Part III
Image Classification and Annotation

Dripping water penetrates the stone

III.1 Introduction

Due to the rapid digitisation and development of the Web, the world is full of digital images. However, without proper classification, these mammoth amount of images are not going to be much helpful, instead, it has caused a huge waste of resources.

Vast amount of research has been done in the past decades to organize digital images into categories so that they can be searched and retrieved conveniently. However, despite the tremendous effort, we are still at the early stage of understanding images.

Basically, image classification is to organize images into different classes based on the features of the images. Image annotation is to label images with different semantic class names, such as trees, air planes, lake, etc. The difference between image classification and image annotation is that image annotation attempts to annotate an image with multiple labels or classify an image into multiple classes. Image annotation is done through *multiple instance learning* (MIL). With MIL, an image is represented with a *bag of features* (BOF), and an image is labelled as positive if any of the instances in the bag is positive. Image classification and annotation are close related, because if an image is correctly classified, it can be annotated and if an image is correctly annotated, it can be properly classified into a class.

Given an image as in Fig. III.1, what we want is to classify it into one of the semantic classes, such as, 'mountain' or 'plants' or 'nature', with a probability or likelihood.

Fig. III.1 An image to be classified into one of the classes

However, what we have is usually a sequence of numeric features computed through certain feature extraction methods described in Part II, such as, a color histogram, or a feature vector (Fig. III.2).

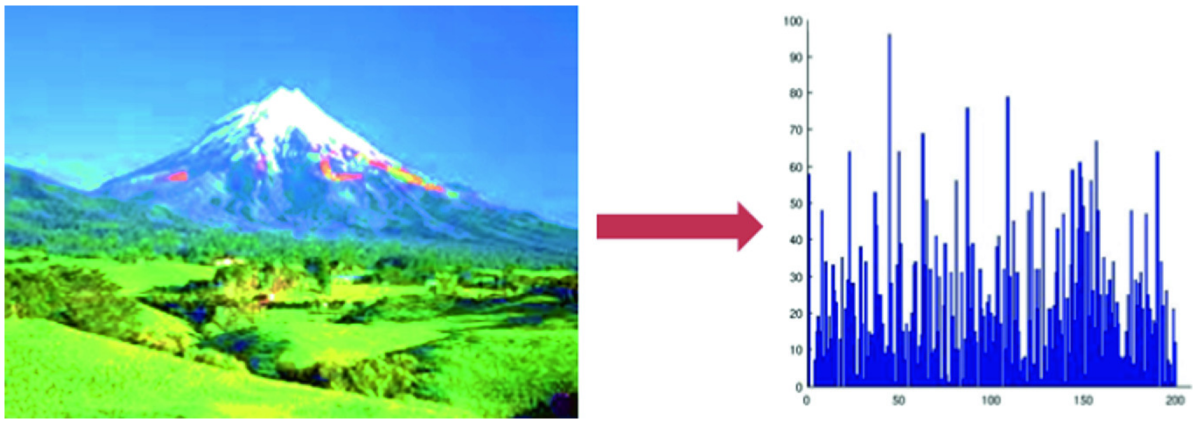

Fig. III.2 An image on the *left* and its color histogram on the *right*

What we can do is to learn from experience or prior knowledge like a human being. Suppose we know the above image is a mountain image, given an unknown image, we can compare its features with the features of the mountain image. If there is a good match (e.g., high probability) between the two feature vectors, we would label or classify the unknown image also as ‘mountain’ (Fig. III.3).

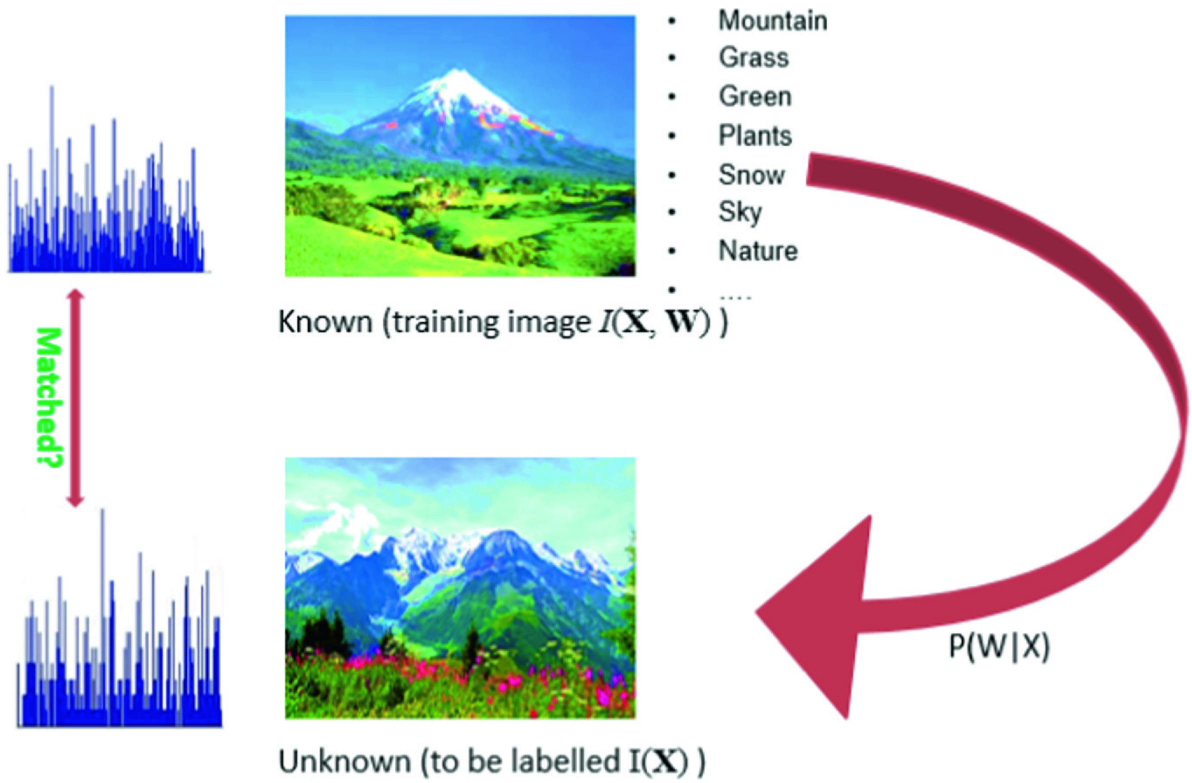

Fig. III.3 Matching between an unknown image with a labelled image

We could use this simple method to identify or retrieve all the mountain images from the database (Fig. III.4). However, this is not going to work well, because the single known image is not a good representation of all the mountain images in the database. Consequently, many mountain images in the database will be misclassified or not retrieved.

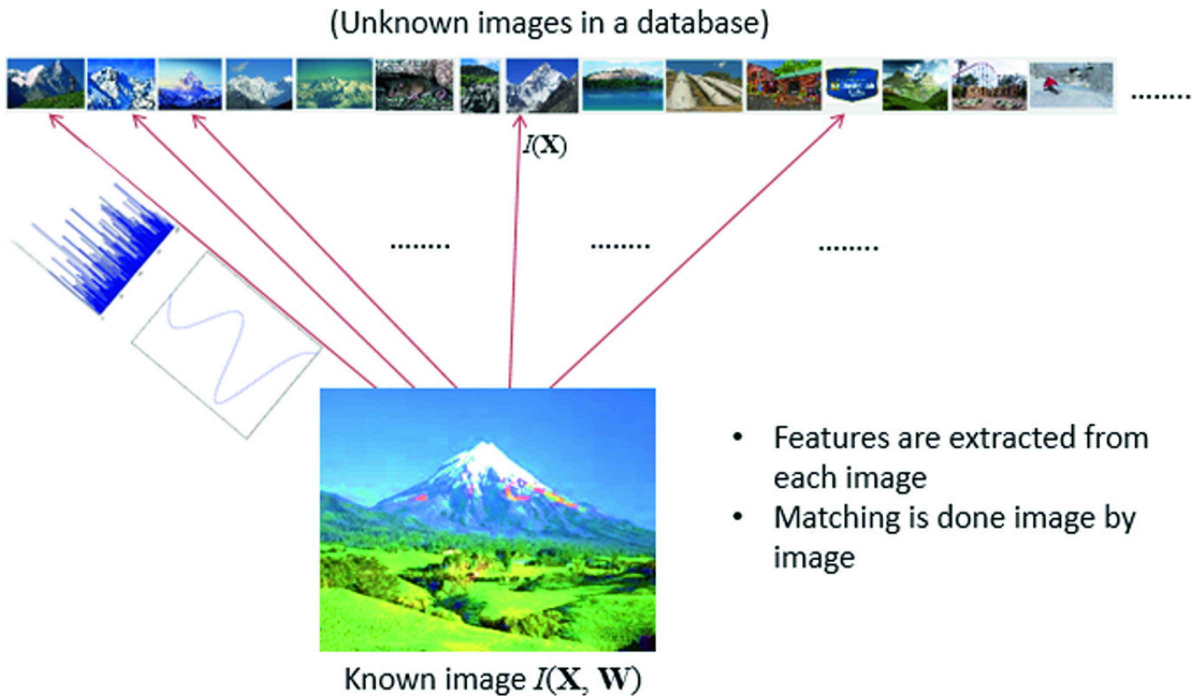

Fig. III.4 Use a labelled image to identify all the mountain images in a database

A much better way to identify all mountain images in a database is to collect large number of sample mountain images and use them to train a classifier. Once trained, the classifier will be able to memorize these sample images and use them to recognize unknown images.

