

CS545 Machine Learning for Signal Processing

Features Part II - ICA and NMF

11 September 2023

Today's lecture

- What comes after PCA
- Independent Component Analysis
 - Going past decorrelation
- Non-Negative Matrix Factorization

PCA and decorrelation

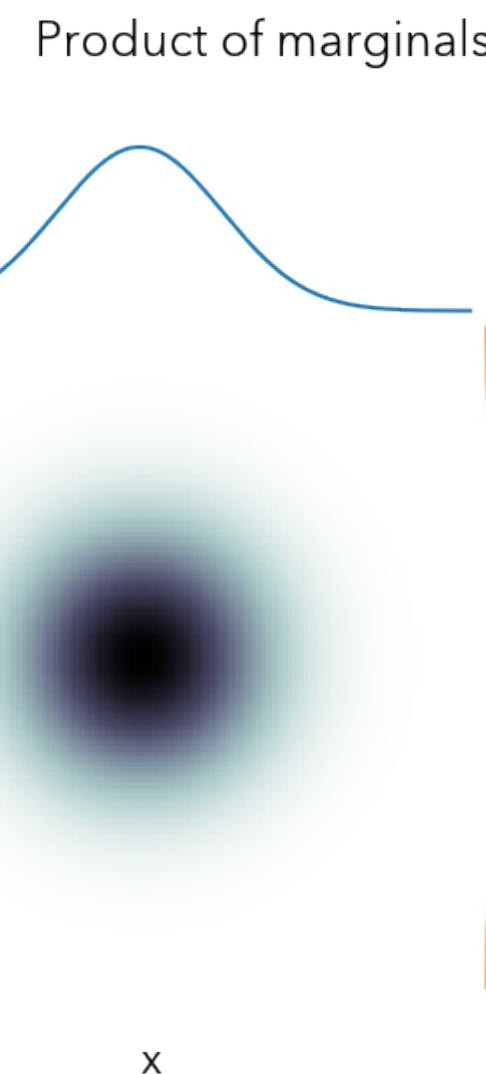
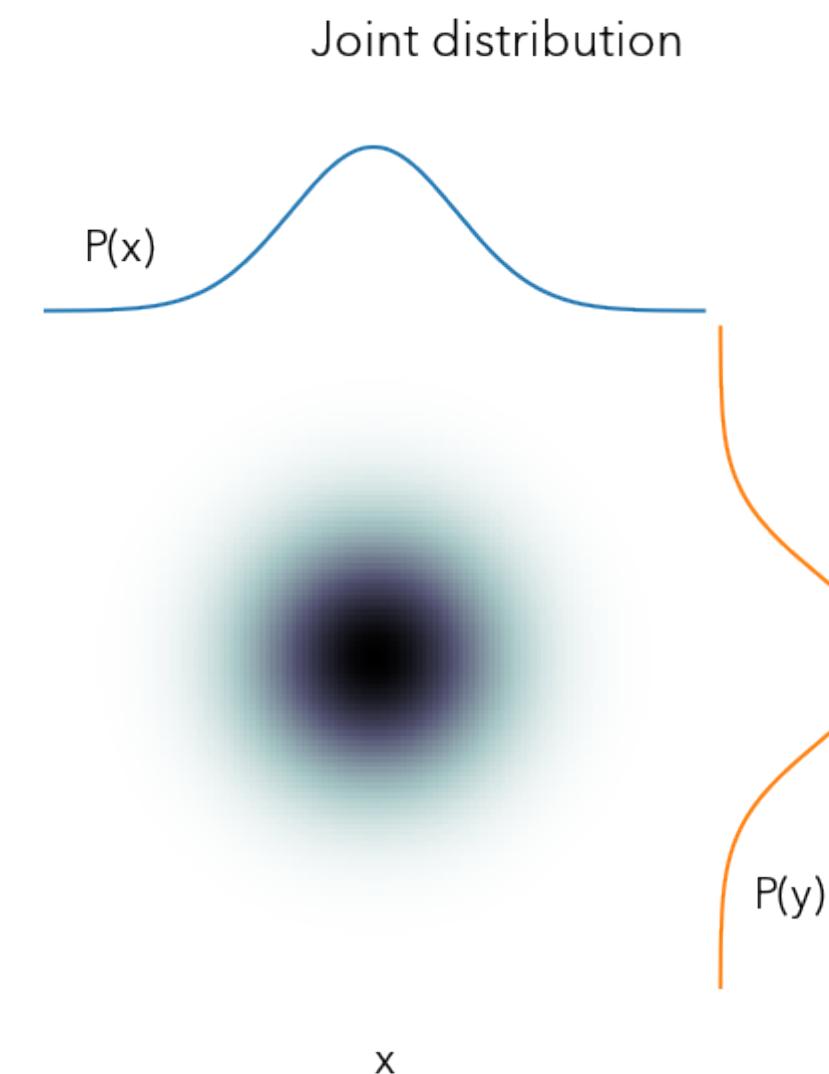
- Goal of PCA
 - Diagonalize the covariance
 - i.e. *Decorrelate* the feature activations
- Why?
 - We want to have the features activated in a *statistically independent* manner
 - So that they capture more structure

