DIMENSIONALITY REDUCTION

J. Elder

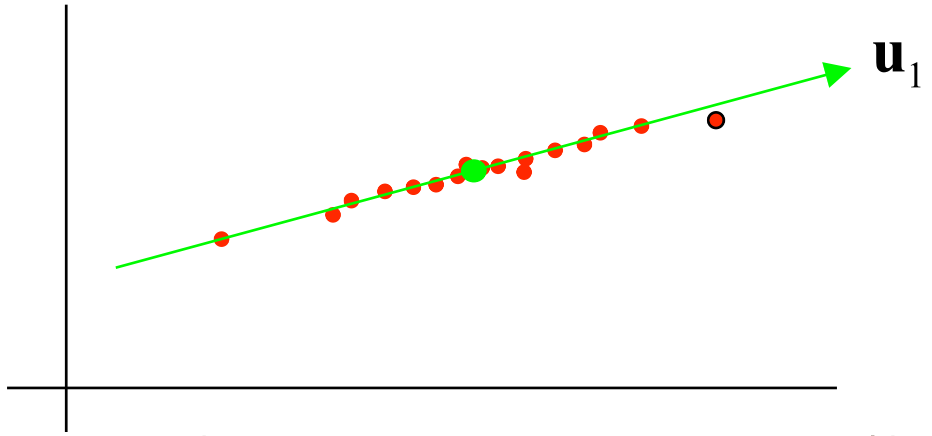
CSE 4404/5327 Introduction to Machine Learning and Pattern Recognition

Credits

- Some of these slides were sourced and/or modified from Simon Prince, University College London

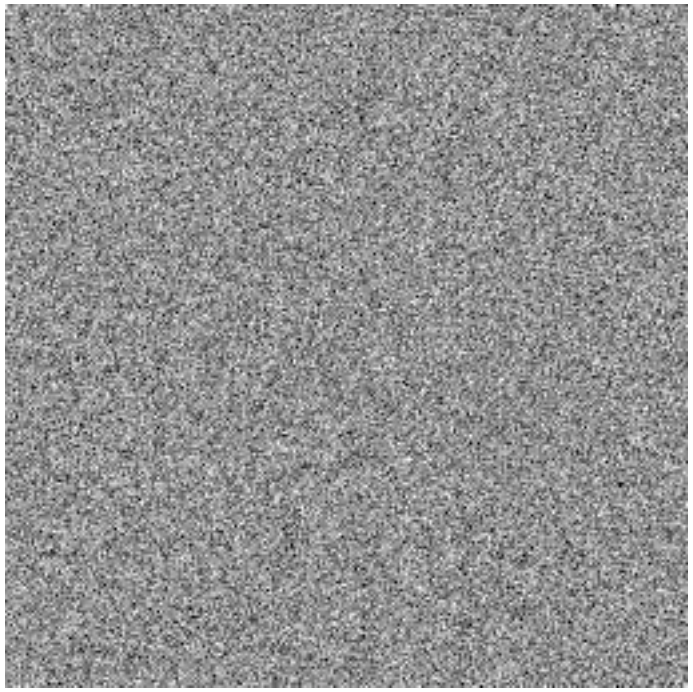
Subspace Models

- Often in machine learning problems, input vectors have high dimensionality *D*
  - for an 1800 x 1200 colour image, *D*≅6.5 million.
  - for a 1-second acoustic voice signal sampled at 5kHz, *D* = 5,000
- There is typically insufficient training data to learn a probabilistic model in such a high-dimensional space.
- Fortunately, these signals usually live in a much smaller subspace, or manifold, of this high-dimensional space.



