**Engineering and Applied Science Programs for Professionals**
**Whiting School of Engineering**
**Johns Hopkins University**
**685.621 Algorithms for Data Science**

This document provides a rollup of the data transformation methods covered in this module. The following list of topics is covered in this module allowing for an understanding of how data is is processed to be used in a compact efficient manner for data representation.

# Contents

# 1 Introduction to Data Transformation

Data is everywhere. Nowadays, it is most likely in digital form that can be processed by a machine. The data transformation cycle is a series of operations on data, mainly including data collection, data input, data transformation, and the data output into a new space. Data transformation may include but is not limited to retrieval, transformation, and/or classification and hope to produce meaningful information. Figure 1 shows a generic example of a data processing system, or specifically a pattern recognition system.

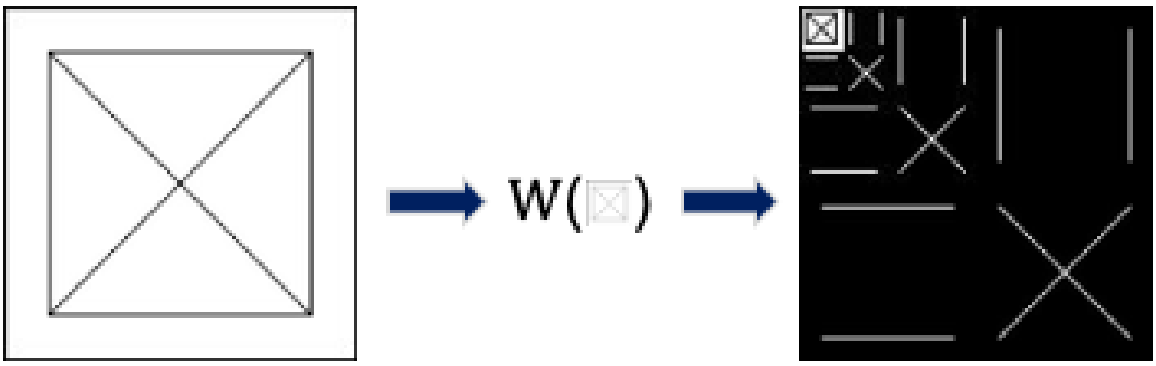



Figure 1: 2-Level wavelet example with an input image containing vertical, diagonal and horizontal edges passed through the wavelet transformed showing the output transform.

In this example of data transformation the input image in Figure 1 is used to show a visual representation of a 2-level wavelet. There are several forms of data transformation which include the mapping of the data from an input space to a new space. While there are several goals for data transformations in the study of data science, specifically in this course, the goal is to represent a larger data set by a small number of values representing the data. This is shown in Figure 2

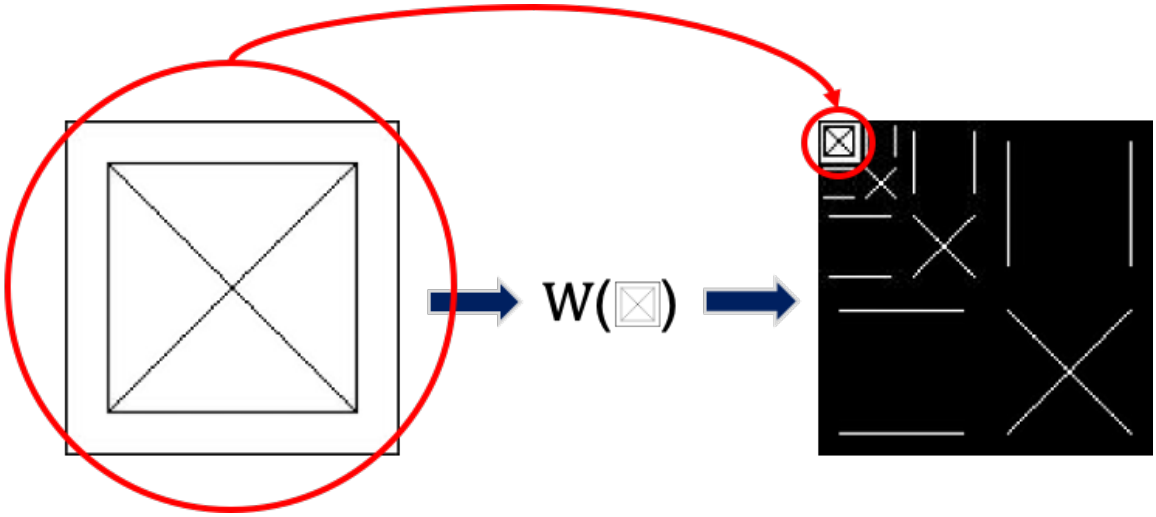



Figure 2: 2-Level wavelet example mapping from a larger input space to a small space allowing for the image to be represented by $1/16^{th}$ the size.

The wavelet example is meant to show how a transformation can be used to show the image in a much smaller format. Careful analysis of the transform shows that the mapping also allows extraction of the vertical, diagonal and horizontal edges of the input image.

## 2 Data Transformations

Now let's discuss how we represent raw data in a manageable form. Features/attributes where large datasets, such as images, can be exploited with a much smaller set of features. We will call this feature generation, where the basic concept of generating features is to transform a given audio, data, image, or video file which contains an extensive number of data values, into a new set of features that will allow the original data files to be represented in a much smaller set of values. It should be noted that the transform alone may not be the only data mapping step in generating the features. This will be shown in the generating features section. If the transform will be discussed later) is suitably chosen the transform domain values can exhibit informational properties about the original input data in a compact vector form. This means that most of the related information is compressed in a relatively small number of values leading to a reduced feature space (Theodoridis and Koutroumbas, 2006). For example, consider the input image in Figure 1 and Figure 2 that is of size 512x512 pixels. This image would contain 262,144 pixel values, mapping the image into a new domain with the use of a wavelet transform can be represented as a much smaller 64x64 image with a significantly smaller number of pixels, 4096. In the new transform space the 4096 values can be used as features. The basic reasoning behind transform-based features is that an appropriate chosen transform can exploit and remove redundancies that usually exist in digital images (Theodoridis and Koutroumbas, 2006). Consider the problem of character recognition, an input image that has been taken can be represented with a much smaller set of values to determine what the actual hand written character is. In the case of JPEG images a compression technique is used which is based on the discrete cosine transform (DCT). Generating features for discriminating between the numbers 0 - 9 using the DCT will eliminate redundant pixel information. When generating features derived from calculating the DCT, most of the energy lies in the frequency bands of the coefficients providing important information for class discrimination. This however leads to a large number of features, which for classification accuracy must be reduced. The data transform methods used in this document are Huffman codes, principal component analysis (PCA), the Karhunen-Loève Transform (KLT), wavelet transform, discrete cosine transform (DCT), and the fast Fourier transform (FFT). While this is not a complete set of transforms, it is a good introduction into the use of transforms for reducing dimensionality as well as generating features.

## 3 Huffman Codes

In computer science and information theory, Huffman coding is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol. It was developed by David A. Huffman while he was a Ph.D. student at MIT, and published in the 1952 paper "A Method for the Construction of Minimum-Redundancy Codes".

Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code (sometimes called "prefix-free codes") (that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol) that expresses the most common characters using shorter strings of bits than are used for less common source symbols. Huffman was able to design the most efficient compression method of this type: no other mapping of individual source symbols to unique strings of bits will produce a smaller average output size when the actual symbol frequencies agree with those used to create the code. A method was later found to do this in linear time if input probabilities (also known as weights) are sorted.

For a set of symbols with a uniform probability distribution and a number of members which is a power of two, Huffman coding is equivalent to simple binary block encoding, e.g., ASCII coding. Huffman coding is such a widespread method for creating prefix codes that the term "Huffman code" is widely used as a synonym for "prefix code" even when such a code is not produced by Huffman's algorithm.

Although Huffman coding is optimal for a symbol-by-symbol coding (i.e. a stream of unrelated symbols) with a known input probability distribution, its optimality can sometimes accidentally be over-stated. For example, arithmetic coding and LZW coding often have better compression capability. Both these methods can combine

an arbitrary number of symbols for more efficient coding, and generally adapt to the actual input statistics, the latter of which is useful when input probabilities are not precisely known or vary significantly within the stream. In general, improvements arise from input symbols being related (e.g., "cat" is more common than "cta").

## 3.1 History

In 1951, David A. Huffman and his MIT information theory classmates were given the choice of a term paper or a final exam. The professor, Robert M. Fano, assigned a term paper on the problem of finding the most efficient binary code. Huffman, unable to prove any codes were the most efficient, was about to give up and start studying for the final when he hit upon the idea of using a frequency-sorted binary tree and quickly proved this method the most efficient.

In doing so, the student outdid his professor, who had worked with information theory inventor Claude Shannon to develop a similar code. Huffman avoided the major flaw of the suboptimal Shannon-Fano coding by building the tree from the bottom up instead of from the top down.

## 3.2 Basic technique

A source generates 4 different symbols a1,a2,a3,a4 with probability 0.4;0.35;0.2;0.05. A binary tree is generated from left to right taking the two less probable symbols, putting them together to form another equivalent symbol having a probability that equals the sum of the two symbols. The process is repeated until there is just one symbol. The tree can then be read backwards, from right to left, assigning different bits to different branches.
The final Huffman code is:
Symbol Code
a1 0
a2 10
a3 110
a4 111
The standard way to represent a signal made of 4 symbols is by using 2 bits/symbol, but the entropy of the source is 1.73 bits/symbol. If this Huffman code is used to represent the signal, then the average length is lowered to 1.85 bits/symbol; it is still far from the theoretical limit because the probabilities of the symbols are different from negative powers of two.The technique works by creating a binary tree of nodes. These can be stored in a regular array, the size of which depends on the number of symbols, n. A node can be either a leaf node or an internal node. Initially, all nodes are leaf nodes, which contain the symbol itself, the weight (frequency of appearance) of the symbol and optionally, a link to a parent node which makes it easy to read the code (in reverse) starting from a leaf node. Internal nodes contain symbol weight, links to two child nodes and the optional link to a parent node. As a common convention, bit '0' represents following the left child and bit '1' represents following the right child. A finished tree has n leaf nodes and n - 1 internal nodes.