Statistical Independence

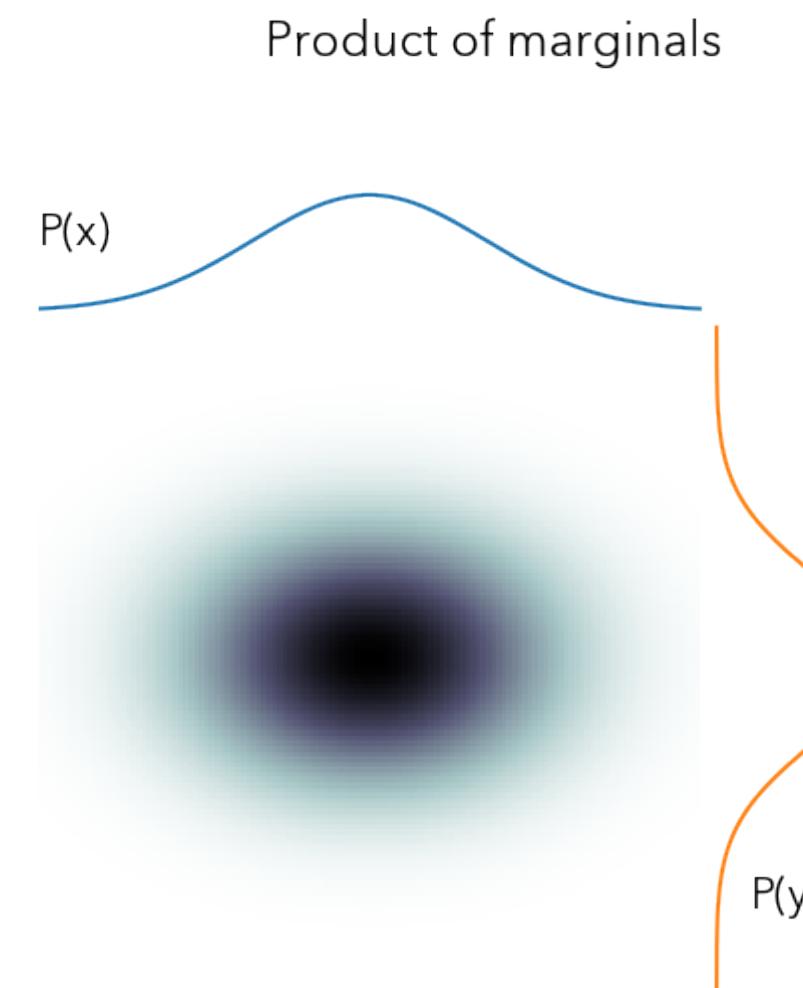
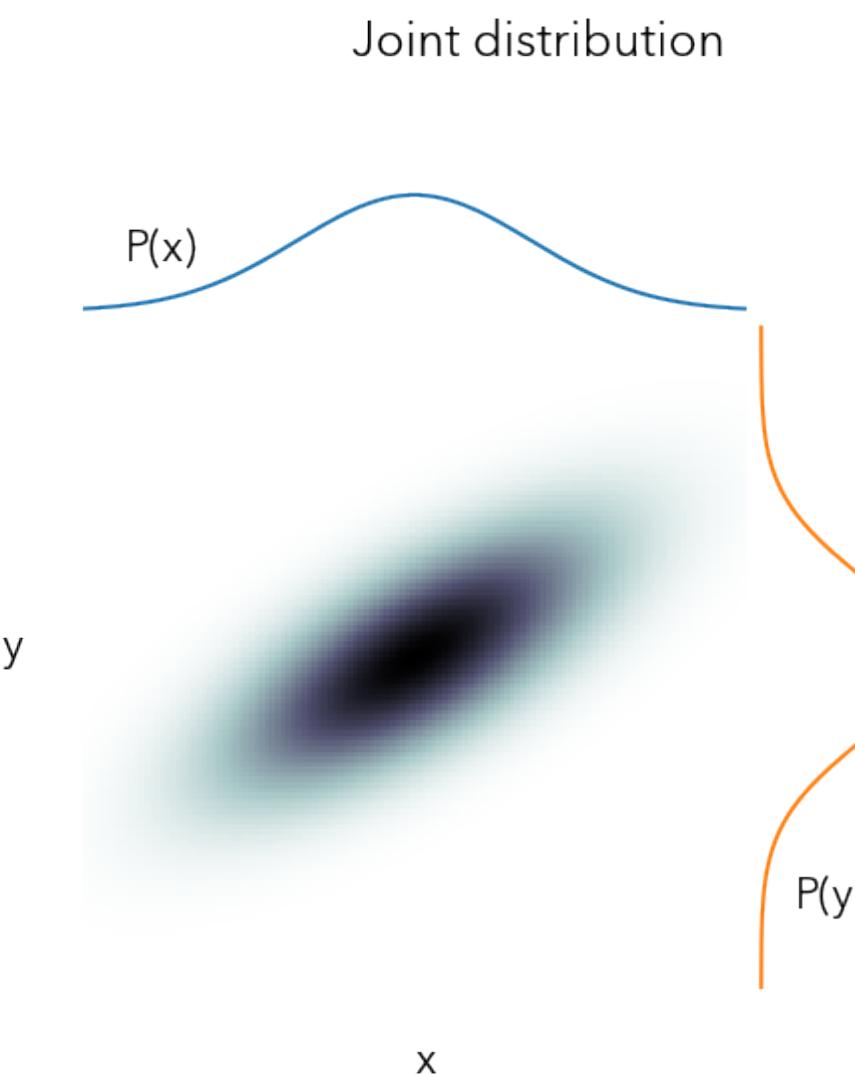
- We defined statistical independence as:

$$P(x, y) = P(x)P(y)$$

Independent variables



Dependent variables



Statistical Independence

- We defined statistical independence as:

$$P(x, y) = P(x)P(y)$$

- Which implies:

$$E\{f(x)g(y)\} = E\{f(x)\}E\{g(y)\}$$

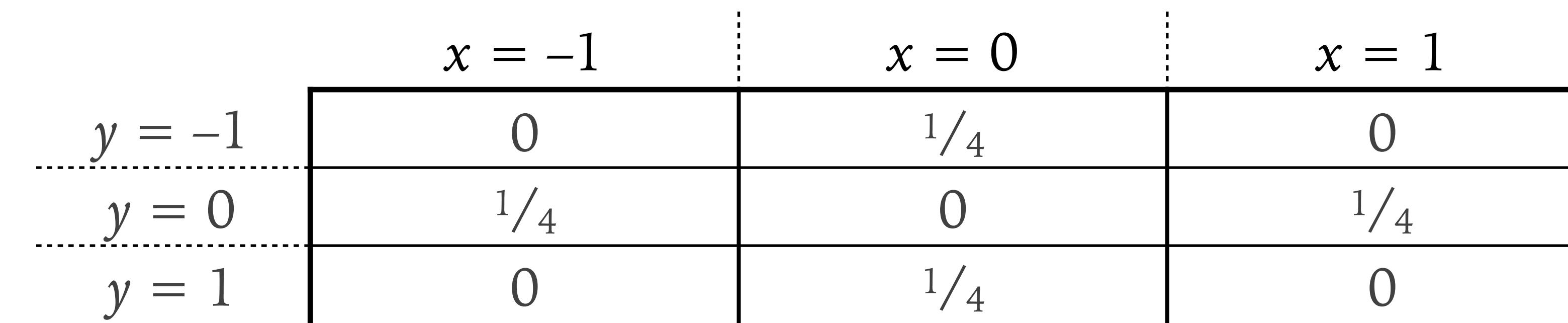
- For all measurable functions f and g
- Essentially independence means that we cannot tell anything about x if we observe y

Decorrelation and Independence

- Decorrelation does not always imply independence!
 - Decorrelation: $E\{xy\} = E\{x\}E\{y\}$
 - Independence: $E\{f(x)g(y)\} = E\{f(x)\}\{g(y)\}$
- But independence always implies decorrelation
 - When f and g are identity functions
 - Independence is a superset of decorrelation

