Feature Classification

CAP 5415, FALL 2019
The simplest element of a complex neural network is a “single neuron”, which is a linear operation on the input image.

To understand where it arises from, we need to consider the basics of statistical decision theory.
Bayes Rule

\[ P(\text{class } i / \text{feature } x) = \frac{P(\text{feature } x / \text{class } i) \times P(\text{class } i)}{P(\text{feature } x)} \]

• Given an observed feature x, Bayes rule allows us to select the most likely class which was the source of the feature.

• Requires knowledge of class conditional distributions of the feature (obtained from training data)

• All Classification techniques (including CNNs) are governed by this underlying principle.

• If we know the statistical distributions, we can use Bayesian Learning

• When distributions are unknown (or difficult to model), we use data driven techniques such as CNNs instead. However, performance will be always limited by Bayes Error Limit.
Bayesian decision strategy

- Assume that $C_1, C_2, \ldots, C_N$ are $N$ possible classes that can be observed
  - The prior probability that class $i$ occurs is denoted by $P(C_i)$.

- Assume that $\mathbf{x} = [x_1 \ x_2 \ \cdots \ x_d]$ is a $d \times 1$ measured (observed) feature vector.

- Question: What is the probability that class $C_i$ is present based on the observed featured vector?

- The class conditional probabilities for the feature vector are denoted by $p(\mathbf{x}|C_i)$
  - This is the distribution of the values of $\mathbf{x}$ that are observed when class $C_i$ is present.

- Overall probability distribution for the feature vector $\mathbf{x}$ is the sum total of the contributions from each class, multiplied by the probability of occurrence of that class, i.e.

  $$p(\mathbf{x}) = \sum_{i=1}^{N} p(\mathbf{x}|C_i)P(C_i)$$

- Therefore, the likelihood that a particular class $i$ was the source of the observed feature vector $\mathbf{x}$, is simply the relative contribution of that class to $p(\mathbf{x})$, i.e.

  $$P(C_i|\mathbf{x}) = \frac{p(\mathbf{x}|C_i)P(C_i)}{p(\mathbf{x})}$$

- Decision Strategy:

  Choose Class $i$ if $P(C_i|\mathbf{x}) > P(C_j|\mathbf{x})$ for all $j \neq i$, 
Distribution Models

- The distributions $p(x|C_i)$ are essentially obtained from training data by collecting lots of samples of $x$ from all class $C_i$

- For simple cases, we can model $p(x|C_i)$ as different types of probability distribution functions (pdfs), and just estimate the necessary parameters from training data.

  - For example, if a scalar feature, $p(x|C_i)$ is gaussian, then the form is given by

    $$ p(x|C_i) = \frac{1}{\sqrt{2\pi\sigma_i}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}} $$

    The distribution for each class is characterized by the mean $\mu_i$ and standard deviation $\sigma_i$

https://galtonboard.com/probability examplesinlife
We obtain 10,000 samples of the features for each class.

This is our “training data” from which we can obtain the “class conditional” distribution for the feature.

The histogram of the data for each class show a gaussian distribution, with approximately equal variances but different means.
Estimating Distribution parameters

- The estimate of the mean is given by \( \mu_k = \frac{1}{M} \sum_{i=1}^{M} x_i^k \)
  - \( M \) is the number of training samples for class \( k \).

- The estimate of the variance is given by \( \sigma_k^2 = \frac{1}{M} \sum_{i=1}^{M} (x_i^k - \mu_k)^2 \)

- The gaussian distributions based on the mean and variance estimated from the training samples are shown above.
  - The normalized histograms and the gaussian models match fairly well.

\[
p(x|C_i) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}
\]
Recall that if \( P(C_1|x) > P(C_2|x) \) we choose class 1; otherwise we choose class 2.

Assume that both have the same variance \( \sigma \). The two class conditional distributions are

\[
p(x|C_1) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu_1)^2}{2\sigma^2}} \quad p(x|C_2) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu_2)^2}{2\sigma^2}}
\]

To choose class 1, we require

\[
e^{-\frac{(x-\mu_1)^2}{2\sigma^2}} > e^{-\frac{(x-\mu_2)^2}{2\sigma^2}}.
\]
Decision Rule