The process essentially begins with the leaf nodes containing the probabilities of the symbol they represent, then a new node whose children are the 2 nodes with smallest probability is created, such that the new node's probability is equal to the sum of the children's probability. With the previous 2 nodes merged into one node (thus not considering them anymore), and with the new node being now considered, the procedure is repeated until only one node remains, the Huffman tree.

The simplest construction algorithm uses a priority queue where the node with lowest probability is given highest priority:

1.Create a leaf node for each symbol and add it to the priority queue. 2.While there is more than one node in the queue: 1.Remove the two nodes of highest priority (lowest probability) from the queue 2.Create a new internal node with these two nodes as children and with probability equal to the sum of the two nodes' probabilities. 3.Add the new node to the queue. 3.The remaining node is the root node and the tree is complete. Since efficient priority queue data structures require $O(\log n)$ time per insertion, and a tree with n leaves has $2n-1$ nodes, this algorithm operates in $O(n \log n)$ time.

If the symbols are sorted by probability, there is a linear-time ($O(n)$) method to create a Huffman tree using two queues, the first one containing the initial weights (along with pointers to the associated leaves), and combined weights (along with pointers to the trees) being put in the back of the second queue. This assures that the lowest weight is always kept at the front of one of the two queues:

1.Start with as many leaves as there are symbols. 2.Enqueue all leaf nodes into the first queue (by probability in increasing order so that the least likely item is in the head of the queue). 3.While there is more than one node in the queues: 1.Dequeue the two nodes with the lowest weight by examining the fronts of both queues. 2.Create a new internal node, with the two just-removed nodes as children (either node can be either child) and the sum of their weights as the new weight. 3.Enqueue the new node into the rear of the second queue. 4.The remaining node is the root node; the tree has now been generated. It is generally beneficial to minimize the variance of codeword length. For example, a communication buffer receiving Huffman-encoded data may need to be larger to deal with especially long symbols if the tree is especially unbalanced. To minimize variance, simply break ties between queues by choosing the item in the first queue. This modification will retain the mathematical optimality of the Huffman coding while both minimizing variance and minimizing the length of the longest character code.

### 3.3 Algorithm

Huffman Algorithm

```
1   n ← 0
2   Q ← C
3   for i ← 1 to n − 1
4       do allocate a new node z
5          left[z] ← x ← Extract-Min[Q]
6          right[z] ← y ← Extract-Min[Q]
7          f[z] ← f[x] + f[y]
8          Insert(Q, z)
9   return Extract-Min(Q)
```

### 3.4 Applications

Arithmetic coding can be viewed as a generalization of Huffman coding; indeed, in practice arithmetic coding is often preceded by Huffman coding, as it is easier to find an arithmetic code for a binary input than for a nonbinary input. Also, although arithmetic coding offers better compression performance than Huffman coding, Huffman coding is still in wide use because of its simplicity, high speed and lack of encumbrance by patents.

Huffman coding today is often used as a "back-end" to some other compression method. DEFLATE (PKZIP's algorithm) and multimedia codecs such as JPEG and MP3 have a front-end model and quantization followed by Huffman coding. While the Huffman codes are used to transform data into binary values for compression the same techniques can be used for representing text data files in the new transformed space for identifying patterns.

## 4 Principal Component Analysis (PCA) and the Karhunen-Loève Transform (KLT)

In this section the differences and uses for generating features is discussed using Principal Component Analysis and the Karhunen-Loève Transform. In PCA, there are assumptions made in the derivation that must be accounted for when scaling of the variables and the applicability. We will see later that PCA can be used for dimensionality reduction, however, caution should be used to ensure information representing the data is preserved in the principal components. In the case of the Karhunen-Loève transform, the coefficients are random variables. The orthogonal basis functions in the KLT are determined by the covariance function and how it is processed. In this section, the goal is to generate features that are uncorrelated. Keep in mind that this is not a full explanation of the differences between PCA and KLT, rather an introduction is presented to ensure the use of PCA for dimensionality and KLT for generating features is introduced.

### 4.1 Principal Component Analysis (PCA)

The idea of feature extraction using PCA (Hotelling, 1933) is to represent a new space in a way to extract mutually uncorrelated features from the current space. The new features are known as the principal components after trans-

form mapping. The dimensionality assessment is accomplished by extracting the principal components from the correlation matrix and retaining only the factors described in Kaiser's criterion (eigenvalues: $\lambda \geq 1$) (Kaiser, 1960). The criterion is used as a guide line to determine the number of principal components to retain by calculating the correlation matrix of the input features. Each observed variable contributes one unit of variance to the total variance in the data set. Hence, any principal component that has an eigenvalue, $\lambda$ greater than one accounts for a greater amount of variance than had been contributed by one variable. Additionally, a principal component that displays an eigenvalue less than one indicates less variance than had been contributed by one variable. The covariance matrix, $\Sigma$, is used to extract eigenvectors, e, retaining only the number of principal components corresponding to Kaiser's criterion.

The basic concept of feature extraction using PCA is to map x onto a new space capable of reducing the dimensionality of the input space. The data is partitioned by variance using a linear combination of 'original' factors. To perform PCA, let $\mathbf{x} = [x_1, x_2, \ldots, x_n] \in X_n$ be a set of training vectors from the n-dimensional input space $X_n$. The set of vectors $\hat{\mathbf{x}} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_m] \in \hat{X}_m$ is a lower dimensional representation of the input training vectors $\mathbf{x}$ in the m-dimensional space $\hat{X}_m$. The vectors $\hat{x}$ are obtained by the linear orthonormal projection

$$\hat{\mathbf{x}} = \mathbf{A}^T(\mathbf{x} - \mu) \tag{1}$$

where $\mathbf{A}$ is an $[n \times m]$ matrix containing the top m eigenvectors and $\mu$ is the mean of the each set of features from $\mathbf{x}$.

### 4.1.1 Karhunen-Loève Transform (KLT)

***NOTE: The below writeup is from Theodoridis and Koutroumbas, 2006, pp. 266-270.***

Let

$$\hat{\mathbf{x}} = \mathbf{A}^T\mathbf{x} \tag{2}$$

From the definition of the correlation matrix we have

$$R_y \equiv E[\mathbf{y}\mathbf{y}^T] = E[\mathbf{A}^T\mathbf{x}\mathbf{x}^T\mathbf{A}] = \mathbf{A}^T R_x \mathbf{A} \tag{3}$$

Where $R_x$ is a symmetric matrix, and hence its eigenvectors are mutually orthogonal. Thus, if the matrix $A$ is chosen so that its columns are the orthonormal eigenvectors $\mathbf{a}_i$, $i = 0, 1, \ldots, N-1$, of $R_x$, then $R_y$ is diagonal

$$R_y = \mathbf{A}^T R_x \mathbf{A} = \Lambda \tag{4}$$

where $\Lambda$ is the diagonal matrix having as elements on its diagonal the respective eigenvalues $\lambda_i$, $i = 0, 1, \ldots, N-1$, of $R_x$. Furthermore, assuming $R_x$ to be positive definite the eigenvalues are positive. The resulting transform is known as the Karhunen-Loève transform, and it achieves our original goal of generating mutually uncorrelated features. The KLT is of fundamental significance in pattern recognition and in a number of signal and image processing applications. Let us look at some of its important properties.

*Mean square error approximation.* From the definition of unitary matrices we have

$$\mathbf{x} = \sum_{i=0}^{N-1} y(i)\mathbf{a}_i \tag{5}$$

and

$$y(i) = \mathbf{a}_i^T\mathbf{x} \tag{6}$$

Let us now define a new vector in the $m$-dimensional subspace

$$\hat{\mathbf{x}} = \sum_{i=0}^{m-1} y(i)\mathbf{a}_i \tag{7}$$

where only $m$ of the basis vectors are involved. Obviously, this is nothing but the projection of $\mathbf{x}$ onto the subspace spanned by the $m$ (orthonormal) eigenvectors involved in the summation. If we try to approximate $\mathbf{x}$ by its projection $\hat{\mathbf{x}}$, the resulting mean square error is given by

$$E\big[\|\mathbf{x}-\hat{\mathbf{x}}\|^2\big] = E\left[\left\|\sum_{i=0}^{N-1} y(i)\mathbf{a}_i\right\|^2\right] \tag{8}$$

Our goal now is to choose the eigenvectors that result in the minimization MSE. From the above equation and taking into account the orthonormality property of the eigenvectors, we have

$$E\left[\left\|\sum_{i=0}^{N-1} y(i)\mathbf{a}_i\right\|^2\right] = E\left[\sum_i\sum_j (y(i)\mathbf{a}_i^T)(y(j)\mathbf{a}_j)\right] = \sum_{i=m}^{N-1} E[y^2(i)] = \sum_{i=m}^{N-1} \mathbf{a}_i^T E[\mathbf{x}\mathbf{x}^T]\mathbf{a}_i \tag{9}$$

Combining the previous two equations and the eigenvector definition, we finally get

$$E\big[\|\mathbf{x}-\hat{\mathbf{x}}\|^2\big] = \sum_{i=m}^{N-1} \mathbf{a}_i^T\lambda_i\mathbf{a}_i = \sum_{i=m}^{N-1}\lambda_i \tag{10}$$

Thus, if we chose in equation (7) the eigenvectors corresponding to the $m$ largest eigenvalues of the correlation matrix, then the error in equation (10) is minimized, being the sum of the $N-m$ smallest eignevalues. Furthermore, it can be shown that this is also the minimum MSE, compared with any other approximation of $x$ and an $m$-dimensional vector. This is the reason the KLT is also know as PCA.