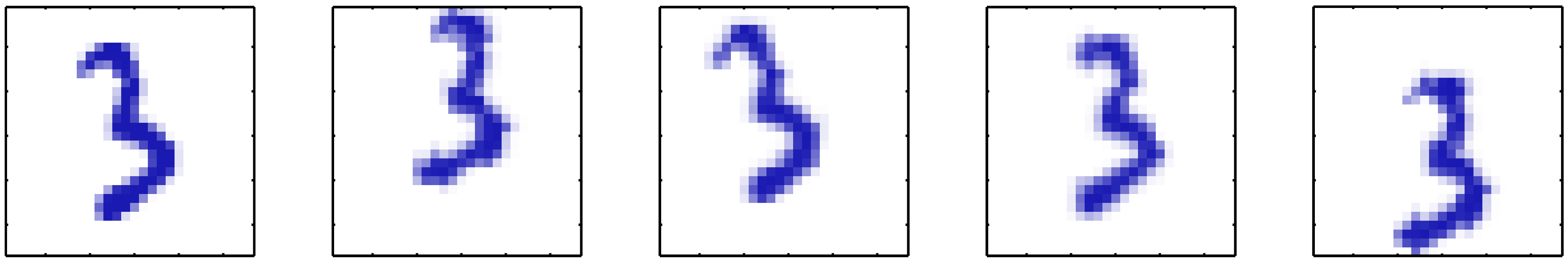
Subspace Models

- For example, you will have to wait a long time before a sample of white noise looks like a natural image.



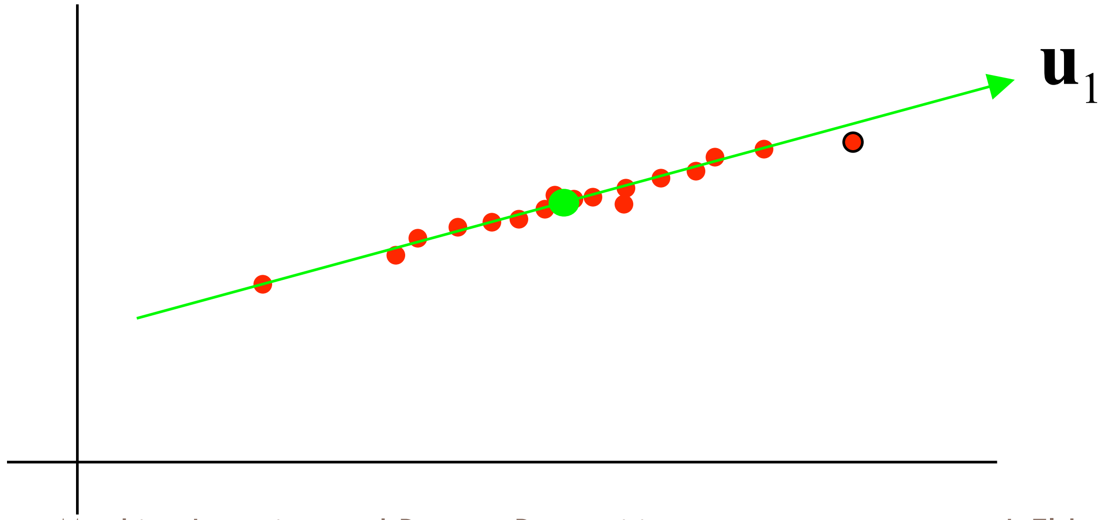
Subspace Models

- e.g., standard transformations (e.g., translations, rotations, scalings) of objects produce images populating a low-dimensional manifold embedded in this high-dimensional space



Subspace Models

- The goal of subspace methods is to discover the low-dimensional subspace in which the data lie and exploit the lower-dimensionality to allow efficient and detailed modeling.