There are generally two types of approaches on training or building a classifier, *generative* versus *discriminative*.

III.1.1 Generative Model

The generative approach is based on an idea similar to Platonic philosophy that there is an abstract concept or model behind every type of objects in this world, such as trees, apples, dogs, human beings etc. It is believed that when people try to recognize a specific type of objects in this world, they actually compare them with this abstract model. Therefore, it is possible to create or work out this abstract model for every type or class of objects. The simplest way to create this kind of model is to collect a set of real world samples and average them, for example, an average apple (Fig. III.5).

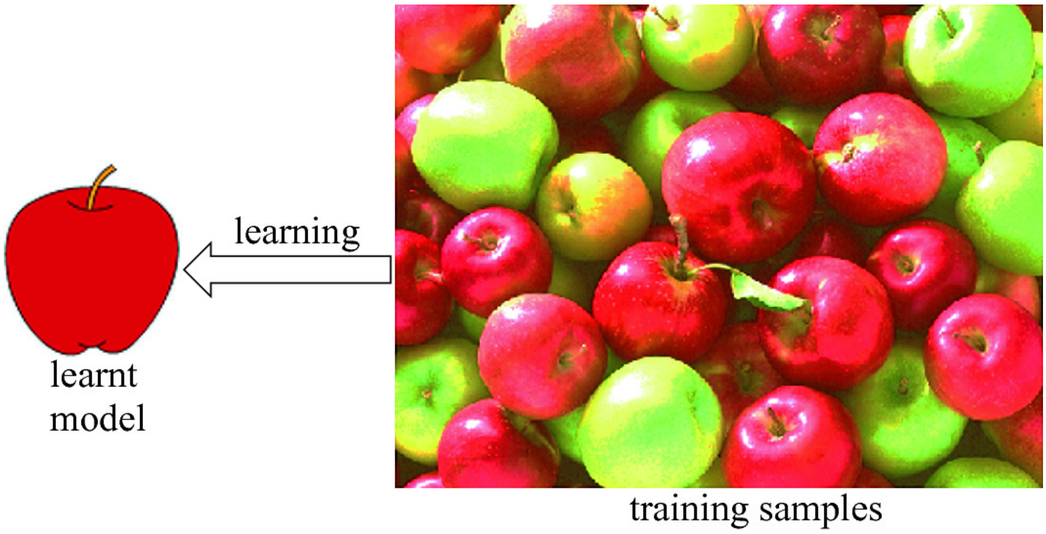

Fig. III.5 Sample apples (*right*) and the learnt apple model (*left*)

However, this can only do simple classification by distinguishing apples from non-fruit objects, while it would be difficult to distinguish apples from other fruits such as peaches, or pears. In practice, a probabilistic distribution is learnt from the collected samples and the distribution is used as the model representing the objects (Fig. III.6). The variety of those probabilistic methods follow the generative approach, including the typical Gaussian mixture model and Bayesian methods.

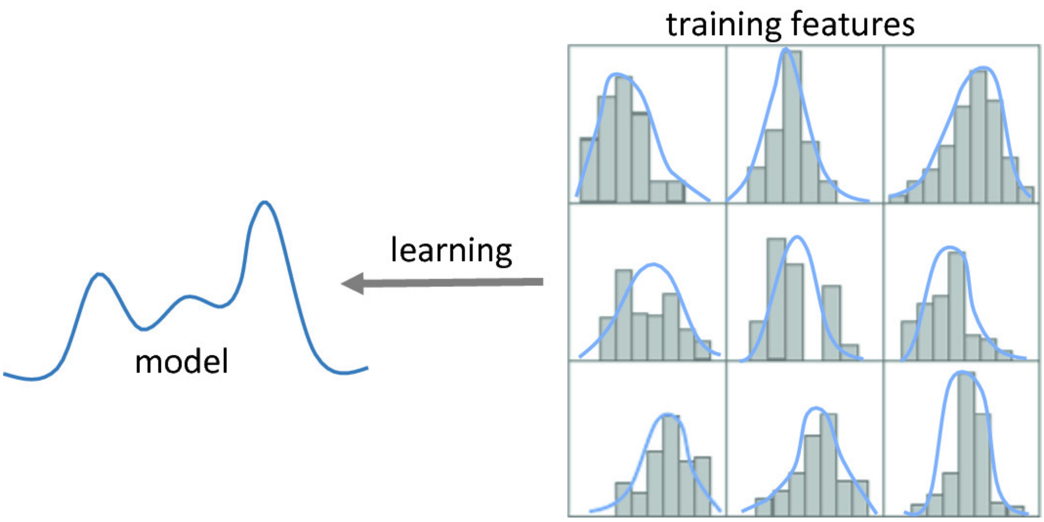

Fig. III.6 Computation of a generative model. Sample images are represented as features (*right*); a mixture distribution model is learnt from those sample features (*left*)

III.1.2 Discriminative Approach

In contrast to the generative approach, the discriminative approach doesn't believe or is unaware of the model behind every type of objects. Instead, discriminative methods do classification by comparing different objects based on their similarity or difference, as opposed to comparing objects with a model in the generative approach. In practice, a large number of sample training data are collected from different classes, and an optimal hyper plane, called the classifier or a machine, is fitted between two classes in a high dimensional feature space (Fig. III.7). The optimal hyper plane is found through a trial and error process by keeping testing the similarity or difference between the training data.

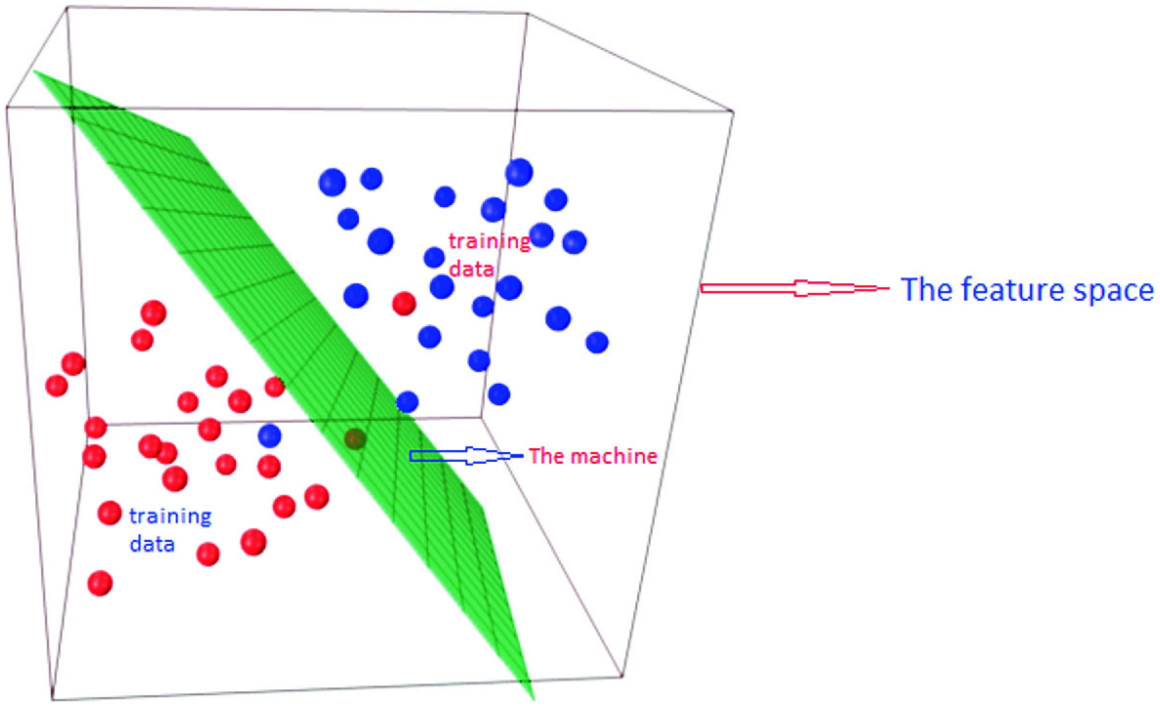

Fig. III.7 A machine is fitted between two classes of data

D. Zhang, *Fundamentals of Image Data Mining*, Texts in Computer Science
https://doi.org/10.1007/978-3-030-69251-3_7

7. Bayesian Classification

Dengsheng Zhang[1]
(1) Federation University Australia, Churchill, VIC, Australia

Dengsheng Zhang
Email: dengsheng.zhang@federation.edu.au

History tells the future.

