.liu@cecs.anu.edu.au

K.L. Chan
Nanyang Technological University 639798, Singapore
e-mail: eklchan@ntu.edu.sg

Y. Ma and G. Guo (eds.), *Support Vector Machines Applications*,
DOI 10.1007/978-3-319-02300-7_5,

video sequence (object tracking), classify the scene in an image (scene recognition), determine the type of human pose in an image (pose recognition) or the action in a video clip (action recognition), detect the irregular visual patterns (unusual event detection) or search for similar images or videos from a large database (image/video retrieval), to name just a few.

Image categorisation has been researched for a long time in the fields of computer vision and pattern recognition. From the perspective of visual features, most of the research, particularly for those conducted 10 years ago or earlier, has been focused on the use of global features, for example, the features based on colour, texture or shape in a whole image. Although many significant research progresses have been made along this line, the performance of generic image categorisation is still far from being satisfactory. A powerful image categorisation model that can be generally applied is still lacking.

In the past several years, the Bag-of-features model [8,47] has attracted intensive attention in the field of visual recognition and achieved great success in a wide range of applications. It has become the state-of-the-art image categorisation model that can be generally applied. The Bag-of-features model can be viewed as a wonderful integration of the Bag-of-words model in the field of text analysis [19] and the local invariant features in the field of computer vision [32, 33]. During the last decade, a number of excellent local invariant features have been developed, for example, the well-known Scale-Invariant Feature Transform (SIFT) feature [28]. With the local invariant features, effective, reliable and robust description of visual content within a small-sized image patch can be obtained. This provides a solid basis for the transplantation of the Bag-of-words model from text analysis, giving birth to the Bag-of-features model. The work on texture classification in [25] is among the earliest work that uses the Bag-of-features model in image categorisation. Generally, the work in [8, 47] is often regarded as the beginning of the Bag-of-features model in generic image categorisation.

A basic Bag-of-features model can be described as follows. First, a set of local image patches is sampled from all training images and characterised by using a local feature descriptor. After that, common visual patterns shared by these local feature descriptors are identified. They mimic the "words" in the Bag-of-words model and are called "visual words." A collection of visual words forms a "visual codebook." Following the Bag-of-words model, each image is represented by a histogram indicating the frequency of occurrences of each visual word in this image. In this way, image categorisation can be performed based on the histogram-based representation, for example, by training a classifier and performing classification. This basic Bag-of-features model has been significantly extended since its introduction and more powerful variants are being used. Figure 5.1 shows an image categorisation system based on the basic Bag-of-features model. The last step "Classification" corresponds to the application of SVMs. However, the previous steps form the basis for this application and more importantly, the development of these steps significantly reshapes the application of SVMs in the image categorisation system. To obtain a clear understanding of this system and the application of SVMs, this chapter will give a brief introduction of the four key components of this system in the following parts.

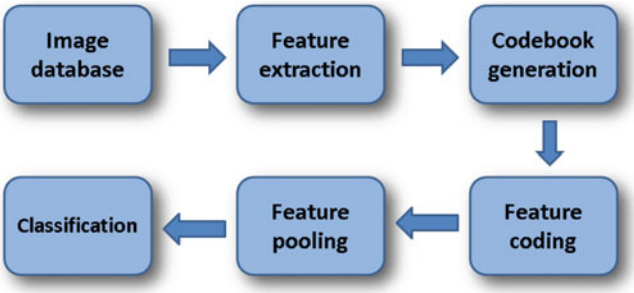

Fig. 5.1 An image categorisation system based on the basic Bag-of-features model

5.1.1 Feature Extraction

As mentioned above, the advances of local invariant features lay the foundation of the birth of the Bag-of-features model. The component "Feature extraction" in Fig. 5.1 is to extract local patches from images and then characterise the visual content therein. This component consists of two steps, feature detection and feature description. Feature detection is to identify the locations in an image where important visual information could exist, which is usually called interest point detection. A number of excellent interest point detection algorithms have been developed. They can reliably and consistently identify interest points in an image even if the image experiences the change of viewpoint, scale or illumination to some extent. The repeatability of identifying the same interest points in the varying conditions is essential to generic image categorisation. A comparative study of the commonly used interest point detectors can be found in [32]. With the recent progress of the Bag-of-features model, it is found that densely sampling local image patches can often lead to better classification performance than performing interest point detection [20]. Dense sampling could extract much more local image patches than interest point detection and thus has the advantage of avoiding missing important visual information that helps classification at the later stage. Dense sampling has now become a common way to extract local patches from images. To deal with the scaling issue, dense sampling with different-sized local patches is often used.

Once local image patches are extracted, a variety of local feature descriptors can be employed. These descriptors are designed to achieve reliable and robust description of the local patches with respect to varying conditions. The best known and the most popular descriptor may be the SIFT descriptor [28]. SIFT actually consists of both feature detector and feature descriptor, but its descriptor is often individually used by researchers to characterise densely sampled local image patches. A systematic evaluation of the performance of existing local descriptors is conducted in [33]. Depending on the size of an image, the total number of local patches extracted from an image can be in the order of thousands or even

tens of thousands. Each of them will be represented by a descriptor, which is a multi-dimensional vector. Note that in this way, each image becomes a bag of orderless feature vectors or "a set of points" in a multi-dimensional vector space.

5.1.2 Visual Codebook

The Bag-of-features model originates from the Bag-of-words model. However, images do not contain words by nature. In order to use the Bag-of-words model, the concept of "visual word" is developed. Visual words can be regarded as a set of common visual patterns shared by the local patches extracted from images. For example, the patterns could be corners, T-junctions, L-junctions or any other frequently observed patterns on the changes of pixel intensities. The k-means clustering may be the most commonly used method to generate visual words. Let $\{\mathbf{x}_1, \mathbf{x}_1, \cdots, \mathbf{x}_n\}$ denote a set of local feature descriptors obtained from the step of feature extraction, where $\mathbf{x} \in \mathbb{R}^d$ is a d-dimensional vector. For example, d is 128 when the SIFT descriptor is used. The k-means clustering aims to find an optimal k-cluster-partitioning $\{C_1, C_2, \cdots, C_k\}$ of these feature descriptors by minimising the sum of the within-cluster variances. Let μ_i denote the mean of the cluster C_i. This partitioning can be shown as an optimisation problem

$$\{C_1, C_2, \cdots, C_k\} = \arg\min \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \|\mathbf{x}_j - \mu_i\|_2^2. \tag{5.1}$$

By solving this problem, the optimal means $\{\mu_1, \mu_2, \cdots, \mu_k\}$ are interpreted as visual words. A collection of the k visual words forms a visual codebook. Throughout this chapter, $\mathbf{V}$ is used to denote a visual codebook and $|\mathbf{V}|$, whose value is k here, denotes the number of visual words in the codebook.

In addition to the k-means clustering algorithm, a variety of more advanced visual codebook generation algorithms have been developed in the past few years. They generally deal with one or more of the following issues related to k-means clustering: (1) How to set the optimal k? To address this issue, methods based on multiple codebook combination, codeword selection and codeword merging have been proposed [53,55]; (2) How to conduct efficient clustering when d or k is large? To speed up clustering in this case, special data structures have been used and this leads to the work of vocabulary tree [36], randomised clustering forests [34] and fast k-means [41]; (3) Can a partitioning better than that given by the k-means clustering be obtained? In this line, fixed-radius partition and mean-shift techniques have been utilised [20]. Also, instead of using Euclidean distance, clustering with other distances has been developed to better handle the histogram structure of SIFT descriptors [56]; (4) Can supervised information be incorporated to obtain better codebooks? When the class label of each image is available, it can be used to design compact and discriminative visual codebooks. To achieve this, information-theoretic method [23] and supervised compact codebook generation [27] have recently been developed.

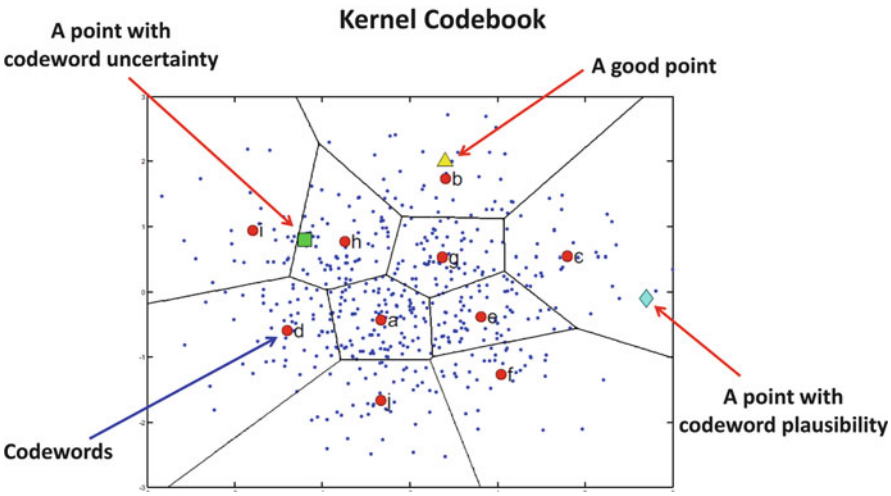

Fig. 5.2 The issues of codeword uncertainty and codeword plausibility. Image courtesy of [49]

5.1.3 Feature Coding

Once a visual codebook is created, it will be used as a basis to represent the visual content of an image. To associate the visual content with the codebook, the most common way is to assign each of the local feature descriptors extracted from an image to one of the visual words. For example, this can be done by evaluating the Euclidean distance between a local descriptor to each visual word and assigning it to the closest word. This is often known as "hard assignment," in which a local descriptor is assigned to one and only one visual word. Although this assignment is conceptually simple and computationally efficient, more advanced assignment methods have been developed in the last few years. Generally, these new methods deal with two issues: (1) how to reduce the quantisation error in the hard assignment in further? (2) how to consider the underlying manifold structure of the local feature descriptors? To resolve the first issue, a number of methods have been developed in the literature, among which kernel codebook and sparse coding are two representative methods. Kernel codebook is a "soft assignment" method, which assigns a local feature descriptor to more than one visual words to reduce quantisation error [49]. In particular, kernel codebook systematically discusses two main drawbacks, codeword uncertainty and codeword plausibility, in hard-assignment methods, as illustrated in Fig. 5.2.

Codeword uncertainty means that hard-assignment methods rigidly assign a local descriptor to one and only one visual word even if it is relevant (close) to multiple different words. For example, for a local descriptor riding on the boundary of two clusters, assigning it to either one of the two corresponding visual words could cause significant quantisation error. Codeword plausibility means that hard-assignment

methods rigidly assign a local descriptor to a visual word even if this descriptor is far from all of the words. Again, this could lead to large quantisation error. To handle the two issues, the kernel codebook proposes to model the degree of relevance between a local descriptor and each visual word. Its assignment scheme can be expressed as

$$\omega_i = \frac{\kappa_\sigma(d(\mathbf{x}, \mathbf{v}_i))}{\sum_{i=1}^{|\mathbf{V}|} \kappa_\sigma(d(\mathbf{x}, \mathbf{v}_i))}, \tag{5.2}$$

where κ denotes a kernel used in kernel density estimation with the width of σ, $\mathbf{x}$ a local descriptor, $\mathbf{v}_i$ the ith visual word, $d(\mathbf{x}, \mathbf{v}_i)$ the distance between $\mathbf{x}$ and $\mathbf{v}_i$, and ω_i the coefficient assigned to $\mathbf{x}$ with respect to $\mathbf{v}_i$. As shown in [49], this soft-assignment method can well resolve the above two issues caused by hard assignment.

Sparse coding is the assignment method that represents the state of the art [57]. It is also a soft-assignment method and has an elegant theoretical framework. The sparsity enforces that only a small number of visual words can be chosen to represent a local descriptor. In addition, sparse coding not only learns the coding coefficient but can also jointly learn the visual codebook. Let $\mathbf{V}_{d\times k} = (\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_k)$ denote a visual codebook consisting of k words. Let $\mathbf{U}_{k\times n} = (\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n)$ denote the coding coefficient matrix, in which $\mathbf{u}_i$ is the coding coefficient for the ith local descriptor $\mathbf{x}_i$. Sparse coding can be expressed as an optimisation problem that aims to minimise the reconstruction error, subject to a sparsity constraint on the coding coefficient.

$$\begin{aligned} \{\mathbf{V}^\star, \mathbf{U}^\star\} &= \arg\min_{\mathbf{V},\mathbf{U}} \sum_{i=1}^{n} \left(\|\mathbf{x}_i - \mathbf{V}\mathbf{u}_i\|_2^2 + \lambda \|\mathbf{u}_i\|_1 \right) \\ &s.t. \quad \|\mathbf{v}_j\| \le 1, \quad j = 1, 2, \cdots, k, \end{aligned} \tag{5.3}$$

where $\|\cdot\|_1$ is the ℓ_1 norm used to impose the sparsity constraint and λ is the regularisation parameter. This optimisation can be efficiently solved in an alternate manner. By fixing $\mathbf{V}$, the coding coefficient $\mathbf{u}_i$ for each local descriptor can be updated one by one by solving an ℓ_1-regularised least-squares problem. By fixing $\mathbf{U}$, the visual codebook $\mathbf{V}$ can be updated by solving a least-squares problem with quadratic constraints.