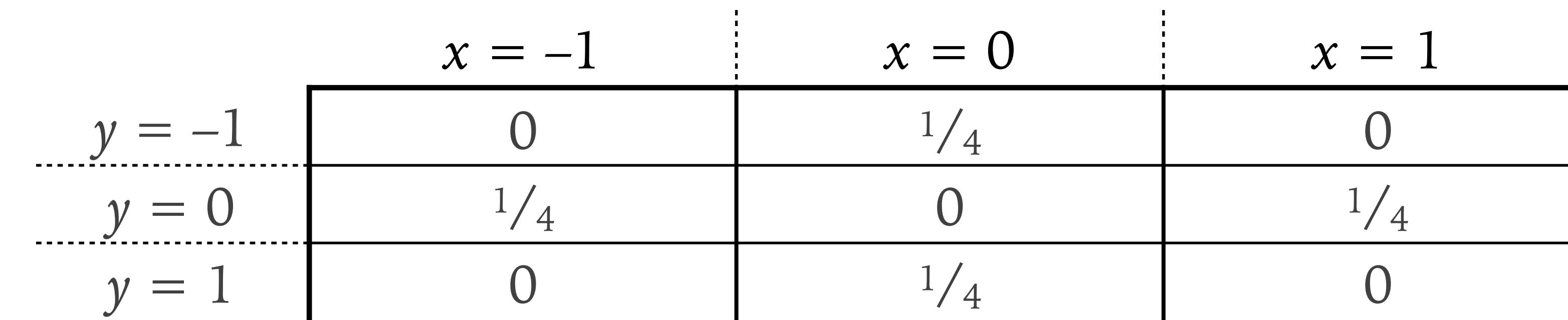
Decorrelation and Independence

- An example with discrete variables
 - Are they correlated?



Decorrelation and Independence

- An example with discrete variables
 - Are they correlated?

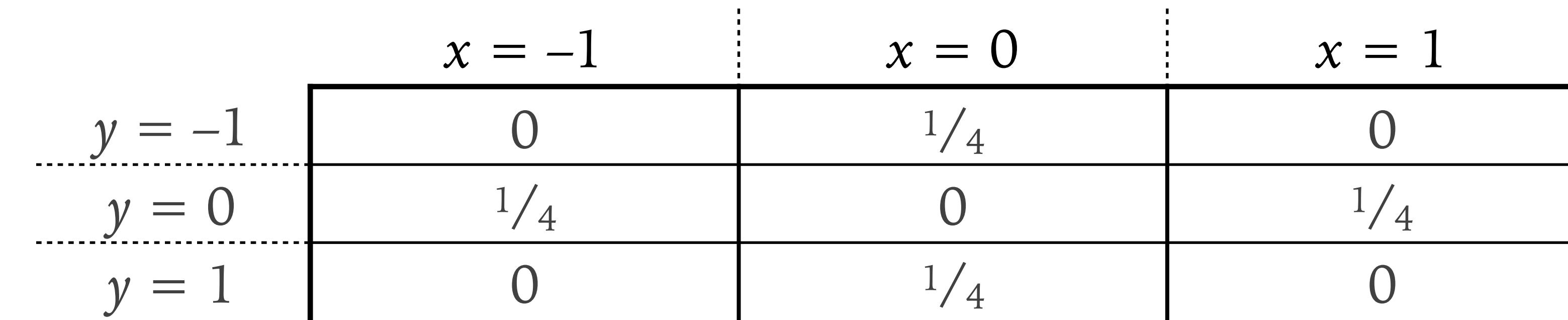


- x, y are uncorrelated

$$E\{xy\} = E\{x\}E\{y\} = 0$$

Decorrelation and Independence

- An example with discrete variables
 - Are they independent?



Decorrelation and Independence

- An example with discrete variables
 - Are they independent?

	$x = -1$	$x = 0$	$x = 1$
$y = -1$	0	$1/4$	0
$y = 0$	$1/4$	0	$1/4$
$y = 1$	0	$1/4$	0

