Chapter 7
Pattern Classification

1

➢Pattern Classification

✓Pattern classification involves using the extracted image features to automatically classify image objects

✓This is done by developing classification algorithms that use the feature information

2

✓The primary uses of pattern classification in image analysis are for computer vision and image compression applications development

✓In the development of a computer vision algorithm, pattern classification is typically the final step

✓In image compression, pattern classification is used to find higher level representation of image data

3

✓**Algorithm Development: Training and Testing Methods**

- To develop a classification algorithm, divide data into a training set and a test set
- The *training set* is used for algorithm development, and the *test set* is used to test the algorithm for success
- This method gives unbiased results, providing confidence in test results

4

- Training and test sets should represent all classes
- Selection of the two sets should be done before development starts
- Maximize the size of the training set to develop the best algorithm, maximize test set size to increase confidence
- Set size depends on many factors, in practice typically split the available images into two equal-sized groups

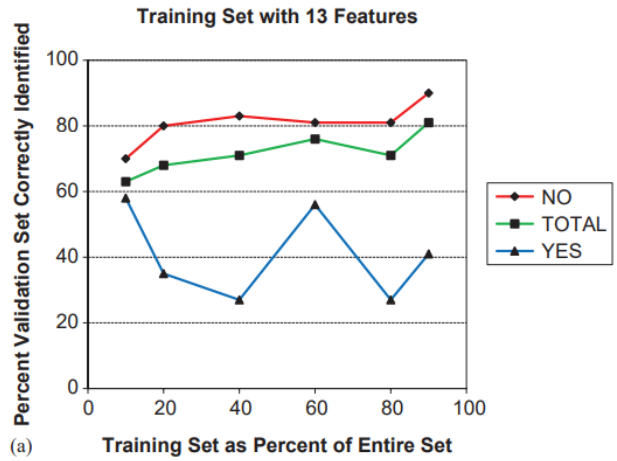
5

- Additionally, during development, training set may be iteratively divided into training set and validation set
- To see how well the current classifier works on an independent set
- Guide the developers defining the parameters for the classifier as well as defining the best classifier for the application
- Development is an iterative process and the validation set is effectively a test set used during a particular iteration
- 50%-80% for training set and 20%-50% for validation set

6

Figure 7.2-1: Example of Increasing the Training Set Size

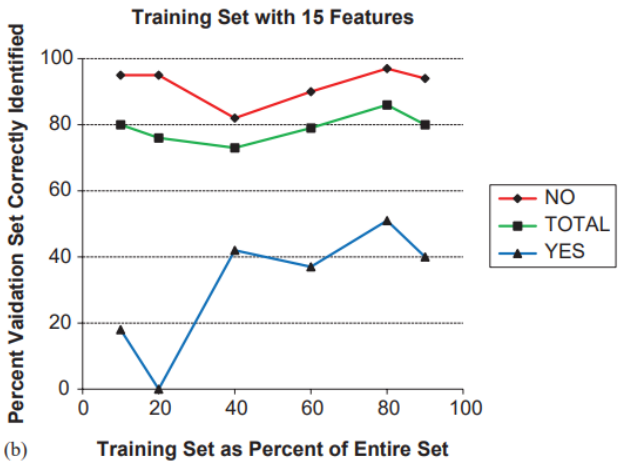
These are results from an experiment to classify skin tumor images



a) Results with 13 features, with the success rate jumping around we are not confident in the feature set being complete

7

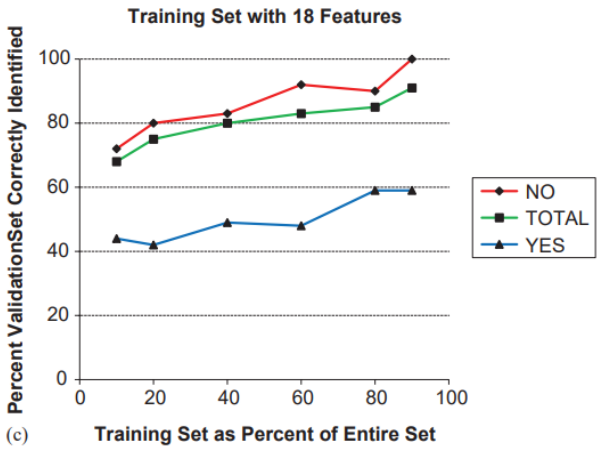
Figure 7.2-1: Example of Increasing the Training Set Size (contd)



b) Results with 15 features, better results but still not increasing consistently