Taking log of both sides, we get

$$-\frac{(x - \mu_1)^2}{2\sigma^2} > -\frac{(x - \mu_2)^2}{2\sigma^2}$$

Decide class 1 is present if $$(x - \mu_1)^2 < (x - \mu_2)^2$$ or

$$\mu_1^2 - 2\mu_1x < \mu_2^2 - 2\mu_2x$$

Rearranging terms, we get $$\mu_1^2 - \mu_2^2 - 2(\mu_2 - \mu_1)x < 0$$, or

$$x > \frac{\mu_1^2 - \mu_2^2}{2(\mu_2 - \mu_1)}$$

i.e. Decide class 1 is present if

$$x > \frac{\mu_1 + \mu_2}{2}$$

Since $\mu_1 = 5$, and $\mu_2 = 0$ we decide in favor of class 1 if $x > 2.5$
Multivariate case

• Now consider the feature vector $\mathbf{x} = [x_1 \ x_2 \ \cdots \ x_d]^T$

• The multivariate Gaussian distribution has the form

$$p(\mathbf{x}|C_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp \left[ -\frac{1}{2} (\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i) \right]$$

Where $\mu_i$ is the class mean, and $\Sigma_i$ is the class covariance matrix.

• Note that these can be estimated from training data as $\mu_i = \frac{1}{N} \sum_{k=1}^{N} x_{ki}$ (i.e. average), and

$$\Sigma_i = \frac{1}{N} \sum_{k=1}^{N} (x_{ki} - \mu_i)(x_{ki} - \mu_i)^T$$

• To choose class i, we want $p(\mathbf{x}|C_i) > p(\mathbf{x}|C_j)$ for all $j \neq i$

• $\log\{p(\mathbf{x}|C_i)\} = -\frac{d}{2} \log(2\pi) - \frac{1}{2} \log(|\Sigma_i|) - \frac{1}{2} \left[ (\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i) \right]$ 

• Ignoring the constant, we choose the class for which

$$g_i(\mathbf{x}) = -\frac{1}{2} \log(|\Sigma_i|) - \frac{1}{2} \left[ (\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i) \right] \quad \text{Mahalanobis Distance}$$

yields the largest value
Linear Classifier

- Consider the case when all classes have the same covariance matrix $\Sigma$

- The discriminant function for the $i$-th class is

$$g_i(x) = -\frac{1}{2} \log(|\Sigma|) - \frac{1}{2} [(x - \mu_i)^T \Sigma^{-1} (x - \mu_i)]$$

Or

$$g_i(x) = -\frac{1}{2} [x^T \Sigma^{-1} x + \mu_i^T \Sigma^{-1} \mu_i - 2x^T \Sigma^{-1} \mu_i] = x^T \Sigma^{-1} \mu_i - \frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i - \frac{1}{2} x^T \Sigma^{-1} x.$$ 

- Since $x^T \Sigma^{-1} x$ for all classes (does not depend on $i$), we choose class $i$ if $g_i(x) > g_j(x)$ for all $j, j \neq i$, i.e.

$$g_i(x) = x^T \Sigma^{-1} \mu_i - \frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i = x^T h_i + b_i$$

Where $h_i = \Sigma^{-1} \mu_i$ and $b_i = -\frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i$

- Therefore, for each class, compute $g_i(x) = x^T h_i + b_i$ for $i=1..K$, and assign $x$ the label of the class for which $g_i(x)$ is the largest.
Consider the three classes “0”, “1”, and “2”.
- Each class has 50 Training Images

Design a feature vector using HCD algorithm
- Use the HCD algorithm to compute the R score for each training images
- For each image, we choose 10 largest positive scores, and 10 largest negative scores to form a 20 x 1 feature vector

Compute the 20 x 20 covariance matrix of the entire data set, and the 20 x 1 mean vectors for each class
- Use these statistics to compute the linear discriminant functions for each class.
• The image of the number “0” is processed by the HCD algorithm to produce the R score.
• The corner and edge regions of the image are identified threshold R.
• The 10 largest values (positive or negative) are selected from each region to form the 20 x 1 feature vector for this image.
• Create a feature vector for every image of each class.
  • Since there are 50 training images per class, the feature data is 20 x 50 matrix for each class: $X_i = [x_{i1}^i \quad x_{i2}^i \quad \cdots \quad x_{i50}^i], \ i=1,2,3$
Class Parameters and statistics

\[ \Sigma \] is the overall covariance matrix of the feature vectors for all three classes.

\[ \mu \] is the overall mean vector for all three classes

The individual class mean vectors are different and show that the center of the three classes are distinct.

\[
\Sigma = \frac{1}{N} \sum_{k=1}^{N} (x_k^i - \mu)(x_k^i - \mu)^T
\]
Linear Discriminant analysis

