Agenda

• Project 4 - Graded and solns in Dropbox
• Project 5 - *NEWS* will be optional
• PCA - eigenfaces cont’d
• Next Thursday, review for final
• 5 projects - worth 15% each
• Will take best 4 scores
• If you’re happy with 4 existing scores, don’t need to do Proj 5 (though I recommend it!)
• Final exam - worth 25%
Proj 5: Face recognition
The space of faces

An image is a point in a high dimensional space

\[ x_i \in \mathbb{R}^d \]

If image is 500x500, \( d = 250000 \)
Why think of images as points?

We know have a simple face recognition algorithm!

Person A  
Person B  
Training images (known labels)

Test image (unknown label)

Given test point, output label of closest training point

This “nearest neighbor” classifier is surprisingly powerful!
1-NN classifier

\[
\{x_i, y_i\}
\]

\[
x, y = ?
\]

\[
Dist(x, x_i) = \sum_{j=1}^{d} (x_i^j - x^j)^2
\]

\[
= (x_i - x)^T (x_i - x)
\]

\[
i^* = \arg \min_i Dist(x, x_i)
\]

\[
y = y_{i^*}
\]
K-NN classifier

1) Find K training points closest to $x$
2) $y =$ most-common label

Meant to alleviate “noisy” training points
The space of faces

An image is a point in a high dimensional space

\[ x_i \in \mathbb{R}^d \]

We know “d” can easily reach a million
Can we store a face library on a cell phone?
Dimensionality reduction

The set of faces is a “subspace” of the set of images

\[ x_i \in \mathbb{R}^d \]

\[ x_i \approx \mu + \alpha_1 u_1 + \alpha_2 u_2 + \ldots + \alpha_k u_k \]

\[ c_i = [\alpha_1, \alpha_2, \ldots, \alpha_k] \]

\[ k \ll d \]

Common values for \( k \) are 50 -100
Do NN-classification storing \( c_i \) rather than \( x_i \)
What do mean ($\mu$) and basis vectors ($u_i$) look like?

\[ x_i \approx \mu + \alpha_1 u_1 + \alpha_2 u_2 + \ldots + \alpha_k u_k \]
Projection and Reconstruction

\[ x_i \approx \mu + \alpha_1 u_1 + \alpha_2 u_2 + \ldots \alpha_k u_k \]

Given a collection of training data \( \{x_i\} \), how do we find the best \( u_1, u_2, \ldots, u_k \)?

We’ll use an algorithm called principle component analysis (PCA)
Assume we have a collection of $n$ high-dimensional points $x_i \in \mathbb{R}^d$. 

Figure 1: A collection of points in $\mathbb{R}^d$ for $d = 3$ that live in a $k = 2$ dimensional subspace
Objective function

Centering: First, let us center these points

\[ x'_i = x_i - \mu \]

\[ \mu = \frac{1}{n} \sum_{i=1}^{n} x_i \]

Maximize variance: Let us find direction \( u \) such that when \( x'_i \) is projected into \( u \), the variance of the data is maximized. Let us define \( u \) to be of length 1. We can then write the projection as \( u^T x'_i \).

\[
\max_u \text{var}(u^T x'_i). \quad ||u|| = 1
\] (3)
Why maximize variance?

We see that the projected data still has a fairly large variance, and the points tend to be far from zero. In order to formalize this, note that given a projection $v$, select the direction $u$ corresponding to the first of the two figures shown above.
Math...

$$\max_u \text{var}(u^T x'_i). \quad ||u|| = 1$$

Interpreting the projected values as samples of a random variable, we can estimate its variance by computing the average squared deviation from its mean:

$$\hat{\text{mean}} \frac{1}{n} \sum_i u^T x'_i = \bar{u}^T (\frac{1}{n} \sum_i x'_i) = u^T 0 = 0$$ is zero

$$\max_u \frac{1}{n} \sum_i (u^T x'_i - 0)^2, \quad ||u|| = 1$$
Math...

\[
\max_u \frac{1}{n} \sum_i (u^T x_i' - 0)^2, \quad ||u|| = 1
\]

\[
\max_u \frac{1}{n} \sum_i (u^T x_i')(u^T x_i')^T, \quad ||u|| = 1
\]

\[
\max_u \frac{1}{n} \sum_i u^T x_i' x_i'^T u, \quad ||u|| = 1
\]

\[
\max_u u^T \left( \frac{1}{n} \sum_i x_i' x_i'^T \right) u, \quad ||u|| = 1
\]

\[
\max_u u^T \Sigma u, \quad u^T u = 1, \quad \Sigma = \frac{1}{n} \sum_i x_i' x_i'^T
\]