7.1 Introduction

All Bayesian classification methods are based on the Bayes' theorem:

$$P\left(A|B\right)=\frac{P(B|A)P\left(A\right)}{P\left(B\right)}=\frac{P(B|A)P\left(A\right)}{P(B|A)P\left(A\right)+P(B|\bar{A})P\left(\bar{A}\right)} \tag{7.1}$$

where

- A and B are random events, and $\bar{A}$ is the complement of A.
- A is a hypothesis to be tested or predicted.
- B is the new data or observation, and it is the new evidence to predict A.
- $P(A)$ is called the *prior* probability, and $P(B|A)$ is called the *likelihood*; they represents our experience or prior knowledge.
- $P(B)$ is the observation probability or the chance to observe event B.
- $P(A|B)$ is called the *posterior* probability.

The idea of Bayes' theorem is to convert the computation of probability of $P(A|B)$ to $P(B|A)$ which is easier to compute. This is extremely helpful when two random events A and B are dependent each other, and the prediction of event A is difficult due to lack of evidence or information; in this situation, the information about *B* can be employed to help predicting *A* more accurately. The information about *B* can usually be obtained from historical data or experience.

This idea of predicting one event using other related events have been practiced by human beings all the time; for example, we use symptoms to predict a disease, use cloud to predict rain, use a man's culture/background to predict his behavior, use rainfall to predict harvest, etc. In the following, we use two simple examples to get some firsthand understanding of how Bayesian theorem works in real-world applications.

For the first example, we are planning for a sports event at a weekend in a local club and we want to know if the weather will be fine at the weekend. We know the weather and humidity are highly related; we can use humidity to help us predicting if the weather is fine so that the sports can go ahead. Formally, let

$$R = \text{Rain} \quad \bar{R} = \text{No rain} \quad H = \text{High humidity} \ (\text{humidity} > 80\%)$$

Suppose we know from history (e.g. bureau of meteorology) the following prior information:

$$P(R) = 35\% \quad P(\bar{R}) = 65\% \quad P(H|\bar{R}) = 30\%$$

Suppose further we know there will be high humidity at the weekend based on the most recent day weather, then we can predict the chance of no rain at the weekend by the Bayes' theorem:

$$P\left(\bar{R}|H\right) = \frac{P(H|\bar{R})P\left(\bar{R}\right)}{P(H)} = \frac{P(H|\bar{R})P\left(\bar{R}\right)}{P(H|\bar{R})P\left(\bar{R}\right) + P(H|R)P(R)}$$
$$= \frac{0.3 \times 0.65}{0.3 \times 0.65 + 1 \times 0.65} = \frac{0.195}{0.845} \approx 23\%$$

This information is useful for making a reasonable decision on if the sports event should go ahead. Notice that without the prior information of $P(\bar{R})$ and $P(H|\bar{R})$, the prediction and decision would have been made arbitrarily. In practice, factors such as humidity, temperature and atmospheric pressure are combined to obtain an even more accurate prediction.

Another example is the application of the Bayes' theorem on image classification. Suppose we have detected some black and white strips (through feature extraction) in an image; based on our experience, we believe it's likely a zebra image. But how likely the image is a zebra, 70% of the chance, 80% or 99%? For many other cases like financial, economic or military situations, this likelihood is crucial to make a right decision. It's clear we need more evidence to determine the accurate likelihood. The answer is in statistics.

If we were able to sample all the images in the world just like a population census in a country, we would be able to calculate the statistics and tell how many non-zebra images would have the black and white strips. This statistic would help us to determine how likely the image with black and white strips is a zebra image. Unfortunately, we are not able to do a census on all the images in the world; all we can do is to sample part of the image population and create an image database, then use the statistics calculated from the image database to approximate those in the image population.

Formally, let

$$Z = \text{ Zebra image} \quad \bar{Z} = \text{non} - \text{Zebra image} \quad BWS = \text{Black and white strips}$$

Now, suppose we know the following probabilities from a training set or an image database (our experience or prior information):

$$P(\text{BWS}|Z) = 1.0 \quad P(Z) = 0.05 \quad P(\text{BWS}|\bar{Z}) = 0.01$$

Then, given the black and white strips in an image, we can predict if there is a zebra in the image by the Bayes' theorem:

$$P(Z|\text{BWS}) = \frac{P(\text{BWS}|Z)P(Z)}{P(\text{BWS})} = \frac{P(\text{BWS}|Z)P(Z)}{P(\text{BWS}|Z)\,P(Z) + P(\text{BWS}|\bar{Z})P\left(\bar{Z}\right)}$$

$$= \frac{1.0 \times 0.05}{1.0 \times 0.05 + 0.01 \times 0.95} = \frac{0.05}{0.0595} \approx 84\%$$

Therefore, we can say there is a high chance that the image is a zebra image and we have high confidence to classify the image into the zebra image category.

Bayes' theorem can be extended to multiple events $A_1, A_2, ..., A_n$:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)} = \frac{P(B|A_i)P(A_i)}{P(B|A_1)P(A_1) + \ldots + P(B|A_n)P(A_n)} \quad (7.2)$$

In this case, B is related to or dependant on multiple other events A_i, and based on what we know about the relationship or dependency between B and each A_i: $P(B|A_i)$, we can predict if a new observation of B is from any of the events A_i.

For example, fever (B) can be caused by many diseases or medical conditions (A_i), such as infection, flu, pneumonia, chickenpox, measles, HIV, meningitis, cancers, malaria and dengue. However, each disease or medical condition has different chances of causing fever: $P(B|A_i)$. Now, given a patient with fever, (7.2) can be used to determine if the patient has flu: $P(A_i|B)$. In clinical practice, however, symptoms are typically combined to nail a disease; e.g. by combining fever with running nose and headache, flu can be diagnosed with very high accuracy.

7.2 Naïve Bayesian Image Classification

7.2.1 NB Formulation

The *naïve Bayesian* (NB) methods are based on a simple application of the above Bayes' theorem on numerical and high-dimensional image data.

- Given a set of N images: $I = \{I_1, I_2, ..., I_N\}$
- And a set of n semantic classes $C = \{C_1, C_2, ..., C_n\}$ (events).
- $I \in C$.
- Each image I is represented by a feature vector $I \sim \mathbf{x} = (x_1, x_2, ..., x_m)$ (observation).

According to Bayes' theorem, the classification or annotation of image I to class C_i is given by

$$P(C_i|I) = P(C_i|\mathbf{x}) = \frac{P(\mathbf{x}|C_i)P(C_i)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|C_i)P(C_i)}{P(\mathbf{x}|C_1)P(C_1) + \ldots + P(\mathbf{x}|C_n)P(C_n)} \quad (7.3)$$

Or

$$P(C_i|I) = P(C_i|\mathbf{x}) = \frac{P(\mathbf{x}|C_i)P(C_i)}{\sum_{k=1}^{n} P(\mathbf{x}|C_k)P(C_k)} \tag{7.4}$$

Because the denominator $P(\mathbf{x}) = \sum_{k=1}^{n} P(\mathbf{x}|C_k) \times P(C_k)$ is independent of class C_i ($i = 1, 2, \ldots, n$) and is a constant, (7.4) can be written as:

$$P(C_i|I) = P(C_i|\mathbf{x}) = \frac{1}{Z} P(\mathbf{x}|C_i) P(C_i) \tag{7.5}$$

where $Z = \sum_{k=1}^{n} P(\mathbf{x}|C_k)P(C_k)$ is a scaling factor. The class of image I can be decided using the *maximizing a posterior* (*MAP*) criterion

$$P\left(C_j|I\right) = \hat{C} = \arg\max_i P(C_i|\mathbf{x}) = \arg\max_i \{P(\mathbf{x}|C_i)P(C_i)\} \tag{7.6}$$

The prior probabilities $P(C_i)$ is usually uniform for all classes; otherwise, they can be found by the frequency or proportion of samples belonging to class C_i among all classes. Therefore, the classification of image I comes down to modeling the likelihood probability of $P(\mathbf{x}/C_i)$.

