In lecture 3, we learned about estimating parametric models and how these then form classifiers and define decision boundaries (lecture 2).

Now we turn back to the question of Dimensionality.

Recall the fish example, where we experimented with the length feature first, then the lightness feature, and then decided upon a combination of the width and lightness.

We developed some intuition saying the more features I add, the better my classifier will be...
In lecture 3, we learned about estimating parametric models and how these then form classifiers and define decision boundaries (lecture 2).

Now we turn back to the question of Dimensionality.

Recall the fish example, where we experimented with the length feature first, then the lightness feature, and then decided upon a combination of the width and lightness.

We developed some intuition saying *the more features I add, the better my classifier will be*...

We will see that in theory this may be, but in practice, this is not the case—the probability of error will increase after a certain number of features (dimensionality) has been reached.
In lecture 3, we learned about estimating parametric models and how these then form classifiers and define decision boundaries (lecture 2).

Now we turn back to the question of Dimensionality.

Recall the fish example, where we experimented with the length feature first, then the lightness feature, and then decided upon a combination of the width and lightness.

We developed some intuition saying *the more features I add, the better my classifier will be*...

We will see that in theory this may be, but in practice, this is not the case—the probability of error will increase after a certain number of features (dimensionality) has been reached.

We will first explore this point and then discuss a set of methods for dimension reduction.
Consider a simple arrangement: you have a sphere of radius $r = 1$ in a space of $D$ dimensions.

We want to compute what is the fraction of the volume of the sphere that lies between radius $r = 1 - \epsilon$ and $r = 1$.

$$V_D(r) = K_D r^D \quad (1)$$

$$V_D(1) - V_D(1 - \epsilon) = 1 - (1 - \epsilon)^D \quad (2)$$
Consider a simple arrangement: you have a sphere of radius $r = 1$ in a space of $D$ dimensions.

We want to compute what is the fraction of the volume of the sphere that lies between radius $r = 1 - \epsilon$ and $r = 1$.

Noting that the volume of the sphere will scale with $r^D$, we have:

$$V_D(r) = K_D r^D \quad (1)$$

where $K_D$ is some constant (depending only on $D$).

$$\frac{V_D(1) - V_D(1 - \epsilon)}{V_D(1)} = \frac{1 - (1 - \epsilon)^D}{1 - \epsilon} \quad (2)$$
Let’s Build Some Better Intuition
Example from Bishop PRML

- Dataset: Measurements taken from a pipeline containing a mixture of oil.
  - Three classes present (different geometrical configuration): homogeneous, annular, and laminar.
  - Each data point is a 12 dimensional input vector consisting of measurements taken with gamma ray densitometers, which measure the attenuation of gamma rays passing along narrow beams through the pipe.
Let’s Build Some Better Intuition
Example from Bishop PRML

- 100 data points of features $x_6$ and $x_7$ are shown on the right.
- **Goal:** Classify the new data point at the ‘x’.
- Suggestions on how we might approach this classification problem?
Let’s Build Some Better Intuition
Example from Bishop PRML

Observations we can make:

- The cross is surrounded by many red points and some green points.
- Blue points are quite far from the cross.
- Nearest-Neighbor Intuition: The query point should be determined more strongly by nearby points from the training set and less strongly by more distant points.
One simple way of doing it is:

- We can divide the feature space up into regular cells.
- For each cell, we associated the class that occurs most frequently in that cell (in our training data).
- Then, for a query point, we determine which cell it falls into and then assign in the label associated with the cell.
One simple way of doing it is:

- We can divide the feature space up into regular cells.
- For each cell, we associated the class that occurs most frequently in that cell (in our training data).
- Then, for a query point, we determine which cell it falls into and then assign the label associated with the cell.
- What problems may exist with this approach?
The problem we are most interested in now is the one that becomes apparent when we add more variables into the mix, corresponding to problems of higher dimensionality.

In this case, the number of additional cells grows exponentially with the dimensionality of the space.

Hence, we would need an exponentially large training data set to ensure all cells are filled.
This severe difficulty when working in high dimensions was coined the **curse of dimensionality** by Bellman in 1961.

The idea is that the volume of a space increases exponentially with the dimensionality of the space.
How does the probability of error vary as we add more features, in theory?

Consider the following two-class problem:

- The prior probabilities are known and equal: $P(\omega_1) = P(\omega_2) = 1/2$.
- The class-conditional densities are Gaussian with unit covariance:

\[
p(x|\omega_1) \sim N(\mu, I) \quad (3) \\
p(x|\omega_2) \sim N(\mu, I) \quad (4)
\]

where $\mu_1 = \mu$, $\mu_2 = -\mu$, and $\mu$ is an $n$-vector whose $i$th component is $(1/i)^{1/2}$.

The corresponding Bayesian Decision Rule is

\[
\text{decide } \omega_1 \text{ if } x^T \mu > 0 \quad (5)
\]
The probability of error is

\[ P(\text{error}) = \frac{1}{\sqrt{2\pi}} \int_{r/2}^{\infty} \exp \left[ -\frac{z^2}{2} \right] dz \]  

(6)

where

\[ r^2 = \|\mu_1 - \mu_2\|^2 = 4 \sum_{i=1}^{n} \left(\frac{1}{i}\right). \]  

(7)

Let’s take this integral for granted… (For more detail, you can look at DHS Problem 31 in Chapter 2 and read Section 2.7.)
Dimensionality and Classification Error?

Some parts taken from G. V. Trunk, TPAMI Vol. 1 No. 3 PP. 306-7 1979

- The probability of error is

\[ P(\text{error}) = \frac{1}{\sqrt{2\pi}} \int_{r/2}^{\infty} \exp \left[ -\frac{z^2}{2} \right] dz \]  

(6)

where

\[ r^2 = \| \mu_1 - \mu_2 \|^2 = 4 \sum_{i=1}^{n} \left( \frac{1}{i} \right). \]  

(7)

- Let’s take this integral for granted... (For more detail, you can look at DHS Problem 31 in Chapter 2 and read Section 2.7.)

- What can we say about this result as more features are added?
The probability of error approaches 0 as $n$ approach infinity because $1/i$ is a divergent series.

More intuitively, each additional feature is going to decrease the probability of error as long as its means are different. In the general case of varying means and but same variance for a feature, we have

$$r^2 = \sum_{i=1}^{d} \left( \frac{\mu_{i1} - \mu_{i2}}{\sigma_i} \right)^2$$ (8)

Certainly, we prefer features that have big differences in the mean relative to their variance.

We need to note that if the probabilistic structure of the problem is completely known then adding new features is not going to decrease the Bayes risk (or increase it).
So, adding dimensions is good....

...in theory.

But, in practice, performance seems to not obey this theory.
Consider again the two-class problem, but this time with unknown means $\mu_1$ and $\mu_2$.

Instead, we have $m$ labeled samples $x_1, \ldots, x_m$.

Then, the best estimate of $\mu$ for each class is the sample mean (recall last lecture).

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x_i$$  \hspace{1cm} (9)

where $x_i$ has been replaced by $x_i$ is $x_i$ comes from $\omega_2$. The covariance matrix is $I/m$. 
Probablity of error is given by

\[
P(\text{error}) = P(x^T \mu \geq 0|\omega_2) = \frac{1}{\sqrt{2\pi}} \int_\gamma^n \exp \left[-\frac{z^2}{2}\right] \, dz
\]  

(10)

because it has a Guassian form as \( n \) approaches infinity where

\[
\gamma_n = E(z) / [\text{var}(z)]^{1/2}
\]  

(11)

\[
E(z) = \sum_{i=1}^{n} \frac{1}{i}
\]  

(12)

\[
\text{var}(z) = \left(1 + \frac{1}{m}\right) \sum_{i=1}^{n} \frac{1}{i} + n/m
\]  

(13)
Probability of error is given by

\[ P(\text{error}) = P(x^T \mu \geq 0 | \omega_2) = \frac{1}{\sqrt{2\pi}} \int_{\gamma_n}^{\infty} \exp\left[-\frac{z^2}{2}\right] dz \]

The key is that we can show

\[ \lim_{n \to \infty} \gamma_n = 0 \tag{14} \]

and thus the probability of error approaches one-half as the dimensionality of the problem becomes very high.
Trunk performed an experiment to investigate the convergence rate of the probability of error to one-half. He simulated the problem for dimensionality 1 to 1000 and ran 500 repetitions for each dimension. We see an increase in performance initially and then a decrease (as the dimensionality of the problem grows larger than the number of training samples).
The discussion on the curse of dimensionality should be enough!

Even though our problem may have a high dimension, data will often be confined to a much lower effective dimension in most real world problems.

Computational complexity is another important point: generally, the higher the dimension, the longer the training stage will be (and potentially the measurement and classification stages).

