Example: SVD of an image

```
>> F = imread('cameraman.tif');
>> F = double(F) / 255; %normalizes values into [0,1]
>> imshow(F);
>> [U, S, V] = svd(F);
>> imshow(U(:,1)*S(1,1)*V(:,1)')
>> k = 10; imshow(U(:,1:k)*S(1:k,1:k)*V(:,1:k)')
```

Rank-1: very block-like: a small entry in $\mathbf{u}_1$ ($\mathbf{v}_1$) makes the whole row (column) small.

Higher ranks still give a 'blocky' behaviour. Still fewer data than the full image, though.

$\|A - U_k S_k V_k^T\|_F =$ sum of squares of neglected singular values.

Examine `diag(S)` — zeros at the end are due to duplicated rows/columns (likely at the top).

Another example: test scores

Consider a matrix of test scores:

	Ex. 1	Ex. 2	Ex. 3	Ex. 4
Student A	4	5	3	5
Student B	2	5	2	3
Student C	3	4	2	5
Student D	4	4	1	5
Student E	1	5	3	3

Suppose A has rank 1, $A = \mathbf{u}\mathbf{v}^T$. How can we interpret the values?

$$A_{ij} = \underbrace{u_i}_{\text{ability of student } i} \quad \underbrace{v_j}_{\text{difficulty of exercise } j}$$

What would a rank-2 table represent instead? Suppose for instance all entries ≥ 0; they can represent 'skill' in two types of exercises, e.g., theory and programming.

$$A_{ij} = \underbrace{(u_1)_i}_{\text{type-1 ability}} \underbrace{(v_1)_j}_{\text{type-1 difficulty}} + \underbrace{(u_2)_i}_{\text{type-2 ability}} \underbrace{(v_2)_j}_{\text{type-2 difficulty}} .$$

SVD approximation

Truncated SVD provides the best rank-1 approximation of given scores:

$$A \approx A_1 = \mathbf{u}_1 \sigma_1 \mathbf{v}_1^T$$

(σ_1, scalar, is just a scaling factor; it could be incorporated in $\mathbf{u}_1$ or $\mathbf{v}_1$.)

A_1 is the approximation with smallest $\|A - A_1\|_F^2 = \sum (A_{ij} - (A_1)_{ij})$.

This has a statistical interpretation: suppose there is a random error in our scores

$$A_{ij} = \underbrace{u_i}_{\text{student ability}} \underbrace{v_j}_{\text{exercise difficulty}} + \underbrace{\epsilon_{ij}}_{\textit{error}}$$

Then, the choices of $\mathbf{u}_1, \mathbf{v}_1$ provided by SVD are the maximum likelihood estimation, if ϵ_{ij} are Gaussian errors with the same variance.

Higher rank approximations with SVD

Approximations of rank larger than 1 could be interpreted as different classes of exercises: for instance,

$$A_{ij} = \underbrace{u_{i1}}_{\substack{\text{mathematical}\\ \text{ability}}} + \underbrace{v_{j1}}_{\substack{\text{mathematical}\\ \text{difficulty}}} + \underbrace{u_{i2}}_{\substack{\text{programming}\\ \text{ability}}} \underbrace{v_{j2}}_{\substack{\text{programming}\\ \text{difficulty}}} + \underbrace{\epsilon_{ij}}_{\textit{error}}$$

However, there is an additional catch: $\mathbf{u}_1, \mathbf{v}_1$ already mean general ability, and the four vectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{v}_1, \mathbf{v}_2$ provided by SVD cannot all have positive entries (by orthogonality).

If all $A_{ij} \geq 0$, then one can prove that $(\mathbf{u}_1)_i \geq 0$, $(\mathbf{v}_1)_j \geq 0$, which matches their interpretation as general ability / difficulty.
(Or both are negative, which is equivalent because $\mathbf{u}_1\mathbf{v}_1^T = (-\mathbf{u}_1)(-\mathbf{v}_1)^T$)

The next terms $\mathbf{u}_2, \mathbf{v}_2$ have mixed signs, and must be interpreted as corrections.

(and so do all the following ones $\mathbf{u}_3, \mathbf{v}_3, \ldots$.)