Since image features are typically numerical and continuous, they need to be discretized before the likelihood modeling. In practice, the following procedure is used to compute the $P(\mathbf{x}|C_i)$ in (7.6).

Training:

- A training database of images from all the n classes $\{C_1, C_2, \ldots, C_n\}$ are created.
- Image features from the training database are clustered into m clusters X_j using a certain vector quantization algorithm.
- Next, a cluster centroid $\mathbf{x}_j$ is computed for each of the clusters $X_{j:}$ $\mathbf{x}_j$, $j = 1, 2, \ldots, m$.
- Then, the likelihood $P(\mathbf{x}_j|C_i)$ is calculated by finding the frequency of samples in X_j belonging to class C_i.

$$P\left(\mathbf{x}_j|C_j\right) = \frac{\text{No. of samples in } X_j \text{ which are from class } C_i}{\text{Total no. of samples in cluster } X_j} \tag{7.7}$$

Classification/Annotation:

- Given a new image I with feature $\mathbf{x}$.
- Match feature $\mathbf{x}$ to the closest cluster centroids $\mathbf{x}_j$*'s*.
- Apply the *MAP* of (7.6) by replacing the likelihood $P(\mathbf{x}|C_i)$ with (7.7) to obtain the posterior probability $P(C_j \mid I)$.

The classification and annotation of an image with naïve Bayesian method is illustrated in Fig. 7.1 [1, 2]. There are two major modules in a NB classifier: *training* and *annotation*, and each of the major modules consists of three submodules. The training module consists of *feature extraction*, *clustering* and *model building*; while the annotation consists of *feature extraction*, *matching* and *decision making* using MAP.

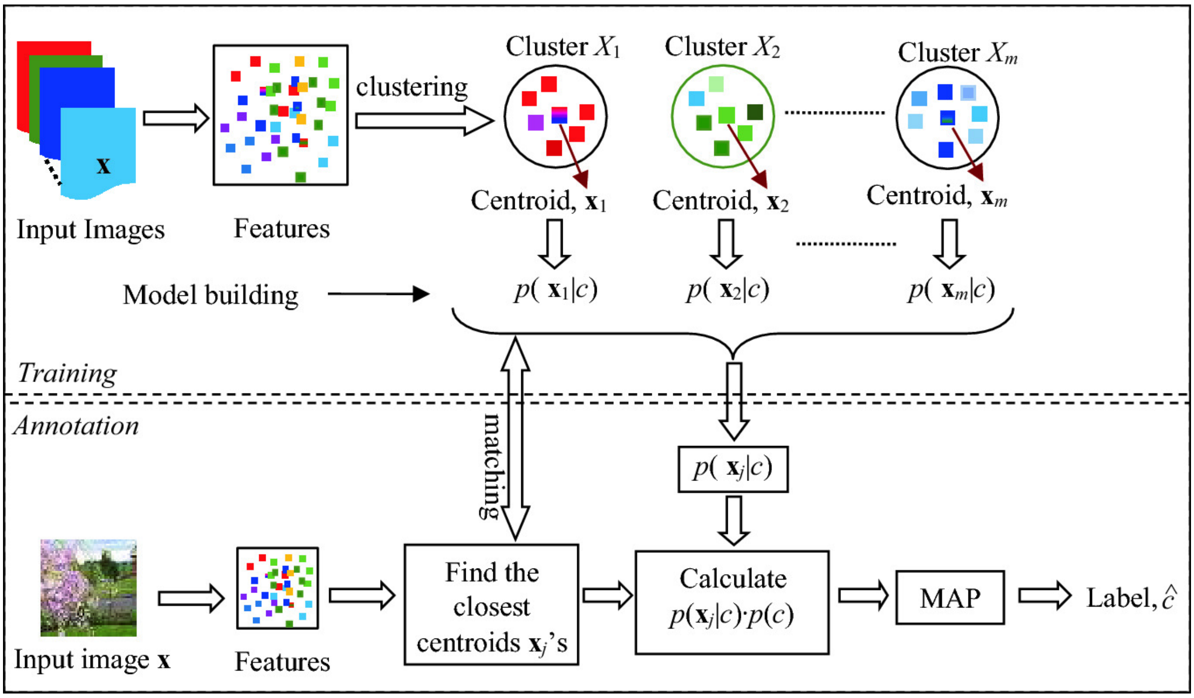

Fig. 7.1 Image classification with naïve Bayesian method

7.2.2 NB with Independent Features

Assume $x_1, x_2, \ldots, x_m$ are independent each other; then the likelihood is given as:

$$P(\mathbf{x}|C_i) = \prod_{j=1}^{m} P(x_j|C_i) \tag{7.8}$$

There are many situations where the features of a data are independent each other, e.g., nominal features extracted by a web crawler, tree, grass, sand and water. Suppose we are using a set of nominal features $\mathbf{x}$ = (*sand*, *water*, *sky*, *people*) to classify a collection of images into 'beach' and 'non-beach' categories; then, (7.8) can be employed to modeling the likelihood in the Bayes' theorem. Often different types of numerical image features are combined into a more powerful feature vector, e.g. $\mathbf{x}$ = (*color*, *shape*, *texture*); again, the likelihood probability in the Bayes' theorem can be computed using (7.8).

7.2.3 NB with Bag of Features

If an image I is segmented into k regions, and each region is represented as a feature vector $\mathbf{x}_j$, $j = 1, 2, \ldots, k$, I can be represented as a *bag of features*: $I = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$. Typically, regions in an image are independent each other; therefore, the conditional probability of $P(I \mid C_i)$ is given by:

$$P(I|C_i) = P(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k|C_i) = \prod_{j=1}^{k} P(\mathbf{x}_j|C_i) \tag{7.9}$$

7.3 Image Annotation with Word Co-occurrence

In the above naïve Bayesian classification, images are not individually labeled; instead, they are simply classified into categories. The categories can be regarded as *implicit image annotation* or collective image annotation. However, individual images can be pre-labeled and the annotation of images can be done explicitly. Vast amount of labeled images are available on the web; they can be employed to annotate new images. One of the earliest works on *explicit image annotation* or individual image annotation is the *word co-occurrence model* (WCC) introduced by Mori et al. [3]. The idea is to establish the relationship between image features and the labels and use the relationship as a likelihood model to label new images. Specifically, features from pre-labeled image are clustered into clusters and a *word histogram* is computed from each cluster as the likelihood model. The idea of their method can be summarized as follows:

1. Collecting training images with pre-labeled key words.

2. Divide each image into parts and extract features from each part.

3. Each divided part inherits all words from its original image.

4. Make clusters from all divided images using vector quantization.

5. Accumulate the frequencies of words of all partial images in each cluster and calculate the likelihood for every word.

6. For an unknown image, divide it into parts, extract their features and match the image parts with the above clusters. Combine the likelihoods of the image parts and determine which words are most plausible.