We seek an understanding of the underlying data to

- Remove or reduce the influence of noisy or irrelevant features that would otherwise interfere with the classifier;
- Identify a small set of features (perhaps, a transformation thereof) during data exploration.
We have \( n \) samples \( \{x_1, x_2, \ldots, x_n\} \).

How can we best represent the \( n \) samples by a single vector \( x_0 \)?
We have \( n \) samples \( \{x_1, x_2, \ldots, x_n\} \).

How can we best represent the \( n \) samples by a single vector \( x_0 \)?

First, we need a distance function on the sample space. Let’s use the Euclidean distance and the sum of squared distances criterion:

\[
J_0(x_0) = \sum_{k=1}^{n} \|x_0 - x_k\|^2
\]  

(15)
We have $n$ samples $\{x_1, x_2, \ldots, x_n\}$.

How can we best represent the $n$ samples by a single vector $x_0$?

First, we need a distance function on the sample space. Let’s use the Euclidean distance and the sum of squared distances criterion:

$$J_0(x_0) = \sum_{k=1}^{n} \|x_0 - x_k\|^2 \quad (15)$$

Then, we seek a value of $x_0$ that minimizes $J_0$. 
We can show that the minimizer is indeed the sample mean:

$$m = \frac{1}{n} \sum_{k=1}^{n} x_k$$  \hspace{1cm} (16)

We can verify it by adding $m - m$ into $J_0$:

$$J_0(x_0) = \sum_{k=1}^{n} \| (x_0 - m) - (x_k - m) \|^2$$  \hspace{1cm} (17)

$$= \sum_{k=1}^{n} \| x_0 - m \|^2 + \sum_{k=1}^{n} \| x_k - m \|^2$$  \hspace{1cm} (18)

Thus, $J_0$ is minimized when $x_0 = m$. Note, the second term is independent of $x_0$.  


So, the sample mean is an initial dimension-reduced representation of the data (a zero-dimensional one).

It is simple, but it not does reveal any of the variability in the data.

Let’s try to obtain a one-dimensional representation: i.e., let’s project the data onto a line running through the sample mean.
Let $e$ be a unit vector in the direction of the line.
The standard equation for a line is then

$$x = m + ae$$

where scalar $a \in \mathbb{R}$ governs which point along the line we are and
hence corresponds to the distance of any point $x$ from the mean $m$. 

$$J_1(a_1, \ldots, a_n, e) = \sum_{k=1}^{n} \| (m + a_k e) - x_k \|^2$$

$$= \sum_{k=1}^{n} a_k^2 \| e \|^2 - 2 \sum_{k=1}^{n} a_k e^T (x_k - m) + \sum_{k=1}^{n} \| x_k - m \|^2$$
Dimension Reduction Version 1

- Let $e$ be a unit vector in the direction of the line.
- The standard equation for a line is then
  \[ x = m + ae \]  
  where scalar $a \in \mathbb{R}$ governs which point along the line we are and hence corresponds to the distance of any point $x$ from the mean $m$.
- Represent point $x_k$ by $m + a_k e$.
- We can find an optimal set of coefficients by again minimizing the squared-error criterion

  \[ J_1(a_1, \ldots, a_n, e) = \sum_{k=1}^{n} \| (m + a_k e) - x_k \|^2 \]  
  \[ = \sum_{k=1}^{n} a_k^2 \| e \|^2 - 2 \sum_{k=1}^{n} a_k e^T (x_k - m) + \sum_{k=1}^{n} \| x_k - m \|^2 \]
Differentiating for $a_k$ and equating to 0 yields

$$a_k = e^T(x_k - m)$$

(22)

This indicates that the best value for $a_k$ is the projection of the point $x_k$ onto the line $e$ that passes through $m$. 
Differentiating for $a_k$ and equating to 0 yields

$$a_k = e^T(x_k - m)$$

(22)

This indicates that the best value for $a_k$ is the projection of the point $x_k$ onto the line $e$ that passes through $m$.

How do we find the best direction for that line?
What if we substitute the expression we computed for the best $a_k$ directly into the $J_1$ criterion:

$$J_1(e) = \sum_{k=1}^{n} a_k^2 - 2 \sum_{k=1}^{n} a_k^2 + \sum_{k=1}^{n} \|x_k - m\|^2$$  \hspace{1cm} (23)$$

$$= -\sum_{k=1}^{n} [e^T(x_k - m)]^2 + \sum_{k=1}^{n} \|x_k - m\|^2$$  \hspace{1cm} (24)$$

$$= -\sum_{k=1}^{n} e^T(x_k - m)(x_k - m)^Te + \sum_{k=1}^{n} \|x_k - m\|^2$$  \hspace{1cm} (25)$$
Define the \textit{scatter matrix} $S$ as

$$S = \sum_{k=1}^{n} (x_k - \mathbf{m})(x_k - \mathbf{m})^T$$

This should be familiar – this is a multiple of the sample covariance matrix.

Putting it in:

$$J_1(e) = -e^T S e + n \sum_{k=1}^{n} \|x_k - \mathbf{m}\|^2$$

The $e$ that maximizes $e^T S e$ will minimize $J_1$. 

\[ \text{(26)} \]
Define the **scatter matrix** $S$ as

$$
S = \sum_{k=1}^{n} (x_k - m)(x_k - m)^T
$$

This should be familiar – this is a multiple of the sample covariance matrix.
Define the scatter matrix $S$ as

$$S = \sum_{k=1}^{n} (x_k - m)(x_k - m)^T$$

(26)

This should be familiar – this is a multiple of the sample covariance matrix.

Putting it in:

$$J_1(e) = -e^T Se + \sum_{k=1}^{n} \|x_k - m\|^2$$

(27)

The $e$ that maximizes $e^T Se$ will minimize $J_1$. 
We use Lagrange multipliers to maximize $e^T Se$ subject to the constraint that $\|e\| = 1$.

$$u = e^T Se - \lambda e^T e$$ (28)
Dimension Reduction Version 1
Solving for $e$

- We use Lagrange multipliers to maximize $e^T Se$ subject to the constraint that $\|e\| = 1$.

$$u = e^T Se - \lambda e^T e$$  \hfill (28)

- Differentiating w.r.t. $e$ and setting equal to 0.

$$\frac{\partial u}{\partial e} = 2Se - 2\lambda e$$  \hfill (29)

$$Se = \lambda e$$  \hfill (30)

- Does this form look familiar?
Dimension Reduction Version 1

Eigenvectors of $S$

\[ Se = \lambda e \]

- This is an eigenproblem.
- Hence, it follows that the best one-dimensional estimate (in a least-squares sense) for the data is the eigenvector corresponding to the largest eigenvalue of $S$.
- So, we will project the data onto the largest eigenvector of $S$ and translate it to pass through the mean.
We’re already done...basically.
Principal Component Analysis

- We’re already done...basically.
- This idea readily extends to multiple dimensions, say $d' < d$ dimensions.
- We replace the earlier equation of the line with

\[
x = m + \sum_{i=1}^{d'} a_i e_i
\]  

(31)

- And we have a new criterion function

\[
J_{d'} = \sum_{k=1}^{n} \left\| \left( m + \sum_{i=1}^{d'} a_{k_i} e_i \right) - x_k \right\|^2
\]  

(32)
Principal Component Analysis

- $J_{d'}$ is minimized when the vectors $e_1, \ldots, e_{d'}$ are the $d'$ eigenvectors of the scatter matrix having the largest eigenvalues.
- These vectors are orthogonal.
- They form a natural set of basis vectors for representing any feature $x$.
- The coefficients $a_i$ are called the principal components.
- Visualize the basis vectors as the principal axes of a hyperellipsoid surrounding the data (a cloud of points).
- Principle components reduces the dimension of the data by restricting attention to those directions of maximum variation, or scatter.
Fisher Linear Discriminant

- Description vs. Discrimination
- PCA is likely going to be useful for representing data.
- But, there is no reason to assume that it would be good for discriminating between two classes of data.
  - ‘Q’ versus ‘O’.
- **Discriminant Analysis** seeks directions that are efficient for discrimination.
Intuition: Projection onto a Line

- Consider the problem of projecting data from $d$ dimensions onto a line.
- Board...
Intuition: Projection onto a Line

- Consider the problem of projecting data from $d$ dimensions onto a line.
- Board...
- Finding the orientation of the line for which the projected samples is the goal of classical discriminant analysis.
Let’s Formalize the Situation

Suppose we have a set of \( n \) \( d \)-dimensional samples with \( n_1 \) in set \( D_1 \), and similarly for set \( D_2 \).

\[
D_i = \{x_1, \ldots, x_{n_i}\}, \quad i = \{1, 2\}
\] (33)
Let’s Formalize the Situation

- Suppose we have a set of \( n \) \( d \)-dimensional samples with \( n_1 \) in set \( \mathcal{D}_1 \), and similarly for set \( \mathcal{D}_2 \).

\[
\mathcal{D}_i = \{ x_1, \ldots, x_{n_i} \}, \quad i = \{1, 2\} \tag{33}
\]

- We can form a linear combination of the components of a sample \( x \):

\[
y = w^T x \tag{34}
\]

which yields a corresponding set of \( n \) samples \( y_1, \ldots, y_n \) split into subsets \( \mathcal{Y}_1 \) and \( \mathcal{Y}_2 \).
Geometrically, This is a Projection

- If we constrain the norm of \( w \) to be 1 (i.e., \( \| w \| = 1 \)) then we can conceptualize that each \( y_i \) is the projection of the corresponding \( x_i \) onto a line in the direction of \( w \).
Geometrically, This is a Projection

- If we constrain the norm of $\mathbf{w}$ to be 1 (i.e., $\|\mathbf{w}\| = 1$) then we can conceptualize that each $y_i$ is the projection of the corresponding $x_i$ onto a line in the direction of $\mathbf{w}$.