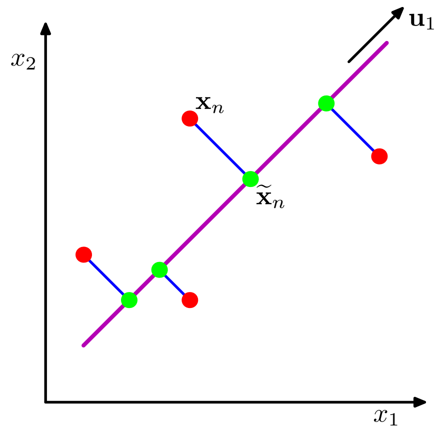
Subspace Models

- We will mainly consider linear subspaces
  - A line if $D=2$
  - A line or a plane if $D=3$
  - A hyperplane of dimensionality $[1,\ldots,D-1]$ for higher D
- But we will also consider some methods to deal with nonlinear manifolds.

Principal Component Analysis

Principal Component Analysis

- PCA finds the linear subspace that
  - maximizes the explained variance
  - equivalently, minimizes the unexplained variance
- PCA can be applied to any multidimensional dataset
  - (data do not have to be Gaussian)



Maximum Variance Formulation

Observations $\left\{\mathbf{x}_\mathbf{n}\right\}, n = 1, \ldots N$

Observation $\mathbf{x}_\mathbf{n}$ is a high-dimensional vector of dimension D

Let $\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_\mathbf{n}$ be the sample mean and $\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{x}_\mathbf{n} - \bar{\mathbf{x}}\right)\left(\mathbf{x}_\mathbf{n} - \bar{\mathbf{x}}\right)^t$ be the sample covariance

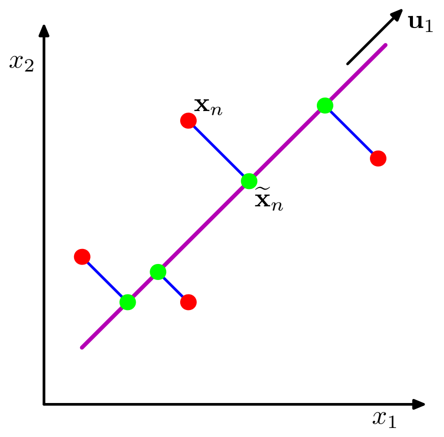
Goal: Project the data onto subspace of dimension $M < D$

Consider a direction in the data space given by unit vector $\mathbf{u}_\mathbf{1}$.

Now imagine projecting all of the data onto this unit vector.

The mean of the projected data is $\mathbf{u}_\mathbf{1}^\mathbf{t}\bar{\mathbf{x}}$.

The variance of the projected data is $\frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{u}_\mathbf{1}^\mathbf{t}\mathbf{x}_\mathbf{n} - \mathbf{u}_\mathbf{1}^\mathbf{t}\bar{\mathbf{x}}\right)^2 = \mathbf{u}_\mathbf{1}^\mathbf{t}\mathbf{S}\mathbf{u}_\mathbf{1}$



Maximum Variance Formulation

We want to select the unit vector $\mathbf{u}_1$ that maximizes the projected variance $\mathbf{u}_1^t\mathbf{S}\mathbf{u}_1$

To do this, we use a Lagrange multiplier λ_1 to maintain the constraint that $\mathbf{u}_1$ be a unit vector.

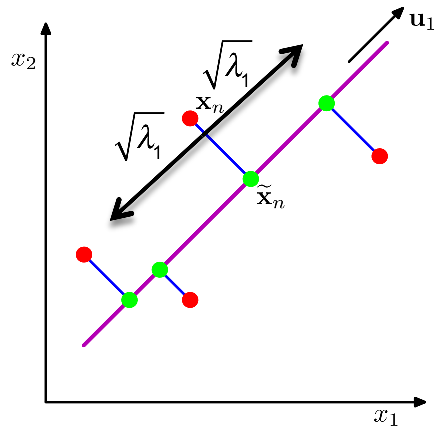
Thus we seek to maximize $\mathbf{u}_1^t\mathbf{S}\mathbf{u}_1 + \lambda_1\left(1 - \mathbf{u}_1^t\mathbf{u}_1\right)$

Setting the derivative with respect to $\mathbf{u}_1$ to 0, we have $S\mathbf{u}_1 = \lambda_1\mathbf{u}_1$

Thus $\mathbf{u}_1$ is an eigenvector of $\mathbf{S}$.