The *algorithm* of the word co-occurrence model is given as following:

- **Collect and label training images**. Given a training dataset of n images $\mathbf{I} = (I_1, I_2,, ..., I_n)$ and each image I_i is pre-labeled with a set of semantic words $\mathbf{w}_i$:

$$(\mathbf{I}, \mathbf{w}) = \{(I_1, \mathbf{w}_1), (I_2, \mathbf{w}_2), \ldots, (I_n, \mathbf{w}_n)\}$$

- **Obtain the vocabulary of the training images**. The semantic vocabulary of the dataset consists of m words:

$$\mathbf{w} = (w_1, w_2, \ldots, w_m)$$

- **Divide training images into blocks**. Each training image is divided into small blocks, and each block inherits all the annotations from its parent image.
- **Vector Quantization (VQ)**.
  - Blocks from all the training images are clustered into v clusters represented by the centroids c_1, c_2, ..., c_v.
  - Each cluster c_i is represented as a feature vector $\mathbf{x}_i$ (each cluster is called a *visual word* or VW, which is corresponding to a region in the training images):

$$\mathbf{c} = (c_1, c_2, \ldots, c_v) = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_v)$$

- **Obtain a word histogram in each cluster**. Because each block has inherited a set of words from its parent image, by counting the occurrence of words, a histogram of words from the vocabulary can be created:

$$P(w_j|c_i) = P\left(w_j|\mathbf{x}_i\right) = (W_1, W_2, \ldots, W_m) \tag{7.10}$$

  where W_j represents the frequency of word w_j in cluster c_i, $P(W_j|c_i)$ represents the likelihood of word w_j.
- **Annotate an unknown image**
  - Given an unknown image I_u, it is also divided into small blocks and the blocks of the unknown image is also clustered into clusters.
  - Each unknown cluster is matched with the VWs, and the nearest l VWs are found for the unknown image.
  - The matching is done by calculating the distance between each feature of the unknown image $\mathbf{x}_u$ and each VW $\mathbf{x}_i$: $||\mathbf{x}_u - \mathbf{x}_i||$

– The annotation of image I_u to a semantic word w_j (j = 1, 2, ..., k) is given by first summing up the histograms of matched clusters c_i (i = 1, ..., l) and then selecting the top k bins as the annotations:

$$P\,(w_1, \ldots, w_k|I_u) = \text{top}\,k\,\text{bins}\,(\sum_1^l P(w_j|c_i)) \tag{7.11}$$

The co-occurrence annotation method can be illustrated in Fig. 7.2 [1, 2]. There are two key differences between Figs. 7.2 and 7.1. The first is in the training module; while the NB builds a model of $p(\mathbf{x}_i|c)$, the WCC builds a model of $p(w|c)$. The second difference in the annotation module; while the NB makes a decision based on MAP, the WCC makes a decision based on top histogram bins which means an image can be classified into several classes.

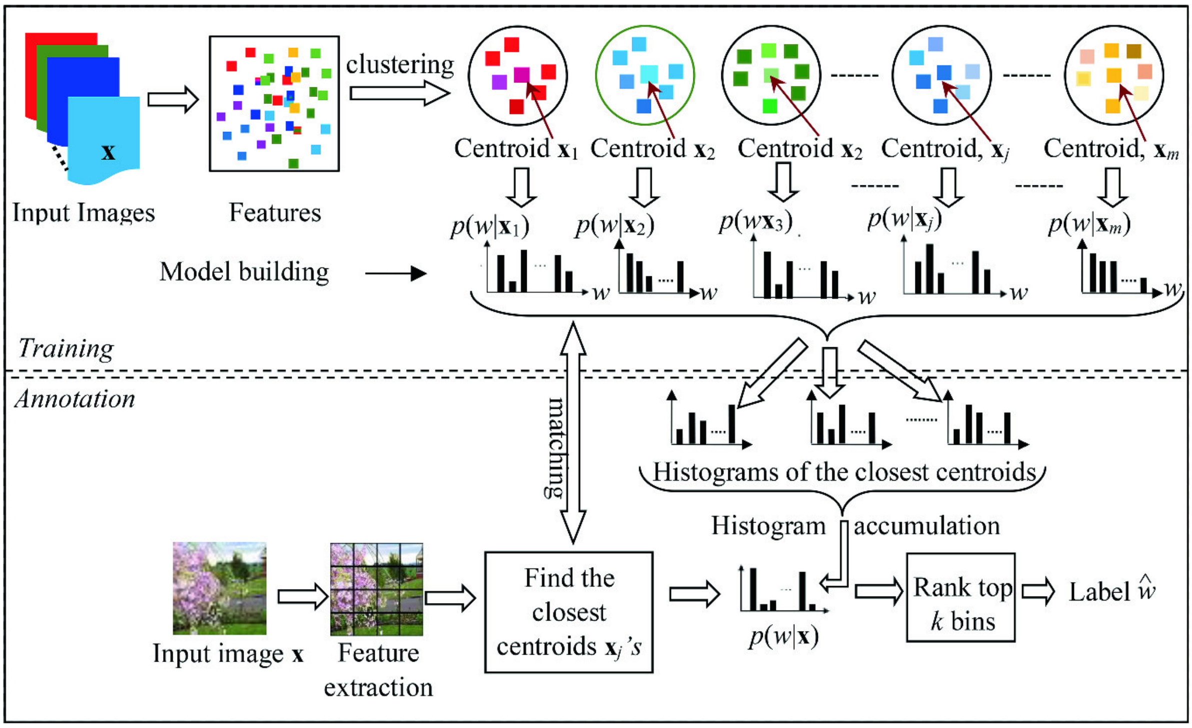

Fig. 7.2 Image annotation with co-occurrence of words

Although the co-occurrence method uses image blocks for the VQ, the blocks can be replaced with pre-segmented image regions. This is because regardless blocks or regions, they are all represented with a feature vector **x**, and the VQ is done based on feature vector **x**.

7.4 Image Annotation with Joint Probability

The word co-occurrence model is a significant development to traditional image classification; it can be generalized into a joint probability model which is described in the following.

- Given a training dataset of n pre-annotated images:

$$(\mathbf{I}, \mathbf{w}) = (I_1, \mathbf{w}_1), (I_2, \mathbf{w}_2), \ldots, (I_n, \mathbf{w}_n)$$

- The semantic vocabulary of the dataset consists of m words:

$$\mathbf{w} = (w_1, w_2, \ldots, w_m)$$

- The annotation or association of an unknown image I to a word w in the vocabulary can be found by the joint probability of $P(w, I)$ or $P(w, \mathbf{x})$, where $\mathbf{x}$ is the feature of I.
- In order to compute $P(w, I)$, a latent variable c is introduced

$$P(w|I) = P(w, I) = P(w|c) \times P(c|I) \tag{7.12}$$

The computation of conditional probabilities $P(w \mid c)$ and $P(c \mid I)$ is given in the following procedure:

- The training images are clustered into v clusters or VWs (*the latent variables*):

$$\mathbf{c} = (c_1, c_2, \ldots, c_v) = (\mathbf{x}_1, \mathbf{x}_2, \ \ldots, \mathbf{x}_v)$$

- An image to be annotated is represented as a histogram or distribution of VWs:

$$P(c_i|I) = (X_1, X_2, \ldots, X_v) \tag{7.13}$$

  where X_i is the frequency of VW $\mathbf{x}_i$ in image I.

- Each cluster is represented as a histogram or distribution of vocabulary words:

$$P\left(w_j|c_i\right) = (W_1, W_2, \ldots, W_m) \tag{7.14}$$

where W_j represents the frequency of word w_j in cluster c_i,

- Finally, the annotation of image I to a word w_j is given by

$$P\left(w_j|I\right) = P\left(w_j|c_i\right) \times P\left(c_i|I\right) \tag{7.15}$$

As discussed in Sect. 7.2, images can be segmented into regions and represented as a *bag of features* for annotation.

- If an image I is represented as a *bag of features* (pre-clustered): $I = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$
- The conditional probability of $P(c_i|I)$ in (7.15) can be computed using *MAP*:

$$P\left(c_i|I\right) = \arg\max_{c_j} P(c_j|I) = \arg\max_{c_j}\{P(I|c_j) \times P(c_j)\} \tag{7.16}$$

where

$$P\left(I|c_j\right) = P\left(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k|c_j\right) = \prod_{l=1}^{k} P(\mathbf{x}_l|c_j) \tag{7.17}$$

The key idea of image annotation based on the joint probability model is the association of semantic words with visual words. This is achieved through the VQ process. Once image features are clustered, each cluster (VW) and the semantic words are bound together or associated because each image feature inherits the semantic word(s) from its parent image. This idea can be illustrated in Fig. 7.3.