- x, y are not statistically independent

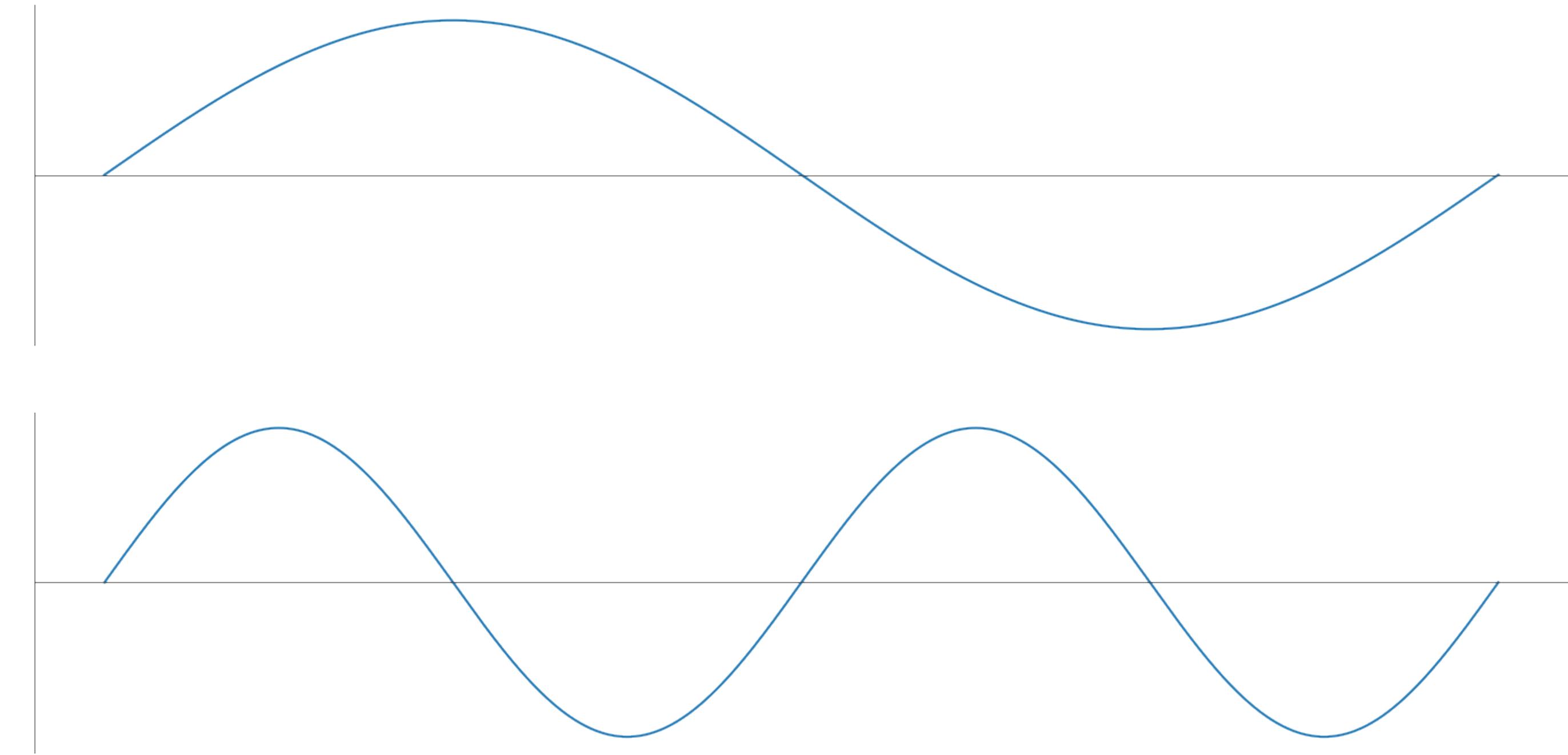
$$E\{x^2y^2\} = 0 \neq E\{x^2\}E\{y^2\} = \frac{1}{4}$$

The signals version

- Are x, y decorrelated?

