- Number of overlapping blocks per ROI: $N = 8 \times 16$ or $16 \times 8 = 128$.
- Number of cells per block: $M = 4$.
- Number of pixels per cell: $A = 8 \times 8 = 64$.
- Number of histogram bins per cell: $H = 8$ or 16.
- Length of final feature vector $\mathbf{V}$: 8 bins/cell $\times$ 4 cells/block $\times$ 8 $\times$ 16 blocks/ROI = 4096 bins = 4096 elements in the feature vector.

3.5.2 Histogram of optical flow feature vector

When applied to spatiotemporal data, HOG features can be obtained using spatiotemporal gradients. Alternatively, optical flow vectors can be used in place of gradient vectors, forming the HOF method.[9,10] The HOG and HOF features are often combined to form HOG-HOF features. These features are commonly extracted at a hierarchy of scales.[10] Instead of extracting these features densely over blocks covering the entire ROI, the HOG-HOF features are sometimes extracted locally about spatiotemporal interest points (STIPs).

3.6 Examples of Classifiers

The paradigm of target classification is as follows, regardless of whether the technique is called template matching, neural, or statistical (Bayesian or frequentist), etc. Image (or signal) data plus temporally synchronized metadata is collected. A potential target is detected in the (e.g., image) data. This target is sometimes referred to as a *blob* or an *unknown*. The blob image is transformed in some manner such as centered, spatially scaled, segmented, or rotated, using the available metadata (e.g., range, roll, pitch). A feature vector (or feature image) is assembled from features extracted from the blob image. Irrelevant and redundant features are discarded. The resultant feature vector $\mathbf{X}$ represents a candidate target. To know which target $\mathbf{X}$ represents requires further processing, which is called *classification*. The candidate target is then assigned to one of r subclasses $\theta_1,\ldots,\theta_r$, where each subclass is a member of a broader class $\theta_i \in \{C_1, C_2,\ldots,C_q\}$ according to the evidence, i.e., the vector of features $\mathbf{X} = \{x_1,\ldots,x_d\}$. For example, a subclass may be T-72 tank at $\sim$45-deg aspect angle, the broader class being T-72 tank.

Supervised classifiers require pairs of input data $\{\mathbf{X}_i, Y_i\}$ where $\mathbf{X}_i$ are the feature vectors, and Y_i are the corresponding labels for the feature vectors. The Y_i can also be expressed as a vector $\mathbf{Y}_i$ with a nonzero entry in the n^{th} position to indicate the n^{th} class type. Unsupervised classifiers require only the set $\{\mathbf{X}_i\}$. They map each unknown $\mathbf{X}$ to a cluster but do not assign a meaningful class label to the cluster. The samples assigned to a particular cluster have common characteristics.

Basic classification algorithms that we will cover next do not do justice to the complexity of the military target classification problem. There are many other issues. What metadata is available, and what is done with it? What

errors are inherent in the metadata? How well is the metadata synched to the image data? What *a priori* information is available? How does the sensor used to collect the training data relate to the operational sensor? What are the operating modes of the sensor? Can modes (e.g., radar mode, camera field-of-view) be switched, or can other sensors be called up to get additional information when needed? Can the ATR control the platform or sensor mode to get a better look at the target? What are the roles of the classifier and the human-in-the-loop? How does the system report its results? What performance is required to successfully complete a mission? How is the mission list of target classes loaded into the system? How does the system know if the sensors are working correctly or if the weather is degrading the data? Does the data in the classifier turn the system into a classified military device? How does the system secure itself if captured?

3.6.1 Simple classifiers

There is nothing wrong with using a simple classifier. Simple classifiers are robust to unforeseen circumstances, and easy to design, test, implement, and maintain. Only when a simple classifier can't meet performance requirements should a more complex classifier be used. Even then, a simple classifier can help determine the added benefit versus cost of a more complex approach.

3.6.1.1 One-class classifiers

A one-class classifier distinguishes targets from non-targets. For a particular mission, everything but the target of interest can be considered a distraction or clutter. The assumption of the one-class classifier is that in the infinitely varying real world clutter blobs are not well described. This is especially true at high resolution rather than for point-like objects and noise-like clutter. Many target detectors can be viewed as one-class classifiers. The one-class classifier cannot provide the posterior probability of targets because information on non-targets is neither available nor convincingly assumed. That is our underlying assumption as ATR developers exploiting a rich set of features. Radar engineers often make the opposite assumption with limited target descriptors or low-resolution data.