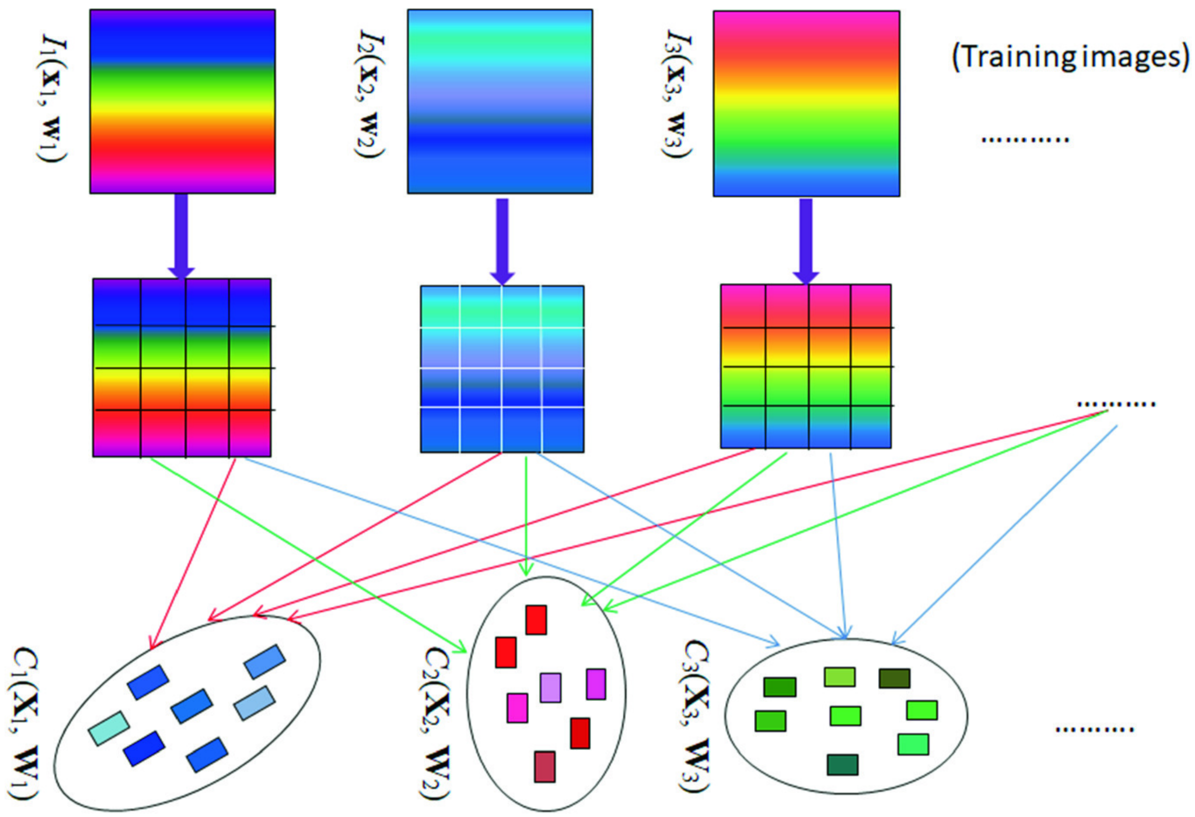

Fig. 7.3 Association of semantic words with block features

Once image features are clustered and visual words are generated, two types of distribution can be created from each cluster: $P(\mathbf{x}|c_i)$ and $P(w|c_i)$. $P(w|c_i)$ is the word distribution in cluster c_i, and it connects the VWs to the semantic vocabulary. $P(\mathbf{x}|c_i)$ is the feature distribution in cluster c_i, and it connects each VW with each of the images in the database (Fig. 7.4). By combining these two important information, new image can be annotated as shown in the above sections.

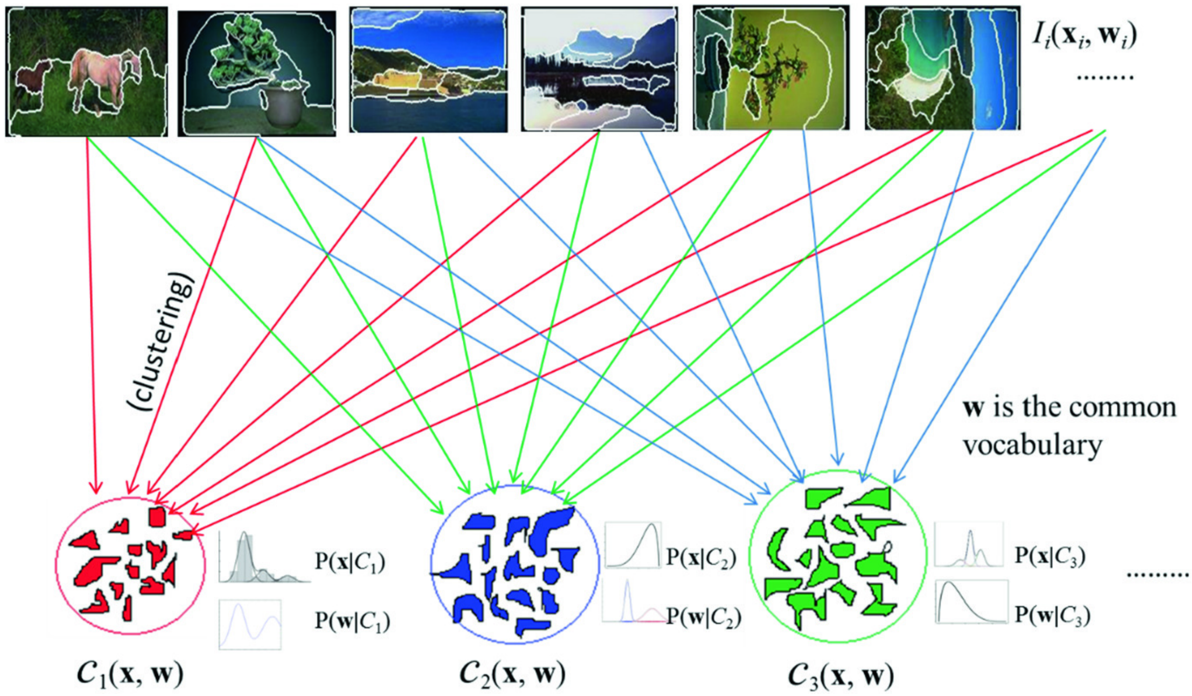

Fig. 7.4 Association of semantic words with region features

7.5 Cross-Media Relevance Model

Although VQ is typical in building the likelihood models, models can also be built 'on the fly' by a set of training images which are relevant to the new observation. The *cross-media relevance model* (CMRM) [4] provides an alternative method to the VQ and is another joint probability model.

Given an image which is represented by a set of blobs: $I = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, the association of I with concept c is given by the joint probability of $p(c, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$:

$$\begin{aligned} p(c, \mathbf{x}_1, \cdots \mathbf{x}_m) &= \sum_{J \in T} p(J) \times p(c, \mathbf{x}_1, \cdots \mathbf{x}_m | J) \\ &= p(J) \times \sum_{J \in T} p(c|J) \times \prod_{i=1}^{m} p(\mathbf{x}_i | J) \end{aligned} \tag{7.18}$$

where

$$p(c|J) = (1 - \alpha_J) \times \frac{\#(c, J)}{|J|} + \alpha_J \times \frac{\#(c, T)}{|T|} \tag{7.19}$$

$$p(\mathbf{x}_i|J) = (1 - \beta_J) \times \frac{\#(\mathbf{x}_i, J)}{|J|} + \beta_J \times \frac{\#(\mathbf{x}_i, T)}{|T|} \tag{7.20}$$

where

- J is an image in the training set T.
- α_J and β_J are the interpolation parameters.
- $\#(c, J)$ is the number of times concept c appears in J.
- $\#(\mathbf{x}_i, J)$ is the number of times blob $\mathbf{x}_i$ appears in J.
- $\#(c, T)$ is the number of times concept c appears in T
- $\#(\mathbf{x}_i, T)$ is the number of times blob $\mathbf{x}_i$ appears in T.

It can be seen from (7.18), given a new observation $I = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, CMRM attempts to find all the relevant images in the training set that have both concept c ($p(c|J) \neq 0$) and feature $\mathbf{x}_i$ ($p(\mathbf{x}_i|J) \neq 0$). A joint probability model is built by aggregating all the models from the relevant images. This is equivalent to build a class model for concept c 'on the fly'. From (7.19) and (7.20), it can be seen that the performance of this model depends on the choice of the weight α_J and β_J. In practice, this can be a difficult decision to make.

7.6 Image Annotation with Parametric Model

One of the classic ways of model building is the parametric method using the *expectation-maximization* (EM) algorithm. The idea of image annotation with parametric model is similar to the CMRM method, that is, to build a model for each of the individual images and aggregate the similar individual models into a class model. However, instead of combining training and annotation into a single process as in the CMRM method, parametric model separates training and annotation into two different processes.