Locality-constrained Linear Coding (LLC) is another important sparse coding method, which takes the underlying manifold structure of local descriptors into account [52]. It argues that locality is more essential than sparsity and that locality induces sparsity. To enforce locality, LLC encodes a descriptor by the visual words nearby. This not only induces the sparsity but also makes similar descriptors tend to share similar coefficients. LLC solves the following optimisation problem,

$$\{\mathbf{V}^\star, \mathbf{U}^\star\} = \arg\min_{\mathbf{V},\mathbf{U}} \sum_{i=1}^{n} \left(\|\mathbf{x}_i - \mathbf{V}\mathbf{u}_i\|_2^2 + \lambda \|\mathbf{d}_i \otimes \mathbf{u}_i\|_2^2 \right)$$

$$s.t.\ \ \mathbf{u}_i^\top \mathbf{1} = 1,\ \ \|\mathbf{v}_j\| \leq 1,\ \ j = 1, 2, \cdots, k, \tag{5.4}$$

where $\mathbf{d}_i$ is a column vector indicating the distance of the descriptor $\mathbf{x}_i$ from each visual word and $\otimes$ denotes a component-wise multiplication. This optimisation problem can also be solved in an alternate manner.

Sharing the spirit of locality in LLC, a computationally more efficient coding method called Localised Soft-assignment Coding (LSC) has recently be developed [26]. LSC proposes to integrate the concept of locality into kernel codebook, by arguing that shorter Euclidean distances are more reliable in the presence of data manifold. LSC extends the kernel codebook to the following form,

$$\omega_i = \frac{\exp(-\beta d(\mathbf{x}, \mathbf{v}_i))}{\sum_{i=1}^{|\mathbf{V}|} \exp(-\beta d(\mathbf{x}, \mathbf{v}_i))},\ \text{where}\ d(\mathbf{x}, \mathbf{v}_i) = \begin{cases} \|\mathbf{x} - \mathbf{v}_i\|_2^2; & \text{if } \mathbf{v}_i \in \mathscr{N}(\mathbf{x}) \\ +\infty; & \text{otherwise.} \end{cases} \tag{5.5}$$

where $\mathscr{N}(\mathbf{x})$ denotes the local neighborhood of $\mathbf{x}$. LSC can achieve comparable coding performance as LLC but incurs much less computational cost.

The advent of sparse and localised coding schemes makes a significant change of the face of the application of SVMs to image categorisation. Together with the feature pooling scheme to be introduced, these coding schemes considerably improve the classification performance of linear SVM classifiers in image categorisation.

5.1.4 Feature Pooling

In order to obtain an image-level representation, the coding coefficients of the local feature descriptors extracted from an image need to be summarised. This process is often called "Pooling." Two ways are usually used, including sum-pooling and max-pooling. Sum-pooling simply adds all the coefficients for each visual word up. Let $\mathbf{z} \in \mathbb{R}^k$ denote the image-level representation for an image I. Sum-pooling can be expressed as

$$\mathbf{z} = \sum_{\mathbf{u}_i \in I} \mathbf{u}_i. \tag{5.6}$$

If $\mathbf{z}$ is set as the mean of the $\mathbf{u}_i$'s, it will be called average pooling. Max-pooling takes the maximum coefficient with respect to each visual word

$$\mathbf{z} = \max_{\mathbf{u}_i \in I} \mathbf{u}_i, \tag{5.7}$$

where the max operation is performed in a dimension-wise manner. The max-pooling can magically make linear SVM classifiers work as well as the nonlinear ones, leading to efficient image categorisation. Formal theoretical analysis has been

attempted to understand why the max-pooling is superior to the sum-pooling [6,26]. More explanation on this magic in this regard will be given in the later part of this chapter from a kernel perspective.

In sum, this section introduces a basic image categorisation system using the Bag-of-features model. In particular, the four key components of this system have been discussed. This paves the way for us to gain a better understanding of the application of SVMs to image categorisation.

5.2 Application of SVMs with Histogram-Based Nonlinear Kernels

When an image-level representation is obtained for each image, an SVM classifier can be trained and used to categorise new images. Since image classes in a categorisation task are usually not linear separable, kernel-based SVM classifiers are generally used in order to obtain good classification performance, especially at the early stage of image categorisation with the Bag-of-features model. In this case the kernel function plays a pivotal role, and therefore identifying and designing appropriate kernel functions has attracted much attention at that stage. Since the histogram-based representation is widely used in that period, the kernel functions that can effectively evaluate the similarity of histograms have been researched and employed.

In the Bag-of-features model, a histogram indicates the frequency of the occurrences of each visual word in an image. It is an efficient approximation to the distribution of different visual patterns in an image. In the literature, a number of measures have been proposed to evaluate the similarity or dissimilarity between histograms. In the work of colour indexing [48], histogram intersection is proposed for object recognition. The work in [43] discusses the histogram dissimilarity measures and applies them to image retrieval. It groups the measures into bin-to-bin measures and cross-bin measures. The former includes Minkowski-form distance, Histogram intersection, Kullback-Leibler divergence, χ^2-statistics, while the latter includes Quadratic-form distance, cumulative histogram distance, and the distance based on distribution parameters. In [7], the SVMs are applied to classify generic images based on colour histograms. That work provides insightful discussion on what kind of kernel functions shall be used to evaluate the similarity of colour histograms. It shows that Non-Gaussian Radial Basis Function (RBF) kernels can achieve better classification performance than the commonly used Gaussian RBF kernels.

Although the SVM classifiers using the above histogram-based kernel functions can produce promising classification performance, training and testing nonlinear SVM classifiers incur more computational load, and this becomes a significant issue for the applications that require real-time classification. In recent years, approaches with the linear kernel have been proposed to achieve the advantage brought by

histogram-based nonlinear kernels. The research in this regard generally follows two lines. The first line is to approximately identify the explicit feature mapping induced by the nonlinear kernels and then a linear SVM can be straightforwardly applied. The other line is to derive a new image representation such that a linear SVM classifier with this new representation can work as well as a nonlinear one. In doing so, these approaches not only well maintain the original classification performance but also considerably decrease the computational load in both training and test stages. In the following parts, the chapter will discuss typical histogram-based kernel functions and the approaches to approximating them via the linear kernel.

5.2.1 Histogram-Based Nonlinear Kernels

This part is focused on three kernel functions that are commonly used to evaluate the similarity of histograms, including the Histogram Intersection Kernel (HIK), the non-Gaussian RBF kernel and the χ^2-RBF kernel.

Recall that $\mathbf{z}$ denotes the representation of an image. Let $\phi(\cdot)$ be the mapping function implicitly induced by a kernel function $\kappa(\cdot,\cdot)$. Let $\mathbf{w}$ and b be the normal and bias of the SVM separating hyperplane. A nonlinear SVM classifier can be expressed as

$$f(\mathbf{z}) = \mathbf{w}^\top \phi(\mathbf{z}) + b = \sum_i \alpha_i y_i \kappa(\mathbf{z}, \mathbf{z}_i) + b, \tag{5.8}$$

where α_i and y_i are the coefficient and the class label for the ith training sample $\mathbf{z}_i$.

5.2.1.1 Histogram Intersection Kernel

Originally proposed in [48] to compare a given image histogram to a pre-defined model histogram for object recognition, histogram intersection has seen an important application to image categorisation with the Bag-of-features model. Let $\mathbf{z}_i$ and $\mathbf{z}_j$ denote the histograms corresponding to images i and j. Intersection of the two histograms is defined as $H(\mathbf{z}_i, \mathbf{z}_j) = \sum_{l=1}^{|\mathbf{V}|} \min(\mathbf{z}_{il}, \mathbf{z}_{jl})$, where $\mathbf{z}_{il}$ is the l-th bin of $\mathbf{z}_i$. A normalised intersection with respect to $\mathbf{z}_j$ can be defined as $H(\mathbf{z}_i, \mathbf{z}_j) = \sum_{l=1}^{|\mathbf{V}|} \min(\mathbf{z}_{il}, \mathbf{z}_{jl}) / \sum_{l=1}^{|\mathbf{V}|} \mathbf{z}_{jl}$. Usually, it is assumed that the sum of the bins in the two histograms is same. In this case, the HIK can be expressed as

$$\kappa(\mathbf{z}_i, \mathbf{z}_j) = \sum_{l=1}^{|\mathbf{V}|} \min(\mathbf{z}_{il}, \mathbf{z}_{jl}). \tag{5.9}$$

The HIK function defined in this way can be written as an inner product in a feature space and therefore it is a Mercer kernel [1]. This makes it suitable for SVM

classifiers which usually need this property to achieve global optimum. Also, as indicated in [48], for two histograms $\mathbf{z}_i$ and $\mathbf{z}_j$ with $\sum_{l=1}^{|\mathbf{V}|} \mathbf{z}_{il} = \sum_{l=1}^{|\mathbf{V}|} \mathbf{z}_{jl} = T$, the HIK has an essential connection with the ℓ_1-norm distance between them. That is,

$$\kappa(\mathbf{z}_i, \mathbf{z}_j) = 1 - \frac{1}{2T} \|\mathbf{z}_i - \mathbf{z}_j\|_1. \tag{5.10}$$

This connection can help understanding the effectiveness of HIK in image categorisation with the Bag-of-features model. For a class of images containing the same object, the size of the area occupied by the object could change, leading to variation on the value of the bins for the visual words associated with the object. Also, the size of background could be different across these images and this will also cause variation on the corresponding bins. However, these variations are not essential and they do not change the object class to which the images belong to. In this case, an ℓ_1-norm distance becomes a better choice because it changes linearly with the variation while a commonly used ℓ_2-norm distance changes quadratically. It is known that the ℓ_2-norm distance corresponds to a linear kernel in the input space. This may partially explain why directly applying a linear SVM classifier to histogram representation often shows inferior classification performance. In fact, the values of the bins of a histogram are not important. Instead, whether a bin is empty or not (indicating whether the corresponding visual word appears in an image or not) matters. This case has been observed in image classification with colour histograms in [7] and the Bag-of-features model in [6].

5.2.1.2 Non-Gaussian RBF Kernel

To control the sensitivity of an RBF kernel function to the difference between two histograms, the work in [7] proposes a set of non-Gaussain RBF kernels in the following form

$$\kappa(\mathbf{z}_i, \mathbf{z}_j) = \exp\left(-\beta \sum_{l=1}^{|\mathbf{V}|} |\mathbf{z}_{il}^a - \mathbf{z}_{jl}^a|^b\right). \tag{5.11}$$

It is easy to see that when $a = 1$ and $b = 2$, the above kernel reduces to a Gaussian RBF kernel. When $a = 1$ and $b = 1$, it gives rise to a Laplacian RBF kernel

$$\kappa(\mathbf{z}_i, \mathbf{z}_j) = \exp\left(-\beta \sum_{l=1}^{|\mathbf{V}|} |\mathbf{z}_{il} - \mathbf{z}_{jl}|\right). \tag{5.12}$$

Assuming that $\mathbf{z}_j$ is obtained by perturbing one empty bin of $\mathbf{z}_i$ by Δh, the non-Gaussian kernel value between them can be written as

$$\kappa(\mathbf{z}_i, \mathbf{z}_j) = \exp\left(-\beta (\Delta h)^{ab}\right). \tag{5.13}$$

Adjusting ab can effectively change the decaying rate of the kernel value with respect to the perturbation. As can be seen, the commonly used Gaussian RBF kernel incurs a quadratic exponential decaying rate while the Laplacian RBF kernel has a linear exponential decaying rate. Experimental study is conducted in [7] to compare different settings of a and b for colour histogram-based image classification. It is found that decreasing the value of a and b can effectively improve the classification performance of SVMs on colour histograms. The best performance is obtained on the Corel image data set when $a = 0.25$ and $b = 1$, and it is significantly better than the performance obtained by a Gaussian RBF kernel. Note that the change of a does not affect the non-Gaussian RBF kernel to be a valid Mercer kernel because we can simply view $\mathbf{z}_{il}^a$ and $\mathbf{z}_{jl}^a$ as the input data. At the same time, b has to be confined between 0 and 2 to make the kernel meet the Mercer's condition. The application of SVMs with the Laplacian RBF kernel has also achieved promising classification performance in image categorisation with the Bag-of-features model.

5.2.1.3 χ^2-Radial Basis Function Kernel

The χ^2-RBF kernel can be traced back to the χ^2-test in mathematical statistics used to compare two distributions. In [44], χ^2 is used to evaluate the dissimilarity between two histograms as

$$\chi^2(\mathbf{z}_i, \mathbf{z}_j) = \sum_{l=1}^{|\mathbf{V}|} \frac{(\mathbf{z}_{il} - \mathbf{z}_{jl})^2}{\mathbf{z}_{il} + \mathbf{z}_{jl}}. \tag{5.14}$$

Based on this measure, the work in [7] defines the χ^2-RBF kernel as

$$\kappa(\mathbf{z}_i, \mathbf{z}_j) = \exp\left(-\beta \sum_{l=1}^{|\mathbf{V}|} \frac{(\mathbf{z}_{il} - \mathbf{z}_{jl})^2}{\mathbf{z}_{il} + \mathbf{z}_{jl}}\right). \tag{5.15}$$

It is not difficult to see that when $\mathbf{z}_j$ is obtained by perturbing one empty bin of $\mathbf{z}_j$ by Δh, the kernel value is $\kappa(\mathbf{z}_i, \mathbf{z}_j) = \exp(-\beta \Delta h)$, which also has a linear exponential decaying rate with respect to Δh. The χ^2-RBF kernel has been proved to be a Mercer kernel in [13]. This kernel has been experimentally compared with other kernels in [58] for image categorisation with an SVM classifier. It is found that the χ^2-RBF kernel can achieve higher classification performance than linear kernel, quadratic kernel and the Gaussian RBF kernel.