• The confusion matrix shows the number of correct and incorrect decisions per class
  • Probability of Correct Classification $P_c = (49 + 44 + 50) / 150 = 95\%$
  • Probability of Error $P_e = 7 / 150 = 5\%$
LDF Algorithm

- Feature vectors extracted from the image are processed by a bank of LDFs.
- Image is assigned to the class which has the largest (maximum) output.

\[
h_i = \Sigma^{-1} \mu_i \quad \text{and} \quad b_i = -\frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i
\]
MMSE classifier

• Well, the world is not always Gaussian!
  • Explore other methods for optimizing the LDF

• Possible Approach – Find an LDF vector which minimizes the squared error between its output and a desired output
  • LDF1 output is +1 for Class 1 input, otherwise produce an output of -1 for all other classes
  • LDF2 output is +1 for Class 2 input, otherwise produce an output of -1 for all other classes
  • LDF3 output is +1 for Class 3 input, otherwise produce an output of -1 for all other classes

\[ x^T h_i = \begin{cases} 
+1 & \text{if } x \text{ belongs to class } i \\
-1 & \text{otherwise}
\end{cases} \]
MMSE solution

- Let $F = [x_1^1 \ldots x_N^1 \ x_1^2 \ldots x_N^2 \ x_1^3 \ldots x_N^3]$ be a matrix whose columns are the feature vectors for all the classes
  - For our example, $x_k^i$ is a 20 x 1 feature vector for the $k$-th image of the $i$-th class, and $N=50$ for each class.
  - Therefore $F$ is a 20 x 150 matrix.

- Define the desired output vector for the LDFs
  - For LDF1, the output should be $+1$ in response to class 1, but $-1$ in response to class 2 and 3
    - $u_1 = [1 \ldots 1 \ -1 \ldots -1 \ -1 \ldots -1]^T$
  - For LDF2, the output should be $+1$ in response to class 2, but $-1$ in response to class 1 and 3
    - $u_2 = [-1 \ldots -1 \ 1 \ldots 1 \ -1 \ldots -1]^T$
  - For LDF3, the output should be $+1$ in response to class 3, but $-1$ in response to class 1 and 1
    - $u_2 = [-1 \ldots -1 \ -1 \ldots -1 \ 1 \ldots 1]^T$

- Therefore, we want

\[
F^T h_1 = u_1 \quad \Rightarrow \quad h_1 = (F F^T)^{-1} F u_1
\]
\[
F^T h_2 = u_2 \quad \Rightarrow \quad h_2 = (F F^T)^{-1} F u_2
\]
\[
F^T h_3 = u_3 \quad \Rightarrow \quad h_3 = (F F^T)^{-1} F u_3
\]
The output of the MMSE LDFs generally follows the desired output for each class.

There is noticeable variations in the output.

Overall classification performance is similar to that of the Bayesian LDF for gaussian distributions.
Data Driven Feature Analysis

- Even for non-gaussian data, one can estimate first and second order moments (mean and variance) to design LDFs.
- Define and optimizing a cost function related to class separation.
- Can be applied to pre-extracted features (such as Harris Corners, or SIFT).
- Can be also applied directly to images to obtain features that are “optimum” for a given cost function.
Fisher Linear Discriminant Functions

R.A. Fisher developed a method for finding a LDF which maximally separates the class means while minimizing the in-class variance.

Consider the data vectors for two classes:

\[ X = [x_1 \ x_2 \ \cdots \ x_N], \quad Y = [y_1 \ y_2 \ \cdots \ y_M] \]

Where \( x_i \) belong to class 1, and \( y_i \) belong to class 2.

The class means are given by

\[ \mu_X = \frac{1}{N} \sum_{k=1}^{N} x_k \quad \text{and} \quad \mu_Y = \frac{1}{M} \sum_{k=1}^{M} y_k \]

The class covariance matrices are given by

\[ \Sigma_X = \frac{1}{N} \sum_{k=1}^{N} (x_k - \mu_X)(x_k - \mu_X)^T \]
\[ \Sigma_Y = \frac{1}{M} \sum_{k=1}^{M} (x_k - \mu_Y)(x_k - \mu_Y)^T \]
Mean and variance of LDF output

• Consider a vector $h$. The projections of class 1 data $(x_i)$ and class 2 data $(y_i)$ on this vector are given by $h^T x_i$ and $h^T y_i$.