- Does the magnitude of $\mathbf{w}$ have any real significance?
For our two-class setup, it should be clear that we want the projection that will have those samples from class $\omega_1$ falling into one cluster (on the line) and those samples from class $\omega_2$ falling into a separate cluster (on the line).
For our two-class setup, it should be clear that we want the projection that will have those samples from class $\omega_1$ falling into one cluster (on the line) and those samples from class $\omega_2$ falling into a separate cluster (on the line).

However, this may not be possible depending on our underlying classes.
What is a Good Projection?

- For our two-class setup, it should be clear that we want the projection that will have those samples from class $\omega_1$ falling into one cluster (on the line) and those samples from class $\omega_2$ falling into a separate cluster (on the line).
- However, this may not be possible depending on our underlying classes.
- So, how do we find the best direction $w$?
Let $\mathbf{m}_i$ be the $d$-dimensional sample mean for class $i$:

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{x \in D_i} x .$$  \hspace{1cm} (35)
Separation of the Projected Means

- Let $\mathbf{m}_i$ be the d-dimensional sample mean for class $i$:

$$
\mathbf{m}_i = \frac{1}{n_i} \sum_{x \in D_i} x .
$$

(35)

- Then the sample mean for the projected points is

$$
\tilde{\mathbf{m}}_i = \frac{1}{n_i} \sum_{y \in Y_i} y
$$

(36)

$$
= \frac{1}{n_i} \sum_{x \in D_i} w^T x
$$

(37)

$$
= w^T \mathbf{m}_i .
$$

(38)

And, thus, is simply the projection of $\mathbf{m}_i$. 
The distance between projected means is thus

$$|\tilde{m}_1 - \tilde{m}_2| = |w^T (m_1 - m_2)|$$

(39)
The distance between projected means is thus

\[ |\tilde{m}_1 - \tilde{m}_2| = |w^T (m_1 - m_2)| \tag{39} \]

The scale of \( w \): we can make this distance arbitrarily large by scaling \( w \).
The distance between projected means is thus

$$|\tilde{m}_1 - \tilde{m}_2| = |w^T (m_1 - m_2)|$$

(39)

- The scale of $w$: we can make this distance arbitrarily large by scaling $w$.
- Rather, we want to make this distance large relative to some measure of the standard deviation. ...This story we’ve heard before.
To capture this variation, we compute the scatter

\[ \tilde{s}^2_i = \sum_{y \in \mathcal{Y}_i} (y - \tilde{m}_i)^2 \]  

which is essentially a scaled sampled variance.
To capture this variation, we compute the scatter

\[ \tilde{s}_i^2 = \sum_{y \in Y_i} (y - \tilde{m}_i)^2 \]  \hspace{1cm} (40)

which is essentially a scaled sampled variance.

From this, we can estimate the variance of the pooled data:

\[ \frac{1}{n} \left( \tilde{s}_1^2 + \tilde{s}_2^2 \right) \]  \hspace{1cm} (41)

\( \tilde{s}_1^2 + \tilde{s}_2^2 \) is called the total **within-class scatter** of the projected samples.
The Fisher Linear Discriminant will select the $w$ that maximizes

$$J(w) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}.$$  \hspace{1cm} (42)

It does so independently of the magnitude of $w$. 

This term is the ratio of the distance between the projected means scaled by the within-class scatter (the variation of the data). Recall the similar term from earlier in the lecture which indicated the amount a feature will reduce the probability of error is proportional to the ratio of the difference of the means to the variance. FLD will choose the maximum...
The Fisher Linear Discriminant will select the $w$ that maximizes

$$J(w) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}.$$  \hspace{1cm} (42)

It does so independently of the magnitude of $w$.

This term is the ratio of the distance between the projected means scaled by the within-class scatter (the variation of the data).

Recall the similar term from earlier in the lecture which indicated the amount a feature will reduce the probability of error is proportional to the ratio of the difference of the means to the variance. FLD will choose the maximum...
The Fisher Linear Discriminant will select the $\mathbf{w}$ that maximizes

$$J(\mathbf{w}) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}. \quad (42)$$

- It does so independently of the magnitude of $\mathbf{w}$.
- This term is the ratio of the distance between the projected means scaled by the within-class scatter (the variation of the data).
- Recall the similar term from earlier in the lecture which indicated the amount a feature will reduce the probability of error is proportional to the ratio of the difference of the means to the variance. FLD will choose the maximum...
- We need to rewrite $J(\cdot)$ as a function of $\mathbf{w}$. 
Define scatter matrices $S_i$:

$$S_i = \sum_{x \in D_i} (x - m_i)(x - m_i)^T$$  \hspace{1cm} (43)

and

$$S_W = S_1 + S_2.$$

(44)
Within-Class Scatter Matrix

- Define scatter matrices $S_i$:
  \[
  S_i = \sum_{x \in D_i} (x - m_i)(x - m_i)^T
  \]  
  (43)

- and
  \[
  S_W = S_1 + S_2
  \]  
  (44)

- $S_W$ is called the **within-class scatter matrix**.
- $S_W$ is symmetric and positive semidefinite.
- In typical cases, when is $S_W$ nonsingular?
Within-Class Scatter Matrix

- Deriving the sum of scatters.

\[
\tilde{s}_i^2 = \sum_{x \in D_i} \left( w^T x - w^T m_i \right)^2 
\]

\[
= \sum_{x \in D_i} w^T (x - m_i) (x - m_i)^T w 
\]

\[
= w^T S_i w 
\]

- We can therefore write the sum of the scatters as an explicit function of \( w \):

\[
\tilde{s}_1^2 + \tilde{s}_2^2 = w^T S_W w 
\]
The separation of the projected means obeys

\[(\bar{m}_1 - \bar{m}_2)^2 = (w^T m_1 - w^T m_2)^2\]  
\[= w^T (m_1 - m_2) (m_1 - m_2)^T w\]  
\[= w^T S_B w\]  

Here, \(S_B\) is called the **between-class scatter matrix**:

\[S_B = (m_1 - m_2) (m_1 - m_2)^T\]

\(S_B\) is also symmetric and positive semidefinite.

When is \(S_B\) nonsingular?
We can rewrite our objective as a function of $w$.

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

This is the generalized Rayleigh quotient.
Rewriting the FLD $J(\cdot)$

- We can rewrite our objective as a function of $w$.
  
  $$J(w) = \frac{w^T S_B w}{w^T S_W w}$$  

  (53)

- This is the generalized Rayleigh quotient.
- The vector $w$ that maximizes $J(\cdot)$ must satisfy
  
  $$S_B w = \lambda S_W w$$  

  (54)

  which is a generalized eigenvalue problem.
Rewriting the FLD $J(\cdot)$

- We can rewrite our objective as a function of $w$.

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$  \hspace{1cm} (53)

- This is the generalized Rayleigh quotient.
- The vector $w$ that maximizes $J(\cdot)$ must satisfy

$$S_B w = \lambda S_W w$$ \hspace{1cm} (54)

which is a generalized eigenvalue problem.
- For nonsingular $S_W$ (typical), we can write this as a standard eigenvalue problem:

$$S_W^{-1} S_B w = \lambda w$$ \hspace{1cm} (55)
Since $\|\mathbf{w}\|$ is not important and $S_B \mathbf{w}$ is always in the direction of $(\mathbf{m}_1 - \mathbf{m}_2)$, we can simplify

$$w^* = S_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$  \hspace{1cm} (56)

$w^*$ maximizes $J(\cdot)$ and is the Fisher Linear Discriminant.
Simplifying

Since $\|w\|$ is not important and $S_Bw$ is always in the direction of $(m_1 - m_2)$, we can simplify

$$w^* = S_W^{-1}(m_1 - m_2)$$  (56)

- $w^*$ maximizes $J(\cdot)$ and is the Fisher Linear Discriminant.
- The FLD converts a many-dimensional problem to a one-dimensional one.
Since $\|w\|$ is not important and $S_B w$ is always in the direction of $(m_1 - m_2)$, we can simplify

$$w^* = S_W^{-1}(m_1 - m_2)$$  \hspace{1cm} (56)

- $w^*$ maximizes $J(\cdot)$ and is the Fisher Linear Discriminant.
- The FLD converts a many-dimensional problem to a one-dimensional one.
- One still must find the threshold. This is easy for known densities, but not so easy in general.
A Classic PCA vs. FLD comparison

- PCA maximizes the total scatter across all classes.
- PCA projections are thus optimal for reconstruction from a low dimensional basis, but not necessarily from a discrimination standpoint.
- FLD maximizes the ratio of the between-class scatter and the within-class scatter.
- FLD tries to “shape” the scatter to make it more effective for classification.