Corrections

$$A = \begin{bmatrix} 3.6 & 5.9 & 6.7 \\ 7.7 & 3.9 & 3.1 \\ 7.9 & 6.7 & 6.6 \\ 2.0 & 6.1 & 7.3 \end{bmatrix}$$

$$= \begin{bmatrix} -0.4 \\ -0.4 \\ -0.6 \\ -0.4 \end{bmatrix} 19.8 \begin{bmatrix} -0.5 & -0.5 & -0.6 \end{bmatrix} + \begin{bmatrix} -0.3 \\ 0.6 \\ 0.2 \\ -0.6 \end{bmatrix} 5.7 \begin{bmatrix} 0.8 & -0.2 & -0.5 \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} 5.1 & 5.4 & 5.7 \\ 4.6 & 4.8 & 5.0 \\ 6.6 & 7.0 & 7.3 \\ 4.9 & 5.2 & 5.4 \end{bmatrix}}_{\text{generic rank-1 scores}} + \underbrace{\begin{bmatrix} -1.5822 & 0.4588 & 0.9922 \\ 3.1354 & -0.9091 & -1.9663 \\ 1.2287 & -0.3563 & -0.7706 \\ -2.9082 & 0.8433 & 1.8238 \end{bmatrix}}_{\text{corrections due to 2 types}}$$

Corrections have mixed signs and reflect relative ability:
$(\mathbf{u}_2)_i \geq 0$: student better at type-A than first approximation would tell;
$(\mathbf{v}_2)_j \geq 0$: exercise is more type-A than first approximation would tell.

The meaning of singular values

Further terms $\mathbf{u}_3\sigma_3\mathbf{v}_3^T, \mathbf{u}_4\sigma_4\mathbf{v}_4^T, \ldots$: further exercise types with further corrections (of decreasing norm).

Errors of these approximations: sums of square of omitted singular values.

$\|A - A_1\|_F^2 = \sigma_2^2 + \sigma_3^2 + \cdots + \sigma_{\text{min}}^2$.
$\|A - A_2\|_F^2 = \sigma_3^2 + \sigma_4^2 + \cdots + \sigma_{\text{min}}^2$.
$\|A - A_3\|_F^2 = \sigma_4^2 + \sigma_5^2 + \cdots + \sigma_{\text{min}}^2$.

What does it mean if $\sigma_1 \gg \sigma_2$? That the first approximation is already good; the rest are minor corrections.
What does it mean if $\sigma_1 \approx \sigma_2 \gg \sigma_3$? That the first correction is more important (in term of how large its entries are), and the successive ones are minor.
And so on.

At some point, further components become indistinguishable from noise in your data.

Another example: text mining

1. Breakfast is the most important meal of the day.
2. I had a peanut butter sandwich for breakfast.
3. I like to eat almonds and peanuts.
4. People normally eat three meals a day.
5. My neighbor got a little dog the other day.
6. Cats and dogs are mortal enemies.
7. You mustn't feed peanuts to your dog.
8. My dog chased a cat in the garden.

Term-document matrix

	1	2	3	4	5	6	7	8
Breakfast	1	1						
Meal	1			1				
Day	1			1	1			
Peanut		1	1				1	
Sandwich		1						
Eat			1					
Almonds			1					
My					1			1
Neighbor					1			
Dog					1	1	1	1
Cat						1		1
Garden								1

Meaning of U and V

Columns of U = concepts (with positive/negative 'scores' for each word).

Columns of V = occurrences of concepts in each sentence.

Representative case: 'perfectly split topics': two subsets of sentences that contain disjoint words.

In NLP (natural language processing), this technique is known as latent semantic analysis, because it identifies 'hidden (latent)' concepts.

Remark: these techniques identify significant components, but do not tell you what these components are: it is up to the user to figure out what type 1 / type 2 exercises are, or topic 1 / topic 2 / topic 3...

Principal component analysis

The analysis of features of a dataset (a matrix A) given by the vectors $\mathbf{u}_k, \mathbf{v}_k$ of its SVD is used in various fields with different names.

One of the most widespread: Principal Component Analysis (PCA), which is usually performed on de-meaned data: replace columns $\mathbf{x}_j$ of A with

$$\hat{\mathbf{x}}_j = \mathbf{x}_j - \frac{1}{n}\left(\sum_{k=1}^{n} \mathbf{x}_k\right), \quad \hat{A} = \begin{bmatrix} \hat{\mathbf{x}}_1 & \hat{\mathbf{x}}_2 & \dots \hat{\mathbf{x}}_n \end{bmatrix}.$$

In this way, all $\mathbf{u}_i$ and $\mathbf{v}_i$ from $\mathtt{svd}(\hat{A})$ have (usually) mixed signs and are interpreted as corrections w.r.t. the mean in one direction or the other: there is no "special status" of the first component $\mathbf{u}_1, \mathbf{v}_1$ anymore, since the data already have mean 0.