We recognize \( \Sigma \) as the sample covariance matrix

\[\Sigma = \Sigma^T \text{ (symmetric)}\]

The \((i,j)\)th entry denotes the covariance between the \((i,j)\)th coordinate of \(x\)
To solve (8), we use lagrange multipliers. We know that the gradient of the objective function (with respect to \( u \)) and the constraint are in the same direction. We can equivalently state they are equal up to a scale factor

\[ \nabla_u u^T \Sigma u = \lambda \nabla_u u^T u \]  

(9)

Using rules of matrix derivatives

\[ 2\Sigma u = 2\lambda u \]  

(10)

\[ \Sigma u = \lambda u \]  

(11)
Done!

\[
\max_u u^T \Sigma u, \quad u^T u = 1, \quad \Sigma = \frac{1}{n} \sum_i x'_i x'^T_i \\
\Sigma u = \lambda u
\]  \hspace{1cm} (8)  \hspace{1cm} (11)

We recognize (11) as an eigenvalue equation. We call the direction \( u \) a *principle component* of the data \( x_i \). Note that an alternate derivation of \( u \) is the direction that minimizes the reconstruction error between a point and its projection.

The optimal direction “\( u \)” is the eigenvector of \( \Sigma \) associated with the largest eigenvalue.
How to find k directions?

“Look for another direction that’s orthogonal to the first one, and which still preserves lots of variance in the projection”

$$\max_u u^T \Sigma u, \quad u^T u = 1, \quad \Sigma = \frac{1}{n} \sum_i x_i' x_i'^T$$  \hspace{1cm} (8)

In order to project to two directions, one can first find the optimal single direction from (8) and call this direction $u_1$. To find $u_2$, one can optimize

$$\max_{u_2} u_2^T \Sigma u_2, \quad u_2^T v_2 = 1, u_2^T u_1 = 0$$  \hspace{1cm} (12)

It turns out that $u_2$ is the eigenvector of $\Sigma$ associated with the second largest eigenvalue.
Summary

PCA: In PCA, one would like to find the vectors $U = [u_1, u_2, \ldots, u_k]$ that preserve the variance of the projected points. One does this by setting $u_1, u_2, \ldots, u_k$ to be the top k eigenvectors (associated with the $k$ largest eigenvalues) of the covariance matrix of centered points $\Sigma = \frac{1}{n} \sum x'_i x'_i^T$

Projection: Let $U_{d \times k} = [u_1, u_2, \ldots, u_k]$ be the matrix of the top $k$ eigenvectors. Let $X_{d \times n} = [x'_1, \ldots, x'_n]$ be the design matrix. Note that $\Sigma = \frac{1}{n} X X^T$. Let $c_i$ be the $k$ coefficients specifying the coordinates of $x_i$ when projected onto $U$. We can write $c_i = U^T x'_i$.

Reconstruction: We can reconstruct the point $x_i$ by computing the linear combination $\hat{x}_i = \mu + U c_i$. 

\[\]
Geometric intuition

Let $A_{d \times d}$ be a square, symmetric matrix ($A = A^T$). Let $u_i$ be an eigenvector $A$ with eigenvalue $\lambda_i$ such that $Au_i = \lambda_i u_i$. $A$ can be decomposed into $A = U \Lambda U^T$ where $U = [u_1, u_2, \ldots, u_k]$ is an orthonormal matrix $U^T U = I$ and $\Lambda$ is a diagonal matrix with entries $\Lambda_{i,i} = \lambda_i$. 
Geometric intuition

We can interpret the decomposition within the context of the linear transformation $Az = UΛU^T z$ for some vector $z$. First, note that the eigenvectors $u_i$ form an orthonormal basis for $R^d$.

1. $c = U^T z$ are the coordinates of the vector $z$ when projected onto the orthonormal basis $U$.
2. $\tilde{c} = Λc$ scales each coordinate $c_i$ by $λ_i$.
3. $U\tilde{c}$ is the “reconstruction” of $z$ with the linear combination $\sum \tilde{c}_i u_i$.