A difference from the KLT results if we compute $A$ in terms of the eigenvectors of the covariance matrix. This transform diagonalizes the covariance matrix $\Sigma_y$,

$$\Sigma_y = A^T\Sigma_x A = \Lambda \tag{11}$$

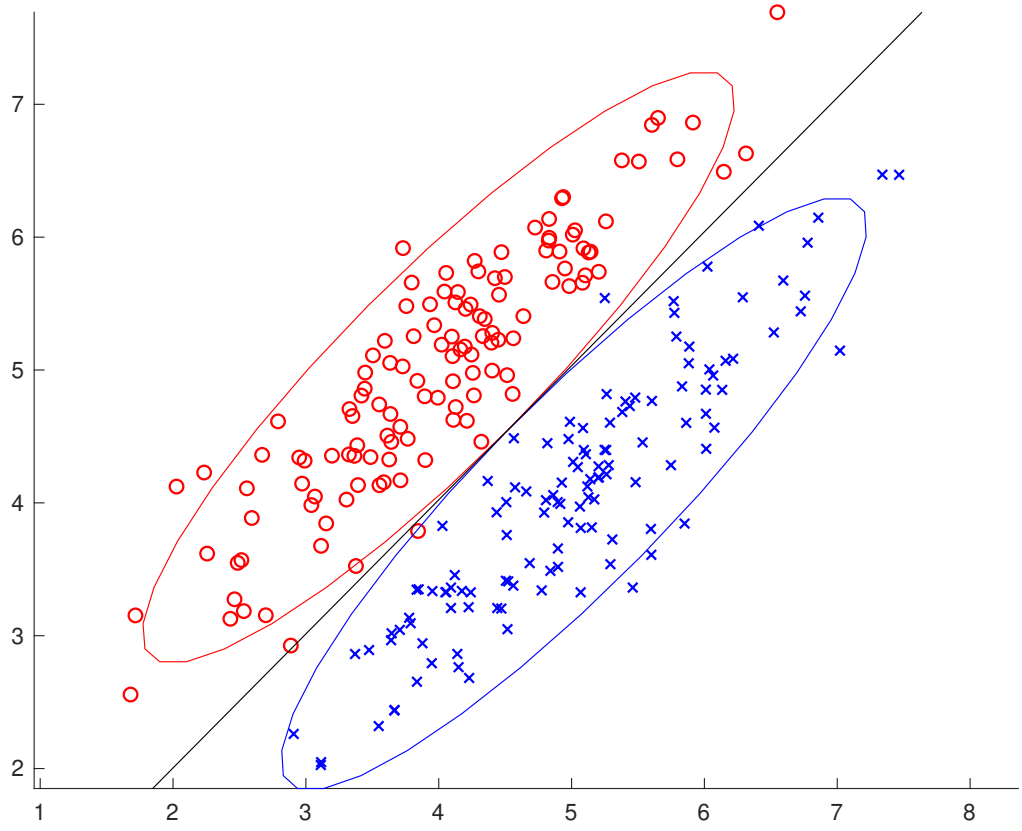
In general, the two are different and coincide for zero mean random vectors. In practice this is usually the case, because if it is not true one can replace each vector by $\mathbf{x}-E\mathbf{x}$. Despite that, it is still interesting to point out a difference between the two varants of the KLT. It can be shown that in this case, the resulting orthonormal basis ($\Sigma_x$ eigenvectors $\hat{a}_i$) guarantees that the mean square error beetween $\mathbf{x}$ and its approximation given by

$$\hat{\mathbf{x}} = \sum_{i=0}^{m-1} y(i)\hat{\mathbf{a}}_i + \sum_{i=m}^{N-1} E\big[y(i)\hat{\mathbf{a}}_i\big], \qquad y(i)\equiv\hat{\mathbf{a}}_i^T\mathbf{x} \tag{12}$$

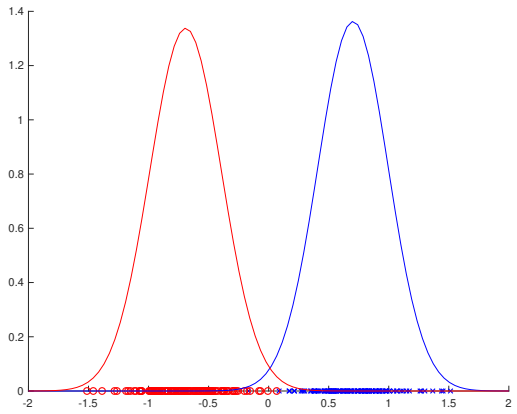
in minimum. In words, the last $N-m$ components are not random but are frozen to their respective mean value.

The optimality of the KLT, with respect to the MSE approximation, leads to excellent information packing properties and offers us a tool to select the $m$ dominant features out of $N$ measurement samples. However, although this may be a good criterion, in may cases it does not necessarily lead to maximum class separability in the lower dimensional subspace. This is reasonable, since the dimensionality reduction is not optimized with respect to class separability. This is demonstrated via the example of Figure 3. The feature vectors in the two classes follow the Gaussian distribution with the same covariance matrix.

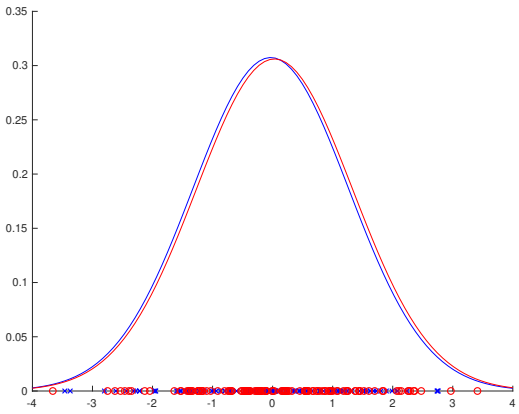
The ellipses show the curves of constant pdf values. We have computed the eigenvectors of the overall correlation matrix, and the resulting eigenvectors are shown in the figure. Eigenvector $\mathbf{a}_i$ is the one that corresponds to the largest eigenvalue. It does not take time for someone to realize that projection on $\mathbf{a}_1$ makes the two classes almost coincide. However, projecting on $\mathbf{a}_2$ keeps the teo classes separable.




(a) Generated Data




(b) Smaller eigenvector projection




(c) Larger eigenvector projection

Figure 3: The KLT is not always best for pattern recognition. In this example, projection on the eigenvector with the larger eigenvalue makes the teo classes coincide. On the other hand, projection on the other eigenvector keeps the class separated.

**Total Variance** Let $E[\mathbf{x}]$ be zero. If this is not the case, the mean can always be subtracted. Let $\mathbf{y}$ be the KLT vector of $\mathbf{x}$. From the respective definitions we have that $\sigma^2_{y(i)} \equiv E[y^2(i)] = \lambda_i$. That is, the eigenvalues of the input correlation matrix are equal matrix are equal to the variances of the transformed features. Thus, selecting those features, $y(i) \equiv \mathbf{a}_i^T\mathbf{x}$, corresponding to the $m$ largest eigenvalues makes their sum variance $\sum_i \lambda_i$ maximum. In other words, the selected $m$ features retain most of the total variance associated with the original random variables $x(i)$. Indeed, the latter is equal tot he trace of $R_x$, which we know from linear algebra to be equal to the sum of the eigenvalues $\sum_{i=0}^{N-1} \lambda_i$. It can be shown that this is a more general property. That is, from all possible sets of $m$ features, obtained via any orthogonal linear transformation on $\mathbf{x}$, the ones resulting from the KLT have the largest sum variance.

**Entropy** We know that the entopy of a process is defined as

$$H_y = -E[\ln p_y(\mathbf{y})] \tag{13}$$

and it is a measure of the randomness of the process. For a zero mean Gaussian multivariable $m$-dimensional process the entropy becomes

$$H_y = \frac{1}{2}E[\mathbf{y}^T R_y^{-1}\mathbf{y}] + \frac{1}{2}\ln|R_y| + \frac{m}{2}\ln(2\pi) \tag{14}$$

However,

$$E[\mathbf{y}^T R_y^{-1}\mathbf{y}] = E[\text{trace}(\mathbf{y}^T R_y^{-1}\mathbf{y})] = E[\text{trace}(R_y^{-1}\mathbf{y}\mathbf{y}^T)] = \text{trace}(I) = m \tag{15}$$

and using the known property from linear algebra the determinant is

$$\ln|R_y| = \ln(\lambda_0\lambda_1 \cdot \lambda_{m-1}) \tag{16}$$

In words, selection of the $m$ features that correspond to the $m$ largest eigenvalues maximizes the entropy of the process. This is expected, because variance and randomness are directly related.

### 4.1.2 Number of Components to Retain (Dillon and Goldstein, 1984)

Principal component analysis is frequently employed for the purpose of generating a reduces set of variates that account for most of the variability in the original data, and that can be used in more substantial subsequent analysis. WE must therefore decide just how many components to retain. Unfortunately there is not universally accepted method for doing so. The decision is largely judgmental and a matter of taste.

A number of procedures for determining how many components to retain have been suggested. These "rules" range from methods that evoke formal significance tests to less formal approaches involving heuristic graphical arguments. The remainder of this section discusses several of the more commonly used procedures.

**Variance - Covariance Input** A reasonable approach for determining the number oof components to retain when factoring a variance-covariance matrix relies on the sampling distribution results. For example, we might suggest that only those components whose associated eigenvalues are statistically different from zero be retained.