Fig. 2. A comparison of principal component analysis (PCA) and Fisher’s linear discriminant (FLD) for a two class problem where data for each class lies near a linear subspace.
Case Study: EigenFaces versus FisherFaces

- Analysis of classic pattern recognition techniques (PCA and FLD) to do face recognition.
- Fixed pose but varying illumination.
- The variation in the resulting images caused by the varying illumination will nearly always dominate the variation caused by an identity change.

Fig. 1. The same person seen under different lighting conditions can appear dramatically different: In the left image, the dominant light source is nearly head-on; in the right image, the dominant light source is from above and to the right.
Case Study: EigenFaces versus FisherFaces

Fact: all of the images of a Lambertian surface, taken from a fixed viewpoint, but under varying illumination, lie in a 3D linear subspace of the high-dimensional image space.
Fact: all of the images of a Lambertian surface, taken from a fixed viewpoint, but under varying illumination, lie in a 3D linear subspace of the high-dimensional image space.

But, in the presence of shadowing, specularities, and facial expressions, the above statement will not hold. This will ultimately result in deviations from the 3D linear subspace and worse classification accuracy.
Method 1: Correlation

- Consider a set of $N$ training images, $\{x_1, x_2, \ldots, x_N\}$.
- We know that each of the $N$ images belongs to one of $c$ classes and can define a $C(\cdot)$ function to map the image $x$ into a class $\omega_c$. 

Pre-Processing – each image is normalized to have zero mean and unit variance.

Why?
- Gets rid of the light source intensity and the effects of a camera’s automatic gain control.

For a query image $x$, we select the class of the training image that is the nearest neighbor in the image space:

$$x^* = \arg \min_{\{x_i\}} \|x - x_i\|$$

then decide $C(x^*)$ (57)

where we have vectorized each image.
Method 1: Correlation  

- Consider a set of \( N \) training images, \( \{x_1, x_2, \ldots, x_N\} \).
- We know that each of the \( N \) images belongs to one of \( c \) classes and can define a \( C(\cdot) \) function to map the image \( x \) into a class \( \omega_c \).
- **Pre-Processing** – each image is normalized to have zero mean and unit variance.
- Why?
Method 1: Correlation

- Consider a set of $N$ training images, $\{x_1, x_2, \ldots, x_N\}$.
- We know that each of the $N$ images belongs to one of $c$ classes and can define a $C(\cdot)$ function to map the image $x$ into a class $\omega_c$.
- **Pre-Processing** – each image is normalized to have zero mean and unit variance.
- Why?
- Gets rid of the light source intensity and the effects of a camera’s automatic gain control.
Method 1: Correlation

- Consider a set of $N$ training images, $\{x_1, x_2, \ldots, x_N\}$.
- We know that each of the $N$ images belongs to one of $c$ classes and can define a $C(\cdot)$ function to map the image $x$ into a class $\omega_c$.
- **Pre-Processing** – each image is normalized to have zero mean and unit variance.
- Why?
- Gets rid of the light source intensity and the effects of a camera’s automatic gain control.
- For a query image $x$, we select the class of the training image that is the nearest neighbor in the image space:

$$x^* = \arg \min_{\{x_i\}} \|x - x_i\|$$

then decide $C(x^*)$ (57)

where we have **vectorized** each image.
Method 1: Correlation

What are the advantages and disadvantages of the correlation based method in this context?
What are the advantages and disadvantages of the correlation based method in this context?

- Computationally expensive.
- Require large amount of storage.
- Noise may play a role.
- Highly parallelizable.
Method 2: EigenFaces

- Quickly recall the main idea of PCA.
- Define a linear projection of the original $n$-d image space into an $m$-d space with $m < n$ or $m \ll n$, which yields new vectors $y$:

$$ y_k = W^T x_k \quad k = 1, 2, \ldots, N $$

(58)

where $W \in \mathbb{R}^{n \times m}$. 
Method 2: EigenFaces

- Quickly recall the main idea of PCA.
- Define a linear projection of the original $n$-d image space into an $m$-d space with $m < n$ or $m \ll n$, which yields new vectors $y$:

$$y_k = W^T x_k \quad k = 1, 2, \ldots, N$$

where $W \in \mathbb{R}^{n \times m}$.

- Define the total scatter matrix $S_T$ as

$$S_T = \sum_{k=1}^{N} (x_k - \mu)(x_k - \mu)^T$$

where $\mu$ is the sample mean image.
Method 2: EigenFaces


The scatter of the projected vectors \( \{y_1, y_2, \ldots, y_N\} \) is

\[
W^T S_T W
\]

(60)
Method 2: EigenFaces

- The scatter of the projected vectors $\{y_1, y_2, \ldots, y_N\}$ is

$$W^T S_T W$$  \hspace{1cm} (60)

- $W_{opt}$ is chosen to maximize the determinant of the total scatter matrix of the projected vectors:

$$W_{opt} = \arg \max_W |W^T S_T W|$$  \hspace{1cm} (61)

$$= \begin{bmatrix} w_1 & w_2 & \ldots & w_m \end{bmatrix}$$  \hspace{1cm} (62)

where $\{w_i \mid i = 1, d, \ldots, m\}$ is the set of $n$-d eigenvectors of $S_T$ corresponding to the largest $m$ eigenvalues.
An Example:

Source: http://www.cs.princeton.edu/~cdecoro/eigenfaces/. (not sure if this dataset include lighting variation...)
What are the advantages and disadvantages of the eigenfaces method in this context?
Method 2: EigenFaces


- What are the advantages and disadvantages of the eigenfaces method in this context?
- The scatter being maximized is due not only to the between-class scatter that is useful for classification but also to the within-class scatter, which is generally undesirable for classification.
- If PCA is presented faces with varying illumination, the projection matrix $W_{opt}$ will contain principal components that retain the variation in lighting. If this variation is higher than the variation due to class identity, then PCA will suffer greatly for classification.
- Yields a more compact representation than the correlation-based method.
Method 3: Linear Subspaces

- Use Lambertian model directly.
- Consider a point $p$ on a Lambertian surface illuminated by a point light source at infinity.
- Let $s \in \mathbb{R}^3$ signify the product of the light source intensity with the unit vector for the light source direction.
- The image intensity of the surface at $p$ when viewed by a camera is

$$E(p) = a(p)n(p)^T s \quad (63)$$

where $n(p)$ is the unit inward normal vector to the surface at point $p$, and $a(p)$ is the albedo of the surface at $p$ (a scalar).
- Hence, the image intensity of the point $p$ is linear on $s$. 
So, if we assume no shadowing, given three images of a Lambertian surface from the same viewpoint under three known, linearly independent light source directions, the albedo and surface normal can be recovered.

Alternatively, one can reconstruct the image of the surface under an arbitrary lighting direction by a linear combination of the three original images.

This fact can be used for classification.
So, if we assume no shadowing, given three images of a Lambertian surface from the same viewpoint under three known, linearly independent light source directions, the albedo and surface normal can be recovered.

Alternatively, one can reconstruct the image of the surface under an arbitrary lighting direction by a linear combination of the three original images.

This fact can be used for classification.

For each face (class) use three or more images taken under different lighting conditions to construct a 3D basis for the linear subspace.