PCA and eigenvalues

Some sources define PCA using instead $\mathbf{u}_i$ = eigenvectors of $\frac{1}{n}\hat{A}\hat{A}^T$ (or sometimes $\frac{1}{n-1}$). This matrix has a statistical interpretation: it is the covariance matrix of the $\mathbf{x}_j$.

In the exercises of the previous lectures, we have seen that if $\hat{A} = USV^T$ then $\hat{A}\hat{A}^T = U(SS^T)U^T$, and this is an eigendecomposition; the factor $1/n$ just scales the eigenvalues, so

```
>> [U, S, V] = svd(Ahat);

>> [U, D] = eig(1/n * Ahat*Ahat'); S = sqrt(n*D);
```

return the exact same matrices.

Warning: in floating-point arithmetic, the first method (SVD) is more accurate.

(In general, all methods that rely on matrices of scalar products like AA^T have stability problems; we will see another instance later in the course.)

What does PCA tell you?

PCA returns a basis, so you to write each data point $\hat{x}_j = x_j - \mu$ as

$$x_j - \mu = \mathbf{u}_1 \alpha_{1j} + \mathbf{u}_2 \alpha_{2j} + \cdots + \mathbf{u}_n \alpha_{nj}.$$

This basis has the property that $\mathbf{u}_1$ is the direction responsible for the largest variation in the data (in the sense of $\sum_j \alpha_{1j}^2$), $\mathbf{u}_2$ is the direction (orthogonal to $\mathbf{u}_1$) responsible for the second largest variation, and so on.

Two alternative formulas to get the matrix of α's:

$$\alpha = U^T \hat{A} = SV^T.$$

Another example: image classification

With the provided files: Yale faces dataset.

```
>> F = readyalefaces_to_tensor();
>> showyalefaces
>> size(F)
ans =
   243 320 11 15
>> imshow(F(:,:,1,1));
>> v = reshape(F(:,:,1,1), 243*320, 1)
>> A = reshape(F, 243*320, 11*15);
>> [U,S,V] = svd(M);
Error using svd
Requested 77760x77760 (45.1GB) array exceeds maximum array
become unresponsive. See array size limit or preference par
>> [U,S,V] = svd(A, 'econ'); %thin SVD
```

Subtract "mean face"

Useful preprocessing: subtract "mean face" from all faces.
$\mathbf{x}_j = j$th image $= j$th column of A.

```
>> meanface = mean(A,2);
>> imagesc(reshape(A(:,52) - meanface, 243, 320));
>> [U, S, V] = svd(A - meanface, 'econ');
>> imagesc(reshape(U(:,1), 243, 320)); % examine them
>> diag(S) % what do the zeros at the end mean?
```

(Let $\widehat{M} = M - \text{meanface} \cdot [1, 1, \ldots, 1]$)

Each image is
$\mathbf{x}_j = \text{meanface} + \mathbf{u}_1\alpha_1 + \mathbf{u}_2\alpha_2 + \cdots + \mathbf{u}_m\alpha_m = \text{meanface} + U\alpha$

Linear combination of fixed (global for the dataset) vectors $\mathbf{u}_i$ with variable (face-dependent) 'scores' (coordinates) α_i.

The vector of 'scores' (coordinates) α is given by
$\alpha = U^{-1}\hat{\mathbf{m}}_j = U^T\hat{\mathbf{x}}_j$.

```
>> interactiverec(M(:,1))
```

Variance around "mean face"

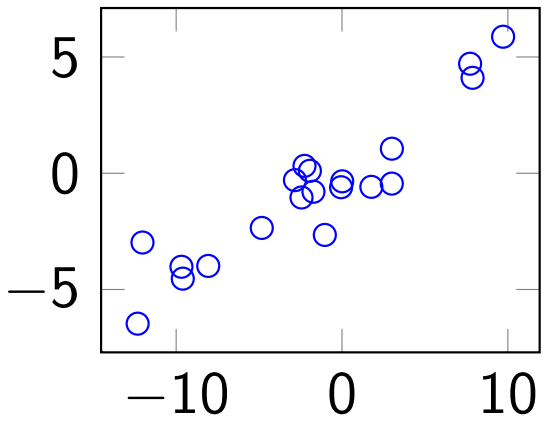
Larger singular components $\iff$ directions (in the "feature space") in which the variance is higher.

We can display the first singular components and try to give them an interpretation (not always clear) in terms of what they represent.

```
>> imagesc(reshape(U(:,1), [243, 320]))
>> imagesc(reshape(U(:,2), [243, 320]))
```

Dimensionality reduction

Let us take de-meaned data $\widehat{\mathbf{x}}_i$, and interpret them as vectors in $\mathbb{R}^n$. Suppose we are allowed only one direction through the origin to plot them. Which one gives the most useful plot?