3.6.1.2 Two-class linear classifiers

A linear classifier computes a single hyperplane to separate one class from another. Suppose that a magical long-range sensor can determine the height and weight of animals. Suppose that 13 animals are imaged. Five of them are elephants, and eight are giraffes. The 13 animals result in a training set of size $m = 13$; $D = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_{13}, Y_{13})\}$, $Y_i \in \{\text{elephant, giraffe}\}$. Each training sample is represented by a two-element feature vector; $\mathbf{X}_i = (w_i, h_i)$, where w denotes animal weight, and h denotes height. Figure 3.8(a) shows a plot of the 13 vectors represented as points in the 2D feature space. The two classes are

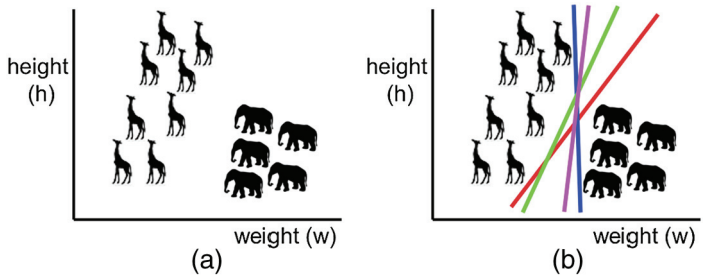

Figure 3.8 (a) Each training sample can be represented by a point in a feature space. This example is for a two-class problem, where training vectors each have two elements. (b) The two classes in this example are linearly separable by any of a number of straight lines.

linearly separable since any of a number of straight lines can be drawn separating them, as shown in Fig. 3.8(b). A separating line can be expressed as $aw + bh = c$. There are many possible solutions for a, b, and c, each resulting in a different separating line. Which solution is deemed optimal depends on the chosen definition of optimal. For example, linear regression defines optimality in terms of means and variances, and uses all 13 points to obtain a solution. As we shall see shortly, an approach called support vector machine uses a different definition of optimality.

Thus, two sets of points are linearly separable in 2D space if they can be completely separated by a single line. This single line is not necessarily unique. Two sets of points are linearly separable in higher-dimensional feature space if they can be separated by a single hyperplane, where a hyperplane is just the multidimensional counterpart of a line.

Let $\mathbf{X}$ be the feature vector of an unknown object, where unknown means not yet assigned a class label. A weight vector $\mathbf{W}$ and threshold b are learned from the labeled training data D. Function f converts the biased dot product of the two vectors into the desired output. The output score of the classifier is

$$out = f(\mathbf{W} \cdot \mathbf{X} - b). \tag{3.2}$$

All output values above zero can be assigned to one class, and all values below zero can be assigned to the other class. When the bias term b is left out of this type of equation, it is assumed to be the last element of the weight vector with a 1 placed into an additional element of the feature vector.

3.6.1.3 Support vector machine

A support vector machine (SVM) constructs a hyperplane separating two classes of linearly separable points.[11] The thesis of the approach is that not all points in feature space are equally important in constructing the separating plane. The points closest to the decision surface are called support vectors. These points are circled in bold in the example of Fig. 3.9(a). They are the most difficult to classify because they are closest to points of the other class.

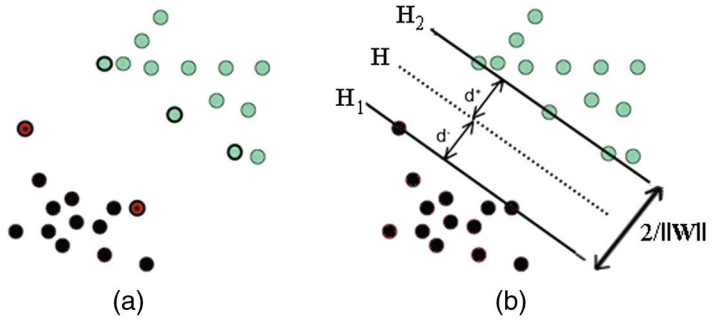

Figure 3.9 (a) Feature vectors from two linearly separable classes. The five support vectors have dark perimeters. (b) The best separating hyperplane, according to SVM, is the one with the most margin, shown here as a dashed line.

Although many lines can separate the black and green points in Fig. 3.9(a), SVM maximizes the margin around the separating hyperplane, as shown in Fig. 3.9(b). The decision function depends only on the support vectors.

The training data set D consists of n points $\mathbf{X}_i$, such that

$$D = \{(\mathbf{X}_i, \mathrm{Y}_i) | \mathbf{X}_i \in R^d, Y_i \in \{-1, +1\}\}, \tag{3.3}$$

where Y_i is either -1 or $+1$, indicating the class of $\mathbf{X}_i$.