In addition to the above kernels, there is a set of special kernels used by the SVM classifiers for image categorisation, which is called "additive kernel" [50]. An additive kernel can be written as a sum of the kernels computed based on each individual dimension of data. This feature makes it be able to work directly with linear SVM classifiers after appropriate manipulation. The additive kernels will be introduced in the later parts of this chapter.

5.2.2 Approximation to Histogram-Based Nonlinear Kernels

One drawback of directly applying the nonlinear kernels introduced in the previous section is the poor scalability for large-scale data sets. However, the application of SVMs calls for highly efficient image classification algorithms in order to handle large-scale tasks.

Nowadays, it is quite often to encounter an image categorisation problem with over tens of thousands of training samples. For example, the commonly used image classification benchmarks, PASCAL [11] and Caltech 101/256 [12, 15], contain around 10,000 images and the more recently developed large-scale benchmarks such as ILSVRC [10] even have millions of images. However, nonlinear SVMs become computationally expensive when training sample size is large. In fact, merely computing the kernel matrix in a nonlinear SVM classifier will take $\mathcal{O}(n^2 d)$ calculations, where n is the number of training samples and d is the dimensionality of image representation. For the state-of-the-art image representation, e.g. Bag-of-features model with a large-sized codebook and spatial pyramid [24], the value of d can be as large as hundreds of thousands. When n is also large, nonlinear SVMs will easily become computationally intractable. In addition, the computational cost of nonlinear SVMs is high in the test stage. The cost of evaluating the decision score is $\mathcal{O}(dn_{sv})$, where n_{sv} is the number of support vectors, which can be very large, for example, a few thousands in practice.

Comparing with the nonlinear SVMs, linear SVMs enjoy much higher computational efficiency. First of all, there exist very efficient training algorithms for linear SVM classifiers [16, 45]. Second, in the test stage, the cost of evaluating the decision score is just $\mathcal{O}(d)$, which can be thousands of times less than that incurred by the nonlinear SVMs. However, for the histogram-based representation in the Bag-of-features model, linear SVMs usually yield poorer performance than its nonlinear counterpart. To achieve both high computational efficiency and excellent classification performance, the recent literature has leveraged the kernel-induced feature mapping to transform a nonlinear SVM classifier to a linear one.

Recall that a kernel function κ can be written as the inner product of the feature mappings $\phi(\cdot)$, that is:

$$\kappa(\mathbf{z}_i, \mathbf{z}_j) = \langle \phi(\mathbf{z}_i), \phi(\mathbf{z}_j) \rangle. \tag{5.16}$$

If $\phi(\cdot)$ can be explicitly obtained, we can simply transform the input data by this mapping and apply a linear SVM on the mapped data. In doing so, a nonlinear SVM can be attained by solving a linear SVM. Unfortunately, the mapping function $\phi(\cdot)$ is generally implicit and may even have infinite dimensions. However, is it possible to develop a sufficiently good approximation to the feature mapping $\phi(\cdot)$? The answer is affirmative. Before systematically introducing the state-of-the-art approximation methods, this section first presents a method that is initially proposed to approximate the commonly used HIK.

5.2.2.1 An Approximation to Histogram Intersection Kernel

Recall that HIK is defined as $\kappa(\mathbf{z}_i, \mathbf{z}_j) = \sum_{l=1}^{d} \min(\mathbf{z}_{il}, \mathbf{z}_{jl})$, where d denotes the dimensionality of a histogram. This kernel is the sum of the values obtained by the nonlinear function $\min(\mathbf{z}_{il}, \mathbf{z}_{jl})$ in each dimension. Note that the function $\min(\cdot,\cdot)$ only takes two scalars as the input.

The work in [31] proposes to approximate the HIK as follows. It first develops a quantised version of the original data. More specifically, that work uniformly quantises the value of each bin into s scales. $q(\cdot)$ denotes the quantisation function and it returns the scale into which the input scalar is quantised. $\mathbf{u}$ is a function that maps the quantised scale $q(z)$ $(q(z) \in \{1, 2, \cdots, s\})$ into an s-dimensional vector. The definition of $\mathbf{u}$ is as follows

$$\begin{aligned} &\mathbf{u}(z) = (u_1,\ u_2,\ \cdots u_k,\ \cdots u_s)^\top, \ \text{where} \\ &u_k = \begin{cases} 1; \text{ if } k \leq q(z) \\ 0; \text{ otherwise.} \end{cases} \end{aligned} \tag{5.17}$$

where u_k denotes the kth dimension of $\mathbf{u}(z)$. Then the function $\min(\cdot,\cdot)$ can be approximated by

$$\min(\mathbf{z}_{il}, \mathbf{z}_{jl}) \approx \alpha \langle \mathbf{u}(\mathbf{z}_{il}), \mathbf{u}(\mathbf{z}_{jl}) \rangle, \tag{5.18}$$

where α is a constant scalar. From the definition of $\mathbf{u}$, we can see that if z is quantised into the $q(z)$th level, the first $q(z)$ dimensions of $\mathbf{u}(z)$ will be "1" and the remaining dimensions will be "0". Thus the inner product of $\mathbf{u}(\mathbf{z}_{il})$ and $\mathbf{u}(\mathbf{z}_{jl})$ will equal $\min(q(\mathbf{z}_{il}), q(\mathbf{z}_{jl}))$.

Note that the function $\mathbf{u}(\cdot)$ essentially develops an explicit feature mapping for the nonlinear function $\min(\cdot,\cdot)$. Since HIK is calculated by simply summing all $\min(\mathbf{z}_{il}, \mathbf{z}_{jl})$ $(l = 1, \cdots, d)$ up, we can define the approximate explicit feature mapping of HIK by concatenating the mapping $\mathbf{u}(\cdot)$ at each dimension of $\mathbf{z}_i$, that is

$$\phi(\mathbf{z}_i) = (\mathbf{u}^\top(\mathbf{z}_{i1}) \ \cdots \ \mathbf{u}^\top(\mathbf{z}_{ik}) \ \cdots \ \mathbf{u}^\top(\mathbf{z}_{id}))^\top. \tag{5.19}$$

From this method, we can see that if a nonlinear kernel can be decomposed into the sum of a set of dimension-wise nonlinear functions with only two scalars as the input, it will be convenient to find an approximate feature mapping for such a nonlinear function. In fact, this kind of kernel is called additive kernel and it has been shown in the recent literature that for this family of kernels, efficient approximate feature mappings can be developed with very good approximation accuracy.

5.2.2.2 Additive Kernel and Its Approximation

An additive kernel can be written as the sum of the kernels computed on each individual dimension of data, that is, $\kappa(\mathbf{z}_i, \mathbf{z}_j) = \sum_{l=1}^{d} k(\mathbf{z}_{il}, \mathbf{z}_{jl})$. We call the function

Table 5.1 Examples of additive kernels and their dimension-wise kernel functions

Linear kernel	$k(\mathbf{z}_{il}, \mathbf{z}_{jl}) = \mathbf{z}_{il}\mathbf{z}_{jl}$
Hellinger's kernel	$k(\mathbf{z}_{il}, \mathbf{z}_{jl}) = \sqrt{\mathbf{z}_{il}\mathbf{z}_{jl}}$
Histogram intersection kernel	$k(\mathbf{z}_{il}, \mathbf{z}_{jl}) = \min(\mathbf{z}_{il}, \mathbf{z}_{jl})$
χ^2 kernel	$k(\mathbf{z}_{il}, \mathbf{z}_{jl}) = 2(\mathbf{z}_{il}\mathbf{z}_{jl})/(\mathbf{z}_{il} + \mathbf{z}_{jl})$

$k(\cdot,\cdot)$ the dimension-wise kernel function (DKF in short) and an additive kernel can be fully determined by its DKF. Note that many commonly used kernels in image classification are additive kernels. Examples of additive kernels and their DKFs are listed in Table 5.1.

The special structure of an additive kernel suggests that its approximate feature mapping can be derived by first finding the approximate mapping of its DKF and concatenating the mappings from all the dimensions. Formally, the lth DKF can be obtained as: $k_l(\mathbf{z}_{il}, \mathbf{z}_{jl}) \approx \langle \phi_l(\mathbf{z}_{il}), \phi_l(\mathbf{z}_{jl}) \rangle$, where $\phi_l(\cdot)$ is the approximate feature mapping for the lth dimension of $\mathbf{z}$. Note that although DKF usually takes the same form in each dimension, its approximate feature mapping $\phi_l(\cdot)$ may be different from dimension to dimension. This is because the feature distribution in each dimension can be different and to capture these differences we may need different approximate feature mappings. Once the mapping $\phi_l(\cdot)$ is obtained for each dimension, an additive kernel can be approximated by $\kappa(\mathbf{z}_i, \mathbf{z}_j) \approx \langle \phi(\mathbf{z}_i), \phi(\mathbf{z}_j) \rangle$, where $\phi(\mathbf{z}_i) = (\phi_1^\top(\mathbf{z}_{i1})\ \phi_2^\top(\mathbf{z}_{i2})\ \cdots\ \phi_d^\top(\mathbf{z}_{id}))^\top$.

In the following parts, we elaborate two representative methods to develop the approximate explicit feature mapping for the DKF: (1) additive kernel principal component analysis (PCA) and (2) homogeneous kernel map.

5.2.2.3 Kernel PCA Approximation to Additive Kernels

The additive kernel PCA method is derived from the classic kernel PCA approximation to nonlinear kernels [54]. The classic kernel PCA approximation firstly calculates the feature mapping on a finite number of samples and then uses Nyström approximation [54] to generalise this mapping to unseen data. In the following parts, we firstly introduce the classic kernel PCA approximation and then discuss its additive kernel extension.

1. Kernel PCA Approximation to Nonlinear Kernels

The quality of an approximate explicit feature mapping can be measured by the incurred approximation error. Formally, this error can be defined as follows:

$$E(\phi) = \int \left(\kappa(\mathbf{z}, \mathbf{z}') - \langle \phi(\mathbf{z}), \phi(\mathbf{z}') \rangle\right)^2 p(\mathbf{z}) p(\mathbf{z}') d\mathbf{z} d\mathbf{z}' \tag{5.20}$$

where $\phi(\mathbf{z})$ is the approximate mapping function which maps a d-dimensional input data to a $\tilde{d}$-dimensional vector. $p(\mathbf{z})$ is the probability density function and it is unknown in general. To make this error term tractable, we can estimate $p(\mathbf{z})$ via non-parametric density estimation method. In specific, $p(\mathbf{z})$ is estimated by using a finite number, m, of samples such that

$$p(\mathbf{z}) = \frac{1}{m}\sum_{i=1}^{m}\delta(\|\mathbf{z}-\mathbf{z}_i\|), \tag{5.21}$$

where the function $\delta(a) = 1$ when $a = 0$ and 0 otherwise. Substituting Eq. (5.21) into Eq. (5.20), we can turn the integral into the summation and it results in the following error term:

$$E(\phi) = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\left(\kappa(\mathbf{z}_i,\mathbf{z}_j) - \langle\phi(\mathbf{z}_i),\phi(\mathbf{z}_j)\rangle\right)^2. \tag{5.22}$$

As shown in [54], the approximate feature mapping for all m samples can be derived in a closed form. Let $\psi_l = (\phi_l(\mathbf{z}_1),\phi_l(\mathbf{z}_2),\cdots,\phi_l(\mathbf{z}_m))^\top$ denote a vector consisting of the lth component of $\phi(\mathbf{z}_i)$, where $i = 1,\cdots,m$. To avoid the redundancy in the approximation, constraints that $\langle\psi_i,\psi_j\rangle = 0$ and $\langle\psi_i,\psi_i\rangle = 1$ for $1 \le i < j \le m$ are imposed. As a result, the solution of minimising the error in Eq. (5.22) can be obtained by solving an eigenvalue problem, which is equivalent to the kernel PCA:

$$\mathbf{K}\psi_l = \lambda_l\psi_l, \tag{5.23}$$

where $\mathbf{K}$ denotes the kernel matrix computed on $\mathbf{z}_1,\cdots,\mathbf{z}_m$. To obtain a $\tilde{d}$-dimensional approximate feature mapping $\phi(\mathbf{z})$ for $\mathbf{z}_1,\cdots,\mathbf{z}_m$, the $\tilde{d}$ eigenvectors of $\mathbf{K}$ corresponding to the largest eigenvalues λ_l are used. For unseen samples, their mappings can be worked out via Nyström approximation:

$$\phi_l(\mathbf{z}) = \frac{\langle\kappa(\mathbf{z},:),\psi_l\rangle}{\lambda_l}, \tag{5.24}$$

where $l = 1,2,\cdots,\tilde{d}$ and $\kappa(\mathbf{z},:) = (\kappa(\mathbf{z},\mathbf{z}_1)\ \kappa(\mathbf{z},\mathbf{z}_2)\ \cdots\ \kappa(\mathbf{z},\mathbf{z}_m))^\top$.