The direction in which they vary the most. A one-dimensional plot is a rank-1 approximation of $\widehat{A}$: given a direction $\mathbf{u}_1$, all plotted vectors are multiples of $\mathbf{u}_1$.

How to retrieve this best multiplicator value (coordinate / score)? As

$$\mathbf{u}_1^T \hat{A} = \mathbf{u}_1^T (\mathbf{u}_1 \sigma_1 \mathbf{v}_1^T + \mathbf{u}_2 \sigma_2 \mathbf{v}_2^T + \dots) = \sigma_1 \mathbf{v}_1^T \quad \text{(orthogonality)}.$$

Dimensionality reduction – 2

The same idea works with multiple directions: a 2D plot approximates $\hat{\mathbf{x}}_i$ with multiples of two directions $\mathbf{u}_1$ and $\mathbf{u}_2$.

The matrix of scores / coordinates of each data vector is given by

$$\begin{bmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \end{bmatrix} \widehat{A} = \begin{bmatrix} \sigma_1 \mathbf{v}_1^T \\ \sigma_2 \mathbf{v}_2^T \end{bmatrix}.$$

The best way to represent the columns of $\hat{M}$ in a k-D space (in the sense that the error $\sum_j \|\hat{\mathbf{m}}_j - \mathbf{p}_j\|^2$ is minimum) is the projection on the plane $(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k)$.

This follows from the Eckart-Young approximation theorem.

```
>> eigenfaces_scatter(F, [1,2])
```

Exploring the dataset via this kind of plots let us figure out also which principal components work best to split the data into classes.

```
>> eigenfaces_scatter(F, [1,3,4])
```

Recognizing new images

Training set: remove an image for each individual.

```
>> T = F(:, :, 1:10, :);
>> T = reshape(T, 243*320, 10*15);
>> meanface = mean(T, 2);
>> [U, S, V] = svd(T - meanface, 0);
```

Test set: new image to be recognized.

```
>> S = F(:, :, 11, 1); %outside training test
>> S = reshape(S, 243*320, 1);
```

Look for "most similar" image along first 5 α-scores.

```
>> training_scores = U(:,1:5)' * (T-meanface);
>> test_scores = U(:,1:5)' * (S-meanface);
>> distances = sum((training_scores - test_scores).^2, 1);
>> [value, position] = min(distances)
```

Images outside of the “face space”

```
B = imread('bart.png'); B = double(B) / 255;
B = reshape(B,243*320,1);
test_scores = U(:,1:5)' * (B-meanface);
interactiverec(B)
distances = sum((training_scores - test_scores).^2, 1);
[value, position] = min(distances)
```

Try also `B = imread('car.png');`

Limitations

Many:

- Distance used: Frobenius distance among images $\Longrightarrow$ does not mean that pictures of the same individual are close.

  Image alignment, shadows, ... may have a large impact.

  Details that result in 'small' Frobenius-distance differences, e.g., facial expressions, are harder to grasp.
- The SVD does not tell you which 'features' are good / bad for person recognition, and which represent e.g. lighting.
- We never used the fact that we have several pictures of the same person: the true structure is a 3-way (or even 4-way) tensor, not a matrix M.
- In fact, SVD is a trick that is difficult to generalize: the same problem with 3 indices A_{ijk} does not have a simple solution. The same problem $\min_{\mathrm{rk}\, X \leq r} \|A - X\|$ with a different norm does not have a simple solution.

Exercises

1. Make further experiments with the dataset. For instance, try recognizing other pictures in the same way (with different training/test splits). How many of them are correctly recognized by our recognition algorithm above?
2. Which components among the first ones of the singular vector basis $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{20}$ are better suited to classify individuals? We can try to answer this question by comparing the variance of the 'scores' $\mathbf{u}_i^T \hat{\mathbf{m}}_j$ for $j \in \{\text{pictures of individual } k\}$ and for $j \in \{\text{pictures of all individuals}\}$. If these scores are uniform across an individual, but varied across the whole dataset, then i is a 'good' component to use.
3. Consider a dataset in which sentences $s_1, \ldots, s_k$ contain all terms of set T_1 exactly once, and none of the (disjoint) set T_2, and sentences $s_{k+1}, \ldots, s_n$ contain all terms of T_2 exactly once and none of T_1. What does its term-document matrix look like? What is its rank, experimentally?

References

Book references Eldén, Sections 6.4, 10.1–10.2, 11.3. (While apparently related at first sight, Chapter 14 uses a different mathematical model.) Demmel, Example 3.4.