In Fig. 3.9(b), the points on the planes H_1 and H_2 are support vectors. The H_1 and H_2 planes are described by the equations

$$\begin{aligned} H_1 &: \mathbf{X} \cdot \mathbf{W} + b = +1, \\ H_2 &: \mathbf{X} \cdot \mathbf{W} + b = -1. \end{aligned} \tag{3.4}$$

According to the SVM method, the hyperplane H provides the best separation between the two sets of points. The distance between hyperplanes H_1 and H_2 is $2/\|\mathbf{W}\|$. Minimizing $\|\mathbf{W}\|$ maximizes the margin. The following constraint is added:

$$\begin{aligned} &\mathbf{X}_i \cdot \mathbf{W} + b \geq +1 \text{ for } \mathbf{X}_i \text{ of the first class } Y_i = +1, \\ &\mathbf{X}_i \cdot \mathbf{W} + b \leq -1 \text{ for } \mathbf{X}_i \text{ of the second class } Y_i = -1. \end{aligned} \tag{3.5}$$

The two equations can be combined to yield

$$\mathbf{Y}_i(\mathbf{X}_i \cdot \mathbf{W} + b) \geq 1 \text{ for all } i, \text{ where } Y_i \in \{-1, +1\}. \tag{3.6}$$

This is a constrained optimization problem that can be solved by the Lagrangian multiplier method. The objective is to find the hyperplanes that maximize the margin by minimizing $\|\mathbf{W}\|^2$, such that the discrimination boundary is conformed:

$$\begin{aligned} &\text{Minimize } \frac{1}{2\|\mathbf{W}\|^2} \text{ such that} \\ &Y_i(\mathbf{X}_i \cdot \mathbf{W}) + b = 1. \end{aligned} \tag{3.7}$$

The problem is formulated in a dual form as

Maximize in η_i:

$$L(\eta) = \sum_{i=1}^{n} \eta_i - \frac{1}{2} \sum_{i,j} \eta_i \eta_j Y_i Y_j \mathbf{X}_i^T \cdot \mathbf{X}_j$$

subject to:

$\eta_i \geq 0$, and to the constraint in b

$$\sum_{i=1}^{n} \eta_i Y_i = 0. \tag{3.8}$$

In this formulation, the input feature vectors only appear inside a dot product. There are two extensions of SVM. Nonlinear SVM modifies the training vectors using a kernel function $k(\mathbf{X}_i,\mathbf{X}_j) = \phi(\mathbf{X}_i) \cdot \phi(\mathbf{X}_j)$ instead of $\mathbf{X}_i \cdot \mathbf{X}_j$. With good selection of a kernel function, nonlinearly separable points can be separated. However, finding such a kernel is not guaranteed. With the kernel method, the actual values of $\phi(\mathbf{X})$ need not be known as long as the dot product is known. Common inner product kernels are given in Table 3.1.

Basic SVM is a binary classifier. Multiclass SVMs are usually implemented by combining several two-class SVMs, such as with a decision tree or ensemble of binary SVM classifiers, each trained to recognize a single class or subclass.

SVMs are often compared to artificial neural networks (ANNs). There are many types of ANNs and several variations of SVM. Basic two-class SVM has a strong foundation in optimization theory, reaches a global minimum, and isn't subject to different performance each time it is re-trained on the same data. SVM code is readily available. SVM doesn't come in a bewildering number of varieties as do ANNs. A basic ANN has some good features, too. It constructs decision surfaces for multiple classes through training and then outputs a posterior probability estimate for each class. It is well suited for very large training sets and is relatively insensitive to noise and mislabeled training vectors. ANNs can be constructed to handle very large multiclass problems (>1000 classes). There are dozens, if not hundreds, of types of ANNs. As noted in Table 3.1, some versions of the SVM are a lot like some ANNs. In conclusion, which one works best depends on the nature of the data and skill of the ATR designer. With both SVMs and ANNs, it is difficult to understand exactly how and why the classifier

Table 3.1 Several inner product kernels that can be used with SVM.[12]

Type of SVM	Inner product kernel $k(\mathbf{X}_i,\mathbf{X}_j)$	Note
Hyperbolic tangent (sigmoid)	$\tanh(\alpha \mathbf{X}_i \cdot \mathbf{X}_j + c)$	Equivalent to a two-layer Perceptron network
Polynomial (homogeneous)	$(\mathbf{X}_i \cdot \mathbf{X}_j)^p$	Power p must be chosen.
Polynomial (inhomogeneous)	$(\mathbf{X}_i \cdot \mathbf{X}_j + 1)^p$	Power p must be chosen.
Gaussian radial basis function (RBF)	$e^{-\frac{1}{2\sigma^2}\|(\mathbf{X}_i - \mathbf{X}_j)\|_2}$	Gaussian width σ must be chosen.