The reader should note, however, that with even moderate sample sizes many of the components will typically be statistically significant. Yet, from a practical viewpoint, some of these significant components account for only a vary small proportion of the total variance. Hence, for reasons of parsimony and practical significance, the statistical criteria should be interpreted loosely with fewer components retained than are statistically significant. In this regard,, some of the graphical procedures soon to be discussed may prove useful.

An alternative and somewhat more ad hoc approach is the percentage of variance criterion. With this approach, the cumulative percentage of eh variance extracted by successive components is the criterion. Note, the cumulative proportion of total variance extracted by a set of components sis given by

$$\frac{\sum_{j=}^{m} l_{(j)}}{\sum_{j=}^{p} l_{(j)}} \tag{17}$$

where $m < p$. The stopping rule is subjective, since the number of components retained is based on some arbitrary determined criterion for the amount of variation accounted for.

**Correlation Input** When factoring a correlation matrix, statistical testing procedures no longer apply, and the retained variance criterion lacks clear meaning. For these reasons, various graphical heuristics have been suggested as a rule of thumb.

Perhaps the most frequent used extraction approach is the "root greater than one" criterion. Originally suggested by Kaiser (1958), this criterion retains those components whose eigenvalues are greater than one. The dimensionality assessment is accomplished by extracting the principal components from the correlation matrix and retaining only the factors described in Kaiser's criterion (eigenvalues: $\lambda \geq 1$). Principal component analysis partitions the data by variance using linear combination of 'original' factors. The rationale for this criterion is that any component should account for more "variance" than any single variable in the standardized test score space.

Another approach, proposed by Cattell (1966), is called the scree test. With this approach, named after the rubble at the bottom of a cliff, the eigenvalues of each component are plotted in successive order of their extraction, and then an elbow in the curve is identified by applying, say, a straightedge to the bottom portion of the eigenvalues to see where they form an approximate straight line. The number of components retained is given by the point at which the components curve above the straight line formed by the smaller eigenvalues. Cattell and Jaspers (1967) suggested that the number of factors be taken as the number immediately before the straight line begins. Figure 4 shows a hypothetical case which four components would be retained. Note that the graph clearly depicts the scree analogy - the first few eigenvalues show the cliff and the rest the rubble. The rationale for the scree test is simple: since the principal component solution extracts components in successive order of magnitude, the substantive factors appear first, followed by the numerous trivial components which account for only a small proportion of the total variance.

Though this approach is relatively simple, complications can surface. First, there may be no obvious break, in which case the test is inclusive. Second, There may be several breaks. This is particularly troublesome when, say, two breaks occur among the first half of the eigenvalues, since it will be difficult to decide which of the breaks reflects the correct number of components.

Horn (1965) has suggested yet another approach for determining the number of components to retain which removes much of the ambiguity associated with the scree test. As in the scree test, the data are factored and the resulting component eigenvalues are plotted in successive order of their magnitude. Next, $K$ setts of $n \times p$ normally and independently distributed (NID) random variates are sampled from a population whose correlation structure is known to be characterized by an identity matrix. Each $n \times p$ data matrix is factored. The average eigenvalue for each extracted component over the $k$ sets of randomly generated data s then computed and plotted in the same graph with the real data. Because of sampling variation, several of the extracted eigenvalues may exceed unity. However, the average curve of eighenvalues can be expressed to cross the ordinate scale at the value 1.0 for component number $p/2$. The principal component solution for the simulated data represents the case in which the eigenvalues under the null hypothesis are unity. Consequently, Horn's criterion is to retain components n the basis of where the reference curve crosses the curve induced from the actual data. **Figure xx** shows a hypothetical application of Horn's criterion based on $p = 30$ variables.

Great Online Resource: http://sebastianraschka.com/Articles/2014_pca_step_by_step.html

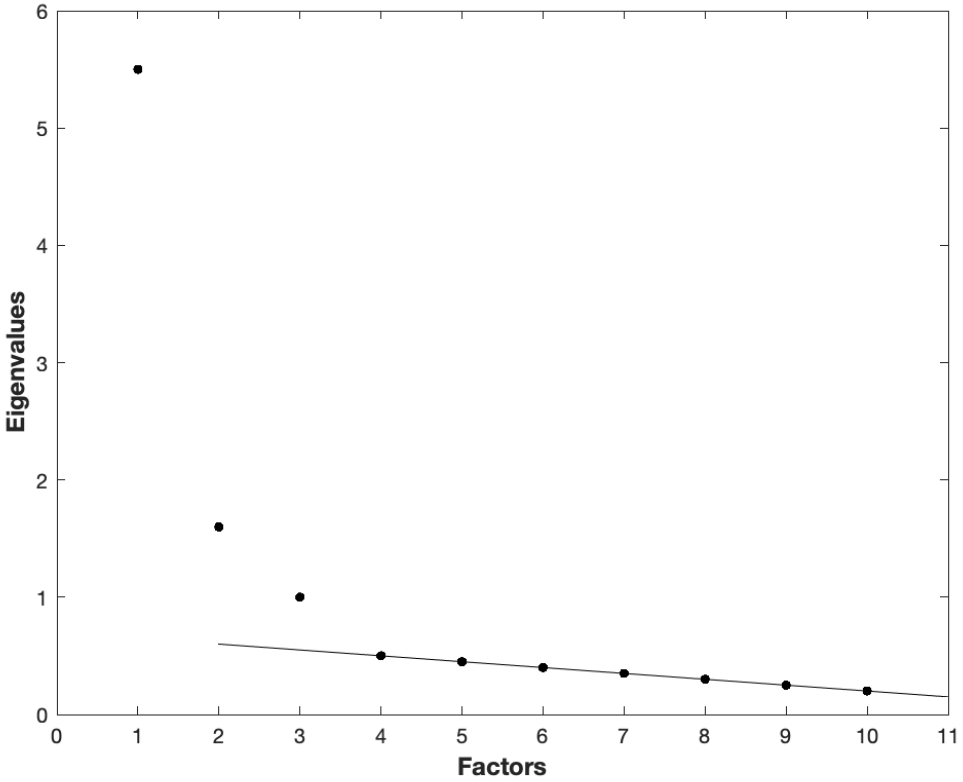
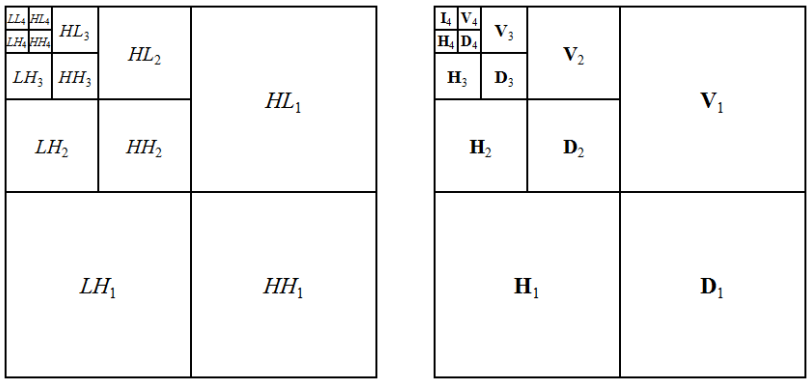



Figure 4: Illustration of the scree test.

# 5 Wavelets

The wavelet transform is a great method for compressing image and audio files. It is also a great for identifying image vertical, diagonal and horizontal characteristics. Research has been employed using wavelet transforms as a preprocess for deriving image features. The image decomposition employed here is based on separable quadrature mirror filters (QMF) originally used in (Farid, 2002). Applying the wavelet transform on a given image maps the image pixel from the spatial domain to the wavelet transform domain, i.e., to the frequency space. The image decomposition splits the frequency space into multiple orientations and levels. For each level, four subbands, $LL$, $LH$, $HL$, and $HH$, are created inheriting the characteristics of the image, identical (**I**), vertical (**V**), horizontal (**H**) and diagonal (**D**) information, correspondingly. The symbol $L$ is denoted as a lowpass filter being used, while $H$ would be the case of a highpass filter.




(a) Subband information (b) Edge information

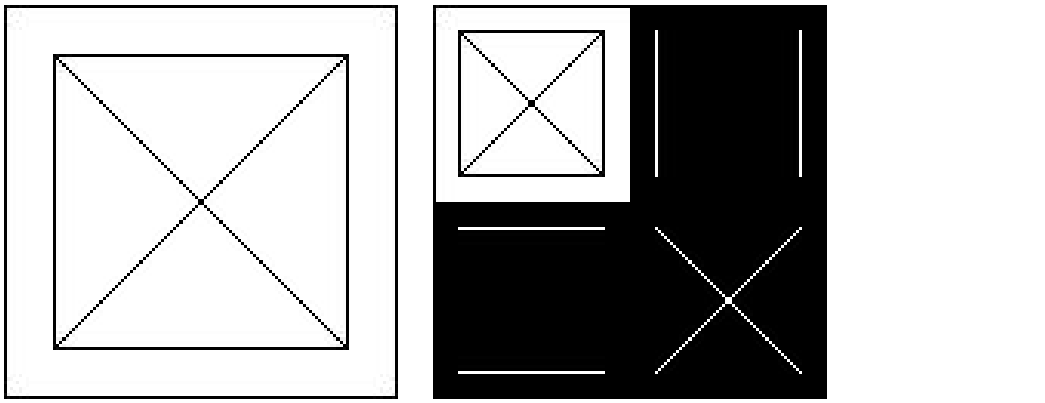
Figure 5: A four-scale wavelet decomposition

The number of iterations applied, also named as a scale or level, is denoted by a subscript $i$.