$$x = \sin(t)$$

$$y = \sin(2t)$$



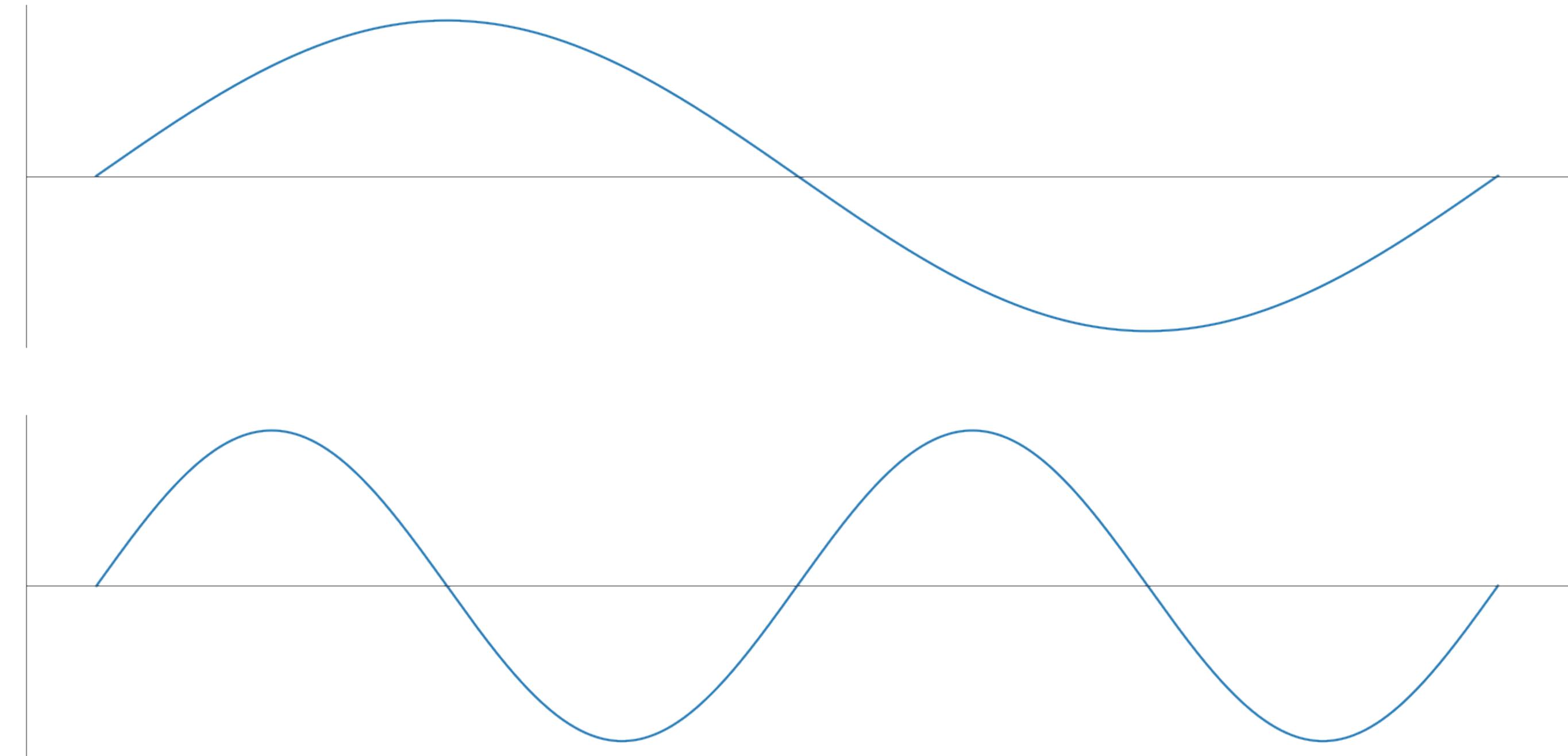
The signals version

- Are x, y decorrelated?

$$x = \sin(t)$$

$$y = \sin(2t)$$

- Yes: $E\{xy\} = 0$
 - But, I can predict one from observing the other
 - i.e., not independent!



So how do we get independence?

- Multiple ways of dealing with the problem
 - Family of algorithms known as ICA
 - *Independent Component Analysis*
- Formal definition:

$$\mathbf{y} = \mathbf{W} \cdot \mathbf{x}$$

Decomposition Analysis Features Input

$$P(y_i, y_j) = P(y_i)P(y_j), \forall i, j$$

Approach 1

- Non-linear decorrelation (assume zero mean inputs from now on)

- Achieve: $E\{f(y_i)g(y_j)\} = 0$
 - for a fixed f and g

- Cichocki-Unbehauen algorithm

- Stops updating when independence holds

do

$$\Delta \mathbf{W} \propto \left(\mathbf{D} - f(\mathbf{y}_i) \cdot g(\mathbf{y}_i^\top) \right) \cdot \mathbf{W}$$

$$\mathbf{W} = \mathbf{W} + \mu \Delta \mathbf{W}$$

repeat

$$\mathbf{D} = \begin{bmatrix} d_1 & & & 0 \\ & \ddots & & \\ 0 & & & d_n \end{bmatrix}$$

$f(x), g(x)$ can be $\tanh(x), x^3, \dots$

Approach 2

- Higher-order “diagonalization”
 - In PCA we diagonalized the covariance matrix
 - which is a $N \times N$ structure (a matrix)

$$\text{Cov}(y)_{i,j} = E\{y_i y_j\} = \kappa_2(y_i, y_j)$$

- In ICA we also diagonalize the *quadricovariance* tensor
 - which is a $N \times N \times N \times N$ structure

$$\begin{aligned} Q(y)_{i,j,k,l} &= \kappa_4(y_i, y_j, y_k, y_l) = E\{y_i y_j y_k y_l\} - \\ &E\{y_i y_j\} E\{y_k y_l\} - E\{y_i y_k\} E\{y_j y_l\} - E\{y_i y_l\} E\{y_j y_k\} \end{aligned}$$

- confused yet?