Left-multiplying by $\mathbf{u}_1^t$, we see that the projected variance $\mathbf{u}_1^t\mathbf{S}\mathbf{u}_1 = \lambda_1$.

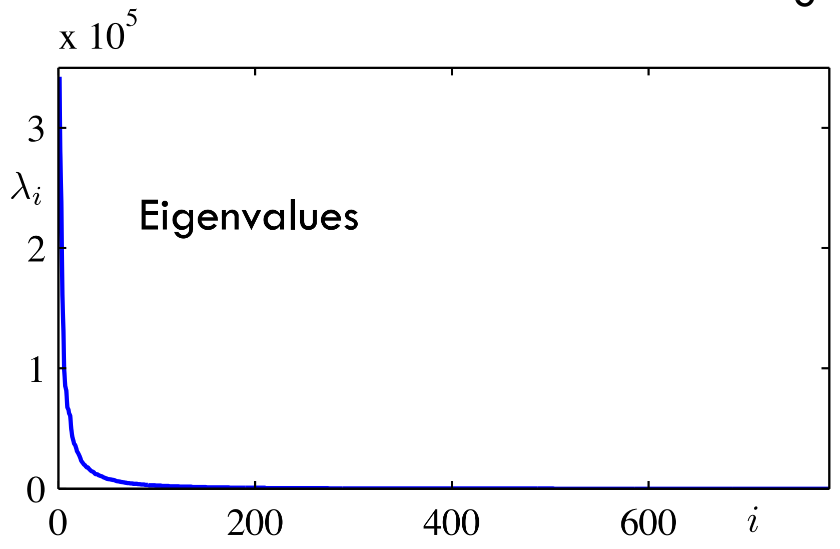
Thus to maximize projected variance, we select the eigenvector with largest associated eigenvalue λ_1.

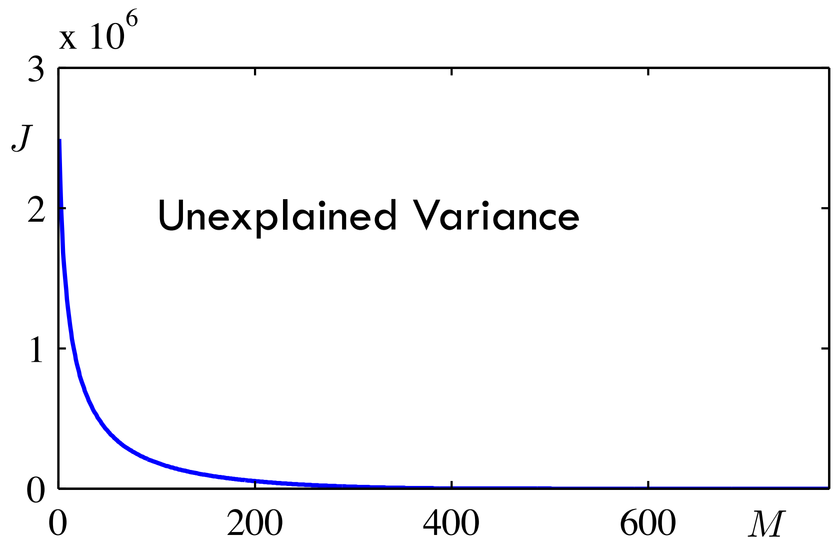


Dimensionality Reduction

- The next direction $\mathbf{u}_2$ can be chosen by maximizing the projected variance in the *D-1* dimensional subspace orthogonal to $\mathbf{u}_1$.
- Thus $\mathbf{u}_2$ is the eigenvector of S with the second-largest eigenvalue, and so on…
- Typically, most of the variance is captured in a relatively small linear subspace.