8

Figure 7.2-1: Example of Increasing the Training Set Size (contd)



(c)

c) Results with 18 features, the consistently increasing success rate as we increase the training set size gives us confidence in the completeness of the feature set being used

9

- One commonly used method to avoid needing to select the training and test sets is the *leave-one-out-method*
- With this technique, all but one of the samples is used for training, and then it is tested on the one that was left out
- This is done as many times as there are samples, and the number misclassified represents the error rate

10

- *Leave-K-out-method* is valid
- *K* is defined based on available resources
- Smaller the value for *K*, the greater the confidence we have in the results
- Leave *K* vectors out and train (develop) the classification algorithm, and then test on the *K* vectors left out
- Results are averaged to predict application success

11

- Correct classification is only one aspect of classification success
- Some types of misclassification may be more "costly" than others
- A *cost function* can be introduced to weight how various misclassifications contribute to the system failure rate

12

✓**Classification Algorithms and Methods**

❖***Nearest neighbor (NN)***:

- The object of interest is compared to every sample in the training set, using distance or similarity measure(s)
- The "unknown" object is classified as the closest sample in the training set
- *NN* is *computationally intensive*, *not very robust*, and can lead to *random classification* in overlapping clusters

13

❖***K-nearest neighbor (K-NN)***:

- *NN* is made more robust by considering the closest K examples in the training set
- Assign the unknown feature vector to the class that occurs most often in the set of K-Neighbors
- *K-NN* method is still very computationally intensive, but alleviates the problem of random classification in overlapping clusters

14

❖***Nearest Centroid***:

- Find the centroids for each class from the samples in the training set, compare the unknown samples to the representative centroids
- Reduces the computational burden
- The comparison is done using any of the distance or similarity measures
- The centroids are calculated by finding the average value for each vector component in the training set

15

EXAMPLE 7.3.1:

Suppose we have a training set of four feature vectors, and we have two classes:

Class A

$$\mathbf{A}_1 = \begin{bmatrix} 3 \\ 4 \\ 7 \end{bmatrix} and\ \mathbf{A}_2 = \begin{bmatrix} 1 \\ 7 \\ 6 \end{bmatrix}$$

Class B:

$$\mathbf{B}_1 = \begin{bmatrix} 4 \\ 2 \\ 9 \end{bmatrix} and\ \mathbf{B}_2 = \begin{bmatrix} 2 \\ 3 \\ 3 \end{bmatrix}$$

16

EXAMPLE 7.3.1 (contd):

The representative vector, centroid, for class A is:

$$\begin{bmatrix} (3+1)/2 \\ (4+7)/2 \\ (7+6)/2 \end{bmatrix} = \begin{bmatrix} 2 \\ 5.5 \\ 6.5 \end{bmatrix}$$

The representative vector, centroid, for class B is:

$$\begin{bmatrix} (4+2)/2 \\ (2+3)/2 \\ (9+3)/2 \end{bmatrix} = \begin{bmatrix} 3 \\ 2.5 \\ 6 \end{bmatrix}$$

17

❖***Template matching***:

- Comparing unknown vectors to classified vectors is also called *template matching*
- Template matching can be applied to the raw image data
  - template is compared to subimages by using a distance or similarity measure
  - a threshold is set on this measure to determine a match
  - useful for applications where the size and orient-ation of the objects is known, and the objects shapes are regular

18

❖***Bayesian analysis***:

- Bayesian theory provides a statistical approach to classification algorithm development
- Requires a complete statistical model of the features; normalize the features to standard normal density
- Provides a classifier that has an *optimal classification rate* – boundaries that separate the classes provide a minimum average error
- Boundaries are called *discriminant functions*