During the training, images in the training set are pre-labeled and pre-classified into different classes. A class model is then built by aggregating individual image models in each class. During the annotation, the model of the new image is built and matched with the class models and the closest classes are selected as the annotations. The procedure of parametric method is shown in Fig. 7.5. The algorithm of this method is as the following:

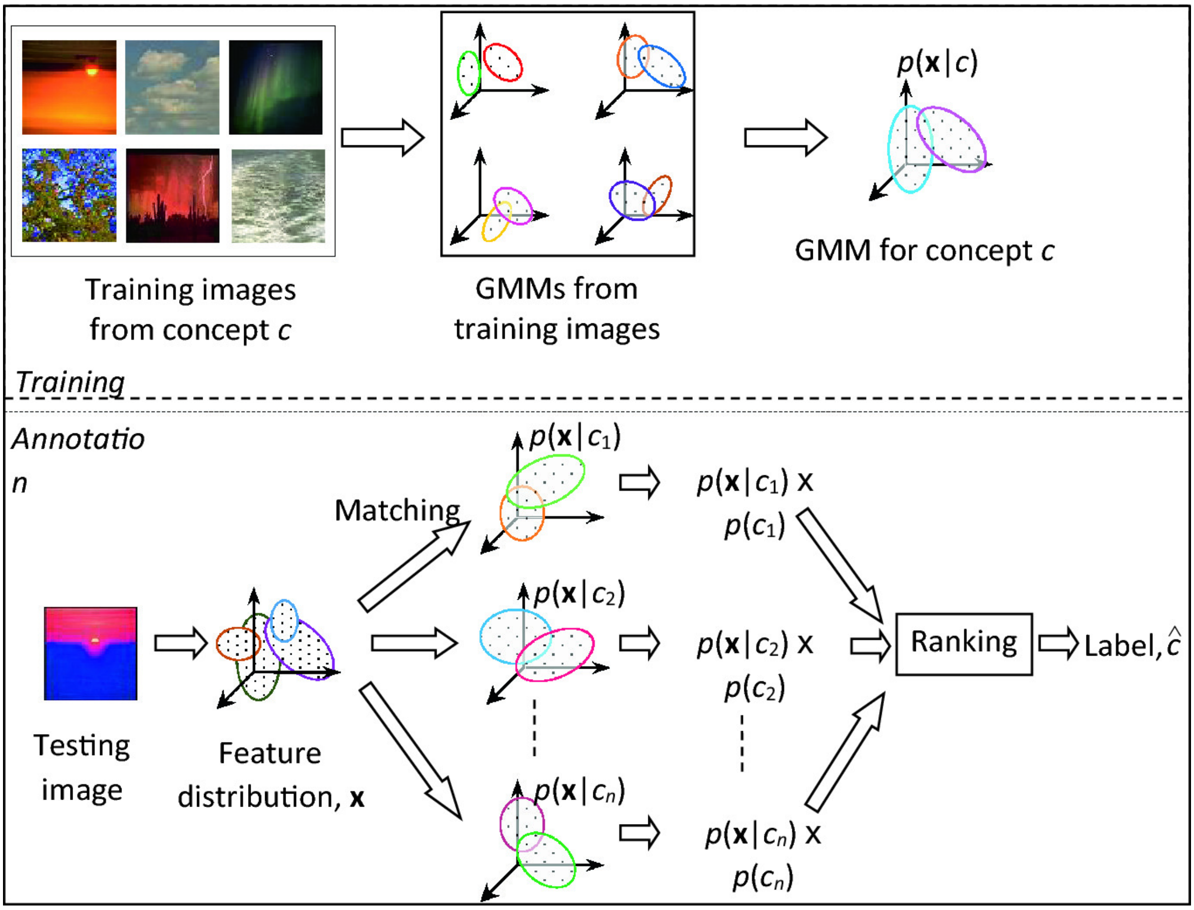

Fig. 7.5 Image annotation with parametric model

- Given a set of N training images: $I_1, I_2, \ldots, I_N$ and a set of n classes $C_1, C_2, \ldots, C_n$
- Features (e.g., block features) from each training image I are clustered within the image
- A Gaussian mixture model (GMM) is learned from the clustering using the EM algorithm:

$$P(\mathbf{x}|I) = \sum_{i=1}^{l} \pi_I^i \mathcal{G}\left(\mathbf{x}, \mu_I^i, \Sigma_I^i\right) \tag{7.21}$$

where

- l is the number of components in the mixture model of image I.
- π_I^i is the weight for the ith component of the mixture model.
- μ_I^i is the mean of the ith component of the mixture model.
- Σ_I^i is the standard deviation of the ith component of the mixture model.

- A Gaussian mixture model for each class C_i is learnt by aggregating (e.g., weighted averaging) all the image models within the class:

$$P(\mathbf{x}|C_i) \sum_{k=1}^{K} \pi_{C_i}^k \mathcal{G}\left(\mathbf{x}, \mu_{C_i}^k, \Sigma_{C_i}^k\right) \tag{7.22}$$

where

- K is the number of components in the mixture model of class C_i.
- $\pi_{C_i}^k$ is the weight for the kth component of the mixture model.
- $\mu_{C_i}^k$ is the mean of the kth component of the mixture model.
- $\Sigma_{C_i}^k$ is the standard deviation of the kth component of the mixture model.

- Given a new observation image $I_u = \mathbf{x}_u$, the annotation of image I_u is given by the *MAP*:

$$\begin{aligned} P\left(C_j|\mathbf{x}_u\right) = \hat{c} &= \arg\max_{C_i} P(C_i|\mathbf{x}_u) \\ &= \arg\max_{C_i}\{P(\mathbf{x}_u|C_i) \times P(C_i)\} \end{aligned} \tag{7.23}$$

The algorithm of the parametric annotation method is illustrated in Fig. 7.5 [2, 5].

7.7 Image Classification with Gaussian Process

In Gaussian mixture, each multidimensional feature vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is regarded as a data point in a R^n space and the mixture model is built based on the statistics of the data points in a cluster.

But a multidimensional data $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ can also be regarded as a discretized function f: X → R and $y = f(\mathbf{x}) = \{x_i = f(d_i) \mid i = 1, 2, \ldots, n\}$. A typical example of such a data is a histogram feature vector. Figure 7.6 shows three normalized histograms (vertical bars) from the same class in red, green and blue, respectively. The corresponding functions approximating the three histograms are shown as colored curves at the top of the histograms.

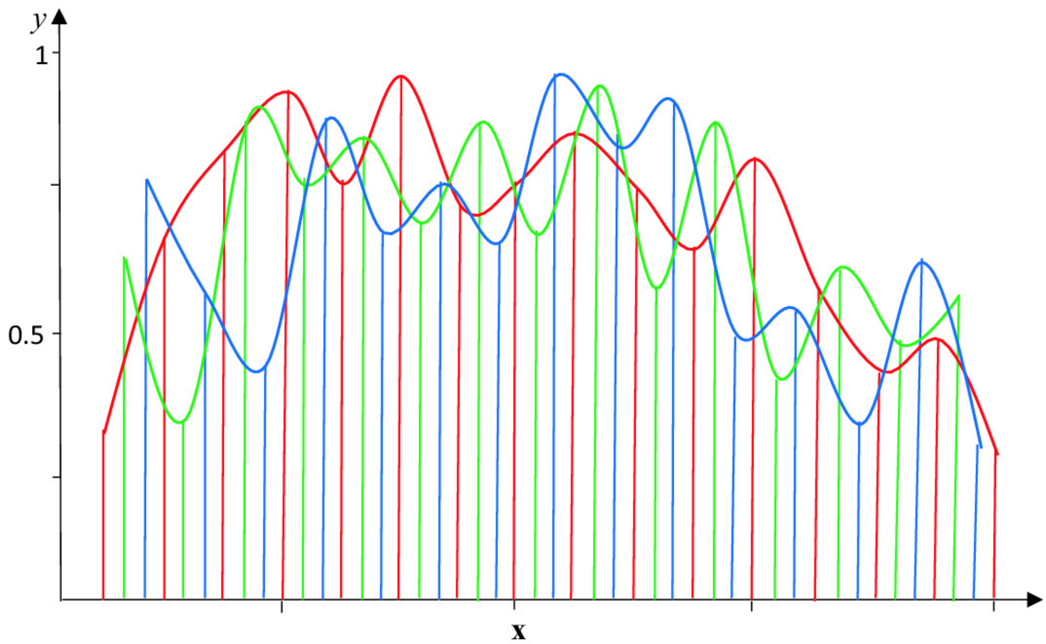

Fig. 7.6 Feature vectors shown as functions. Three histograms shown as vertical color bars and their respective functions shown as colored curves on the top

If we plot all the histogram features $f(\mathbf{x}_i)$ from a class in a single coordinate system, we would see all the data fall within a band and form a cluster. Like in the linear regression which attempts to fit a line to a cluster of data points, we can also fit a curve to this cluster of data points and use this curve as the model to predict new instances. This approach is the idea behind the Gaussian process or GP, which is demonstrated in Fig. 7.7 [6].