Recursively applying QMF to $LL_i$, Figure 5 presents an $i^{th}$-scale wavelet decomposition, where $i = 1, 2, 3, 4$. Suppose, there is a set of $n$ images. For each image, a four-scale three-orientation QMF is utilized. A coefficient located within a wavelet transformed image is denoted by its edge characteristics and its position, i.e., $v_{j,i}(x, y)$, $h_{j,i}(x, y)$, $d_{j,i}(x, y)$, where $i = 1, 2, 3, 4$, $j = 1, 2, \ldots, n$, and $(x, y)$ is the row and column index within a characteristic block. The coordinate $(x, y)$ denotes the coefficient location within a block. Given an example image with vertical, horizontal, and diagonal lines as in Figure 6(a), Figure 6(b) to 6(e) demonstrate the wavelet decomposition results for each iteration from 1 to 4. The various scales of the wavelet coefficients are normalized for visualization purposes in Figure 6.

# 6 Discrete Fourier Transforms (DFT)

The Discrete Fourier transform (DFT) is the most widely used application in the fields of science and engineering, such as mathematics (linear systems, random process, probability, boundary-value systems), physics (quantum mechanics, optics, acoustics, astronomy), chemistry (spectroscopy, crystallography), and engineering (digital signal and image processing, telecommunications, computer vision). The theory of trigonometric series can be dated back to the beginning of 18th century. In 1747, Euler represented the movements of the planets in the form of a trigonometric series, which actually contained what is now called the Fourier series (Euler, 1760). In 1754, d'Alambert represented the reciprocal value of the mutual distance of two planets as a series of cosine functions (d'Alambert, 1754). H. H. Goldstine attributed to Carl Friedrich Gauss an algorithm, developed in 1805, similar to the fast Fourier transform (FFT) for the computation of the coefficients of a finite Fourier series (Goldstine, 1977). In 1807,



(a) Scale 0 (b) Scale 1

(c) Scale 2 (d) Scale 3 (e) Scale 4

Figure 6: Wavelet decompositions on an example image

Fourier discovered that a wide class of signals could be generated by summing scaled sine and cosine functions. These early techniques provided ideal tools for analyzing both periodic signals and the more general class of stationary signals. In 21st century, discrete Fourier transform remains one of the most frequently applied tools in science and technology. This is also because of the development in fast Fourier transform algorithms. The history of FFT development, dates back to Gauss, can be found in ((Goldstine, 1977; Caglayan, 2008).

Consider the N-point discrete Fourier Transform (DFT) of a signal $x(n), n = 0, 1, 2, \cdot, N-1$ is defined by

$$Y(k) = \sum_{n=0}^{N-1} x(n)w(n,k), \qquad k = 0, 1, \cdot, N-1 \tag{18}$$

Similarly, the inverse Fourier transform of the sequence $x(n) = (x_1, x_2, \cdot, x_{N-1}), n = 0, 1, 2, \cdot, N-1$ can be given by

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} Y(k)w^{-1}(n,k), \qquad n = 0, 1, \cdot, N-1 \tag{19}$$

where $w(n,k)$ are the complex roots of unity, that is

$$w(n,k) = e^{-2i\pi k/N}, \qquad n, k = 0, 1, \cdot, N-1 \tag{20}$$

Both equations (18) and (19) can be expressed in vector matrix form as

$$Y_N = \mathbf{F_N} x_N \tag{21}$$

$$x_N = \frac{1}{N} \mathbf{F_N^{-1}} Y_N \tag{22}$$

DFT matrix of size $N \times N$ composed of the twiddle factors as:

$$\mathbf{F_N} = \begin{bmatrix} w(0,0) & w(0,1) & w(0,2) & \cdots & w(0,N-1) \\ w(1,0) & w(1,1) & w(1,2) & \cdots & w(1,N-1) \\ w(1,0) & w(1,1) & w(1,2) & \vdots & w(2,N-1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w(N-1,0) & \cdots & \cdots & \cdots & w(N-1,N-1) \end{bmatrix} \tag{23}$$

Figure 7 illustrates the basis function of the discrete Fourier transform when $N = 16$, in which DFT matrix coefficients in each row is represented by a linear combination of each element of Fourier space, i.e. sines and cosines.

The DFT is used in signal processing to identify frequencies of interest. The identification of frequencies allows the use of digital filters to eliminate noise and audio components outside of the frequency of interest. For example if human vocals are needed to be extracted a bandpass filter can be used to extract vocal components of audio file around 8k Hz. The DFT has been extended to 2-dimensional files such as images. The frequencies extracted using the 2-dimensional DFT used with images can smooth an image by removing high frequencies in an image. Caglayan in 2008 showed his work using the DFT in extracting features from a large volume of images for the use of determining images that have been modified from their original form..

# 7 Discrete Cosine Transforms (DCT)

The standard DCT used in JPEG compression has two properties, i.e., the directional and frequency distributions of $8 \times 8$ blocks within an image (Rao and Yip, 1990). In JPEG compression on a two dimensional (2-D) signal, the zig-zag scan shown in Figure 8(a) is used to take advantage of the frequency distributions of the DCT shown in Figure 8(b) (Brown and Shepherd, 1995, pp. 224). The DCT decomposition divides the coefficients into low, medium and high frequencies. Figure 8(c) shows the breakdown of the vertical, diagonal and horizontal directions of the coefficients. In this research both the frequencies and directions of the DCT are investigated to generate

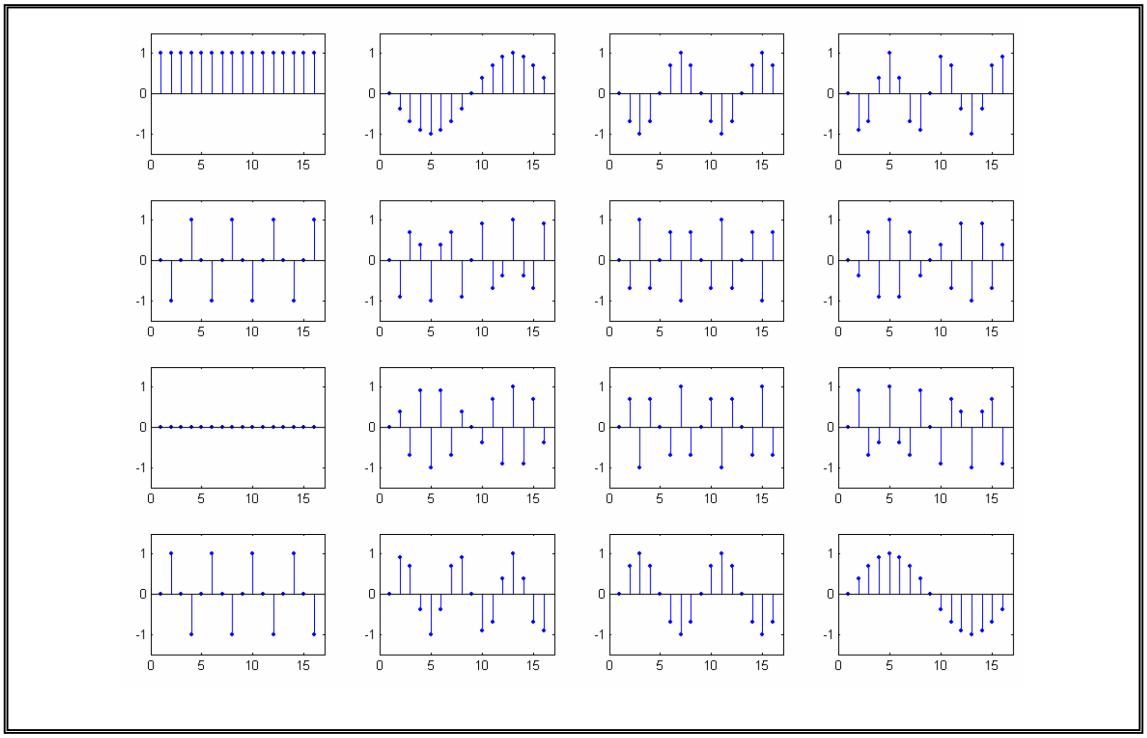


Figure 7: 16-point basis functions of discrete Fourier transform

features. Figure 8(d) shows an $8 \times 8$ image with a horizontal edge between black and white pixels. The corresponding 2-D DCT of Figure 8(d) is shown in Figure 8(g) which has coefficients that are prominent along the first column. In Figure 8(e) an image is shown with a diagonal edge between black and white pixels with a corresponding 2-D DCT shown in Figure 8(h) which has coefficients located along the diagonal. In Figure 8(f) an image is shown with a vertical edge between black and white pixels with a corresponding 2-D DCT shown in Figure 8(i) which has coefficients located along the first row.

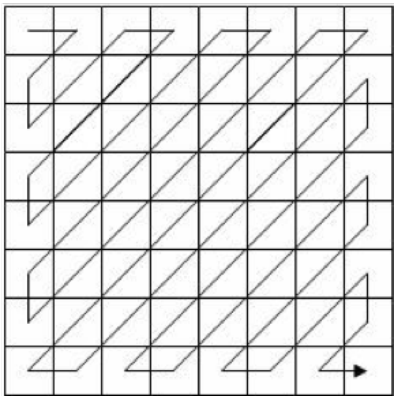
### 7.0.1 JPEG Image Representation Background

In this section, the basic structure of the JPEG image format and the steps in the compression process are described. This is followed by a brief introduction of JPEG image embedding methods.