19

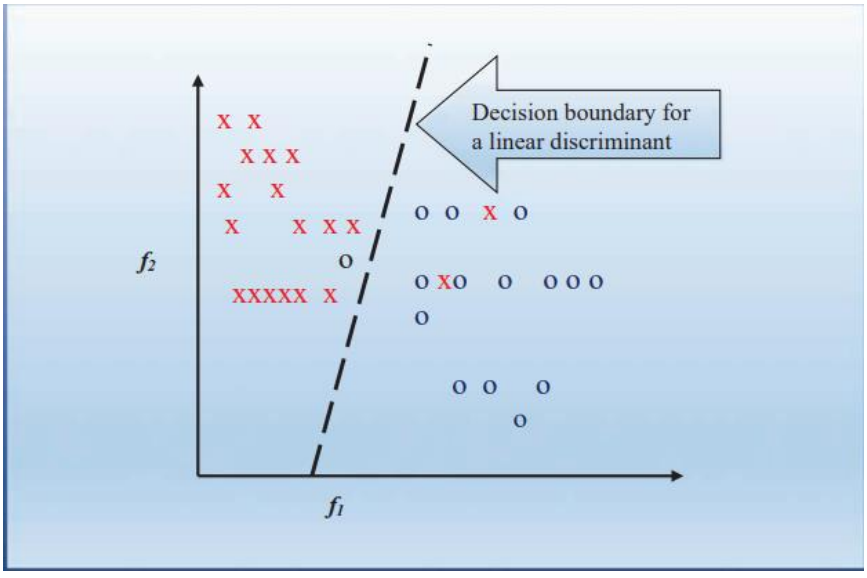


FIGURE 7.4-1
A linear discriminant separating two classes. This shows a two-dimensional feature space defined by feature vectors, $\mathbf{F} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$, and two classes represented by *x* and *o*. Note that this linear discriminant misclassifies two *x*'s and one *o*.

- The *n*-dimensional form of a linear function is called a *hyperplane*
- Discriminant functions can also be curved or piece-wise linear
- Generally, the forms are limited to those that can be defined analytically via equations; such as circles, ellipses, parabolas, or hyperbolas

21

❖***Neural networks and Deep Learning***:

- Neural networks are based on a simple model of the nervous system in biological organisms, thus *artificial neural networks*
- These biological models are based on a simple processing element called a neuron
- Neurons function by outputting a weighted sum of the inputs, and these weights are generated during the learning or training phase

22

- The element's output function is called the *activation function*, and the basic types are:

1. The *identity function*, the input equals the output
2. A *threshold function* where every input greater than the threshold value outputs a 1, and less than the threshold outputs 0
3. A *sigmoid function*, an S-shaped curve, which is nonlinear and was used most often
4. The ReLU function operates by outputting the input directly if it is positive; otherwise, it outputs zero. It has become the default activation function for many types of neural networks – vanishing gra

23

- Single layer networks are limited in their capabilities and nonlinear functions are required to take advantage of the power available in multilayer networks
- ReLu overcomes the vanishing gradient problem, allowing models to learn faster and perform better

- The neural network consists of:

1. Input layer
2. Output layer
3. Hidden layers

24

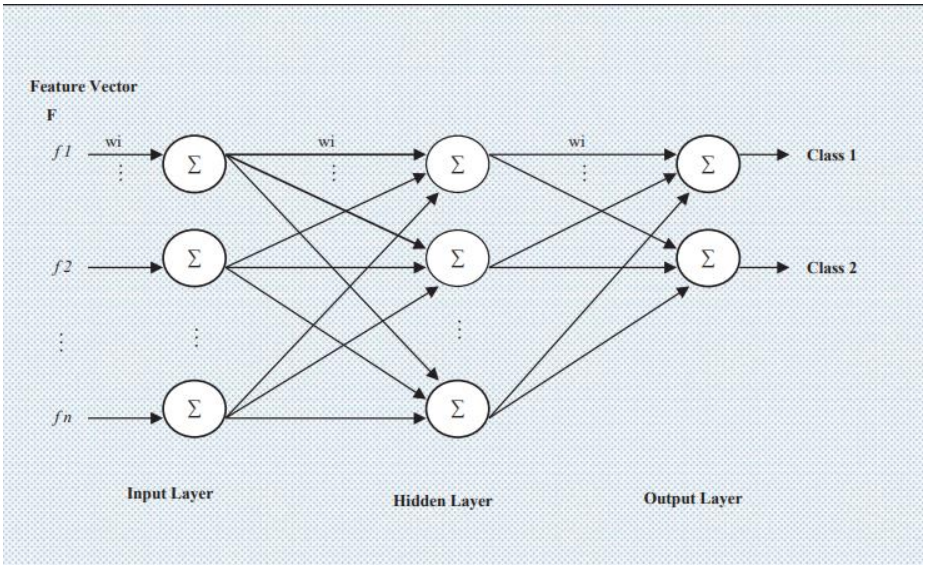


FIGURE 7.5-2
Neural network. Here, we show a neural network architecture with an input layer, where the feature values are input, one hidden layer and the output layer. In this case, the output layer corresponds to two classes. The circles are the processing elements, neurons, and the arrows represent the connections. Associated with each connection is a weight (w_i, shown only across the top for clarity) by which the signal is multiplied. These weights are adjusted during the training (learning) phase.