One problem with this approximation is its computational cost. The complexity of calculating $\kappa(\mathbf{z},:)$ is $\mathcal{O}(md)$ and the complexity of calculating $\phi_l(\mathbf{z})$ for the whole $\tilde{d}$-dimensional mapping is $\mathcal{O}(m\tilde{d})$. In total, the complexity is $\mathcal{O}(m(d+\tilde{d}))$. Since the kernel function $\kappa(\cdot,\cdot)$ takes two high-dimensional vectors as input, the mapping from these two vectors to the kernel value can be complex. In this case, a large number of samples are usually needed to achieve reasonably good approximation, making m a number at the order of thousands. Consequently, the calculation of this approximate feature mapping can be very time-consuming in practice.

2. Kernel PCA Approximation to Additive Kernels

Fortunately, for additive kernels the above method can be modified to achieve much lower computational cost [40]. More specifically, we can build the dimension-wise approximation through Eq. (5.24) for each DKF of an additive kernel. The work in [40] adopts this idea and builds the approximate feature mapping by the following algorithm:

1. For each dimension l, compute the corresponding $m \times m$ kernel matrix $\mathbf{K}_l$. The (i,j)th entry of $\mathbf{K}_l$ is calculated by $\mathbf{K}_l(i,j) = k(\mathbf{z}_{il}, \mathbf{z}_{jl})$, where $k(\cdot,\cdot)$ is the DKF of the additive kernel.
2. For each dimension l, compute the $\tilde{d}_l$ (e.g. $\tilde{d}_l = 10$) largest eigenvalues of $\mathbf{K}_l$ $\{\lambda_{l,1}, \cdots, \lambda_{l,\tilde{d}_l}\}$ and their associated eigenvectors. In total, this step will generate $d \times \tilde{d}_l$ eigenvalues for all the d DKFs.
3. Sort the $d \times \tilde{d}_l$ eigenvalues and keep the $\tilde{d}$ largest ones. They are re-numbered as $\{\lambda_{11}, \lambda_{12}, \cdots, \lambda_{1n_1}; \lambda_{21}, \cdots, \lambda_{2n_2}; \cdots; \lambda_{d1}, \cdots, \lambda_{dn_d},\}$, where $\sum_{i=1}^{d} n_i = \tilde{d}$. Let $\{\psi_{11}, \psi_{12}, \cdots, \psi_{1n_1}; \psi_{21}, \cdots, \psi_{2n_2}; \cdots; \psi_{d1}, \cdots, \psi_{dn_d},\}$ be the associated eigenvectors. For a test sample $\mathbf{z}_t$, its feature mapping is obtained by concatenating the mapping for each DKF. For a given l, this mapping is defined as

$$\phi_{li}(\mathbf{z}_{tl}) = \frac{\langle k(\mathbf{z}_{tl}, :), \psi_{li} \rangle}{\lambda_{li}}, \tag{5.25}$$

where $i = 1, \cdots, n_l$ and $k(\mathbf{z}_{tl}, :) = (k(\mathbf{z}_{tl}, \mathbf{z}_{1l}), \cdots, k(\mathbf{z}_{tl}, \mathbf{z}_{ml}))^\top$. Note that in this method, the dimensions of the approximated feature mapping $n_1, \cdots, n_d$ can be different. As argued in [40], this scheme can be more adaptive to the distribution of each dimension of the input data.

The computational complexity of mapping a sample with the obtained feature mapping $\phi(\cdot)$ is still $\mathcal{O}(m(d + \tilde{d}))$. However, since the approximation is now for a much simpler kernel function (a DKF with two scalar inputs only), a much smaller number of samples and mapping dimensions are usually sufficient to attain a good approximation. In [40], it is reported that 128 samples and the same number of mapping dimensions have been sufficient, which is in contrast to the requirement of thousands of samples in the classic kernel PCA. Moreover, one can further leverage the quantisation trick to quantise $\mathbf{z}_{tl}$ into finite levels, denoted by $q(\mathbf{z}_{tl})$, and pre-compute the value of $\phi_{li}(q(\mathbf{z}_{tl}))$. In this way, the computational complexity can be further reduced.

5.2.2.4 Homogeneous Kernel Map Approximation to Additive Kernels

One drawback of the kernel PCA approximation to additive kernels lies in the fact that it is *data dependent*. That is, the approximate mapping function needs to be learned from a set of training samples. As a result, an extra training step is

required before applying the approximation to an additive kernel. In the following part, we will introduce another approximation method called "homogeneous kernel map," which is *data independent*. Before introducing this method, we need to first review a method called "random Fourier kernel approximation" which inspires the homogeneous kernel map.

1. Random Fourier Kernel Approximation to Stationary Kernels

Among the commonly used kernels in the literature, there is another family of kernels called stationary or translational invariant kernel [42]. It is formally defined as a kernel satisfying the following relationship:

$$\kappa(\mathbf{z}_i, \mathbf{z}_j) = \kappa(\mathbf{z}_i + \mathbf{c}, \mathbf{z}_j + \mathbf{c}). \tag{5.26}$$

A property of stationary kernels is that there always exists a function $\mathscr{K}$ to make $\kappa(\mathbf{z}_i, \mathbf{z}_j) = \mathscr{K}(\mathbf{z}_j - \mathbf{z}_i)$ valid. To prove this property, we can simply set $\mathbf{c} = -\frac{\mathbf{z}_i + \mathbf{z}_j}{2}$ and substitute it to Eq. (5.26):

$$\begin{aligned} \kappa(\mathbf{z}_i, \mathbf{z}_j) &= \kappa\Big(-\frac{\mathbf{z}_i + \mathbf{z}_j}{2} + \mathbf{z}_i, -\frac{\mathbf{z}_i + \mathbf{z}_j}{2} + \mathbf{z}_j\Big) \\ &= \kappa\Big(\frac{\mathbf{z}_i - \mathbf{z}_j}{2}, -\frac{\mathbf{z}_i - \mathbf{z}_j}{2}\Big) \equiv \mathscr{K}(\mathbf{z}_j - \mathbf{z}_i). \end{aligned} \tag{5.27}$$

It has been proved that by Bochner's theorem [42] for any Positive Definite function $\mathscr{F}(\mathbf{z})$, there exists a non-negative measure p, such that $\mathscr{F}(\mathbf{z})$ is its Fourier transform:

$$\mathscr{F}(\mathbf{z}) = \int_{\omega} p(\omega) e^{-j\langle \omega, \mathbf{z} \rangle} d\omega. \tag{5.28}$$

Note that $\mathscr{K}(\mathbf{z}_j - \mathbf{z}_i)$ is a Positive Definite function because the corresponding $\kappa(\mathbf{z}_i, \mathbf{z}_j)$ is assumed to be a Mercer kernel, which always produces a Positive Semi-Definite kernel matrix. This suggests that $\mathscr{K}(\mathbf{z}_j - \mathbf{z}_i)$ can be represented as the Fourier transform of non-negative function p,

$$\mathscr{K}(\mathbf{z}_j - \mathbf{z}_i) = \int_{\omega} p(\omega) e^{-j\langle \omega, (\mathbf{z}_j - \mathbf{z}_i) \rangle} d\omega. \tag{5.29}$$

Since both $\mathscr{K}$ and p are real, we can replace $e^{-j\langle \omega, \mathbf{z} \rangle}$ by $\cos(\langle \omega, \mathbf{z} \rangle)$ and rewrite Eq. (5.29) into

$$\begin{aligned} \mathscr{K}(\mathbf{z}_j - \mathbf{z}_i) &= \int_{\omega} p(\omega) \cos\left(\langle \omega, (\mathbf{z}_j - \mathbf{z}_i) \rangle\right) d\omega \\ &= \int_{\omega} p(\omega) \left(\langle \cos(\omega, \mathbf{z}_i) \rangle \cos(\langle \omega, \mathbf{z}_j \rangle) + \sin(\langle \omega, \mathbf{z}_j \rangle) \sin(\langle \omega, \mathbf{z}_i) \rangle\right) d\omega. \end{aligned} \tag{5.30}$$

If $\mathscr{K}$ is properly scaled, $p(\omega)$ can be viewed as a probability density function. Hence, Eq. (5.30) can be seen as the expectation of $\cos(\langle\omega,(\mathbf{z}_j-\mathbf{z}_i)\rangle)$, that is, $\mathscr{K}(\mathbf{z}_j-\mathbf{z}_i) = E_\omega(\cos(\langle\omega,(\mathbf{z}_j-\mathbf{z}_i)\rangle))$. This result motivates the work in [42] to use the empirical mean to approximate the expectation. More specifically, they randomly draw m frequency components $\{\omega_i\}$, where $i = 1,\cdots,m$, from the distribution $p(\omega)$ and use the average of $\cos(\langle\omega_i,(\mathbf{z}_j-\mathbf{z}_i)\rangle)$ to approximate $E(\cos(\langle\omega,(\mathbf{z}_j-\mathbf{z}_i)\rangle))$ as

$$E(\cos(\langle\omega,(\mathbf{z}_j-\mathbf{z}_i)\rangle)) \approx \frac{1}{m}\sum_{i=1}^{m}\cos(\langle\omega_i,(\mathbf{z}_j-\mathbf{z}_i)\rangle) \triangleq \langle\phi(\mathbf{z}_i),\phi(\mathbf{z}_j)\rangle, \tag{5.31}$$

where $\phi(\mathbf{z}) = \frac{1}{\sqrt{m}}(\cos(\langle\omega_1,\mathbf{z}\rangle),\cdots,\cos(\langle\omega_m,\mathbf{z}\rangle),\sin(\langle\omega_1,\mathbf{z}\rangle),\cdots,\sin(\langle\omega_m,\mathbf{z}\rangle))^\top$. It defines an approximate explicit feature mapping and this method is called random Fourier kernel map in [42]. Note that this approximation is data independent—the only input of this approximation method is $p(\omega)$, which can be obtained via the Fourier transform of the given kernel function.

2. Homogeneous Kernel Map Approximation to Additive Kernels

Inspired by the random Fourier kernel approximation and the kernel PCA approximation to additive kernels, the work in [50] develops a unified framework to build *data-independent* approximate feature mapping for a large family of additive kernels, known as γ-homogeneous kernels. Formally, an additive kernel $\kappa(\cdot,\cdot)$ is γ-homogeneous if its DKF $k(\cdot,\cdot)$ satisfies

$$\forall c \geq 0:\ k(c\mathbf{z}_{il},c\mathbf{z}_{jl}) = c^\gamma k(\mathbf{z}_{il},\mathbf{z}_{jl}). \tag{5.32}$$

By choosing $c = 1/\sqrt{\mathbf{z}_{il}\mathbf{z}_{jl}}$, the DKF of a γ-homogeneous kernel can be written as

$$\begin{aligned} k(\mathbf{z}_{il},\mathbf{z}_{jl}) &= c^{-\gamma}k(c\mathbf{z}_{il},c\mathbf{z}_{jl}) = (\mathbf{z}_{il}\mathbf{z}_{jl})^{\frac{\gamma}{2}}k\left(\sqrt{\frac{\mathbf{z}_{il}}{\mathbf{z}_{jl}}},\sqrt{\frac{\mathbf{z}_{jl}}{\mathbf{z}_{il}}}\right) \\ &= (\mathbf{z}_{il}\mathbf{z}_{jl})^{\frac{\gamma}{2}}\mathscr{K}(\log(\mathbf{z}_{il})-\log(\mathbf{z}_{jl})), \end{aligned} \tag{5.33}$$

where the scalar function $\mathscr{K}(\cdot)$ is called "kernel signature" and it is defined as:

$$\mathscr{K}(\lambda) = k(e^{\frac{\lambda}{2}},e^{-\frac{\lambda}{2}}). \tag{5.34}$$

Note that the role of kernel signature resembles the function $\mathscr{K}$ in the random Fourier map method previously introduced. In fact, the derivation of the algorithm in [50] bears a similarity with the one for random Fourier kernel approximation.