• The average value of the LDF output for each class is given by
  \[
  \mu_1 = \frac{1}{N} \sum_{i=1}^{N} h^T x_i = h^T \left( \frac{1}{N} \sum_{i=1}^{N} x_i \right) \\
  \mu_2 = \frac{1}{M} \sum_{i=1}^{M} h^T y_i = h^T \left( \frac{1}{M} \sum_{i=1}^{M} y_i \right)
  \]

  \[
  \mu_1 = h^T \mu_x \\
  \mu_2 = h^T \mu_y
  \]

• The variance of the LDF output for class 1 is given by
  \[
  \sigma_1^2 = \frac{1}{N} \sum_{i=1}^{N} |h^T x_i - \mu_1|^2 = \frac{1}{N} \sum_{k=1}^{N} |h^T x_i - h^T \mu_x|^2 \\
  = \frac{1}{N} \sum_{i=1}^{N} h^T (x_i - \mu_x)(x_i - \mu_x)^T h \\
  = h^T \left( \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu_x)(x_i - \mu_x)^T \right) h \\
  \sigma_1^2 = h^T \Sigma_x h
  \]

• Similarly, the variance of the LDF output for class 2 is given by
  \[
  \sigma_2^2 = h^T \Sigma_y h
  \]
Fisher criterion

• To separate the projections of the two classes, we maximize

\[ J(h) = \frac{|\mu_1 - \mu_2|^2}{\sigma_1^2 + \sigma_2^2} = \frac{|h^T \mu_x - h^T \mu_y|^2}{h^T \Sigma_x h + h^T \Sigma_y h} = \frac{h^T (\mu_x - \mu_y)(\mu_x - \mu_y)^T h}{h^T (\Sigma_x + \Sigma_y) h} \]

• The ratio of quadratic terms of the form \( J(h) = \frac{h^T Bh}{h^T Ah} \) is known as a Rayleigh Quotient

• The solution for \( h \) which maximizes this ratio is the dominant eigenvector of \( A^{-1}B \), i.e

\[ A^{-1}B h_{opt} = \lambda_{max} h_{opt} \]

Proof is left to the interested student to investigate!
Fisher LDF

• The vector that maximizes $J(h) = \frac{|\mu_1 - \mu_2|^2}{\sigma_1^2 + \sigma_2^2} = \frac{h^T(\mu_x - \mu_y)(\mu_x - \mu_y)^T h}{h^T(\Sigma_x + \Sigma_y)h}$ is therefore the dominant eigenvector of

$$\left(\Sigma_x + \Sigma_y\right)^{-1}(\mu_x - \mu_y)(\mu_x - \mu_y)^T h$$

• Therefore

$$\left(\Sigma_x + \Sigma_y\right)^{-1}(\mu_x - \mu_y)(\mu_x - \mu_y)^T h = \lambda h$$

• However, $(\mu_x - \mu_y)^T h$ is a scalar, say $\alpha$. This yields

$$\alpha\left(\Sigma_x + \Sigma_y\right)^{-1}(\mu_x - \mu_y) = \lambda h$$

• In other words, $h = \frac{\alpha}{\lambda}(\Sigma_x + \Sigma_y)^{-1}(\mu_x - \mu_y)$.  
  • The scale factor $\frac{\alpha}{\lambda}$ does not matter since Fisher criterion is a ratio which is not affected by this term.

• Thus the optimum LDF which maximizes the Fisher criterion is $h_{opt} = \left(\Sigma_x + \Sigma_y\right)^{-1}(\mu_x - \mu_y)$
Example using Class 1 and 2 from digit data set

- The distributions of output produced by the Fisher LDF have less overlap.
- The Bayesian LDF has 5 errors for class 1, while the Fisher LDF has 1 error.
Multi-Class Fisher LDFs

- The goal is to measure new features such that the class means are well separated, while the variance within each class is minimized.