Approach 2

- Conceptually, we then perform a tensor singular value decomposition
- Comon's algorithm
 - 1) Do PCA
 - Imposes decorrelation (gets us halfway there)
 - 2) Find unitary transform (rotation) that minimizes fourth order cross-cumulants

Approach 3

- Information theoretic optimization

- Minimize mutual information: $I(\mathbf{y}) = \sum H(y_k) - H(\mathbf{y})$

- Which implies minimizing:
$$D(\mathbf{y}) = -\int P(\mathbf{y}) \log \frac{P(\mathbf{y})}{\prod P(y_k)}$$

- Iterative rule: $\Delta \mathbf{W} \propto (\mathbf{I} - f(\mathbf{y}) \cdot \mathbf{y}^\top) \cdot \mathbf{W}$
 - Looks familiar?

Approaches 4, 5, ...

- Maximum likelihood
- FastICA
 - A fast fixed-point algorithm
- Neural nets
 - Directly optimize KL divergence/Mutual information
- Negentropy
 - A measure of non-gaussianity
- ...

What approach works best?*

- Algebraic algorithms
 - HSVD, cumulant tensors, etc.
 - Consistent but computationally demanding
- Iterative algorithms (neural net-ish)
 - Non-linear decorrelation, infomax, etc.
 - Small, fast, but prone to numerical instabilities
- FastICA
 - Fixed-point algorithm
 - Quite robust and reliable

* Never ask that question, there is never a right answer

So what does ICA do?