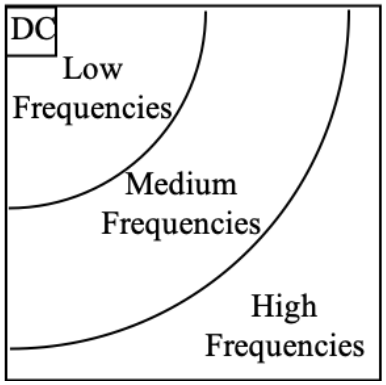
The Joint Photographic Experts Group (JPEfG) format uses lossy compression to achieve high levels of compression on images with many colors (Elysium Ltd., 2004). JPEG is an international standard for still image compression, and is widely used for compressing gray scale and color images. JPEG images are commonly used for storing digital photos, and publishing Web graphics; tasks for which slight reductions in the image quality are barely noticeable. Due to the loss of quality during the compression process, JPEGs should be used only where image file size is important (Murry and vanRyper, 1994; Brown and Shepherd, 1995).

The JPEG encoder, shown in figure below, performs compression with the following sequential steps: image preprocessing (divides the input image into $8 \times 8$ blocks), forward DCT of each $8 \times 8$ block, quantization with scaling factor, separation of DC and AC coefficients, prediction of the DC coefficient and zig-zag scan the AC coefficients and Huffman encoder (there is a separate encoder for the DC and AC coefficients).

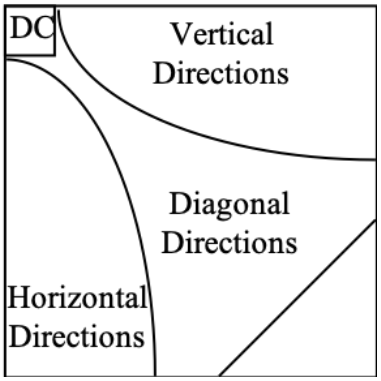
In JPEG decoding, all steps from the encoding process are reversed. The following procedure is a short description of the JPEG baseline systems.



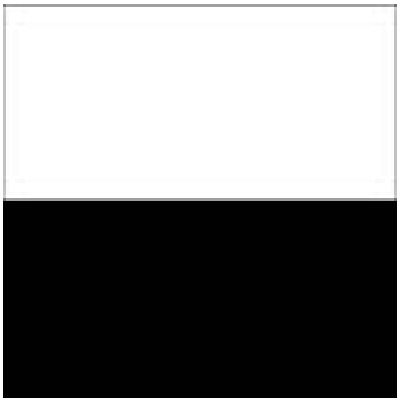
(a) JPEG Zig-Zag Pattern



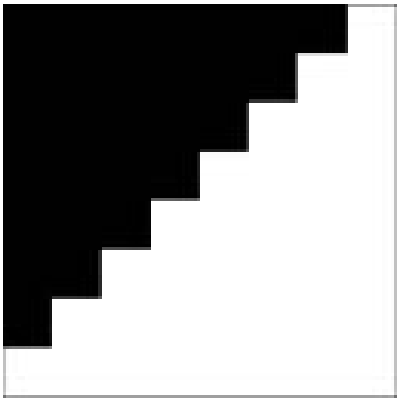

(b) DCT Frequencies



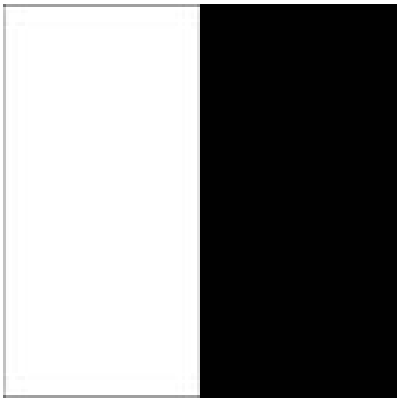

(c) DCT Directions



(d) Horizontal Image

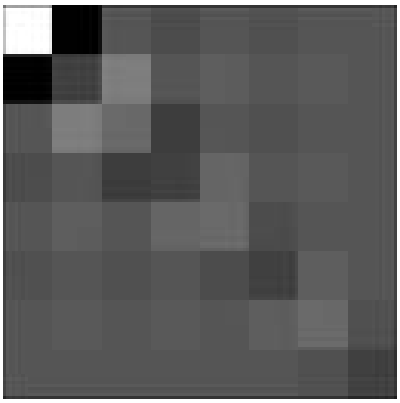


(e) Diagonal Image

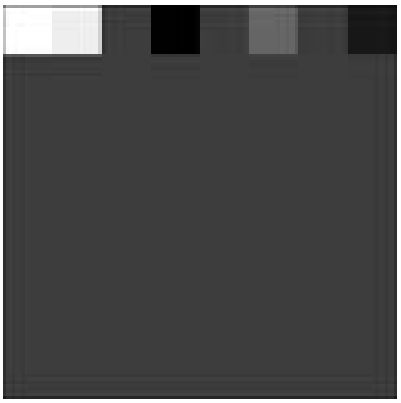


(f) Vertical Image

(g) DCT Horizontal



(h) DCT Diagonal



(i) DCT Vertical

Figure 8: DCT decomposition (a) zig-zag scan pattern (b) low, medium and high frequency distributions (c) vertical, diagonal and horizontal directions (d) $8 \times 8$ image with a horizontal edge between pixels (e) $8 \times 8$ image with a diagonal edge between pixels (f) $8 \times 8$ image with a vertical edge between pixels (g) 2-D DCT representation of horizontal image (h) 2-D DCT representation of diagonal image (i) 2-D DCT representation of vertical image.

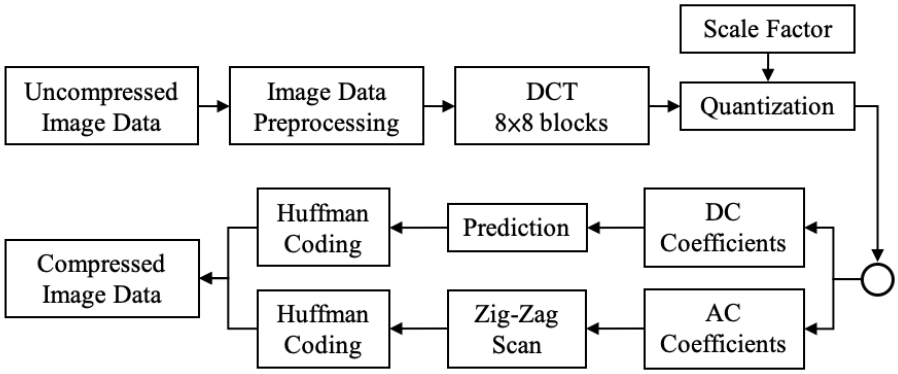



Figure 9: Block Diagrams of Sequential JPEG Encoder.

**Preprocessing block** - Subdivides the image into blocks of $8 \times 8$ pixels and level-shift the original pixel values from the range $[0, 225]$ to the range $[-128, +127]$ by subtracting 128. The shifting procedure is a preprocessing step for the DCT calculation.

**Forward DCT block** - Perform a two dimensional discrete cosine transform (DCT) on each level-shifted block B from the Preprocessing block step. The two dimension DCT is defined as

$$C(n_1, n_2, k_1, k_2) = \begin{cases} \frac{1}{\sqrt{N_1 N_1}} & \text{if } k_1 = k_2 = 0 \\ \frac{1}{\sqrt{N_1 N_2}} \cos\left(\frac{\pi(2n_1+1)k_1}{2N_1}\right) \cos\left(\frac{\pi(2n_2+1)k_2}{2N_2}\right) & \text{if } 1 \leq k_1 \leq N_1 - 1 \text{ and } 1 \leq k_2 \leq N_2 - 1 \end{cases} \tag{24}$$

where $0 \leq n_1 \leq N_1 - 1$ and $0 \leq n_2 \leq N_2 - 1$. Assuming that $N_2 = 1$, will give $k_2 = n_2 = 0$ or simply non existent. This will result in the following updated equation

$$C(n, k) = \begin{cases} \frac{1}{\sqrt{N}} & \text{if } k = 0 \\ \frac{1}{\sqrt{N}} \cos\left(\frac{\pi(2n+1)k}{2N}\right)\Big) & \text{if } 0 \leq n \leq N - 1 \text{ and } 1 \leq k \leq N - 1 \end{cases} \tag{25}$$

This provides an $N \times N$ matrix for use to transform an $N \times N$ input matrix. The transform is performed on the two dimensional matrix $B$ as $CBC^T$. In the case when matrix multiplication is not used the following equation can be used to take the two-dimensional discrete cosine transform of the matrix $B$

$$B\_DCT_{k_1,k_2} = b_{k_1} b_{k_2} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} B_{k_1,k_2} \cos\left(\frac{\pi(2n_1+1)k_1}{2N_1}\right) \cos\left(\frac{\pi(2n_2+1)k_2}{2N_2}\right) \tag{26}$$

for $1 \leq k_1 \leq N_1 - 1$ and $1 \leq k_2 \leq N_2 - 1$

where

$$b_{k_1} = \begin{cases} \frac{1}{\sqrt{N_1}} & \text{if } k_1 = 0 \\ \sqrt{\frac{2}{N_1}} & \text{if } 1 \leq k_1 \leq N_1 - 1 \end{cases} \tag{27}$$

and

$$b_{k_2} = \begin{cases} \frac{1}{\sqrt{N_2}} & \text{if } k_2 = 0 \\ \sqrt{\frac{2}{N_2}} & \text{if } 1 \leq k_2 \leq N_2 - 1 \end{cases} \tag{28}$$

The transform helps to remove data redundancy by mapping data from a spatial domain to the frequency domain. No compression has been achieved in this stage, but by changing representation of the information contained in the image block it makes the data more suitable for compression. In the JPEG encoding process, $N_1 = N_2 = 8$.