For recognition, compute the distance of a new image to each linear subspace and choose the face corresponding to the shortest distance.
Method 3: Linear Subspaces

- Pros and Cons?
Method 3: Linear Subspaces

- Pros and Cons?
- If there is no noise or shadowing, this method will achieve error free classification under any lighting conditions (and if the surface is indeed Lambertian).
- Faces inevitably have self-shadowing.
- Faces have expressions...
- Still pretty computationally expensive (linear in number of classes).
Method 4: FisherFaces

- Recall the Fisher Linear Discriminant setup.
- The between-class scatter matrix

\[
S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T
\]  
(64)

- The within-class scatter matrix

\[
S_W = \sum_{i=1}^{c} \sum_{x_k \in D_i} (x_k - \mu_i)(x_k - \mu_i)^T
\]  
(65)

- The optimal projection \(W_{opt}\) is chosen as the matrix with orthonormal columns which maximizes the ratio of the determinant of the between-class scatter matrix of the projected vectors to the determinant of the within-class scatter of the projected vectors:

\[
W_{opt} = \arg \max_W \frac{|W^T S_B W|}{|W^T S_W W|}
\]  
(66)
Method 4: FisherFaces

- The eigenvectors \( \{w_i| i = 1, 2, \ldots, m\} \) corresponding to the \( m \) largest eigenvalues of the following generalized eigenvalue problem comprise \( W_{\text{opt}} \):

\[
S_B w_i = \lambda_i S_W w_i, \quad i = 1, 2, \ldots, m \tag{67}
\]

- This is a multi-class version of the FLD, which we will discuss in more detail.
Method 4: FisherFaces

- The eigenvectors \( \{ w_i | i = 1, 2, \ldots, m \} \) corresponding to the \( m \) largest eigenvalues of the following generalized eigenvalue problem comprise \( W_{opt} \):

\[
S_B w_i = \lambda_i S_W w_i, \quad i = 1, 2, \ldots, m
\]  

(67)

- This is a multi-class version of the FLD, which we will discuss in more detail.

- In face recognition, things get a little more complicated because the within-class scatter matrix \( S_W \) is always singular.
Method 4: FisherFaces

- The eigenvectors \( \{ w_i | i = 1, 2, \ldots, m \} \) corresponding to the \( m \) largest eigenvalues of the following generalized eigenvalue problem comprise \( W_{\text{opt}} \):

\[
S_B w_i = \lambda_i S_W w_i, \quad i = 1, 2, \ldots, m \quad (67)
\]

- This is a multi-class version of the FLD, which we will discuss in more detail.

- In face recognition, things get a little more complicated because the within-class scatter matrix \( S_W \) is always singular.

- This is because the rank of \( S_W \) is at most \( N - c \) and the number of images in the learning set are commonly much smaller than the number of pixels in the image.

- This means we can choose \( W \) such that the within-class scatter is exactly zero.
Method 4: FisherFaces

To overcome this, project the image set to a lower dimensional space so that the resulting within-class scatter matrix $S_W$ is nonsingular.

How?
Method 4: FisherFaces

- To overcome this, project the image set to a lower dimensional space so that the resulting within-class scatter matrix $S_W$ is nonsingular.
- How?
- PCA to first reduce the dimension to $N - c$ and the FLD to reduce it to $c - 1$.
- $W_{opt}^T$ is given by the product $W_{FLD}^T W_{PCA}^T$ where

$$W_{PCA} = \arg \max_W |W^T S_T W|$$

(68)

$$W_{FLD} = \arg \max_W \frac{|W^T W_{PCA}^T S_B W_{PCA} W|}{|W^T W_{PCA}^T S_W W_{PCA} W|}$$

(69)
Experiment 1 and 2: Variation in Lighting

- Hypothesis: face recognition algorithms will perform better if they exploit the fact that images of a Lambertian surface lie in a linear subspace.
- Used Hallinan’s Harvard Database which sampled the space of light source directions in 15 degree increments.
- Used 330 images of five people (66 of each) and extracted five subsets.
- Classification is nearest neighbor in all cases.
Experiment 1 and 2: Variation in Lighting

3.1 Variation in Lighting
The first experiment was designed to test the hypothesis that under variable illumination, face recognition algorithms will perform better if they exploit the fact that images of a Lambertian surface lie in a linear subspace. More specifically, the recognition error rates for all four algorithms described in Section 2 are compared using an image database constructed by Hallinan at the Harvard Robotics Laboratory [14], [15]. In each image in this database, a subject held his/her head steady while being illuminated by a dominant light source. The space of light source directions, which can be parameterized by spherical angles, was then sampled in $15^\circ$ increments. See Fig. 3.

From this database, we used 330 images of five people (66 of each). We extracted five subsets to quantify the effects of varying lighting. Sample images from each subset are shown in Fig. 4.

- **Subset 1** contains 30 images for which both the longitudinal and latitudinal angles of light source direction are within $15^\circ$ of the camera axis, including the lighting direction coincident with the camera's optical axis.
- **Subset 2** contains 45 images for which the greater of the longitudinal and latitudinal angles of light source direction are $30^\circ$ from the camera axis.
- **Subset 3** contains 65 images for which the greater of the longitudinal and latitudinal angles of light source direction are $45^\circ$ from the camera axis.
- **Subset 4** contains 85 images for which the greater of the longitudinal and latitudinal angles of light source direction are $60^\circ$ from the camera axis.
- **Subset 5** contains 105 images for which the greater of the longitudinal and latitudinal angles of light source direction are $75^\circ$ from the camera axis.

For all experiments, classification was performed using a nearest neighbor classifier. All training images of an individual were used to train the classifier, and all test images of an individual were used to test the classifier. The results are summarized in Table 1. The error rates were calculated using the leave-one-out cross-validation method.

Fig. 3. The highlighted lines of longitude and latitude indicate the light source directions for Subsets 1 through 5. Each intersection of a longitudinal and latitudinal line on the right side of the illustration has a corresponding image in the database.
Experiment 1: Variation in Lighting


In this section, we present and discuss each of the afore-mentioned face recognition techniques using two different databases. Because of the specific hypotheses that we wanted to test about the relative performance of the considered algorithms, many of the standard databases were inappropriate. So, we have used a database from the Harvard Robotics Laboratory in which lighting has been systematically varied. Secondly, we have constructed a database at Yale that includes variation in both facial expression and lighting.

3.1 Variation in Lighting

The first experiment was designed to test the hypothesis that under variable illumination, face recognition algorithms will perform better if they exploit the fact that images of a Lambertian surface lie in a linear subspace. More specifically, the recognition error rates for all four algorithms described in Section 2 are compared using an image database constructed by Hallinan at the Harvard Robotics Laboratory [14], [15]. In each image in this database, a subject held his/her head steady while being illuminated by a dominant light source. The space of light source directions, which can be parameterized by spherical angles, was then sampled in 15 increments. See Fig. 3.

From this database, we used 330 images of five people (66 of each). We extracted five subsets to quantify the effects of varying lighting. Sample images from each subset are shown in Fig. 4.

Subset 1 contains 30 images for which both the longitudinal and latitudinal angles of light source direction are within 15° of the camera axis, including the lighting direction coincident with the camera's optical axis.

Subset 2 contains 45 images for which the greater of the longitudinal and latitudinal angles of light source direction are 30° from the camera axis.

Subset 3 contains 65 images for which the greater of the longitudinal and latitudinal angles of light source direction are 45° from the camera axis.

Subset 4 contains 85 images for which the greater of the longitudinal and latitudinal angles of light source direction are 60° from the camera axis.

Subset 5 contains 105 images for which the greater of the longitudinal and latitudinal angles of light source direction are 75° from the camera axis.

For all experiments, classification was performed using a nearest neighbor classifier. All training images of an individual were from a single subset. See Fig. 4.
Experiment 1: Variation in Lighting – Extrapolation

- Train on Subset 1.
- Test of Subsets 1, 2, and 3.
Comparisons

Case Study: Faces

Experiment 2: Variation in Lighting – Interpolation

- Train on Subsets 1 and 5.
- Test of Subsets 2, 3, and 4.

![Diagram showing error rates for different methods and subsets.]

### Table: Interpolating between Subsets 1 and 5

<table>
<thead>
<tr>
<th>Method</th>
<th>Reduced Space</th>
<th>Error Rate (%)</th>
<th>Subset 2</th>
<th>Subset 3</th>
<th>Subset 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenface</td>
<td>4</td>
<td>53.3</td>
<td>75.4</td>
<td>52.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>11.11</td>
<td>33.9</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>Eigenface</td>
<td>4</td>
<td>31.11</td>
<td>60.0</td>
<td>29.4</td>
<td></td>
</tr>
<tr>
<td>w/o 1st 3</td>
<td>10</td>
<td>6.7</td>
<td>20.0</td>
<td>12.9</td>
<td></td>
</tr>
<tr>
<td>Correlation</td>
<td>129</td>
<td>0.0</td>
<td>21.54</td>
<td>7.1</td>
<td></td>
</tr>
<tr>
<td>Linear Subspace</td>
<td>15</td>
<td>0.0</td>
<td>1.5</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Fisherface</td>
<td>4</td>
<td>0.0</td>
<td>0.0</td>
<td>1.2</td>
<td></td>
</tr>
</tbody>
</table>
Experiment 1 and 2: Variation in Lighting

- All of the algorithms perform perfectly when lighting is nearly frontal. However, when lighting is moved off axis, there is significant difference between the methods (spec. the class-specific methods and the Eigenface method).
Experiment 1 and 2: Variation in Lighting