The work in [50] proves that a γ-homogeneous kernel $\kappa(\cdot,\cdot)$ is positive definite if, and only if, its signature $\mathscr{K}(\cdot)$ is a positive definite function. With this result, the Bochner's theorem in Eq. (5.28) can be readily applied to the kernel signature:

$$k(\mathbf{z}_{il},\mathbf{z}_{jl}) = (\mathbf{z}_{il}\mathbf{z}_{jl})^{\frac{\gamma}{2}}\mathscr{K}(\lambda) = (\mathbf{z}_{il}\mathbf{z}_{jl})^{\frac{\gamma}{2}}\int_{-\infty}^{\infty} e^{-j\omega\lambda}q(\omega)d\omega$$
$$= \int_{-\infty}^{\infty}\left(e^{-j\omega\log(\mathbf{z}_{jl})}\sqrt{\mathbf{z}_{jl}^{\gamma}q(\omega)}\right)^{\star}\left(e^{-j\omega\log(\mathbf{z}_{il})}\sqrt{\mathbf{z}_{il}^{\gamma}q(\omega)}\right)d\omega, \quad (5.35)$$

where $\lambda = \log\frac{\mathbf{z}_{il}}{\mathbf{z}_{jl}}$ and $q(\omega)$ is the Fourier transform of the kernel signature $\mathscr{K}(\lambda)$. In this case, by defining

$$\phi_{\omega}(\mathbf{z}_{il}) = e^{-j\omega\log(\mathbf{z}_{il})}\sqrt{\mathbf{z}_{il}^{\gamma}q(\omega)}, \quad (5.36)$$

it is easy to verify that

$$k(\mathbf{z}_{il},\mathbf{z}_{jl}) = \langle\phi_{\omega}(\mathbf{z}_{il}),\phi_{\omega}(\mathbf{z}_{jl})\rangle, \quad (5.37)$$

where $\langle\cdot,\cdot\rangle$ is the inner product in a Hilbert space.

However, the feature mapping ϕ_{ω} has infinite dimensions. To handle this situation, the work in [50] proposes an idea that is similar to the discrete Fourier transform. They first approximate the kernel signature function $\mathscr{K}(\cdot)$ by its periodic version with the assumption that the function $\mathscr{K}(\lambda)$ has a restrictive domain of λ, which can usually be satisfied in practice. According to the theory of Fourier transform, the transform of a periodic signal is discrete. Then they simply choose the first $\tilde{d}_l$ frequency components as the feature mapping for the lth DKF.

5.2.2.5 Experimental Comparison

We quote the experimental result in [50] to give an intuition on the effectiveness of the aforementioned approximate explicit feature mappings. In this experiment, two methods, additive kernel PCA approximation in Sect. 5.2.2.3 and homogeneous kernel map in Sect. 5.2.2.4, are compared with the baseline using the original nonlinear kernel. The evaluation is carried out on Caltech-101 data set with SIFT as the local feature descriptor. A visual codebook with 600 visual words is created and a $1\times 1+2\times 2+4\times 4$ spatial pyramid [24] is used. The result is shown in Table 5.2. As seen, the approximate explicit feature mappings achieves comparable or even better classification performance than the original nonlinear kernels, but with much less training time. The speed advantage of the homogeneous kernel maps over the additive Kernel PCA approximation is due to the fact that the latter requires an extra training step to learn the mapping function. In addition, an interesting observation is that the $\gamma = 1/2$ variant of the homogeneous kernel performs much better than the original nonlinear kernel. Note that this variant is equivalent to calculating the square root of the data first and then applying the nonlinear kernel. The square rooting operation makes the value in each bin of the histogram more stable and reduces the negative impact of the "burstiness" phenomenon, which will be discussed in Sect. 5.3.

Table 5.2 This table is quoted from [50]

			χ^2-kernel		JS kernel		HIK	
Appro. mapping	Dimension	SVM solver	Acc.	Time	Acc.	Time	Acc.	Time
Original	–	LIBSVM	64.1	1769	64.2	6455	62.3	1729
			(± 0.6)	(± 16)	(± 0.5)	(± 39)	(± 0.3)	(± 19)
Homogeneous	3	LIBLINEAR	64.4	312	64.2	483	64.6	367
			(± 0.6)	(± 14)	(± 0.4)	(± 52)	(± 0.5)	(± 18)
Homogeneous	5	LIBLINEAR	64.2	746	64.1	804	63.7	653
			(± 0.5)	(± 58)	(± 0.4)	(± 82)	(± 0.6)	(± 54)
$\frac{1}{2}$-Homogeneous	3	LIBLINEAR	67.2	380	67.3	456	67.0	432
			(± 0.7)	(± 10)	(± 0.6)	(± 54)	(± 0.6)	(± 55)
Add.-kernel-PCA	3	LIBLINEAR	64.3	682	64.5	1351	62.7	741
			(± 0.5)	(± 67)	(± 0.6)	(± 93)	(± 0.4)	(± 41)
Add.-kernel-PCA	5	LIBLINEAR	64.1	1287	64.1	1576	62.6	1374
			(± 0.5)	(± 156)	(± 0.5)	(± 158)	(± 0.4)	(± 257)

It compares different approximate explicit feature mappings with respect to a number of original nonlinear kernels. The comparison is based on the classification performance and the incurred training time of the linear or nonlinear SVM classifiers. The "Dimension" means the dimensions of the mapping used to approximate each DFK of an additive kernel

5.3 Application of SVMs with Max-Pooling-Based Linear Kernel

The motivation of using nonlinear kernel-based SVMs is that they tend to achieve better performance than linear SVMs for image categorisation with histogram-based image representation. However, is it possible to design an image representation with which linear SVMs can achieve the performance comparable or even better than the nonlinear counterparts? If it is, the computational issue can be readily removed by adopting linear SVMs for large-scale image classification. Note that in some sense designing a new image representation for a linear kernel can also be viewed as inventing a new nonlinear kernel at the level of original image data. To show this, we can abstract the process of forming an image representation as a function $e(\cdot)$. A kernel defined based on the original image data can then be expressed as $\hat{\kappa}(I_i, I_j) = \langle e(I_i), e(I_j) \rangle$. Thus, if we change the image representation $e(\cdot)$, we virtually change the image-level kernel $\hat{\kappa}$.

The work in [57] is among the earliest ones that take the above approach. It combines sparse coding and max-pooling to obtain image representation and use it for image categorisation. That work shows that with this representation, a simple linear SVM classifier has been able to attain the performance superior to the traditional ones in which a nonlinear kernel is employed. Later, the work in [5] further discovers that the max-pooling step is the key to the success of the system in [57]. A comprehensive experimental study is conducted in [5] to compare various combinations of coding and pooling methods. Their results are quoted in Table 5.3. From the result, three conclusions can be drawn:

1. The use of max-pooling significantly improves the classification performance of linear SVMs for all the coding methods. The improvement is even significant for the simplest hard-assignment coding. In that case, a pooled coding vector only indicates the presence or absence of a visual word in the associated image;
2. By using the max-pooling, linear SVMs can achieve comparable or even better performance than the counterpart which adopts the sum-pooling and then nonlinear kernel-based SVMs.
3. Once the max-pooling is used, the classification performance obtained by linear and nonlinear SVMs becomes similar.

Table 5.3 This table is quoted from [5]

	Performance on Scene-15		Performance on Caltech-101	
Coding method and kernel	Sum-pooling	Max-pooling	Sum-pooling	Max-pooling
Hard-assignment + linear kernel	51.4 ± 0.9 %	64.3 ± 0.9 %	73.9 ± 0.9 %	80.1 ± 0.6 %
Hard-assignment + HIK kernel	64.2 ± 1.0 %	64.3 ± 0.9 %	80.8 ± 0.4 %	80.1 ± 0.6 %
Soft-assignment + linear kernel	57.9 ± 1.5 %	69.0 ± 0.8 %	75.6 ± 0.5 %	81.4 ± 0.6 %
Soft-assignment + HIK kernel	66.1 ± 1.2 %	70.6 ± 1.0 %	81.2 ± 0.4 %	83.0 ± 0.7 %
Sparse coding + linear kernel	61.3 ± 1.3 %	71.5 ± 1.1 %	76.9 ± 0.6 %	83.1 ± 0.6 %
Sparse coding + HIK kernel	70.3 ± 1.3 %	71.8 ± 1.0 %	83.2 ± 0.4 %	84.1 ± 0.5 %

It lists the classification performance obtained by combining different pooling and coding methods

In sum, we can conclude that max-pooling is an effective way to produce better image representation and that linear SVMs show excellent performance in classifying the max-pooled coding vectors.

However, why can max-pooling obtain such a "magic" performance? To understand this, several interpretations have been proposed in the literature. In the following parts, we discuss three representative ones.

The work in [4] explains the superior performance of max-pooling by showing that it can generate more discriminative image representations. Assuming the case of binary classification, that work compares the class separability of each individual feature generated through max-pooling and sum-pooling. The class separability is defined as:

$$\psi = \frac{|E(\mathbf{z}_k|C_1) - E(\mathbf{z}_k|C_2)|}{\text{var}(\mathbf{z}_k)}, \tag{5.38}$$

where $\mathbf{z}_k$ denotes the value of the kth dimension of a pooled coding vector. It can be obtained by $\mathbf{z}_k = \frac{1}{n}\sum_{i=1}^{n}\mathbf{u}_{ik}$ for sum-pooling or $\mathbf{z}_k = \max_{i=1}^{n}\mathbf{u}_{ik}$, where n is the number of coding vectors in an image and $\mathbf{u}_{ik}$ denotes the coding value at the kth dimension of the ith coding vector. C_1 and C_2 denote two classes. $E(\mathbf{z}_k|C_1)$ and $E(\mathbf{z}_k|C_2)$ are the expectation of $\mathbf{z}_k$ in each class, respectively. $\text{var}(\mathbf{z}_k)$ denotes the variance of $\mathbf{z}_k$. To simplify their analysis, they assume that the $\{\mathbf{u}_{1k}, \cdots, \mathbf{u}_{ik}, \cdots, \mathbf{u}_{nk}\}$ are i.i.d. random variables.

For hard-assignment coding, the coding value $\mathbf{u}_{ik}$ is assumed to be drawn from a Bernoulli distribution in which $\mathbf{u}_{ik} = 1$ with probability α and $\mathbf{u}_{ik} = 0$ with probability $1 - \alpha$. In the context of hard-assignment coding, this means that a coding value will be "activated" $(= 1)$ with probability α. Based on this assumption the class separability with respect to max-pooling and sum-pooling is derived as follows:

$$\psi_{\text{sum}} = \frac{|\alpha_1 - \alpha_2|\sqrt{n}}{\sqrt{\alpha_1(1-\alpha_1)} + \sqrt{\alpha_2(1-\alpha_2)}} \tag{5.39}$$

$$\psi_{\text{max}} = \frac{|(1-\alpha_1)^n - (1-\alpha_2)^n|}{\sqrt{(1-(1-\alpha_1)^n)(1-\alpha_1)^n} + \sqrt{(1-(1-\alpha_2)^n)(1-\alpha_2)^n}}, \tag{5.40}$$

where α_1 and α_2 denote $P(\mathbf{u}_{ik} = 1|C_1)$ and $P(\mathbf{u}_{ik} = 1|C_2)$, respectively, that is, the probability of a coding value being "activated" in each class. As previously defined, n is the number of coding vectors in an image and it is called "pool cardinality" in Fig. 5.3. By evaluating the class separability for max-pooling and sum-pooling with different α_1 and α_2, it is found that when the activation probability α_1 and α_2 are low, the ratio of α_1 to α_2 is large, and the value of n is small, max-pooling can achieve better class separability than sum-pooling. This explains the superior performance of max-pooling.

However, the above analysis cannot be readily used to explain the better discrimination achieved by max-pooling for the coding schemes where continuous coding coefficient is used, for example, the sparse coding. [4] To refine the analysis,

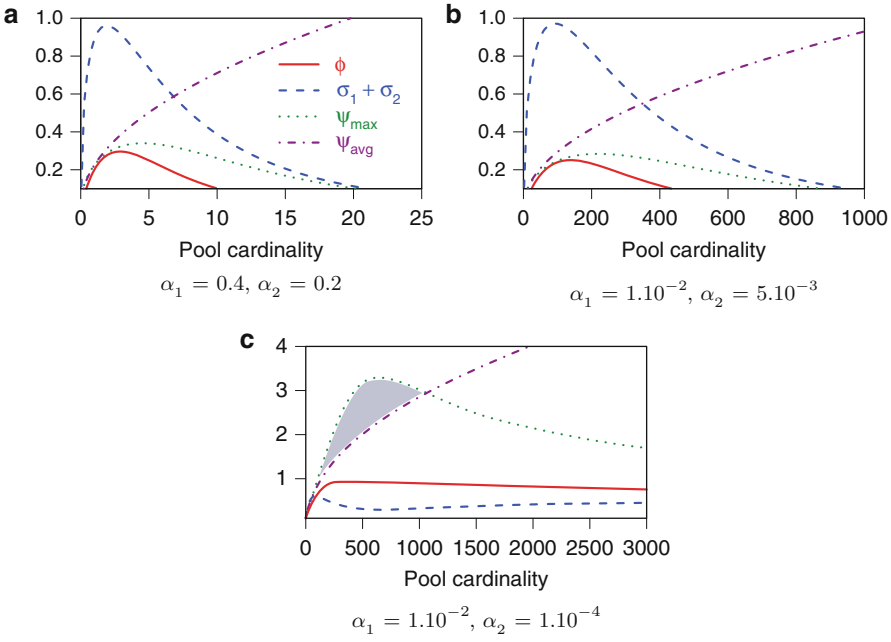

Fig. 5.3 $\phi(n) = |E(\mathbf{z}_k|C_1) - E(\mathbf{z}_k|C_2)| = |(1-\alpha_1)^n - (1-\alpha_2)^n|$ denotes the difference between the expectation of the max-pooled features in classes C_1 and C_2, and σ_1 and σ_2 are the standard deviations. $\psi_{\max}$ and ψ_{avg} are the class separability defined in Eq. (5.39). (**a**) When α_1 and α_2 are relatively large, the peak of the class separability is achieved at small pool cardinalities; (**b**) With the decreasing value of α_1 and α_2, the peak becomes wider (note the change of scale in the x axis); (**c**) When α_1 and α_2 are both small and $\alpha_1 \gg \alpha_2$, $\psi_{\max}$ becomes larger than ψ_{avg} for some cardinalities (indicated by the shaded area). Image courtesy of [4]

the work in [5] further assumes that the distribution of a coding value in an image is the mixture of two distributions: a distribution corresponding to the local descriptors from the object and a distribution corresponding to the local descriptors from the background [5]. The mixture weights of the two distributions vary from image to image. Through a more involved analysis, it is shown in [4, 5] that under certain conditions, max-pooling creates immunity to the variation in the mixture weights and thus it can lead to better classification performance.