**Quantization** - Quantize the DCT coefficients block obtained from the previous step using the quantization table $Q$. The quantization table is a matrix used to divide the transformed block for compression purpose by reducing the amplitude of the DCT coefficient values and increasing the number of zero valued coefficients. The Huffman encoder takes advantage of these quantized values. When $Qs$ is represented the value s is a scalar multiple, called the scale (or quality) factor, which defines the amount of compression within the image. Higher values of $s$ yield higher compression. The below figure shows an instance of the typical quantization matrix $Qs$.

$$Qs = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 6 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{29}$$

A set of four quantization tables are specified by the JPEG standard (Independent JPEG Group, 1998). After quantization, most of the DCT coefficients in the $8 \times 8$ blocks are truncated to zero values. It is the principal of lossiness in the JPEG transform-based encoder.

**DC Coefficient Coding** - The first coefficient, coefficient 1 (upper left) in figure b) below is called the "DC coefficient", short for the direct current coefficient, and represents the average brightness (intensity) of the component block. To encode the DC coefficient, the JPEG standard utilizes a Huffman difference code table that categorizes the value according to the number of $k$ bits that are required to represent its magnitude. The value of the element is encoded with $k$ bits.

**AC Coefficients Coding** - The remaining 63 coefficients are the "AC coefficients", short for the alternating current coefficients. The Huffman code assigns short (binary) codewords to each AC coefficient. The AC coefficient encoding scheme is slightly more elaborate than the one for the DC coefficient. For each AC array, a run-length of 0 elements is recorded. When encountering a non-zero element, the length of $0s$ is recorded and the number of $k$ bits to represent the magnitude of the element is determined. The run-length and $k$ bits are used as a category in the JPEG default Huffman table for assigning a code.

Using a zig-zag run encoder converts the $8 \times 8$ array of DCT coefficients into a column vector of length $k$ (zig-zag goes from left to right and top to bottom). The "zig-zag" scan attempts to trace the DCT coefficients according to their significance, shown in figure below.

The coefficient ordering sequence is shown below.

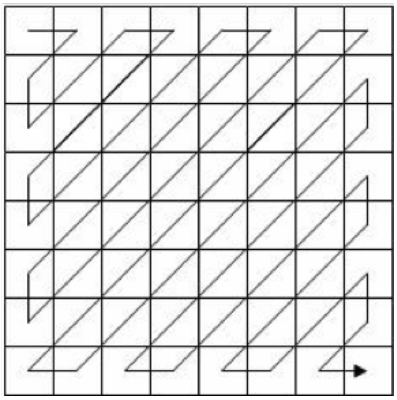


Figure 10: DCT decomposition zig-zag structure for an $8 \times 8$ block.

$$\begin{bmatrix} 1 & 2 & 6 & 7 & 15 & 16 & 28 & 29 \\ 3 & 5 & 8 & 14 & 17 & 27 & 30 & 43 \\ 4 & 9 & 13 & 18 & 26 & 31 & 42 & 44 \\ 10 & 12 & 19 & 25 & 32 & 41 & 45 & 54 \\ 11 & 20 & 24 & 33 & 40 & 46 & 53 & 55 \\ 21 & 23 & 34 & 39 & 47 & 52 & 56 & 61 \\ 22 & 35 & 38 & 48 & 51 & 57 & 60 & 62 \\ 36 & 37 & 49 & 50 & 58 & 59 & 63 & 64 \end{bmatrix} \tag{30}$$

The Huffman encoding reduces the number of bits needed to store each of the 64 integer coefficients. For example, when a true color uncompressed image of size $512 \times 512$ pixels is stored the file size is 769 kilobytes. However, this same image store as a JPEG at a quality factor of 75, the image is stored in 200 kilobytes or smaller. The Huffman encoding tables for the DC and AC coefficients can be found in Gonzalez and Woods (2007), Elysium Ltd. (2004), Independent JPEG Group (1998), and JPEG (1994).

One of the primary reasons using image embedding methods for creating stego files is due to the number of redundant portions within a digital image. The vast number of JPEG images on the Internet makes them ideal cover images for hiding secrete data and transmitting them as stego images. In JPEG steganography, the stego message is converted to binary values and embedded into DC and AC coefficients prior to Huffman encoding. By embedding at this stage, the stego message can be extracted without losing the message. The embedding methods range from simple embedding techniques that alter the least significant bits (LSB) of the coefficients such as JP Hide (Latham, 1999) and JSteg (Upham, 1993) to more complicated embedding techniques that maintain natural histograms of the coefficients such as; F5 (Westfeld, 2001; 2003), JP Hide (Latham, 1999), JSteg (Upham, 1993), Model-base (Sallee, 2003; 2006), Model-based Version 1.2 (Sallee, 2008a), OutGuess (Provos, 2004), Steganos (2008), StegHide (Hetzl, 2003) and UTSA (Agaian et al., 2006). The six tools selected provide a set of embedding methods that differ in embedding strategy. Investigation of these methods has provided an insight into six different and unique embedding capacities, embedding patterns and the appearance of the individual feature spaces. Another reason for selecting these particular tools is in previous research and existing steganalysis tools, these 6 embedding methods have been used for analysis (Provos and Honeyman, 2003; Lyu and Farid, 2004; Kharrazi et al., 2005; Shi et al., 2005; Xuan et al., 2005; Fu et al., 2006; Pevny and Fridrich, 2007).

In summary, a useful property of JPEG is that the degree of lossiness can be varied by adjusting the quality factor s (scale of the quantization table), shown in Figure 2.3. The ease of file sharing with JPEG images and its popularity over the internet has made JPEG image format a desirable cover file for many stego methods. Each embedding method leaves a signature that can be identified by various statistical measures. The next section describes feature generations methods used to identify changes made to a JPEG image.

# 8 Generating Features

In generating features it is important to understand the data being processed, images, video, audio, acoustic, radiometric, sonar, medical, banking, etc. In each of these data types the data can also be represented in various forms. Let consider an audio file, it can be represented as analog, various bits, frequencies, collection microphones, stereo, Dolby Digital, communication channel, etc. Based on the data type, the transformation needed will determine how well the features will represent the data in a compressed fashion. Let's consider taking the FFT of a standard 44.1kHz musical audio signal the vocals can be represented with a few real and imaginary coefficients.

## 8.1 Statistical methods (mean, moment, standard deviation, skewness, kurtosis)

When studying first order statistics, the use is clear in understanding the data mean, standard deviation and other statistics. By using the first order statistics an initial assessment give the data analyst a great deal of information. With the use of transforms (DFT, DCT, Wavelets, etc.) the statistical methods can be used to extract characteristics the transforms extract. The statistics calculated over the data vectors $\mathbf{x} = x_i = [x_1, x_2, \ldots, x_n] \in X_n$ are used to generate the features that represent the transformed raw data in a much small form. The table below lists five statistics: mean, standard deviation, skewness, and kurtosis along with their calculation.

Table 1: Statistics for Generating Features

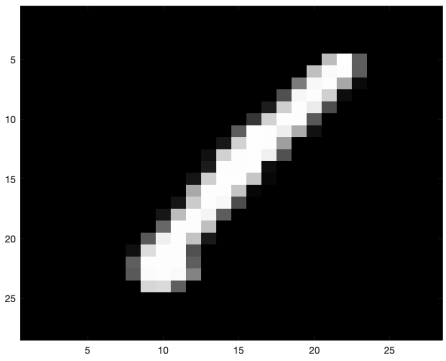
| **Test Statistics** | **Statistical Function $\boldsymbol{F}(\cdot)$** |
|---|---|
| Mean | $F_\mu(\mathbf{x}) = \mu(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} x_i$ |
| Standard Deviation | $F_\sigma(\mathbf{x}) = \left(\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \mu(\mathbf{x})\right)^2\right)^{1/2}$ |
| Skewness | $F_\gamma(\mathbf{x}) = \gamma(\mathbf{x}) = \dfrac{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \mu(\mathbf{x})\right)^3}{\sigma(\mathbf{x})^3}$ |
| Kurtosis | $F_\kappa(\mathbf{x}) = \kappa(\mathbf{x}) = \dfrac{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \mu(\mathbf{x})\right)^4}{\sigma(\mathbf{x})^4}$ |

The of both transforms and statistics allow a large data set to be reduce to a smaller set of data that is an order or two orders of magnitude smaller.

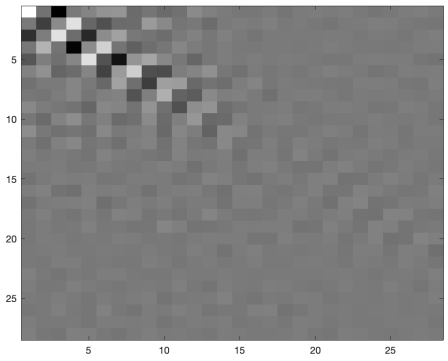
Figure 11 shows the numbers and the corresponding DCT. It should be noted how the diagonal and horizontal characteristics of the numerical data shows up on the 2-dimensional DCT images. Making a comparison with Figure 8, specifically, subfigures d, e, f, g, h and i, it can be seen that the vertical, horizontal and diagonal characteristics of the images come through clearly.