- All of the algorithms perform perfectly when lighting is nearly frontal. However, when lighting is moved off axis, there is significant difference between the methods (spec. the class-specific methods and the Eigenface method).
- Empirically demonstrated that the Eigenface method is equivalent to correlation when the number of Eigenfaces equals the size of the training set (Exp. 1).
Experiment 1 and 2: Variation in Lighting

- All of the algorithms perform perfectly when lighting is nearly frontal. However, when lighting is moved off axis, there is significant difference between the methods (spec. the class-specific methods and the Eigenface method).
- Empirically demonstrated that the Eigenface method is equivalent to correlation when the number of Eigenfaces equals the size of the training set (Exp. 1).
- In the Eigenface method, removing the first three principal components results in better performance under variable lighting conditions.
Experiment 1 and 2: Variation in Lighting

• All of the algorithms perform perfectly when lighting is nearly frontal. However, when lighting is moved off axis, there is significant difference between the methods (spec. the class-specific methods and the Eigenface method).
• Empirically demonstrated that the Eigenface method is equivalent to correlation when the number of Eigenfaces equals the size of the training set (Exp. 1).
• In the Eigenface method, removing the first three principal components results in better performance under variable lighting conditions.
• Linear Subspace has comparable error rates with the FisherFace method, but it requires 3x as much storage and takes three times as long.
Experiment 1 and 2: Variation in Lighting

• All of the algorithms perform perfectly when lighting is nearly frontal. However, when lighting is moved off axis, there is significant difference between the methods (spec. the class-specific methods and the Eigenface method).
• Empirically demonstrated that the Eigenface method is equivalent to correlation when the number of Eigenfaces equals the size of the training set (Exp. 1).
• In the Eigenface method, removing the first three principal components results in better performance under variable lighting conditions.
• Linear Subspace has comparable error rates with the FisherFace method, but it requires 3x as much storage and takes three times as long.
• The Fisherface method had error rates lower than the Eigenface method and required less computation time.
Experiment 3: Yale DB

- Uses a second database of 16 subjects with ten images of images (5 varying in expression).

![Graph showing error rate comparison between different recognition algorithms.](image_url)
Experiment 3: Comparing Variation with Number of Principal Components


The relative performance of the algorithms is self evident in Fig. 9. The Fisherface method had error rates that were better than half that of any other method. It seems that the Fisherface method chooses the set of projections which performs well over a range of lighting variation, facial expression variation, and presence of glasses.

Note that the Linear Subspace method faired comparatively worse in this experiment than in the lighting experiments in the previous section. Because of variation in facial expression, the images no longer lie in a linear subspace. Since the Fisherface method tends to discount those portions of the image that are not significant for recognizing an individual, the resulting projections $W$ tend to mask the regions of the face that are highly variable. For example, the area around the mouth is discounted, since it varies quite a bit for different facial expressions. On the other hand, the nose, cheeks, and brow are stable over the within-class variation.
Can PCA outperform FLD for recognition?

- There may be situations in which PCA might outperform FLD.
- Can you think of such a situation?
Can PCA outperform FLD for recognition?


- There may be situations in which PCA might outperform FLD.
- Can you think of such a situation?

![Diagram showing two classes embedded in two different distributions.](image)

- When we have few training data, then it may be preferable to describe total scatter.
Can PCA outperform FLD for recognition?

- They tested on the AR face database and found affirmative results.

![Graphs showing performance curves for different dimensionality tests.](image-url)
Multiple Discriminant Analysis

- We can generalize the Fisher Linear Discriminant to multiple classes.
- Indeed we saw it done in the Case Study on Faces. But, let’s cover it with some rigor.
- Let’s make sure we’re all at the same place: How many discriminant functions will be involved in a $c$ class problem?
Multiple Discriminant Analysis

- We can generalize the Fisher Linear Discriminant to multiple classes.
- Indeed we saw it done in the Case Study on Faces. But, let’s cover it with some rigor.
- Let’s make sure we’re all at the same place: How many discriminant functions will be involved in a \( c \) class problem?
- **Key:** There will be \( c - 1 \) projection functions for a \( c \) class problem and hence the projection will be from a \( d \)-dimensional space to a \( (c - 1) \)-dimensional space.
- \( d \) must be greater than or equal to \( c \).
The projection is accomplished by \( c - 1 \) discriminant functions:

\[
y_i = w_i^T x \quad i = 1, \ldots, c - 1 \tag{70}
\]

which is summarized in matrix form as

\[
y = W^T x \tag{71}
\]

where \( W \) is the \( d \times (c - 1) \) projection function.
The generalization for the Within-Class Scatter Matrix is straightforward:

\[
S_W = \sum_{i=1}^{c} S_i \tag{72}
\]

where, as before,

\[
S_i = \sum_{x \in D_i} (x - m_i)(x - m_i)^T
\]

and

\[
m_i = \frac{1}{n_i} \sum_{x \in D_i} x .
\]
The between-class scatter matrix $S_B$ is not so easy to generalize.
The between-class scatter matrix $S_B$ is not so easy to generalize.

Let’s define a *total mean vector*

$$m = \frac{1}{n} \sum x = \frac{1}{n} \sum_{i=1}^{c} n_i m_i$$  \hspace{1cm} (73)

Recall then *total scatter matrix*

$$S_T = \sum_{x} (x - m)(x - m)^T$$  \hspace{1cm} (74)
Then it follows that

\[ S_T = \sum_{i=1}^{c} \sum_{x \in D_i} (x - m_i + m_i - m)(x - m_i + m_i - m)^T \]  

(75)

\[ = \sum_{i=1}^{c} \sum_{x \in D_i} (x - m_i)(x - m_i)^T + \sum_{i=1}^{c} \sum_{x \in D_i} (m_i - m)(m_i - m)^T \]  

(76)

\[ = S_W + \sum_{i=1}^{c} n_i (m_i - m)(m_i - m)^T \]  

(77)
MDA Between-Class Scatter Matrix

Then it follows that

\[ S_T = \sum_{i=1}^{c} \sum_{x \in D_i} (x - m_i + m_i - m)(x - m_i + m_i - m)^T \]  

(75)

\[ = \sum_{i=1}^{c} \sum_{x \in D_i} (x - m_i)(x - m_i)^T + \sum_{i=1}^{c} \sum_{x \in D_i} (m_i - m)(m_i - m)^T \]  

(76)

\[ = S_W + \sum_{i=1}^{c} n_i (m_i - m)(m_i - m)^T \]  

(77)

So, we can define this second term as a generalized between-class scatter matrix.

The total scatter is the sum of the within-class scatter and the between-class scatter.
We again seek a criterion that will maximize the between-class scatter of the projected vectors to the within-class scatter of the projected vectors.

Recall that we can write down the scatter in terms of these projections:

\[
\tilde{m}_i = \frac{1}{n_i} \sum_{y \in Y_i} y \quad \text{and} \quad \tilde{m} = \frac{1}{n} \sum_{i=1}^{c} n_i \tilde{m}_i \tag{78}
\]

\[
\tilde{S}_W = \sum_{i=1}^{c} \sum_{y \in Y_i} (y - \tilde{m}_i)(y - \tilde{m}_i)^T \tag{79}
\]

\[
\tilde{S}_B = \sum_{i=1}^{c} n_i (\tilde{m}_i - \tilde{m})(\tilde{m}_i - \tilde{m})^T \tag{80}
\]
Then, we can show

\[ \tilde{S}_W = W^T S_W W \quad (81) \]
\[ \tilde{S}_B = W^T S_B W \quad (82) \]
Then, we can show

\[ \tilde{S}_W = W^T S_W W \]  \hspace{1cm} (81)
\[ \tilde{S}_B = W^T S_B W \]  \hspace{1cm} (82)

A simple scalar measure of scatter is the \textbf{determinant} of the scatter matrix. The determinant is the product of the eigenvalues and thus is the product of the variation along the principal directions.

\[ J(W) = \frac{|\tilde{S}_B|}{|\tilde{S}_W|} = \frac{|W^T S_B W|}{|W^T S_W W|} \]  \hspace{1cm} (83)
MDA Solution

- The columns of an optimal $W$ are the generalized eigenvectors that correspond to the largest eigenvalues in

$$S_B w_i = \lambda_i S_W w_i$$

(84)
MDA Solution

- The columns of an optimal $W$ are the generalized eigenvectors that correspond to the largest eigenvalues in

$$S_B w_i = \lambda_i S_W w_i$$  (84)

- If $S_W$ is nonsingular, then this can be converted to a conventional eigenvalue problem. Or, we could notice that the rank of $S_B$ is at most $c - 1$ and do some clever algebra...
The columns of an optimal $W$ are the generalized eigenvectors that correspond to the largest eigenvalues in

$$S_B w_i = \lambda_i S_W w_i$$  \hspace{1cm} (84)

If $S_W$ is nonsingular, then this can be converted to a conventional eigenvalue problem. Or, we could notice that the rank of $S_B$ is at most $c - 1$ and do some clever algebra...