- The separation between the classes is increased by maximizing the distance of each class center from the overall center (mean) of the entire data.
Between-Class Distance

- Assume that $C_1, C_2, \ldots, C_N$ are $M$ possible classes
  - The training data for class $i$ is denoted by $x_k^i$
  - The class means are given by $\mu_i = \frac{1}{N} \sum_{k=1}^{N} x_k^i$
  - The covariance matrix for each class is given by $\Sigma_i = \frac{1}{N} \sum_{k=1}^{N} (x_k^i - \mu_i)(x_k^i - \mu_i)^T$
  - The overall mean of the data is simply $\mu = \frac{1}{M} \sum_{i=1}^{M} \mu_i$

- Now consider the output of the LDF vector $h$ i.e., $g_i(x_k^i) = h^T x_k^i$. It is easy to show that the mean value of the output for class $i$ is equal to $m_i = h^T \mu_i$

and the overall mean of the output of the LDF (for all data) is $m = h^T \mu$

- Therefore the overall distances between the class means and the overall center of the data is given by
  \[
  between\ class\ distance = \sum_{i=1}^{M} (m_i - m)^2 = \sum_{i=1}^{M} (h^T \mu_i - h^T \mu)^2 = \sum_{i=1}^{M} |h^T (\mu_i - \mu)|^2
  \]
  \[
  = \sum_{i=1}^{M} h^T (\mu_i - \mu)(\mu_i - \mu)^T h
  \]
  \[
  = h^T \left[ \sum_{i=1}^{M} (\mu_i - \mu)(\mu_i - \mu)^T \right] h = h^T Bh
  \]

  where $B = \sum_{i=1}^{M} (\mu_i - \mu)(\mu_i - \mu)^T$ is the Between Class Scatter Matrix.
Within Class Scatter

- The variance within each class is given by
  \[
  \sigma_i^2 = \frac{1}{N} \sum_{k=1}^{N} |h^T x_k^i - \mu_i|^2
  \]
  \[
  = h^T \left( \frac{1}{N} \sum_{k=1}^{N} (x_k^i - \mu_i)(x_k^i - \mu_i)^T \right) h
  \]
  \[
  \sigma_i^2 = h^T \Sigma_i h
  \]
- Total in-class variance is \( \sum_{i=1}^{M} \sigma_i^2 = \sum_{i=1}^{M} h^T \Sigma_i h = h^T \left( \sum_{i=1}^{M} \Sigma_i \right) h = h^T Ah \)
- \( A = \sum_{i=1}^{M} \Sigma_i \) is the total in-class covariance matrix
- Therefore, We wish to maximize \( \frac{\text{between class distance}}{\text{in-class variance}} = \frac{\sum_{i=1}^{M} (m_i-m)^2}{\sum_{i=1}^{M} \sigma_i^2} = \frac{h^T Bh}{h^T Ah} \)
- Solution: The optimum LDF vectors \( (h) \) are the dominant eigen-vectors of \( A^{-1}B \)
Three class Example

- Compute mean vectors for each class and overall mean of data
- Compute between class scatter matrix $B$. \[ B = \sum_{i=1}^{M} (\mu_i - \mu)(\mu_i - \mu)^T \]
- Compute covariance matrix for each class, and their sum $A$. \[ A = \sum_{i=1}^{M} \Sigma_i \]
- Compute $A^{-1}B$ and obtain its eigenvectors and eigenvalues
  - Since there are three classes, the rank of $B$ is 2, and there are only 2 non-zero eigen-values
  - $\lambda_1 = 19.29$, $\lambda_2 = 1.61$
- Select the dominant eigenvectors corresponding to the largest eigen-values as the LDF vectors.
  - Eigenvector $h_1$ corresponds to $\lambda_1$, and Eigenvector $h_2$ corresponds to $\lambda_2$
Classification Process

• Using the two Fisher LDFs, the three classes are easy to separate.
  • Define a $d \times 2$ matrix $H = [h_1 \ h_2]$

• Transform the HCD feature vectors $x_k^i$ into the Fisher LDF space $f_k^i = H^T x_k^i$
  • Note that $f_k^i$ is a $2 \times 1$ vector with the output of LDF1 ($x^T h_1$) as its first element and the output of LDF2 ($x^T h_2$) as its second element.

• Transform the class means and covariance matrices to the Fisher LDF space as well
  • $m_i = H^T \mu_i, \quad S_i = H^T \Sigma_i H$

• The data is classified by assigning it to the class with the smallest Mahalanobis Distance
  • For each test vector $k$, compute $d_i = (f_k^i - m_i)^T S_i^{-1} (f_k^i - m_i)$ for all $i$ classes.
  • Assign the test vector the label of the class for which $d_i$ is smallest

• Results identical to the Bayesian LDF are obtained.

<table>
<thead>
<tr>
<th>Truth\Decision</th>
<th>Class 1 “0”</th>
<th>Class 2 “1”</th>
<th>Class 3 “2”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1 “0”</td>
<td>49</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Class 2 “1”</td>
<td>6</td>
<td>44</td>
<td>0</td>
</tr>
<tr>
<td>Class 3 “2”</td>
<td>0</td>
<td>0</td>
<td>50</td>
</tr>
</tbody>
</table>
The Fisher LDF algorithm can be applied directly to images.