Figure 3: We can interpret the linear transformation $Az$ as a sequence of projection, scaling, and reconstruction steps.
If $d$ is large, it maybe difficult to compute and store $\Sigma_{d \times d}$. We show how one can use the SVD of $X$ to compute $U$ without computing $\Sigma$. Let us compute the SVD of $X = U'\Sigma'V'T$. We use the accents to differentiate these matrices from $\Sigma$ and $U$ computed earlier.

Recall the following fact:

Any (possibly non-square matrix) $A_{m \times n}$ can be factored $A = U\Sigma V^T$, where $U_{m \times m}$ an orthornomal matrix ($UU^T = I$) and $V_{n \times n}$ is also an orthonormal matrix ($VV^T = I$). $\Sigma_{m \times n}$ is a diagonal matrix with entries $\sigma_i \geq 0$. These $\sigma_i$ are called singular values of $A$. 

We use this result to compute $\Sigma$.

PCA-via-the-SVD
PCA-via-the-SVD

\[ X_{d \times n} = [x'_1, \ldots x'_n] \] be the design matrix.

We know that the sample covariance matrix \( \Sigma \) can be written as

\[
\Sigma = \frac{1}{n} \sum_{i} x'_i x'_i T
\]

\[ = \frac{1}{n} X X^T \] (14)

\[ = \frac{1}{n} (U' \Sigma' V' T) (U' \Sigma' V' T)^T \] (15)

\[ = \frac{1}{n} (U' \Sigma' (V' T V') \Sigma' U' T) \] (16)

\[ = \frac{1}{n} (U' \Sigma^2 U' T) \] (17)
PCA-via-the-SVD: Because we know that $\Sigma = U \Lambda U^T$, we conclude that $\Lambda = \frac{1}{n} \Sigma'^2$ and $U = U'$. From Sec. 1, we know that we can the optimal projection directions $u_1, u_2, \ldots, u_k$ are equal to the top $k$ eigenvectors of $\Sigma$. To avoid computing $\Sigma$, which often will not fit in memory, we can set the directions to be $u'_1, u'_2, \ldots, u'_k$, the top $k$ left singular vectors of the design matrix $X$, where $X = U' \Sigma' V'^T$.

Will need this for Proj5!
How to pick k?

Recall objective function = variance of projected data

\[
\max_u u^T \Sigma u, \quad u^T u = 1, \quad \Sigma = \frac{1}{n} \sum_i x'_i x'_i^T
\]  (8)

We know solution will be an eigenvector \( u_i \). The value of the objective function will be

\[
u^T_i \Sigma u_i = u^T_i (\lambda_i u_i) = \lambda_i (u^T_i u_i) = \lambda_i
\]

Hence eigenvalues specify the variance of data along the direction \( u_i \).
How to pick $k$?

Eigenvalues specify the variance of data along the eigenvector directions.

If data is truly low-dimensional, we’ll see a big drop at some $k$.
Algorithm for eigenfaces

1) Given training images, process them all with PCA. Store k coefficients for each training image

2) Given a new test image, calculate its k coefficients

3) Find closest training image in k-dimensional space
Face detection versus recognition

Person A

Person B
Can we use PCA technique for detection?
Can we use PCA technique for detection?

Non-faces should have high reconstruction error

$$\|x - \hat{x}\|^2$$

$$\hat{x} = \mu + Uc$$

$$c = U^T x$$
Window scanning

Find regions with low reconstruction error
Call them faces
Applications of PCA

\[ x_i \approx \mu + \alpha_1 u_1 + \alpha_2 u_2 + \ldots + \alpha_k u_k \]

Idea of expressing a high-dimensional vector as a linear combination of a few basis vectors very powerful
Morphable Face Model

Use subspace to model elastic 2D or 3D *shape* variation (vertex positions), in addition to *appearance* variation
Morphable Face Model

\[ S_{model} = \sum_{i=1}^{m} \alpha_i S_i \quad T_{model} = \sum_{i=1}^{m} b_i T_i \]

\[ s = \alpha_1 \cdot + \alpha_2 \cdot + \alpha_3 \cdot + \alpha_4 \cdot + \ldots = S \cdot a \]

\[ t = \beta_1 \cdot + \beta_2 \cdot + \beta_3 \cdot + \beta_4 \cdot + \ldots = T \cdot b \]

3D models from Blanz and Vetter ’99

http://www.youtube.com/watch?v=nice6NYb_WA