# 9 Dimensionality Reduction

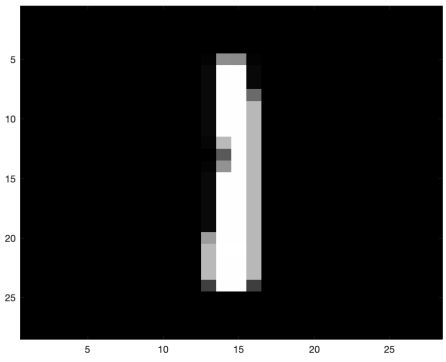
Another approach to reducing the dimension of the input features is to use a transformed space instead of the original feature space. For example using a transformation $f(x)$ that maps the data points $\mathbf{x}$ of the input space, $x[n]$, into a reduced dimensional space $x'[p]$, where $n > p$, creates features in a new space that may have better discriminatory properties. Classification is based on the new feature space rather than the input feature space. The advantage of feature extraction with the use of dimensionality reduction as described here over feature selection is that no information from any of the elements of the measurement vector is removed. In some situations feature extraction is easier than feature selection. A disadvantage of feature extraction is that it requires the determination of a suitable transformation $f(x)$. Some methods include principal component analysis (Hotelling, 1933; Dillon and



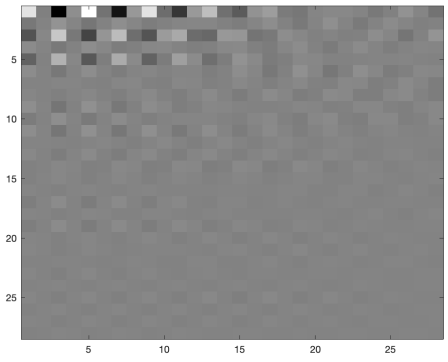

(a) Number 1 Index 1



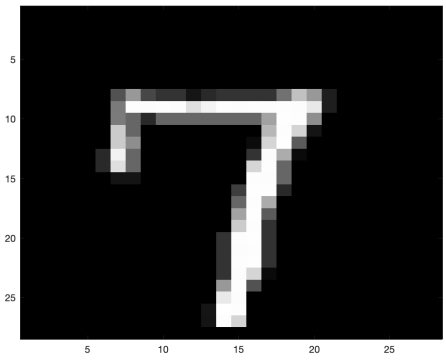

(b) Corresponding DCT



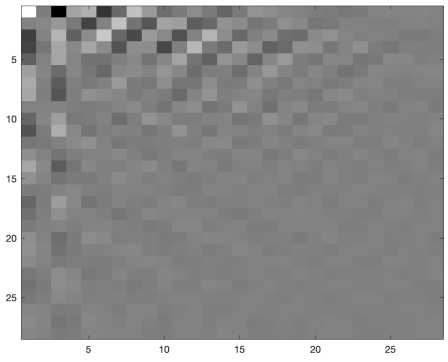

(c) Number 1 Index 3




(d) Corresponding DCT




(e) Number 7 Index 7




(f) Corresponding DCT

Figure 11: Numerical Images and corresponding DCT (a) image number 1 index 1 (b) corresponding DCT (c) image number 1 index 3 (d) corresponding DCT (e) image number 1 index 7 (f) corresponding DCT.

Goldstein, 1984) and kernel principal component analysis (Scholkopf et al., 1998; Bishop, 2006). If the transformation chosen is too complex, the ability to generalize from a small data set will be poor. On the other hand, if the transformation chosen is too simple, it may constrain the decision boundaries to a form that is inappropriate to discriminate between classes. Another disadvantage is that all features are used, even if some of them have noise like characteristics. This might be unnecessarily expensive in term of computation (van der Heijden et al., 2004). It should be noted that the transformation used for the input features in the training of the classification model should also be used for the testing features.

Previously in Section 4 PCA was used to generate features, in this section the reduction in dimensionality can also be accomplished using PCA. How ever PCA alone may not be sufficient, it may require the used of factor analysis in which the factors are rotated in order to group the appropriate variables based on their correlations. Examples will be given, however the exact details are outside of the scope of this document. In the Machine Learning I module linear discriminant analysis will be described which will describe dimensionality reduction in more detail using PCA.

### 9.1 Kernel PCA

The Kernel Principal Component Analysis (Kernel PCA) is the non-linear extension of the ordinary linear PCA (Scholkopf et al., 1998). The input training vectors is $\mathbf{x} = [x_1, x_2, \cdots, x_\lambda] \in \Upsilon^n$ are mapped by a nonlinear transformation $\phi(\cdot) : X \rightarrow F$ to a new dimensional feature space $F \in \Upsilon^\lambda$. The mapping $\phi(\cdot)$ is represented in the kernel PCA by a kernel function $K(\cdot, \cdot)$ which defines an inner product in $\Upsilon^\lambda$. This yields a non-linear (kernel) projection of data which has a general definition as

$$\hat{\mathbf{z}} = \hat{\mathbf{A}}^T K(\mathbf{x}_i, \mathbf{x}_j) + \mathbf{b} \tag{31}$$

where $\hat{A}$ is an $[\lambda \times p]$ matrix containing the top $p$ values, $\mathbf{b}$ is a bias vector and $\hat{\mathbf{z}} \in \Upsilon^p$ is the vector of extracted features. The eigenvectors are not computed directly from the kernel matrix $K(\cdot, \cdot)$. The kernel matrix must be centered as follows:

$$\mathbf{K}_c = K(\mathbf{x}_i, \mathbf{x}_j) - 1_{[\lambda \times \lambda]} K(\mathbf{x}_i, \mathbf{x}_j) - K(\mathbf{x}_i, \mathbf{x}_j) 1_{[\lambda \times \lambda]} + 1_{[\lambda \times \lambda]} K(\mathbf{x}_i, \mathbf{x}_j) 1_{[\lambda \times \lambda]} \tag{32}$$

where $1_{[\lambda \times \lambda]}$ is a $[\lambda \times \lambda]$ matrix in which every value is $1/\lambda$. The eigenvalues, $\lambda$, and eigenvectors, $\mathbf{e}$, are determined with the use of $\mathbf{K}_c$. The bias vector $\mathbf{b}$ is computed as:

$$\mathbf{b} = \hat{\mathbf{A}}^T \Big( 1_{[\lambda \times \lambda]} K(\mathbf{x}_i, \mathbf{x}_j) 1_\lambda - K(\mathbf{x}_i, \mathbf{x}_j) 1_{[\lambda \times \lambda]} \Big) \tag{33}$$

where $1_\lambda$ is an $[\lambda \times 1]$ vector with each element equal to $1/\lambda$.

## 10 References


[1] Background story: Profile: David A. Huffman, Scientific American, September 1991, pp. 54-58.

[2] Bishop, Christopher M., *Pattern Recognition and Machine Learning*, Springer, 2006

[3] Brown, W. and Shepherd, B. J., *Graphics File Formats: Reference and Guide*, Greenwich, CT: Manning Publications, 1995

[4] Caglayan, O., *Digital Multimedia System Security Based on Transform Domain Techniques*, Dissertation, 2008

[5] Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronal L., and Stein, Clifford, *Introduction to Algorithms*, 3rd Edition, MIT Press, 2009

[6] d'Alambert, J.R., *Researchers sur diferentes points importants du systeme du monde*, vol. 2, pp. 66, 1754

[7] Dillon, and Goldstein, M.. *Multivariate Analysis Methods and Applications*, John Wiley, 1984

[8] Duda, Richard O., Hart, Peter E., and Stork, David G., *Pattern Classification*, 2nd Edition, John Wiley & Sons, Inc., 2001

[9] Duhamel, P., Hollman, H., *Existence of a 2n FFT algorithm with a number of multiplications lower than* $2^{n+1}$ Electron. Lett., vol 20, pp. 690-692, 1984

[10] Duin, Robert P.W., Tax, David and Pekalska, Elzbieta, *PRTools*, http://prtools.tudelft.nl/

[11] Elysium Ltd., *Home of the JPEG committee—*, Retrieved February 20, 2005, http://www.JPEG.org/, 2004

[12] Euler, L., *Nova Acta Acad. Sci. Petrop.*, v(1), 14, 435-542-84, publ. 1760

[13] Farid, H., *Detecting Hidden Messages Using Higher-Order Statistical Models*, IEEE International Conference on Images Processing, Volume 2, Rochester, NY, pp. 905-908, 2002

[14] Franc, Vojtech and Hlavac, Vaclav, *Statistical Pattern Recognition Toolbox*, https://cmp.felk.cvut.cz/cmp/software/stprtool/index.html

[15] Goldstine, H.H., *A History of Numerical Analysis from the 16th Through the 19th Century*, pp. 249-253, New York, Springer-Verlag, 1977 (See also M. T. Heideman, D. H. Johnson, and C. S. Burrus, "Gauss and the history of the fast Fourier transform," IEEE ASSP Magazine 1 (4), 14–21 (1984))

[16] Gonzalez, R. C. and Woods R., *Digital Image Processing*, 3rd Edition. Upper Saddle River, NJ: Prentice Hall, 2007

[17] Hotelling, H., *Analysis of a complex of statistical variables into principal components*, Journal of Educational Psychology, Number 24, pp. 417–441, 1933

[18] Huffman, D.A., "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098-1102.

[19] Independent JPEG Group, *Independent JPEG Group*, Retrieved February 20, 2005, http://www.ijg.org/, 1988

[20] JPEG, *Information technology - Digital compression and coding of continuous-tone still images: Requirements and guidelines*, ISE/IEC IS 10918-1, American National Standards Institute, Retrieved June 15, 2008 http://www.w3.org/Graphics/JPEG/itu-t81.pdf., 1994

[21] Kaiser, H. F., *The varimax criterion for analytic rotation in factor analysis*, Psychometrika, 23(3), pp. 187-200, 1958

[22] Kaiser, H. F., *The application of electronic computers to factor analysis*, Educational and Psychological Measurement, Volume 20, Number 1, pp. 141-151, 1960

[23] Machine Learning at Waikato University, *WEKA*, https://www.cs.waikato.ac.nz/ ml/index.html

[24] Murry, J. D. and vanRyper, W., *Encyclopedia of Graphics File Formats*, Sebastopol, CA: O'Reilly & Associates, Inc., 1994

[25] Rao, K. P. and Yip, P., *Discrete Cosine Transform Algorithms, Advantages, Applications*, San Diego, CA: Academic Press, Inc., 1990

[26] Theodoridis, S. and Koutroumbas, K.. *Pattern Recognition Third Edition*, San Diego, CA: Academic Press, 2006