Digit '3' Dataset





Computational Cost

- Computing full eigenvector decomposition is $O(D^3)$.
- If we only need the first M eigenvectors, the cost is $O(MD^2)$.
- However, this could still be very expensive if D is large

  $e.g.,$ For an 1800×1600 image and $M = 100$, $O(650 \text{ million})$
- For classification or regression, this is precisely the situation where we need PCA, to reduce the number of parameters in our model and therefore prevent overlearning!

Computational Cost

- In many cases, the number of training vectors *N* is much smaller than *D*, and this leads to a trick:

Let $\mathbf{X}$ be the $N \times D$ centred data matrix whose nth row is given by $\left(\mathbf{x_n} - \bar{\mathbf{x}}\right)^t$.

Then the sample covariance matrix is $\mathbf{S} = \frac{1}{N}\mathbf{X^tX}$.

and the eigenvector equation is $\frac{1}{N}\overbrace{\mathbf{X^tX}}^{D \times D}\mathbf{u_i} = \lambda_i \mathbf{u_i}$

Pre-multiplying both sides by $\mathbf{X}$ yields $\frac{1}{N}\mathbf{XX^t}\left(\mathbf{Xu_i}\right) = \lambda_i\left(\mathbf{Xu_i}\right)$

Now letting $\mathbf{v_i} = \mathbf{Xu_i}$, we have

$\frac{1}{N}\overbrace{\mathbf{XX^t}}^{N \times N}\mathbf{v_i} = \lambda_i \mathbf{v_i}$ $\longleftarrow$ **Much smaller eigenvector problem!**

Computational Cost

- To find the eigenvectors of **S**, we premultiply by **X**^{t}:

$$\frac{1}{N}\underbrace{\mathbf{X}\mathbf{X}^t}_{N\times N}\mathbf{v}_i = \lambda_i \mathbf{v}_i \rightarrow \underbrace{\left(\frac{1}{N}\mathbf{X}^t\mathbf{X}\right)}_{S}\left(\mathbf{X}^t\mathbf{v}_i\right) = \lambda_i\left(\mathbf{X}^t\mathbf{v}_i\right)$$

and, normalized to unit length, the eigenvectors are $\mathbf{u}_i = \frac{1}{\sqrt{N\lambda_i}}\mathbf{X}^t\mathbf{v}_i$

Note that these *N* eigenvectors live in the *N*-dimensional subspace spanned by the training images.

Other Applications of PCA

Other Applications of PCA

- We have motivated PCA as a method for reducing the dimensionality of the input space and therefore the number of parameters that must be learned for classification or regression.
- This will help to reduce overlearning.
- But there are other applications of PCA…

Standardization

- Input vectors are often heterogeneous in that values might vary widely on some dimensions relative to others.
- This is particularly true when input vectors are composed of different kinds of measurements, perhaps measured in different units.
- Example:
  - We may try to classify a patient in a hospital setting based upon:
    - Age (years)
    - Resting pulse (beats per minute)
    - Body temperature (degrees Celsius)
    - …
- For pattern recognition algorithms to work well, it is often important that the data be standardized along these different dimensions.

Pre-Whitening

Let U be the $D \times D$ matrix whose columns are the D orthonormal eigenvectors $\mathbf{u}_i$ of S.

Let Λ be the $D \times D$ diagonal matrix whose diagonal elements Λ_{ii} are the associated eigenvalues λ_i.