The work in [26] points out that for soft-assignment coding, the coding coefficient can be viewed as the membership of a local feature descriptor with respect to different visual words, that is, $P(\mathbf{v}_j|\mathbf{x}_i)$, where $\mathbf{x}_i$ is the ith local descriptor in an image and $\mathbf{v}_j$ is the jth visual word. Hence, each dimension in the max-pooled coding vector can be related to the maximum membership score of the corresponding word, and this score is proved to be the lower bound of the probability of finding at least one local descriptor belonging to this word:

$$P(\mathbf{v}_j|\mathscr{A}) = 1 - \prod_{i=1}^{n}(1 - P(\mathbf{v}_j|\mathbf{x}_i)) \geq \max_{i=1,\cdots,n} P(\mathbf{v}_j|\mathbf{x}_i), \tag{5.41}$$

where $\mathscr{A}$ denotes the set of n local feature descriptors in an image. Assuming the i.i.d property for the local descriptors, the probability of finding at least one descriptor belonging to the jth visual word can be calculated by $1-\prod_{i=1}^{n}(1-P(\mathbf{v}_j|\mathbf{x}_i))$. As argued in [26], directly computing this probability involves the product of $(1-P(\mathbf{v}_j|\mathbf{x}_i))$ for all $\mathbf{x}_i$ and each of them could bring in noise, making the result unreliable. In contrast, computing the lower bound via the max-pooling scheme tends to obtain a more reliable estimate since it only considers the largest term.

Based on this interpretation, the work in [26] further generalises the max-pooling to mixed-order max-pooling to model the higher order occurrence information of a visual word in an image, that is, the probability that a word occurs more than k times. As shown in that work, the classification performance can be further improved by using the mix-order max-pooling.

The analysis in [26] can also be generalised to other coding schemes with continuous coding coefficient, if the coefficient can be related to the membership score. From this viewpoint, sum-pooling will not be suitable for this coding scheme since it may accumulate low membership scores to a high one, which, however, is not a good indication of the presence of a visual word in such a case. For example, by the sum-pooling, 100 local descriptors with coding value 0.002 for a given bin will produce the pooled value of 0.2, which appears to have the same effect of observing one local descriptor with coding value 0.2. However, it is clear that the former has much weaker indication of the presence of the corresponding visual word.

Another interpretation of the good classification performance of linear SVMs with the max-pooling can be obtained from the "burstiness" phenomenon, which was initially discovered in the text classification [30] and was later observed in image retrieval with the Bag-of-features model [18]. This phenomenon indicates that "a visual pattern appears often more frequently than a statistically independent model would predict" [18]. Basically, it suggests that if one visual pattern appears once, it will be more likely to occur again in the same image. Intuitively, it can be understood in the way that the occurrence of some visual concepts will produce many repetitive local visual patterns. For example, the occurrence of a wall will produce many local patches corresponding to bricks. However, the size of an object often changes dramatically from image to image due to scale variation, which makes the occurrence frequency of its corresponding local visual patterns unstable. As a result, the sum-pooling of the coding vectors suffers from this instability because it essentially reflects the occurrence frequency of the visual words. For linear SVMs, its decision function is merely a weighted sum of the pooled coding values and thus it tends to be affected by the variation of the occurrence frequency. In contrast, nonlinear kernel SVMs can mitigate the adverse effect of burstiness phenomenon by its nonlinear operation. For example, in HIK $\kappa(\mathbf{z}_i,\mathbf{z}_j)=\sum_{l=1}^{d}\min(\mathbf{z}_{il},\mathbf{z}_{jl})$, if a statistically less probable large value occurs at $\mathbf{z}_{il}$ due to the burstiness, its impact will be capped by $\mathbf{z}_{jl}$. However, if max-pooling is used, the linear SVMs will no longer be affected by the occurrence frequency of a visual word. In some sense, the max-pooling scheme builds a better image-level kernel and it helps the linear SVMs achieve better classification performance.

5.4 Application of SVMs with Point-Set Kernels

5.4.1 Introduction to Point-Set Kernels

Recall that in the Bag-of-features model a set of local feature descriptors is extracted from an image. Hence, an image can essentially be viewed as a point set in a multi-dimensional feature descriptor space. In addition to building a visual codebook and generating an image representation, another line of research measures the similarity between two images directly based on the associated point sets. This is desirable from the perspective of SVM application because the similarity can be used as a kernel function to perform nonlinear SVM classification. Compared with the approach of building visual codebooks, this approach can lead to a more compact and conceptually simpler classification system. Better classification result could even be achieved when appropriate point-set kernels are employed.

A variety of point-set kernels have been developed in the recent literature on machine learning and computer vision. Generally speaking, point-set kernels are constructed in two ways. One is to evaluate the similarity between the points (e.g., local feature descriptors) in two sets via a common kernel (often called a local or base kernel) and then combine the kernels to obtain a point-set kernel. The simplest one may be the sum-match kernel that sums the local kernels between every pair of points in the two sets. However, in the presence of outliers, this kernel cannot effectively reflect the set similarity because good matches between points are often buried by a large number of bad matches. In [51], a sum-max kernel is proposed, which averages the maximum local kernel value from each point in one set with respect to the points in the other set. In [29], a sum-exponent kernel is put forward. It computes the sum of the local kernels over all pairs of points in the two sets after raising each kernel value to the power p. This allows it to adjust the weight of each local kernel in the summation instead of treating them equally. The aforementioned sum-match kernel and sum-max kernel can be regarded two special cases of the sum-exponent kernel. The work in [3] suggests considering only the local kernels between the points that can be truly matched by a matching algorithm and discusses the way to make the obtained point-set kernel to satisfy the Mercer's condition. In [46], a general family of set kernels is derived based upon local kernels. The proposed kernel can combine local kernels in a linear or nonlinear way, where the nonlinear combination is achieved by mapping each point set onto a high-dimensional matrix space. In addition, the work in [37] considers the case where each point corresponds to a pixel in an image or a voxel of a video sequence. Taking into account the location information of each point, a neighborhood kernel is defined to compare each pair of points and the point-set kernel is defined as the average of all the neighborhood kernels.

The other way to construct a point-set kernel is to estimate a probability distribution of the points in each set and use the similarity between the two distributions to define a set-level kernel. The work in [21] uses the Bhattacharyya's affinity between two distributions to define a kernel for sets of vectors. To ensure

the kernel to have sufficient representational power, that work maps the vectors onto a kernel-induced feature space and computes the Bhattacharyya's affinity of the probability distributions in the feature space. Similarly, in [35] a Gaussian Mixture Model (GMM) is used to obtain a probabilistic model of each set and the Kullback–Leibler divergence between the probabilistic models is used to define a point-set kernel. The work in [9] is motivated by comparing the distributions of two point sets before and after the two sets are merged. Intuitively, a strengthened distribution will be obtained if the two point sets are similar. Following this idea, that work develops point-set kernels by studying the properties of the concatenation of two point sets. In addition, the above way of constructing a point-set kernel can be related to building a kernel based on a generative model, which has been widely used as a means to combining generative models with discriminative classifiers. Fisher kernel [17] may be the most commonly used one in this regard and it has been applied to image classification recently [38, 39].

For image categorisation with the Bag-of-features model, a systematic study of point-set kernels is conducted in [51]. Among the existing point-set kernels developed in this area, the Pyramid Matching Kernel (PMK) [14], the Efficient Matching Kernel (EMK) [2] and the Fisher kernel [38] are three representative methods. This following parts will introduce the work in [51]. After that, it will be focused on the PMK, the EMK and the Fisher kernel.

5.4.2 *A Kernel Recipe to Local Feature-Based Image Recognition*

A general kernel-based approach is proposed in [51] for image recognition with local feature descriptors. Its motivation is to combine the representation power of local features with the excellent discriminative capability of the SVM classifiers. To achieve this goal, that work proposes a class of new kernels for point sets based on the commonly used kernels, or equally the local kernels mentioned above. Let $\mathscr{A}_i = \{\mathbf{x}_{i1}, \mathbf{x}_{i2}, \cdots, \mathbf{x}_{in_i}\}$ and $\mathscr{A}_j = \{\mathbf{x}_{j1}, \mathbf{x}_{j2}, \cdots, \mathbf{x}_{jn_j}\}$ denote the two point sets associated with images i and j, where $\mathbf{x}$ is a local feature descriptor. A measure evaluates the similarity of the set $\mathscr{A}_i$ with respect to the set $\mathscr{A}_j$ is defined as

$$m(\mathscr{A}_i|\mathscr{A}_j) = \frac{1}{n_i}\sum_{p=1}^{n_i}\max_{q=1}^{n_j}\kappa_l(\mathbf{x}_{ip}, \mathbf{x}_{jq}), \tag{5.42}$$

where $\kappa_l(\mathbf{x}, \mathbf{x}')$ plays the role of a local kernel and any existing kernel can be used for it. This measure finds the best match (in terms of the value of the local kernel) of each local feature in $\mathscr{A}_i$ in the set $\mathscr{A}_j$ and computes the average. Note that $m(\mathscr{A}_i|\mathscr{A}_j)$ does not equal $m(\mathscr{A}_j|\mathscr{A}_i)$ in general. With such a measure, the work in [51] defines a new class of kernel as

$$\kappa(\mathscr{A}_i, \mathscr{A}_j) = \frac{1}{2}\left[m(\mathscr{A}_i|\mathscr{A}_j) + m(\mathscr{A}_j|\mathscr{A}_i)\right]. \tag{5.43}$$

This kernel is a symmetrical function over the local features. Both the non-Gaussian RBF kernel and the χ^2-RBF kernel in Sect. 5.2.1 have been shown in [51] as a valid local kernel. In the experimental study, that work compares a nearest-neighbour classifier with a predefined distance metric and the SVM classifier using the corresponding point-set kernel. The results on object and face recognition data sets demonstrate the superiority of the SVM classifier with the new class of point-set kernels. However, as pointed out in [29], this class of kernels does not necessarily satisfy the Mercer's condition due to the use of the "max" operation.

5.4.3 Pyramid Matching Kernel

The work of PMK aims to develop a kernel function for SVM classification that can efficiently measure the similarity of two point sets [14]. It can be regarded as combining a set of local kernels to produce a point-set kernel. As indicated by its name, the PMK builds a pyramid of multi-resolution histograms to quantise the points in the two sets to be compared. At each resolution, the HIK, introduced in Sect. 5.2.1.1, is used as the local kernel to evaluate the similarity of the two sets. The similarity from different levels is then linearly combined to obtain the set-level similarity, with different weights used for the multiple resolutions.

Recall that $\mathbf{x}$ denotes a local feature descriptor in a d-dimensional space. The PMK partitions the volume occupied by the descriptors in the d-dimensional space using a set of bins with a gradually increasing size. For example, the side length of the d-dimensional bins is doubled at each resolution level. By appropriately scaling the data, the smallest bin size ensures that each individual descriptor will reside in its own bin, whereas the largest bin size ensures that all the descriptors will be contained in the same bin. Assigning the local descriptors into these bins leads to a hierarchy of multi-resolution histograms, and the bin values of this set of histograms vary with different local descriptor sets. Recall that $\mathscr{A}$ denotes a set of local descriptors extracted from an image. By concatenating the multi-resolution histograms for the set $\mathscr{A}$, a long vector can be obtained as $\phi(\mathscr{A}) = (\mathbf{h}_{-1}(\mathscr{A}), \mathbf{h}_0(\mathscr{A}), \cdots, \mathbf{h}_L(\mathscr{A}))^\top$, where $\mathbf{h}_{-1}$ indicates the histogram with the highest resolution for which no descriptor is matched (falling into the same bin) and $\mathbf{h}_L$ indicates the histogram with the lowest resolution for which all local feature descriptors stay in the same bin.