- Most learning algorithms are form of Gradient Descent which searches the multidimensional error space for a local minimum
- It can be applied to any differentiable function by moving in the opposite direction of the gradient

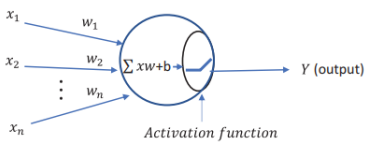


FIGURE 7.5-1
Neuron and activation function. A typical neuron, or processing element, in a neural network. The neuron first calculates the weighted sum and then sends the sum to the activation function that controls the output. The extra term in the summation, b, is called the bias term, and it can be used to deal with bias in unbalanced training sets.

26

- The Gradient is the direction in which the function is changing the fastest;
  - By moving away from it, we can find minimum
- It can be applied to any differentiable function by moving in the opposite direction of the gradient

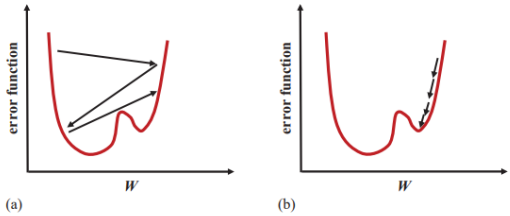


FIGURE 7.5-3
Neural network learning with gradient descent. Gradient descent works by following the path toward the minimum derivative, where the slope of the tangent line is minimum. We represent the network weights by W and show a simple two-dimensional function for illustration. (a) With too large of a learning rate, we may miss the minimum and oscillate and (b) with too small a learning rate, it may take many, many iterations and then we may get stuck in a local minimum

- With a 2-D function for simplicity, the learning takes place by adjusting the weights so that error moves towards minimum
- Or to think of this is to follow the path of decreas-ing slope, the derivative, until it reaches zero
- The learning rate parameter determines how fast the values are changed when adjusting weights
  - Large learning rate, there will be larger change per iteration, which may **cause overshoot** and create undesirable oscillation in the error or success measurements
  - Too small of LR will take much longer, and it may get stuck in a local minimum

28

- The main distinguishing characteristics of the neural network are:

1. The *architecture*, includes the number of inputs, outputs and hidden layers, and how they are all connected
2. The *activation function*, determines operation of neuron
3. The *learning algorithm,* determines weights by calculating error and then adjusting the weights to improve the error

29

- The training continues for a specific number of iterations, or until a specified error criterion is reached
- The feature vectors are preprocessed with a PCT to decorrelate the input data
- Neural networks have been used successfully in many applications, for example in the recognition of hand written characters

30

- NN training proceeds as follows:

1. Determine the number of iterations, i, for which we will train the NN. Start with i =1 (one iteration = one training *cycle* or one training *epoch)*
2. From the training set, randomly select the iteration training set , train_*i,* and put the others aside for validation
3. Train the model using train_*i*
4. Test the model on validation_*i* and record desired success metrics
5. Increment i and go back to step (2) and continue until desired number of iterations is achieved or until the minimum error is reached

31

- Longer training – improved results
- Avoid overtraining – trained to a specific data and may NOT generalized to the classes desired
  - Variance in the validation result will be large
  - Change the network architecture and/or training parameters