- Then the transformation

  $$\mathbf{y}_n = \Lambda^{-1/2} U^t \left(\mathbf{x}_n - \overline{\mathbf{x}} \right)$$

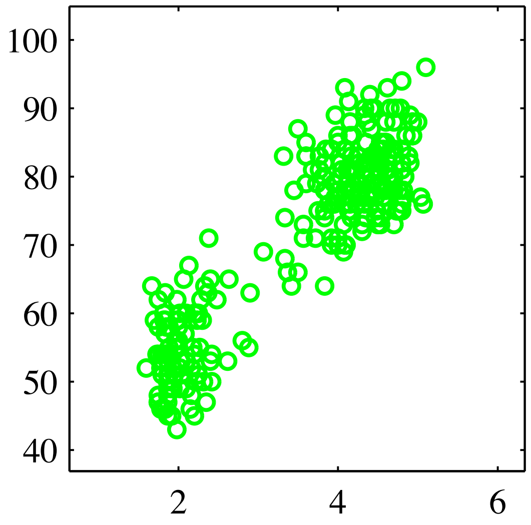
  does three things:

1. Shifts the data to the origin, so that the transformed data have zero mean.
2. Rotates the data into the principal axes, **decorrelating the data** (diagonal covariance)
3. Scales the data by the inverse standard deviation along each principal axis, thus normalizing the variance in all directions (covariance = identity matrix).

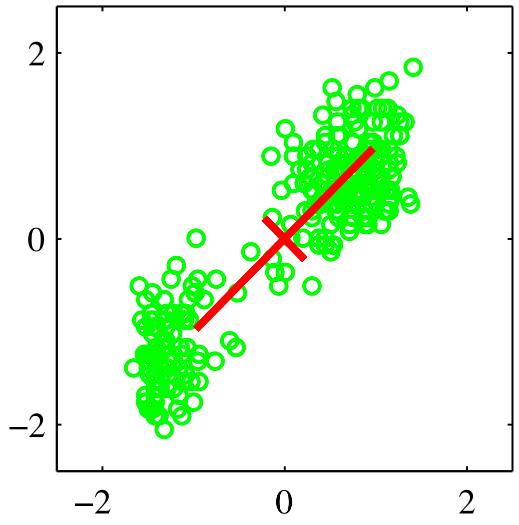
Pre-Whitening

$$y_i = \frac{x_i - \bar{x}_i}{\sigma_i}$$

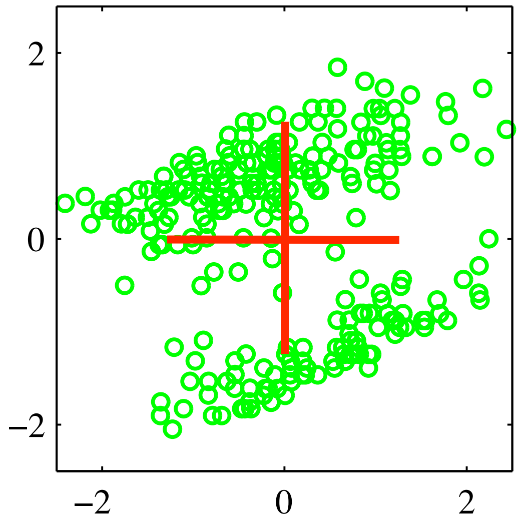
$$\mathbf{y}_n = \Lambda^{-1/2} U^t \left(\mathbf{x}_n - \bar{\mathbf{x}} \right)$$



Original Data



Normalized to 0-mean and unit variances (z-scores)



Whitened (unit covariance)

Compression

- PCA can be used to compress data. Since the *D* eigenvectors $\mathbf{u}_i$ of the covariance matrix form a complete orthonormal basis, any input $\mathbf{x}_n$ can be described by a linear combination of these eigenvectors:

$$\mathbf{x}_n = \sum_{i=1}^{D} \alpha_{ni} \mathbf{u}_i$$

- Taking the inner product with $\mathbf{u}_j$, we obtain $\alpha_{ni} = \mathbf{x}_n^t \mathbf{u}_j$, and so

$$\mathbf{x}_n = \sum_{i=1}^{D} \left(\mathbf{x}_n^t \mathbf{u}_i \right) \mathbf{u}_i .$$

- In other words, the input vector is simply the sum of the linear projections onto the eigenvectors. Note that describing $\mathbf{x}_n$ in this way still requires *D* numbers (the α_{ni}): no compression yet!