Instead of extracting features first, treat the images as the vectors $\mathbf{x}_k^i$.

- For 28 x 28 images, the size of the data vectors and the class mean vectors is 784 x 1.
  - These can be reshaped back into 28 x 28 images.
  - The matrices $\Sigma_i$, $A$, and $B$ are all of size 784 x 784.
Visualizing the covariance matrices

- The $784 \times 784$ class covariance matrices can be visualized as images
  - Structure in these matrices indicate which elements of the data vectors are statistically related
  - Vector representation does not change/alter 2D structural relations in images
Fisher LDFs

- The Rank of $B$ is still 2, so $A^{-1}B$ has only two eigen-vectors that correspond to non-zero eigen-values
  - $\lambda_1 = 356, \lambda_2 = 211$

- The two eigen-vectors can be reshaped into the 28x28 images shown above
  - These are also referred to as “filters” or “kernels” in the context of direct application to 2D images

- Each of the image vectors can be projected on these two LDF vectors to get optimally separated features.
Example Results

Data is perfectly separated with compact clusters and large distances between the classes.

- Can you compute the effective Fisher Ratio?

- To classify an image, compute the 2 Fisher features using the two LDFs, and then compute its Mahalanobis distance to each of the class centers in the Fisher Feature space.

- Directly optimizing the Fisher Criterion using the images produces LDFs which yield the “best” features for the given task.

- The need to extract “engineered” features is avoided.
Linear Basis Representation of Images

• Consider a set of images in vector notation $x_1, x_2, \cdots x_N$
  • The image vectors are of length $d \times 1$

• Assume that these image vectors can be represented as a linear combination of set of orthonormal vectors, i.e.

$$x_k = \sum_{i}^{d} a_{i,k} \phi_i$$

where

$$\phi_i^T \phi_i = \begin{cases} 
1 & i = j \\
0 & i \neq j 
\end{cases}$$

• Define $\Phi = [\phi_1 \phi_2 \phi_d]$, and $a_k = [a_{1,k} \ a_{2,k} \cdots \ a_{d,k}]$
  • $\Phi^T \Phi = \Phi \Phi^T = I$ (orthonormal basis set)

• The $k$-th image vector is now given by $x_k = \Phi a_k$
Principle Components Analysis (Karhunen Loeve Expansion)

• To find $\Phi$, we compute the eigen-vectors of the auto-correlation matrix of the data

• Consider the set of images $x_1(m,n), x_2(m,n), \ldots x_N(m,n)$.
  • In vector notation these are denoted by $x_1 \ x_2 \ \cdots \ x_N$

• Define the $d \times N$ data matrix $X = [x_1 \ x_2 \ \cdots \ x_N]$

• The correlation matrix of the data is then given by $R = XX^T$
  • $R$ is a $d \times d$ matrix.

• $R$ can be expressed in terms of its eigenvectors $\phi_i$ and eigen values $\lambda_i$ as
  $$R = \sum_{i=1}^{d} \lambda_i \phi_i \phi_i^T$$

• Select the $M$ dominant eigen-vectors and define the $d \times M$ matrix $\Phi = [\phi_1 \ \phi_2 \ \cdots \ \phi_M]$

• Then the image $x_k$ can be approximated as
  $$\hat{x}_k = \Phi a_k$$
  where the coefficients of linear combination are given by $a_k = \Phi^T x_k$
Consider the digit data set with 3 classes, and 50 image per class.

The autocorrelation matrix $R$ has a rank of 150. Therefore, there are only 150 non-zero eigen-values.

- In fact, the plot shows that there are less than 25 eigenvalues that are not close to zero.

We will use 25 eigen-vectors to approximate the images of the three classes.
The top 25 eigen images are shown in the montage.

The approximation of images for each class shows good reconstruction.
PCA Feature Space

- The plot shows the relation of the three classes using the top 2 PCA features.
- While PCA features are optimum for image representation in a “minimum MSE” sense, they are not good for DISCRIMINATION.
- Fisher Features are good for discrimination, but may not be good for representation.
- Classifier design is often about balancing the trade off between representation and discrimination.

\[
\Phi = [\phi_1 \quad \phi_2] \\
\alpha_k = \Phi^T x_k
\]