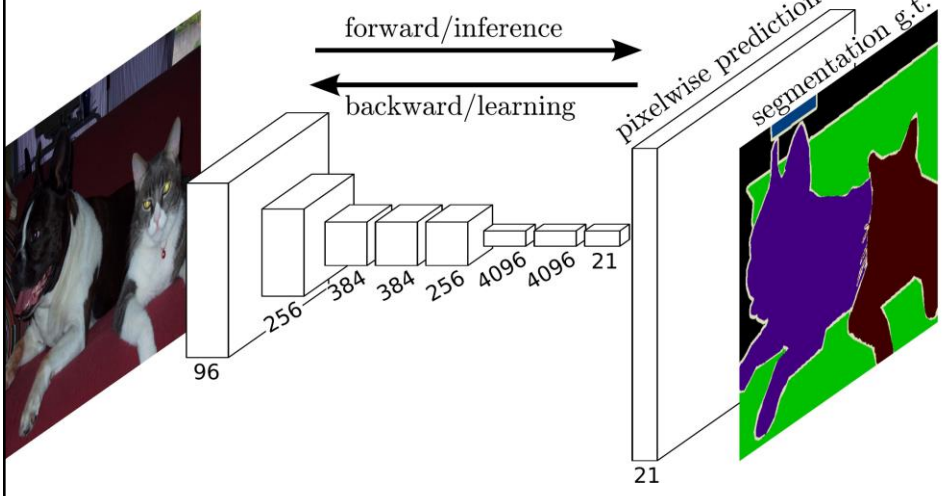
32

- CNN training works differently proceeds as follows:
- A CNN performs both the functions of feature extraction and pattern classification and takes the image itself as input
  - By using convolution filters to extract various spatial features, and the convolution filter coefficients are the weights that are adjusted during the training/learning phase
- A convolution layer in a CNN is specified by the number and size of the filters or convolution masks
  - The deeper we get into the network, the more specific and complex the corresponding convolution filters are

33

- A network that consists of only convolution layers, along with up- and downsampling is referred as fully convolutional network (FCN)
- A typical CNN consists of convolution layers followed by downsampling layers, which can then be upsampled to fully connected layers by “flattening” or vectorising the data
- Deep learning requires large number of input images for training – for limited data, data augmentation (geometric operations: resizing, rotation, flipping, translation, zooming and elastic deformation) is performed

34



35

- For further study, other methods for pattern classification are available such as:

  1. More Artificial intelligence approaches
  2. Structural approaches
  3. Fuzzy logic approaches
  4. Genetic algorithms

36

METRICS

- Imagine you are training a spam; How to measure the model performance?
  - It is more important to correctly identify all actual spam emails, even if mistakenly flags some of the legitimate emails

37

		Predicted	
		0	**1**
Actual	**0**	TN	FP
	1	FN	TP

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

38

Metrics

- True Positive (TP): Sick person classified as sick
- False Positive (FP): healthy person classified as sick
- True Negative (TN): healthy person classified as healthy
- False Negative (FN): sick person classified as healthy

39

- **Accuracy:** Most basic metric used to evaluate a classification model
- Represent percentage of correct prediction made by your model
- Misleading for imbalanced data: 1000 mails. 10 spam, accuracy is 99%

```
Accuracy = (TP + TN) / (TP + TN + FP + FN)
```

- **Precision**: Measure the proportion of true positive predictions among all positive predictions
- High Precision: Great; your model rarely flags innocent emails as spam
- Low Precision: Oops! Your model is firing of false alarm, flagging many legitimate emails as spam

```
Precision = TP / (TP + FP)
```

40

- **Sensitivity (Recall)** measure the proportion of true positive predications among all actual positive instances
- what percentage of actual spam emails were correctly identified
- High Recall: Fantastic! your model catches most of the spam emails
- Low Recall: Uh oh! Your model is letting some spam slip through the cracks

```
Recall = TP / (TP + FN)
```

- **F1-Score:** Precision and recall have an inverse relationship

```
F1 = 2 * (precision * recall) / (precision + recall)
```

- Harmonic mean of Precision and Recall

41

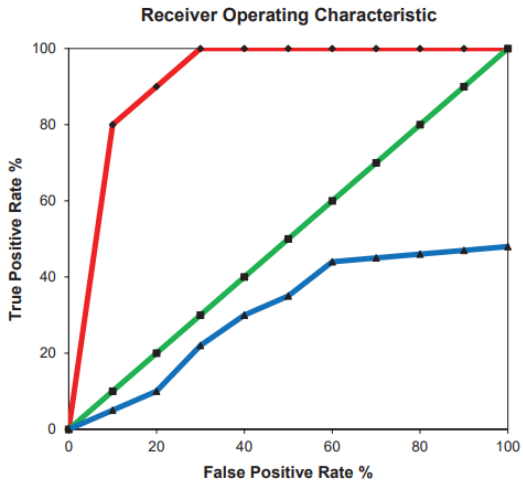


FIGURE 7.6-1
Receiver operating characteristic (ROC) curve. The red curve represents a good ROC, the area under the curve (AUC) is close to 1. The green line represents a marginal ROC, it is the cutoff for where the system has any value – here, the AUC is 0.5, and it represents a system that is equivalent to random guessing. The blue curve has an AUC less than 0.5, so the system has no value, it classifies more samples incorrectly than correctly.

42