Compression

- But suppose we only use the first M eigenvectors to code **x**$_n$. We could then reconstruct an approximation to **x**$_n$ as:

$$\mathbf{x}_n \simeq \sum_{i=1}^{M} z_{ni}\mathbf{u}_i + \sum_{i=M+1}^{D} b_i\mathbf{u}_i$$

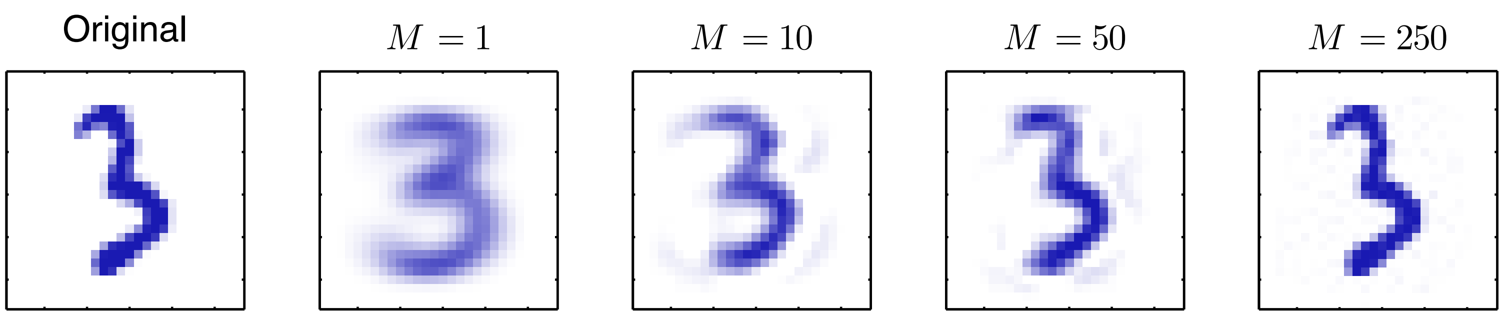
- Now to describe **x**$_n$, we only have to transmit M numbers (the z$_{ni}$). (Note that the b$_i$ are common to all inputs – not a function of **x**$_n$.)
- It can be shown that for minimum expected squared error, the optimal basis U is indeed the eigenvector basis, and the optimal coefficients are:

$$z_{nj} = \mathbf{x}_n^t\mathbf{u}_j \qquad b_j = \bar{\mathbf{x}}^t\mathbf{u}_j$$

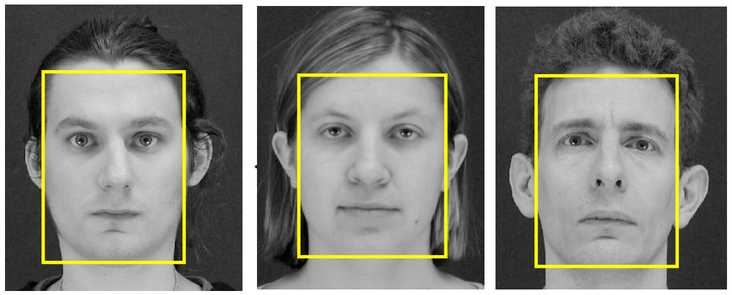
- In other words, we approximate **x**$_n$ as the sum of:
  - The projections of the input vector on the *M* eigenvectors with largest associated eigenvalues, and
  - The projections of the mean vector on the remaining *D* - *M* eigenvectors.