The solution for $W$ is, however, not unique and would allow arbitrary scaling and rotation, but these would not change the ultimate classification.
**Observation:** To apply PCA and FLD on images, we need to first “vectorize” them, which can lead to high-dimensional vectors. Solving the associated eigen-problems is a very time-consuming process.
**Observation:** To apply PCA and FLD on images, we need to first “vectorize” them, which can lead to high-dimensional vectors. Solving the associated eigen-problems is a very time-consuming process.

**So, can we apply PCA on the images directly?**
Observation: To apply PCA and FLD on images, we need to first “vectorize” them, which can lead to high-dimensional vectors. Solving the associated eigen-problems is a very time-consuming process.

So, can we apply PCA on the images directly?

Yes!

This will be accomplished by what the authors’ call the image total covariance matrix.
Define $\mathbf{A} \in \mathbb{R}^{m \times n}$ as our image.

Let $\mathbf{w}$ denote an $n$-dimensional column vector, which will represent the subspace onto which we will project an image.

$$\mathbf{y} = \mathbf{A}\mathbf{w}$$

which yields $m$-dimensional vector $\mathbf{y}$.

You might think of $\mathbf{w}$ this as a “feature selector.”

We, again, want to maximize the total scatter...
IMPCA: Image Total Scatter Matrix


- We have $M$ total data samples.
- The sample mean image is

$$M = \frac{1}{M} \sum_{j=1}^{M} A_j$$ (86)

- And the projected sample mean is

$$\tilde{m} = \frac{1}{M} \sum_{j=1}^{M} y_j$$ (87)

$$= \frac{1}{M} \sum_{j=1}^{M} A_j w$$ (88)

$$= Mw$$ (89)
The scatter of the projected samples is

\[ \tilde{S} = \sum_{j=1}^{M} (y_j - \tilde{m})(y_j - \tilde{m})^T \]

(90)

\[ = \sum_{j=1}^{M} [(A_j - M)w] [(A_j - M)w]^T \]

(91)

\[ tr(\tilde{S}) = w^T \left( \sum_{j=1}^{M} (A_j - M)^T (A_j - M) \right) w \]

(92)
So, the **image total scatter matrix** is

\[
S_I = \sum_{j=1}^{M} (A_j - M)^T (A_j - M)
\]  \hspace{1cm} (93)
IMPCA: Image Total Scatter Matrix

- So, the **image total scatter matrix** is

\[
S_I = \sum_{j=1}^{M} (A_j - M)^T (A_j - M)
\]  

(93)

- And, a suitable criterion, in a form we’ve seen before is

\[
J_I(w) = w^T S_I w
\]  

(94)
So, the **image total scatter matrix** is

\[
S_I = \sum_{j=1}^{M} (A_j - M)^T (A_j - M) \quad (93)
\]

And, a suitable criterion, in a form we’ve seen before is

\[
J_I(w) = w^T S_I w \quad (94)
\]

We know already that the vectors \( w \) that maximize \( J_I \) are the orthonormal eigenvectors of \( S_I \) corresponding to the largest eigenvalues of \( S_I \).
Dataset is the ORL database
- 10 different images taken of 40 individuals.
- Facial expressions and details are varying.
- Even the pose can vary slightly: 20 degree rotation/tilt and 10% scale.
- Size is normalized (92 × 112).

The first five images of each person are used for training and the second five are used for testing.
### IMPCA: Experimental Results

**IMPCA varying the number of extracted eigenvectors (NN classifier):**

<table>
<thead>
<tr>
<th>Projection vector number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum distance</td>
<td>73.0</td>
<td>83.0</td>
<td>86.5</td>
<td>88.5</td>
<td>88.5</td>
</tr>
<tr>
<td>Nearest neighbor</td>
<td>85.0</td>
<td>92.0</td>
<td>93.5</td>
<td>94.5</td>
<td>94.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Projection vector number</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum distance</td>
<td>88.5</td>
<td>90.0</td>
<td>90.5</td>
<td>91.0</td>
<td>91.0</td>
</tr>
<tr>
<td>Nearest neighbor</td>
<td>95.0</td>
<td>95.0</td>
<td>95.5</td>
<td>93.5</td>
<td>94.0</td>
</tr>
</tbody>
</table>
IMPCA: Experimental Results

IMPCA vs. EigenFaces vs. FisherFaces for Recognition

<table>
<thead>
<tr>
<th>Recognition rate</th>
<th>Eigenfaces</th>
<th>Fisherfaces</th>
<th>IMPCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum distance</td>
<td>89.5% (46)</td>
<td>88.5% (39)</td>
<td>91.0%</td>
</tr>
<tr>
<td>Nearest neighbor</td>
<td>93.5% (37)</td>
<td>88.5% (39)</td>
<td>95.5%</td>
</tr>
</tbody>
</table>
### IMPCA: Experimental Results


#### IMPCA vs. EigenFaces vs. FisherFaces for Recognition

<table>
<thead>
<tr>
<th>Recognition rate</th>
<th>Eigenfaces</th>
<th>Fisherfaces</th>
<th>IMPCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum distance</td>
<td>89.5% (46)</td>
<td>88.5% (39)</td>
<td>91.0%</td>
</tr>
<tr>
<td>Nearest neighbor</td>
<td>93.5% (37)</td>
<td>88.5% (39)</td>
<td>95.5%</td>
</tr>
</tbody>
</table>

#### IMPCA vs. EigenFaces vs. FisherFaces for Speed

<table>
<thead>
<tr>
<th></th>
<th>Feature extraction time</th>
<th>Classification time</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenfaces (37)</td>
<td>371.79</td>
<td>5.16</td>
<td>376.95</td>
</tr>
<tr>
<td>Fisherfaces (39)</td>
<td>378.10</td>
<td>5.27</td>
<td>383.37</td>
</tr>
<tr>
<td>IMPCA (112 × 8)</td>
<td>27.14</td>
<td>25.04</td>
<td>52.18</td>
</tr>
</tbody>
</table>
There is a clear speed-up because the amount of computation has been greatly reduced.
There is a clear speed-up because the amount of computation has been greatly reduced.

However, this speed-up comes at some cost. What is that cost?
There is a clear speed-up because the amount of computation has been greatly reduced.

However, this speed-up comes at some cost. What is that cost?

Why does IMPCA work in this case?

What is IMPCA really doing?
So far, we have covered a few methods for dimension reduction that all make the underlying assumption the data in high dimension lives on a planar manifold in the lower dimension.

- The methods are easy to implement.
- The methods do not suffer from local minima.

Recently, we introduced an eigenvector method—called locally linear embedding (LLE)—for the problem of nonlinear dimensionality reduction. This problem is illustrated by the nonlinear manifold in Figure 1. In this example, the dimensionality reduction by LLE succeeds in identifying the underlying structure of the manifold, while projections of the data by PCA or metric MDS map far away data points to nearby points in the plane. Like PCA and MDS, our algorithm is simple to implement, and its optimizations do not involve local minima. At the same time, however, it is capable of generating highly nonlinear embeddings. Note that mixture models for local dimensionality reduction, which cluster the data and perform PCA within each cluster, do not address the problem considered here—namely, how to map high dimensional data into a single global coordinate system of lower dimensionality.

In this paper, we review the LLE algorithm in its most basic form and illustrate a potential application to audio-visual speech synthesis.
Locally Linear Embedding
Source: Saul and Roweis, An Introduction to Locally Linear Embedding, 2001

- So far, we have covered a few methods for dimension reduction that all make the underlying assumption the data in high dimension lives on a planar manifold in the lower dimension.
  - The methods are easy to implement.
  - The methods do not suffer from local minima.
- But, what if the data is non-linear?

![Diagram of a 3D manifold and its 2D embedding](image)

Figure 1: The problem of nonlinear dimensionality reduction, as illustrated for three dimensional data (B) sampled from a two dimensional manifold (A). An unsupervised learning algorithm must discover the global internal coordinates of the manifold without signals that explicitly indicate how the data should be embedded in two dimensions. The shading in (C) illustrates the neighborhood-preserving mapping discovered by LLE.
In non-linear dimension reduction, one must discover the global internal coordinates of the manifold without signals that explicitly indicate how the data should be embedded in the lower dimension (or even how many dimensions should be used).
In non-linear dimension reduction, one must discover the global internal coordinates of the manifold without signals that explicitly indicate how the data should be embedded in the lower dimension (or even how many dimensions should be used).

The LLE way of doing this is to make the assumption that, given enough data samples, the local frame of a particular point $x_i$ is linear. LLE proceeds to preserve this local structure while simultaneously reducing the global dimension (indeed preserving the local structure gives LLE the necessary constraints to discover the manifold).
Suppose we have $n$ real-valued vectors $x_i$ in dataset $D$ each of dimension $D$. We assume these data are sampled from some smooth underlying manifold.
Suppose we have $n$ real-valued vectors $x_i$ in dataset $\mathcal{D}$ each of dimension $D$. We assume these data are sampled from some smooth underlying manifold.