Based on the representation of $\phi(\mathscr{A})$, the PMK evaluates the similarity between the two sets $\mathscr{A}_i$ and $\mathscr{A}_j$ (or equally the similarity of images i and j) as

$$\kappa(\mathscr{A}_i, \mathscr{A}_j) = \sum_{l=0}^{L} \omega_l n_l, \tag{5.44}$$

where n_l is the number of "new" matches found at the level l and ω_l is a weight indicating the contribution of a match at level l to the final similarity. Note that a new match at level l means that a pair of local descriptors falls into the same bin of a histogram at level l but resides in different bins for any level lower than l (lower levels correspond to the histograms with higher resolutions). The weight for new matches decreases with the increase of level. This is used to emphasise that the matches at higher-resolution histograms are more important for similarity evaluation. The number of matched points at level l, denoted by $\hat{n}_l$, is computed by histogram intersection as

$$\hat{n}_l = \sum_{i=1}^{T_l} \min(\mathbf{h}_{li}(\mathscr{A}), \mathbf{h}_{li}(\mathscr{B})), \tag{5.45}$$

where T_l is the number of bins in the histogram $\mathbf{h}_l(\cdot)$. Based on $\hat{n}_l$, the number of new matches at each level l can be conveniently worked out as $n_l = \hat{n}_l - \hat{n}_{l-1}$. In addition, to remove the impact of the cardinality of a point set (e.g., a large-sized point set usually has more descriptors to be matched with the descriptors in other sets), a normalised version of the PMK is defined as

$$\kappa_{nrml}(\mathscr{A}_i, \mathscr{A}_j) = \frac{\kappa(\mathscr{A}_i, \mathscr{A}_j)}{\sqrt{\kappa(\mathscr{A}_i, \mathscr{A}_i)\kappa(\mathscr{A}_j, \mathscr{A}_j)}}. \tag{5.46}$$

The PMK is proved to be a Mercer kernel in [14] and it can be efficiently computed once the two sets of local feature descriptors are given. With the use of histogram interaction as the local kernel, the PMK works well with two sets having different cardinalities and can effectively conduct partial matching. This property is important for image categorisation because (1) the number of local descriptors from different images is often different; (2) local descriptors can disappear due to occlusion or the change of view angles, object pose, and scale; (3) irrelevant or noisy descriptors may appear due to the presence of background clutter. As experimentally demonstrated in that work, the pyramid matching used by the PMK can well approximate the result obtained by applying optimal matching between the points in two sets. The advantages of the PMK over some of the existing point-set kernels are summarised in [14] as (1) it is computationally more efficient; (2) it is proved to be positive-definite; (3) it does not need to fit a parametric model to data; (4) it can handle two sets of different cardinalities. As shown in the experimental study in [14], this kernel can demonstrate excellent classification performance in image categorisation tasks.

5.4.4 *Efficient Matching Kernel*

The motivation of the EMK [2] is to speed up the training and test phases in which a point-set kernel is used. Recall that a typical way to construct a point-set kernel is to combine the local kernels over all pairs of points, leading to a sum-match kernel

$$\kappa(\mathscr{A}_i, \mathscr{A}_j) = \frac{1}{|\mathscr{A}_i|} \frac{1}{|\mathscr{A}_j|} \sum_{\mathbf{x} \in \mathscr{A}_i} \sum_{\mathbf{x}' \in \mathscr{A}_j} \kappa_l(\mathbf{x}, \mathbf{x}'), \tag{5.47}$$

where $\mathscr{A}_i$ denotes a set of points, $\mathbf{x}$ is a point (e.g., a local feature descriptor in a d-dimensional space) in this set and $|\mathscr{A}_i|$ is the cardinality of the set. Computing such a kernel has a computational complexity of $\mathscr{O}(|\mathscr{A}_i||\mathscr{A}_j|d)$. This complexity increases to $\mathscr{O}(|\mathscr{A}_i||\mathscr{A}_j|dn^2)$ when training an SVM classifier with a set of n training images. This makes the application of SVMs with such a kernel inefficient in handling a large-sized training set. Also, such a complexity prevents the obtained SVM classifier from efficiently classifying unseen images. Assuming that the local kernel $\kappa_l(\mathbf{x}, \mathbf{x}')$ can be expressed as an inner product between $\psi(\mathbf{x})$ and $\psi(\mathbf{x}')$ in a finite-dimensional kernel-induced feature space, the above sum-match kernel can be rewritten as

$$\kappa(\mathscr{A}_i, \mathscr{A}_j) = \left\langle \frac{1}{|\mathscr{A}_i|} \sum_{\mathbf{x} \in \mathscr{A}_i} \psi(\mathbf{x}), \frac{1}{|\mathscr{A}_j|} \sum_{\mathbf{x}' \in \mathscr{A}_j} \psi(\mathbf{x}') \right\rangle \triangleq \left\langle \Psi(\mathscr{A}_i), \Psi(\mathscr{A}_j) \right\rangle, \tag{5.48}$$

where it is defined that $\Psi(\mathscr{A}_i) = \frac{1}{|\mathscr{A}_i|} \sum_{\mathbf{x} \in \mathscr{A}_i} \psi(\mathbf{x})$. As proposed in the work of EMK, if the implicit mapping $\psi(\mathbf{x})$ is approximated by an explicit mapping $\phi(\mathbf{x})$, the sum-match kernel evaluation can be avoided and a linear SVM classifier will be sufficient. This will remove the aforementioned computational issue and at the same time maintain the classification performance brought by the kernel trick.

Let $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ be a set of predefined basis vectors in a d-dimensional space. Their images under the mapping $\psi(\cdot)$ form a matrix $\mathbf{B} = (\psi(\mathbf{x}_1), \cdots, \psi(\mathbf{x}_m))$. The work of EMK approximates $\psi(\mathbf{x})$ with its projection into the space spanned by $\mathbf{B}$. The project coefficient $\mathbf{z}$ can be obtained by minimising the reconstruction error

$$\mathbf{z}^\star = \arg\min_{\mathbf{z} \in \mathbb{R}^m} \|\psi(\mathbf{x}) - \mathbf{B}\mathbf{z}\|_2^2. \tag{5.49}$$

The optimal solution $\mathbf{z}^\star$ can be analytically expressed as $\mathbf{z}^\star = \left(\mathbf{B}^\top \mathbf{B}\right)^{-1} \left(\mathbf{B}^\top \psi(\mathbf{x})\right) = \left(\mathbf{B}^\top \mathbf{B}\right)^{-1} \mathbf{k}_\mathbf{B}(\mathbf{x})$, where $\mathbf{k}_\mathbf{B}(\mathbf{x})$ is a vector consisting of the (local) kernel values between $\mathbf{x}$ and each of the m basis vectors. Let $\mathbf{K}_\mathbf{B}$ denote the $m \times m$ (local) kernel matrix computed over all the basis vectors. An approximate kernel is defined with the projection as

$$\kappa_{\text{appro}}(\mathbf{x},\mathbf{x}') = (\mathbf{B}\mathbf{z}^\star)^\top \left(\mathbf{B}\mathbf{z}'^\star\right) = \mathbf{k}_\mathbf{B}(\mathbf{x})^\top \mathbf{K}_\mathbf{B}^{-1}\mathbf{k}_\mathbf{B}(\mathbf{x}') = \mathbf{k}_\mathbf{B}(\mathbf{x})^\top \mathbf{C}^\top \mathbf{C}\mathbf{k}_\mathbf{B}(\mathbf{x}'), \quad (5.50)$$

where $\mathbf{C}$ is a matrix satisfying $\mathbf{C}^\top\mathbf{C} = \mathbf{K}_\mathbf{B}^{-1}$. In doing so, the mapping of the approximate kernel can be explicitly obtained as $\phi(\mathbf{x}) = \mathbf{C}\mathbf{k}_\mathbf{B}(\mathbf{x})$. With this mapping, the approximate point-set kernel can be defined as an inner product

$$\kappa_{\text{appro}}(\mathscr{A}_i,\mathscr{A}_j) = \left\langle \frac{1}{|\mathscr{A}_i|}\sum_{\mathbf{x}\in\mathscr{A}_i}\phi(\mathbf{x}), \frac{1}{|\mathscr{A}_j|}\sum_{\mathbf{x}'\in\mathscr{A}_j}\phi(\mathbf{x}')\right\rangle \triangleq \left\langle \Phi(\mathscr{A}_i)\Phi(\mathscr{A}_j)\right\rangle, \quad (5.51)$$

where it is defined that $\Phi(\mathscr{A}_i) = \frac{1}{|\mathscr{A}_i|}\sum_{\mathbf{x}\in\mathscr{A}_i}\phi(\mathbf{x}) = \frac{1}{|\mathscr{A}_i|}\sum_{\mathbf{x}\in\mathscr{A}_i}\mathbf{C}\mathbf{k}_\mathbf{B}(\mathbf{x})$. Since this mapping can be explicitly obtained, it becomes unnecessary to evaluate the kernel matrix for κ_{appro} and a linear classifier can be employed. As can be seen, when the number of basis vectors, m, is not large, the computation at the training and test phases can be considerably reduced.

The last issue is to decide the matrix $\mathbf{B}$ which consists of the m basis vectors. By randomly selecting l local feature descriptors, the work of EMK jointly learns the optimal $\mathbf{B}$ and the projection coefficient $\mathbf{z}$ for each descriptor by minimising the total reconstruction error

$$\{\mathbf{B}^\star,\mathbf{z}_1^\star,\cdots,\mathbf{z}_m^\star\} = \arg\min \sum_{i=1}^{l} \|\psi(\mathbf{x}_i) - \mathbf{B}\mathbf{z}_i\|_2^2. \quad (5.52)$$

Applying the result of $\mathbf{z}^\star = \left(\mathbf{B}^\top\mathbf{B}\right)^{-1}\left(\mathbf{B}^\top\psi(\mathbf{x})\right)$ again, the variable $\mathbf{z}$ is removed and an optimisation problem solely for $\mathbf{B}$ is obtained as

$$\mathbf{B}^\star = \arg\min\left(-\sum_{i=1}^{l}\mathbf{k}_\mathbf{B}(\mathbf{x}_i)^\top\mathbf{K}_\mathbf{B}^{-1}\mathbf{k}_\mathbf{B}(\mathbf{x}_i)\right). \quad (5.53)$$

The optimal $\mathbf{B}$ is solved by a gradient descent algorithm. This completes the derivation of the EMK. As seen, the EMK method has no constraint on the type of local kernels and therefore can be widely applied.

In [2], a linear SVM classifier with the EMK is compared with a linear SVM classifier and a nonlinear SVM classifier with the Gaussian RBF kernel. For the latter two, the input is the histogram representation obtained with respect to a predefined visual codebook. In the experiment, the number of basis vectors m is 1,000 and 100,000 local feature descriptors are randomly sampled to optimise the matrix $\mathbf{B}$. The results on three benchmark data sets including Scene-15, Caltech-101 and Caltech-256 show that the proposed EMK can help the linear SVM classifier produce better classification performance than the other two SVM classifiers in comparison. Also, it leads to much higher computational efficiency in both training and test stages, especially when the number of training images is large.

5.4.5 *Fisher Kernel*

Fisher kernel [17] provides a way to compare samples induced by a generative model $p(\mathbf{x}|\theta)$. It maps a sample to a feature vector in the gradient space of the model parameters θ. The intuition is that similar samples induce similar log-likelihood gradients of the model parameters. Let $\mathbf{x}_i$ and $\mathbf{x}_j$ denote two samples. Fisher kernel is defined as

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{g}(\mathbf{x}_i)^\top \mathbf{U}^{-1} \mathbf{g}(\mathbf{x}_j), \tag{5.54}$$

where $\mathbf{g}(\mathbf{x}) = \nabla_\theta \log(p(\mathbf{x}|\theta))$ is the gradient vector describing the changing direction of θ to better fit the model. $\mathbf{U}$ is the Fisher information matrix that weights this similarity measure, for example, normalising the dynamic range of the components of the gradient vector [38].

Fisher kernel has recently been successfully applied to image categorisation with the Bag-of-features model [22, 38]. In this application, $\mathbf{x}$ ($\mathbf{x} \in \mathbb{R}^d$) denotes a local feature descriptor extracted from a set of training images. Based on these descriptors, a GMM with k components is learned

$$p(\mathbf{x}|\theta) = \sum_{i=1}^{k} w_i \mathcal{N}(\mathbf{x}|\mu_i, \Sigma_i) \tag{5.55}$$

where w_i is the mixture weight, μ_i the mean vector and Σ_i the covariance matrix of the ith component. These model parameters are compactly represented by $\theta = \{w_i, \mu_i, \Sigma_i\}_{i=1}^k$. All the covariance matrices Σ_i are assumed to be diagonal and expressed as $\Sigma_i = \{\sigma_{i1}^2, \sigma_{i2}^2, \cdots, \sigma_{id}^2\}$. Conceptually, each Gaussian component can be understood as a visual word.