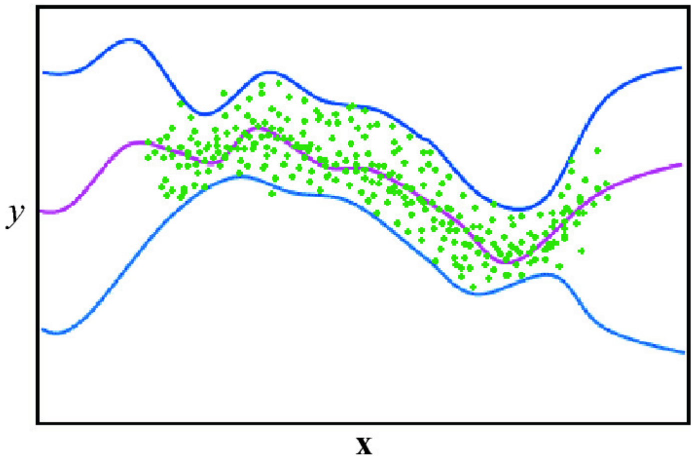

Fig. 7.7 Cluster of multidimensional data (green) and the approximation function of the data shown in pink

Now, given a set of data $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ from a certain class C and each feature vector $\mathbf{x}_i$ is a D-dimensional data point in space $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{iD})$. A matrix $\mathbf{X} = D \times N = (\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_D)^{\mathrm{T}}$ can be created as following:

$$\mathbf{X} = \begin{bmatrix} x_{11}, x_{21}, \ldots\ldots, x_{N1} \\ x_{12}, x_{22}, \ldots\ldots, x_{N2} \\ \ldots\ldots\ldots \\ x_{1D}, x_{2D}, \ldots\ldots, x_{ND} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \\ \ldots \\ \mathbf{d}_D \end{bmatrix} \tag{7.24}$$

where $\mathbf{d}_j$ is a jth-dimensional vector. Since the elements of each $\mathbf{d}_j$ are samples from (or follow) a normal distribution, $N(\mu_j, \sigma_j)$, $\mathbf{X}$ is a Gaussian process and $\mathbf{X} \sim N(\boldsymbol{\mu}_{\mathbf{X}}, \mathbf{K}_{\mathbf{XX}})$, where $\boldsymbol{\mu}_{\mathbf{X}}$ and $\mathbf{K}_{\mathbf{XX}}$ are the *mean* and *variance* which are determined by (7.25) and (7.26), respectively.

$$\boldsymbol{\mu}_{\mathbf{X}} = \begin{bmatrix} \mu(\mathbf{d}_1) \\ \mu(\mathbf{d}_2) \\ \ldots \\ \mu(\mathbf{d}_D) \end{bmatrix} \tag{7.25}$$

$$\mathbf{K}_{\mathbf{XX}} = \begin{bmatrix} k(\mathbf{d}_1, \mathbf{d}_1), k(\mathbf{d}_1, \mathbf{d}_2) \ldots\ldots, k(\mathbf{d}_1, \mathbf{d}_D) \\ k(\mathbf{d}_2, \mathbf{d}_1), k(\mathbf{d}_2, \mathbf{d}_2) \ldots\ldots, k(\mathbf{d}_2, \mathbf{d}_D) \\ \ldots\ldots\ldots \\ k(\mathbf{d}_D, \mathbf{d}_1), k(\mathbf{d}_D, \mathbf{d}_2) \ldots\ldots, k(\mathbf{d}_D, \mathbf{d}_D) \end{bmatrix} \tag{7.26}$$

where $k(\mathbf{d}_i, \mathbf{d}_j)$ is a *kernel* function which is typically the *covariance* function.

To predict a new data or a new set of data $\mathbf{X}_*$, $\mathbf{X}$ and $\mathbf{X}_*$ are concatenated and the concatenated data is a new GP which follows the following normal distribution:

$$f\begin{pmatrix} \mathbf{X}_* \\ X \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_{\mathbf{X}_*} \\ \mu_{\mathbf{X}} \end{pmatrix}, \begin{pmatrix} \Sigma_{\mathbf{X}_*\mathbf{X}_*}, \Sigma_{\mathbf{X}_*\mathbf{X}} \\ \Sigma_{\mathbf{XX}_*}, \Sigma_{\mathbf{XX}} \end{pmatrix}\right) \tag{7.27}$$

Then, the probability of the new data $\mathbf{X}_*$ given the observed data $\mathbf{X}$ is given by (7.28):

$$p(\mathbf{X}_*|\mathbf{X}) = \mathcal{N}\left(\mu_{\mathbf{X}_*} + \mathbf{K}_{\mathbf{X}_*\mathbf{X}}\mathbf{K}_{\mathbf{XX}}^{-1}(\mathbf{X} - \mu_{\mathbf{X}}), \mathbf{K}_{\mathbf{X}_*\mathbf{X}_*} - \mathbf{K}_{\mathbf{X}_*\mathbf{X}}\mathbf{K}_{\mathbf{XX}}^{-1}\mathbf{K}_{\mathbf{XX}_*}\right) \tag{7.28}$$

The proof of (7.28) is given in Appendix [7–9].

7.8 Summary

This chapter introduces the first image classification method: Bayesian classification. Several important and interesting application of Bayesian classifier are described and demonstrated in details, including NB, word co-occurrence model, CMRM, parametric model and Gaussian process method. The key features of Bayesian classifiers can be summarized as the following

1. **Generative**. A Bayesian classifier is a typical generative model, and it assumes the distribution or model of likelihood probability is known. This likelihood probability model is typically obtained through learning from known samples.

2. **Intuitive**. Compared with many other black box-based classifiers such as SVM and ANN, Bayesian classifiers are intuitive, and results are easily interpreted by a human being. The basic idea the Bayesian method is to use our prior experience to forecast or predict new events. In this sense, we all make decision using Bayesian classifiers.

3. **Robust**. A Bayesian classifier generates a result in probabilistic form instead of deterministic form. Probabilistic prediction is more robust than deterministic prediction, e.g., a 70% chance of rain forecast is more likely to be correct than a rain/no-rain forecast.

4. **Nonlinear**. The boundary of a Bayesian classifier is nonlinear because the prediction is based on the data distributions and the distribution models can be any shape.

However, the downside of Bayesian classifier is that there needs large number of data samples to have a reasonable accurate estimation of data distribution.

7.9 Exercises

1. Use the example code in the following web page to generate a Gaussian mixture model: https://au.mathworks.com/help/stats/gmdistribution.html#mw_4758a58e-5bc7-4eda-b261-83521d63d1ce. First, try the `gmdistribution` function with the following code. Turn the graph to different angles and also to flat (2D) to view the model in more details. Then try different *mu* and *sigma* values to create more GMM models.

```
mu = [1 2;-3- 5];
sigma = cat(3,[2 .5],[1 1]);
gm = gmdistribution(mu,sigma);
ezsurf(@(x,y)pdf(gm,[x y]),[-10 10],[-10
10]);
```

2. Use the example code from this link: https://au.mathworks.com/help/stats/fitgmdist.html and try the `fitgmdist` function with the following code. Turn the graph to different angles and also to flat (2D) to view the model in more details. Now, try `fitgmdist` with more components and different *mu* and *sigma* values to create more complex GMM models.

```
mu1 = [1 2];
sigma1 = [2 0; 0 .5];
mu2 = [-3 -5];
sigma2 = [1 0; 0 1];
rng('default')
r1 = mvnrnd(mu1,sigma1,1000);
r2 = mvnrnd(mu2,sigma2,1000);
X = [r1; r2];
gm = fitgmdist(X,2)
ezsurf(@(x,y)pdf(gm,[x y]),[-8 6],[-8 6])
```

3. Use the code from the following web page to compute the posterior probabilities of a GMM model which you have generated from the above exercises. Explain the graph using the colors and color bar. Write a report on the GMM models, the posterior probabilities and tell how they can be used for image analysis (hints: an image or an image region is a GMM model, and a GMM model is characterized or defined by its parameters e.g. *mu* and *Sigma*). https://au.mathworks.com/help/stats/gmdistribution.posterior.html.

References

1. Islam M (2009) *SIRBOT—semantic image retrieval based on object translation*, Ph.D. thesis, Monash University
2. Zhang D, Islam M, Lu G (2012) A review on automatic image annotation techniques. Patt Recogn 45(1):346–362
   [Crossref]
3. Mori Y, Takahashi H, Oka R (1999) *Image-to-word transformation based on dividing and vector quantizing images with words*. Proceedings of the 1st international workshop on multimedia intelligent storage and retrieval management, ACM Press