Furthermore, we assume that each data point and its neighbors lie on or close to a locally linear patch of the manifold.
LLE: Neighborhoods
Source: Saul and Roweis, An Introduction to Locally Linear Embedding, 2001

- Suppose we have $n$ real-valued vectors $x_i$ in dataset $D$ each of dimension $D$. We assume these data are sampled from some smooth underlying manifold.
- Furthermore, we assume that each data point and its neighbors lie on or close to a locally linear patch of the manifold.
- LLE characterizes the local geometry of these patches by linear coefficients that reconstruct each data point from its neighbors.
  - If the neighbors form the $D$-dimensional simplex, then these coefficients form the barycentric coordinates of the data point.
  - In the simplest form of LLE, one identifies $K$ such nearest neighbors based on the Euclidean distance.
  - But, one can use all points in a ball of fixed radius, or even more sophisticated metrics.
We can write down the reconstruction error with the following cost function:

\[ J_{LLE_1}(W) = \sum_i \left| x_i - \sum_j W_{ij} x_j \right|^2 \] (95)

Notice that each row of the weight matrix \( W \) will be nonzero for only \( K \) columns; i.e., \( W \) is a quite sparse matrix. I.e., if we define the set \( N(x_i) \) as the \( K \) neighbors of \( x_i \), then we enforce

\[ W_{ij} = 0 \quad \text{if} \quad x_j \notin N(x_i) \] (96)

We will enforce that the rows sum to 1, i.e., \( \sum_j W_{ij} = 1 \). (This is for invariance.)
Note that the constrained weights that minimize these reconstruction errors obey important symmetries: for any data point, they are invariant to rotations, rescalings, and translations of that data point and its neighbors.
LLE: Neighborhoods
Source: Saul and Roweis, An Introduction to Locally Linear Embedding, 2001

- Note that the constrained weights that minimize these reconstruction errors obey important symmetries: for any data point, they are invariant to rotations, rescalings, and translations of that data point and its neighbors.

- This means that these weights characterize intrinsic geometric properties of the neighborhood as opposed to any properties that depend on a particular frame of reference.
Note that the constrained weights that minimize these reconstruction errors obey important symmetries: for any data point, they are invariant to rotations, rescalings, and translations of that data point and its neighbors.

This means that these weights characterize intrinsic geometric properties of the neighborhood as opposed to any properties that depend on a particular frame of reference.

If we suppose the data lie near a smooth nonlinear manifold of dimension $d \ll D$. Then, to a good approximation, there exists a linear mapping (a translation, rotation, and scaling) that maps the high dimensional coordinates of each neighborhood to global internal coordinates on the manifold.
So, we expect that the characterization of the local neighborhoods \((W)\) in the original space to be equally valid for the local patches on the manifold.

In other words, the same weights \(W_{ij}\) that reconstruct point \(x_i\) in the original space should also reconstruct it in the embedded manifold coordinate system.

First, let’s solve for the weights. And, then we’ll see how to use this point to ultimately compute the global dimension reduction.
Solving for the Weights

Source: Saul and Roweis, An Introduction to Locally Linear Embedding, 2001

- Solving for the weights $W$ is a constrained least-squares problem.
- Consider a particular $x$ with $K$ nearest neighbors $\eta_j$ and weights $w_j$ which sum to one. We can write the reconstruction error as

$$
\epsilon = \left| x - \sum_j w_j \eta_j \right|^2
$$

(97)

$$
= \left| \sum_j w_j (x - \eta_j) \right|^2
$$

(98)

$$
= \sum_{jk} w_j w_k C_{jk}
$$

(99)

where $C_{jk}$ is the covariance $(x - \eta_j)(x - \eta_k)^{T}$. 
Solving for the Weights
Source: Saul and Roweis, An Introduction to Locally Linear Embedding, 2001

- We need to add a Lagrange multiplier to enforce the constraint $\sum_j w_j = 1$ and then the weights can be solved in closed form.
- The optimal weights, in terms of the local covariance matrix, are

$$w_j = \frac{\sum_k C_{jk}^{-1}}{\sum_{lm} C_{lm}^{-1}} \quad \text{.(100)}$$
We need to add a Lagrange multiplier to enforce the constraint \( \sum_j w_j = 1 \) and then the weights can be solved in closed form.

The optimal weights, in terms of the local covariance matrix, are

\[
    w_j = \frac{\sum_k C^{-1}_{jk}}{\sum_{lm} C^{-1}_{lm}}.
\]  

(100)

But, rather than explicitly inverting the covariance matrix, one can solve the linear system

\[
    \sum_k C_{jk} w_k = 1
\]  

(101)

and rescale the weights so that they sum to one.
Stage 2: Neighborhood Preserving Mapping
Source: Saul and Roweis, An Introduction to Locally Linear Embedding, 2001

- Each high-dimensional input vector $x_i$ is mapped to a low-dimension vector $y_i$ representing the global internal coordinates on the manifold.

- LLE does this by choosing the $d$-dimension coordinates $y_i$ to minimize the embedding cost function:

\[
J_{\text{LLE}_2}(y) = \sum_i \left| y_i - \sum_j W_{ij} y_j \right| \tag{102}
\]

- The basis for the cost function is the same—locally linear reconstruction errors—but here, the weights $W$ are fixed and the coordinates $y$ are optimized.
Stage 2: Neighborhood Preserving Mapping
Source: Saul and Roweis, An Introduction to Locally Linear Embedding, 2001

This defines a quadratic:

\[ J_{\text{LLE}}(y) = \sum \left| y_i - \sum_j W_{ij} y_j \right| \]

\[ = \sum_{ij} M_{ij} (y_i^T y_j) \]

where

\[ M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki} W_{kj} \]

with \( \delta_{ij} \) is 1 if \( j \neq i \) and 0 otherwise.
To remove the degree of freedom for translation, we enforce the coordinates to be centered at the origin:

$$\sum_i y_i = 0$$  \hspace{1cm} (106)

To avoid degenerate solutions, we constrain the embedding vectors to have unit covariance, with their outer-products satisfying

$$\frac{1}{n} \sum_i y_i y_i^T = I$$  \hspace{1cm} (107)

The optimal embedding is found by the bottom $d+1$ non-zero eigenvectors, i.e., those $d+1$ eigenvectors corresponding to the smallest but non-zero $d+1$ eigenvalues. The bottom eigenvector is the unit vector the corresponds to the free translation, it is discarded.
LLE ALGORITHM

1. Compute the neighbors of each data point, $\tilde{X}_i$.

2. Compute the weights $W_{ij}$ that best reconstruct each data point $\tilde{X}_i$ from its neighbors, minimizing the cost in eq. (1) by constrained linear fits.

3. Compute the vectors $\tilde{Y}_i$ best reconstructed by the weights $W_{ij}$, minimizing the quadratic form in eq. (2) by its bottom nonzero eigenvectors.
Example 1
Source: Saul and Roweis, An Introduction to Locally Linear Embedding, 2001

K=12.
Example 2
Source: Saul and Roweis, An Introduction to Locally Linear Embedding, 2001

K = 4.
Example 3
Source: Saul and Roweis, An Introduction to Locally Linear Embedding, 2001

If the lip images described a nearly linear manifold, these two methods would yield similar results; thus the significant differences in these embeddings reveal the presence of nonlinear structure.
Computing nearest neighbors scales $O(Dn^2)$ in the worst case. But, in many situations, space partitioning methods can be used to find the $K$ nearest neighbors in $O(n \log n)$ time.
Computing nearest neighbors scales $O(Dn^2)$ in the worst case. But, in many situations, space partitioning methods can be used to find the $K$ nearest neighbors in $O(n \log n)$ time.

- Computing the weights is $O(DnK^3)$. 
LLE Complexity

Source: Saul and Roweis, An Introduction to Locally Linear Embedding, 2001

- Computing nearest neighbors scales $O(Dn^2)$ in the worst case. But, in many situations, space partitioning methods can be used to find the $K$ nearest neighbors in $O(n \log n)$ time.
- Computing the weights is $O(DnK^3)$.
- Computing the projection is $O(dn^2)$.
LLE Complexity
Source: Saul and Roweis, An Introduction to Locally Linear Embedding, 2001

- Computing nearest neighbors scales $O(Dn^2)$ in the worst case. But, in many situations, space partitioning methods can be used to find the $K$ nearest neighbors in $O(n \log n)$ time.
- Computing the weights is $O(DnK^3)$.
- Computing the projection is $O(dn^2)$.
- All matrix computations are on very sparse matrices and can thus be implemented quite efficiently.