Example

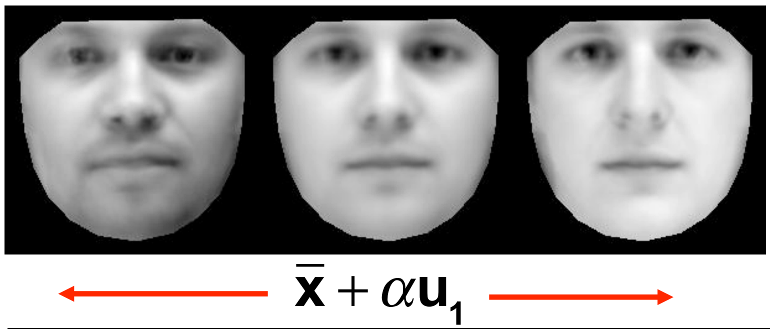
- Suppose we need to transmit images for the handwritten digit '3'.
- Each image is 28 x 28 pixels: each transmission costs 784 bytes.
- Instead: the transmitter and receiver both store the D principal components (eigenvectors).
- The transmitter then sends only the *M* scalar projections onto the first *M* of these eigenvectors.
- If each projection is stored as an 8-byte floating point number, the cost of each transmission is 8 x M bytes.

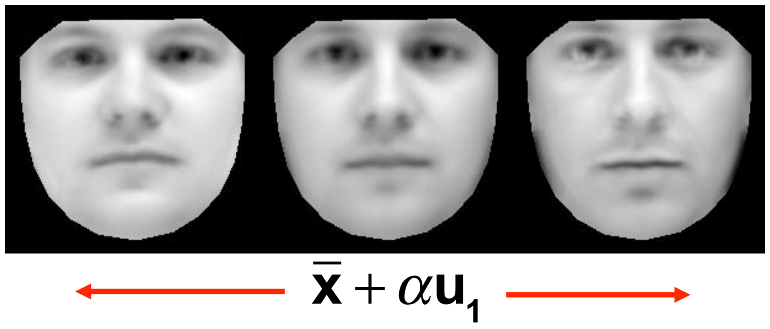


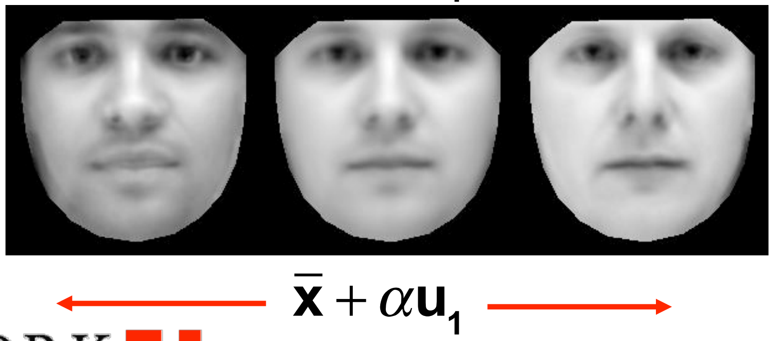
Modeling

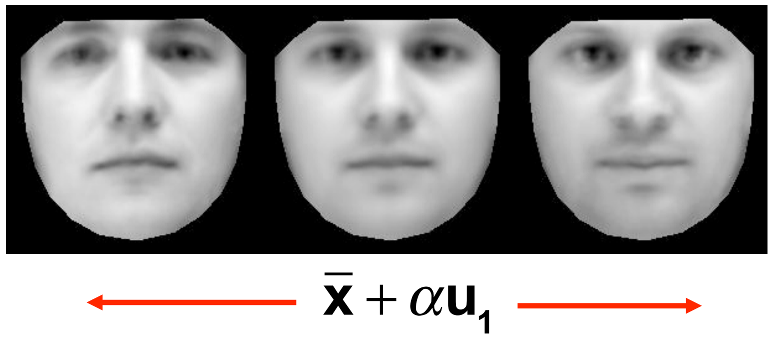


Low-dimensional model of variation of registered objects such as faces


$\bar{\mathbf{x}} + \alpha \mathbf{u}_1$


$\bar{\mathbf{x}} + \alpha \mathbf{u}_1$


$\bar{\mathbf{x}} + \alpha \mathbf{u}_1$


$\bar{\mathbf{x}} + \alpha \mathbf{u}_1$