Let $\mathscr{A} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ be a set of local feature descriptors extracted from an image. Let $L(\mathbf{x}|\theta) = \log(p(\mathbf{x}|\theta))$ denote the log-likelihood. The gradient vector $\mathbf{g}(\mathbf{x})$ is a concatenation of the partial derivatives of L with respect to all the parameters, that is, $\mathbf{g}(\mathbf{x}) = (\nabla_{w_i} L(\mathbf{x}|\theta);\ \nabla_{\mu_i} L(\mathbf{x}|\theta);\ \nabla_{\Sigma_i} L(\mathbf{x}|\theta))$. The partial derivatives with respect to the tth local descriptor, $\mathbf{x}_t$, are worked out as:

$$\begin{aligned}
\frac{\partial L(\mathbf{x}_t|\theta)}{\partial w_i} &= \left(\frac{\gamma_t(i)}{w_i} - \frac{\gamma_t(1)}{w_1} \right); \ \text{for } i \geq 2, \\
\frac{\partial L(\mathbf{x}_t|\theta)}{\partial \mu_{ij}} &= \gamma_t(i) \left(\frac{\mathbf{x}_{tj} - \mu_{ij}}{\sigma_{ij}^2} \right), \\
\frac{\partial L(\mathbf{x}_t|\theta)}{\partial \sigma_{ij}} &= \gamma_t(i) \left(\frac{(\mathbf{x}_{tj} - \mu_{ij})^2}{\sigma_{ij}^3} - \frac{1}{\sigma_{ij}} \right),
\end{aligned} \tag{5.56}$$

where $i = 1, \cdots, k$ and $j = 1, \cdots, d$ index the Gaussian components and each component of a vector. The term $\gamma_t(i)$ is defined as $\gamma_t(i) = \frac{w_i \mathcal{N}(\mathbf{x}_t|\mu_i, \sigma_i)}{\sum_{l=1}^{k} w_i \mathcal{N}(\mathbf{x}_t|\mu_i, \sigma_i)}$. By summing the gradient vectors with respect to the n local descriptors, an image-level representation can be obtained as

$$\mathbf{z} = \mathbf{U}^{-\frac{1}{2}} \sum_{t=1}^{n} \mathbf{g}(\mathbf{x}_t), \tag{5.57}$$

where $\mathbf{U}^{-\frac{1}{2}}$ is used to normalise the dynamic range of each dimension of the gradient vector. Let $\mathbf{z}$ and $\mathbf{z}'$ denote the image representation for images I and I'. A linear kernel between two images can be defined as

$$\kappa(I, I') = \langle \mathbf{z}, \mathbf{z}' \rangle = \sum_{p=1}^{n_i} \sum_{q=1}^{n_j} \mathbf{g}(\mathbf{x}_p)^\top \mathbf{U}^{-1} \mathbf{g}(\mathbf{x}'_q), \tag{5.58}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are the local descriptors from images I and I', respectively. It can be seen that this kernel can be regarded as a point-set kernel between the two images. This point-set kernel is in the form of a sum-match kernel, where the local kernel is the Fisher kernel applied to local feature descriptors.

As pointed out in [38], the gradient representation of Fisher kernel on GMM can be related to the Bag-of-features model in image categorisation. The histogram representation of the Bag-of-features model considers only the number of occurrences of each visual word, which corresponds to the zeroth-order statistics. In contrast, the Fisher kernel additionally considers the first- and second-order statistics. This produces a higher-dimensional image representation even when the number of visual words is small, which could be helpful for classification. Moreover, for this representation a linear kernel has been able to perform well, avoiding the use of costly nonlinear kernels.

The work in [38] verifies the advantage of the Fisher-kernel-induced image representation on two databases, an in-house database and PASCAL VOC2006. The GMM is trained in an unsupervised way by using the local feature descriptors extracted from all images. Linear SVMs and Sparse Logistic Regression (SLR) are investigated. Classification with the Fisher-kernel-induced image representation achieves comparable or even better performance than the results reported in the literature. In addition, the Fisher-kernel-induced image representation can bring computational advantages because it can achieve excellent classification performance with small-sized visual codebooks.

5.5 Conclusion

Kernel is the soul of the SVM classifiers and the place where the prior knowledge of an application is accommodated. The performance of SVM classifiers in an application largely depends on the appropriateness and efficiency of the employed

kernels. Without exception, this is also true for the application of SVMs to image categorisation with the Bag-of-features model. This chapter takes a unique perspective to review the development of kernels in this application. Focused on two typical image representations, histogram and point set, the chapter introduces the representative kernels used by the SVM classifiers for each of them. Also, the progress of the use of kernels for each image representation has been briefly shown. For the histogram representation, we can see the trend of avoiding explicitly using kernels, with the advent of advanced coding and pooling techniques as well as powerful kernel approximation methods. This trend has also been seen in the point-set-based representation through the development of EMK. The driving force of these changes is just the applications of SVMs, which need efficient image classification methods to handle large-scale tasks, reduce system complexity and improve recognition performance. We can expect that novel kernels and the novel ways of using kernels will continue emerging with the applications of SVMs to image recognition.

References

1. Barla, A., Odone, F., Verri, A.: Histogram intersection kernel for image classification. In: ICIP, vol. 3, pp. 513–516 (2003)
2. Bo, L., Sminchisescu, C.: Efficient match kernel between sets of features for visual recognition. In: Neutral Information Proceeding Systems, pp. 135–143 (2009)
3. Boughorbel, S., Tarel, J.-P., Fleuret, F.: Non-mercer kernels for svm object recognition. In: BMVC, pp. 1–10 (2004)
4. Boureau, Y., Ponce, J., LeCun, Y.: A theoretical analysis of feature pooling in vision algorithms. In: Proceedings of International Conference on Machine learning (ICML'10), pp. 111–118 (2010)
5. Boureau, Y.-L., Bach, F., LeCun, Y., Ponce, J.: Learning mid-level features for recognition.In: Computer Vision Pattern Recognition, pp. 2559–2566 (2010)
6. Boureau, Y.-L., Bach, F., LeCun, Y., Ponce, J.: Learning mid-level features for recognition. In: Computer Vision Pattern Recognition, pp. 2559–2566 (2010)
7. Chapelle, O., Haffner, P., Vapnik, V.: Support vector machines for histogram-based image classification. IEEE Trans. Neural Netw. **10**(5), 1055–1064 (1999)
8. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: ECCV, pp. 1–22 (2004)
9. Cuturi, M., Vert, J.-P.: Semigroup kernels on finite sets. In: Neutral Information Proceeding Systems, vol. 17, pp. 329–336 (2004)
10. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: Computer Vision Pattern Recognition, pp. 248–255 (2009)
11. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL visual object classes challenge (VOC2007) results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html (2007)
12. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. Comput. Vis. Image Und. **106**(1), 59–70, (2007)
13. Fowlkes, C., Belongie, S., Chung, F.R.K., Malik, J.: Spectral grouping using the nyström method. IEEE T. Pattern Anal. Mach. Intell. **26**(2), 214–225, (2004)

14. Grauman, K., Darrell, T.: The pyramid match kernel: discriminative classification with sets of image features. In: International Conference on Computer Vision, vol. 2, pp. 1458–1465 (2005)
15. Griffin, G., Holub, A., Perona, P.: Caltech-256 Object Category Dataset. Tech. Report, California Institute of Technology (2007)
16. Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S.S., Sundararajan, S.: A dual coordinate descent method for large-scale linear svm. In: Cohen, W.W., McCallum, A., Roweis, S.T. (eds.) ICML, vol. 307 of ACM International Conference Proceeding Series, pp. 408–415. ACM (2008)
17. Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. In: Neutral Information Proceeding Systems, pp. 487–493 (1998)
18. Jégou, H., Douze, M., Schmid, C.: On the burstiness of visual elements. In: Computer Vision and Pattern Recognition 2009, pp. 1169–1176 (2009)
19. Joachims, T.: A statistical learning model of text classification for support vector machines. In: SIGIR, pp. 128–136 (2001)
20. Juriem, F., Triggs, B.: Creating efficient codebooks for visual recognition. In: International Conference on Computer Vision, vol. 1, pp. 604–610 (2005)
21. Kondor, R., Jebara, T.: A kernel between sets of vectors. In: ICML, pp. 361–368 (2003)
22. Krapac, J., Verbeek, J., Jurie, F.: Modeling spatial layout with fisher vectors for image categorization. In: International Conference on Computer Vision, pp. 1487–1494 (2011)
23. Lazebnik, S., Raginsky, M.: Supervised learning of quantizer codebooks by information loss minimization. IEEE T. Pattern Anal. Mach. Intell. **31**(7), 1294–1309 (2009)
24. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: Computer Vision and Pattern Recognition, vol. 2, pp. 2169–2178 (2006)
25. Leung, T.K., Malik, J.: Representing and recognizing the visual appearance of materials using three-dimensional textons. Int. J. Comput. Vision **43**(1), 29–44 (2001)
26. Liu, L., Wang, L., Liu, X.: In defense of soft-assignment coding. In: International Conference on Computer Vision, pp. 2486–2493 (2011)
27. Liu, L., Wang, L., Shen, C.: A generalized probabilistic framework for compact codebook creation. In: Computer Vision and Pattern Recognition, pp. 1537–1544 (2011)
28. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision, pp. 1150–1157 (1999)
29. Lyu, S.: Mercer kernels for object recognition with local features. In: Computer Vision and Pattern Recognition, vol. 2, pp. 223–229 (2005)
30. Madsen, R.E., Kauchak, D., Elkan, C.: Modeling word burstiness using the dirichlet distribution. In: International Conference on Machine learning, pp. 545–552 (2005)
31. Maji, S., Berg, A.C., Malik, J.: Classification using intersection kernel support vector machines is efficient. In: 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), Anchorage, 24–26 June 2008, pp. 1–8. IEEE Computer Society (2008). http://dx.doi.org/10.1109/CVPR.2008.4587630
32. Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. Int. J. Comput. Vision **60**(1), 63–86 (2004)
33. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Gool, L.V.: A comparison of affine region detectors. Int. J. Comput. Vision **65**(1–2), 43–72 (2005)
34. Moosmann, F., Triggs, B., Jurie, F.: Fast discriminative visual codebooks using randomised clustering forests. In: Neutral Information Proceeding Systems, pp. 985–992 (2006)
35. Moreno, P.J., Ho, P., Vasconcelos, N.: A kullback-leibler divergence based kernel for svm classification in multimedia applications. In: Neutral Information Proceeding Systems (2003)
36. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: Computer Vision and Pattern Recognition, vol. 2, pp. 2161–2168 (2006)
37. Parsana, M., Bhattacharya, S., Bhattacharyya, C., Ramakrishnan, K.R.: Kernels on attributed pointsets with applications. In: Platt et al.(eds.) Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3–6, 2007. Curran Associates (2008)

38. Perronnin, F., Dance C.R.: Fisher kernels on visual vocabularies for image categorization. In: Computer Vision and Pattern Recognition, pp. 1–8 (2007)
39. Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: ECCV, vol. 4, pp. 143–156 (2010)
40. Perronnin, F., Sánchez, J., Liu, Y.: Large-scale image categorization with explicit data embedding. In: The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010), San Francisco, 13–18 June 2010, pp. 2297–2304. IEEE (2010). http://dx.doi.org/10.1109/CVPR.2010.5539914
41. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: Computer Vision and Pattern Recognition (2007)
42. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: Platt et al. (eds.) Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3–6, 2007. Curran Associates (2008)
43. Rubner, Y. Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. Int. J. Comput. Vision **40**(2), 99–121 (2000)
44. Schiele, B., Crowley, J.L.: Object recognition using multidimensional receptive field histograms. In: ECCV, vol. 1, pp. 610–619 (1996)
45. Shalev-Shwartz, S., Singer, Y., Srebro, N.: Pegasos: primal estimated sub-gradient solver for svm. In: ICML (2007)
46. Shashua, A., Hazan, T.: Algebraic set kernels with application to inference over local image representations. In: Neutral Information Proceeding Systems, pp. 1257–1264 (2004)
47. Sivic, J., Zisserman, A.: Video Google: a text retrieval approach to object matching in videos. In: Proceedings of the International Conference on Computer Vision, vol. 2, pp. 1470–1477 (2003)
48. Swain, M.J., Ballard, D.H.: Color indexing. Int. J. Comput. Vision **7**(1), 11–32 (1991)
49. van Gemert, J., Geusebroek, J.-M., Veenman, C.J., Smeulders, A.W.M.: Kernel codebooks for scene categorization. In: ECCV, vol. 3, pp. 696–709 (2008)
50. Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. IEEE Trans. Pattern Anal. Mach. Intell. **34**(3), 480–492 (2012)
51. Wallraven, C., Caputo, B., Graf, A.B.A.: Recognition with local features: the kernel recipe. In: International Conference on Computer Vision, pp. 257–264 (2003)
52. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T.S., Gong, Y.: Locality-constrained linear coding for image classification. In: Computer Vision and Pattern Recognition, pp. 3360–3367 (2010)
53. Wang, L.: Toward a discriminative codebook: codeword selection across multi-resolution. In: Computer Vision and Pattern Recognition, pp. 1–8 (2007)
54. Williams, C., Seeger, M.: Using the nystrm method to speed up kernel machines. In: Neutral Information Proceeding Systems (2001)
55. Winn, J.M.: Criminisi, A.: Minka, T.P.: Object categorization by learned universal visual dictionary. In: International Conference on Computer Vision, pp. 1800–1807 (2005)
56. Wu, J., Rehg, J.M.: Beyond the euclidean distance: creating effective visual codebooks using the histogram intersection kernel. In: International Conference on Computer Vision, pp. 630–637 (2009)
57. Yang, J., Yu, K., Gong, Y., Huang, T.S.: Linear spatial pyramid matching using sparse coding for image classification. In: Computer Vision and Pattern Recognition, pp. 1794–1801 (2009)
58. Zhang, J. Marszalek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: a comprehensive study. Int. J. Comput. Vision **73**(2), 213–238 (2007)