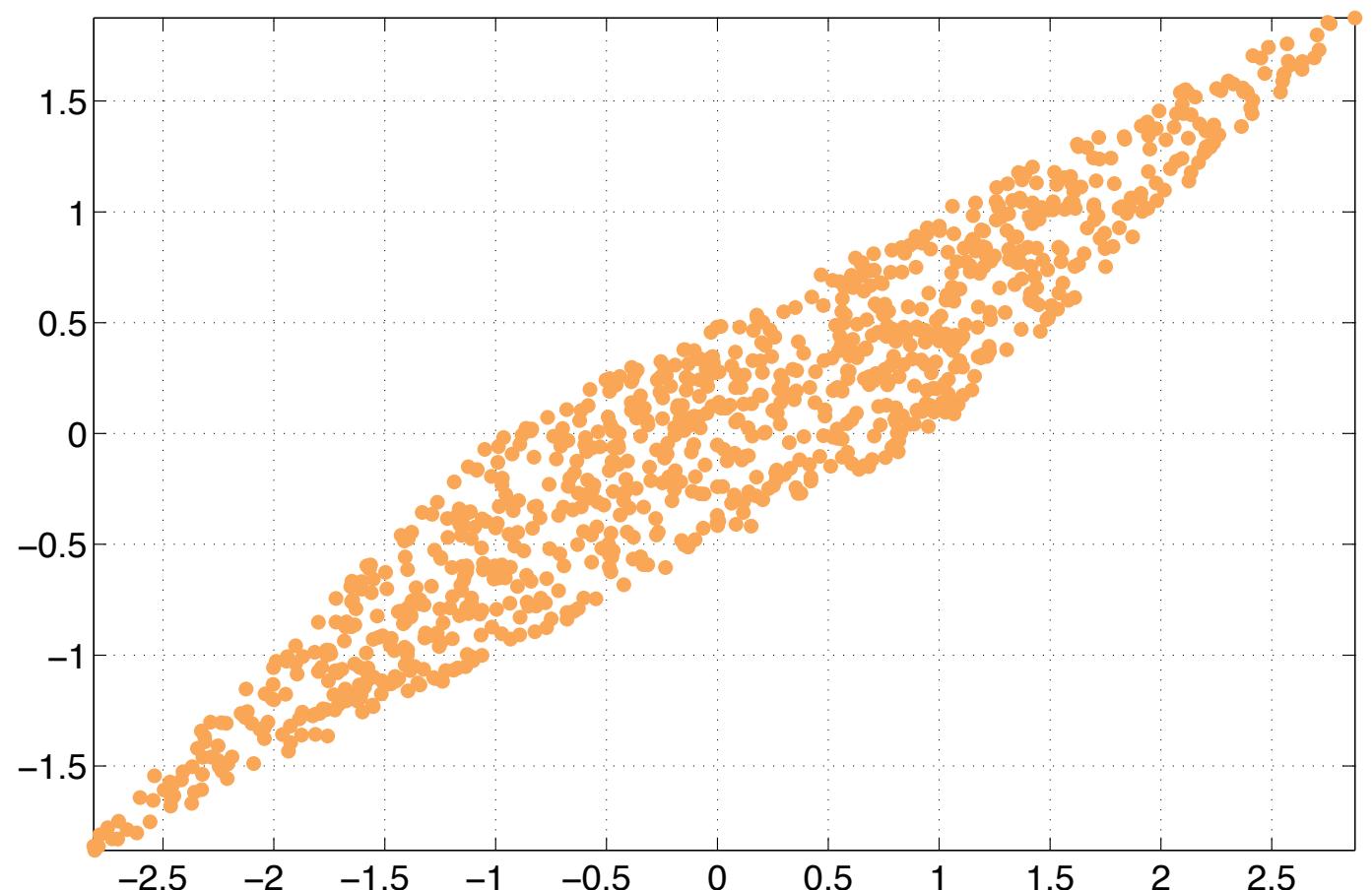
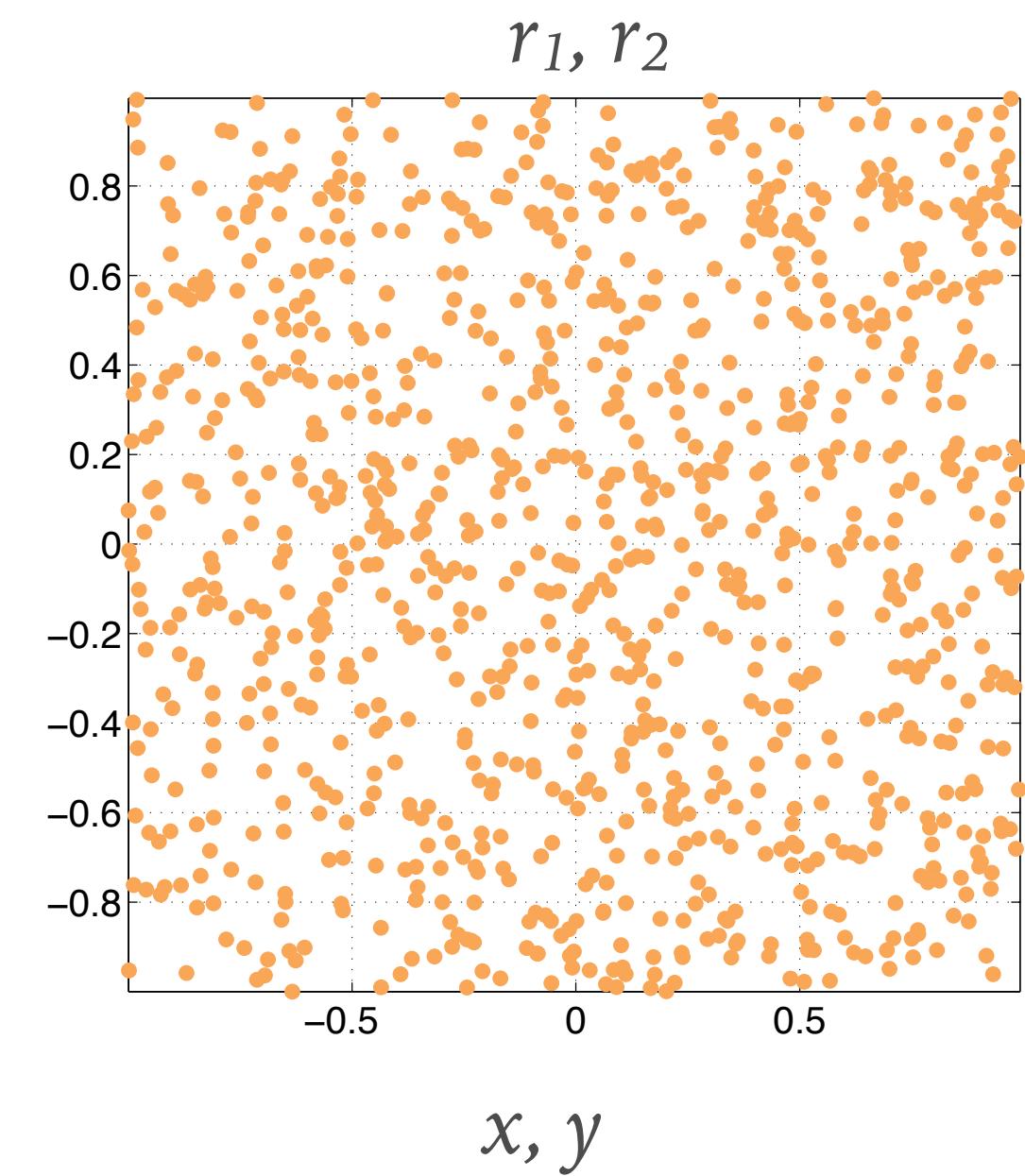
- Lets take two uniform RVs and mix them

$$r_1, r_2 \sim U(-1,1)$$

$$x = 2r_1 + r_2$$

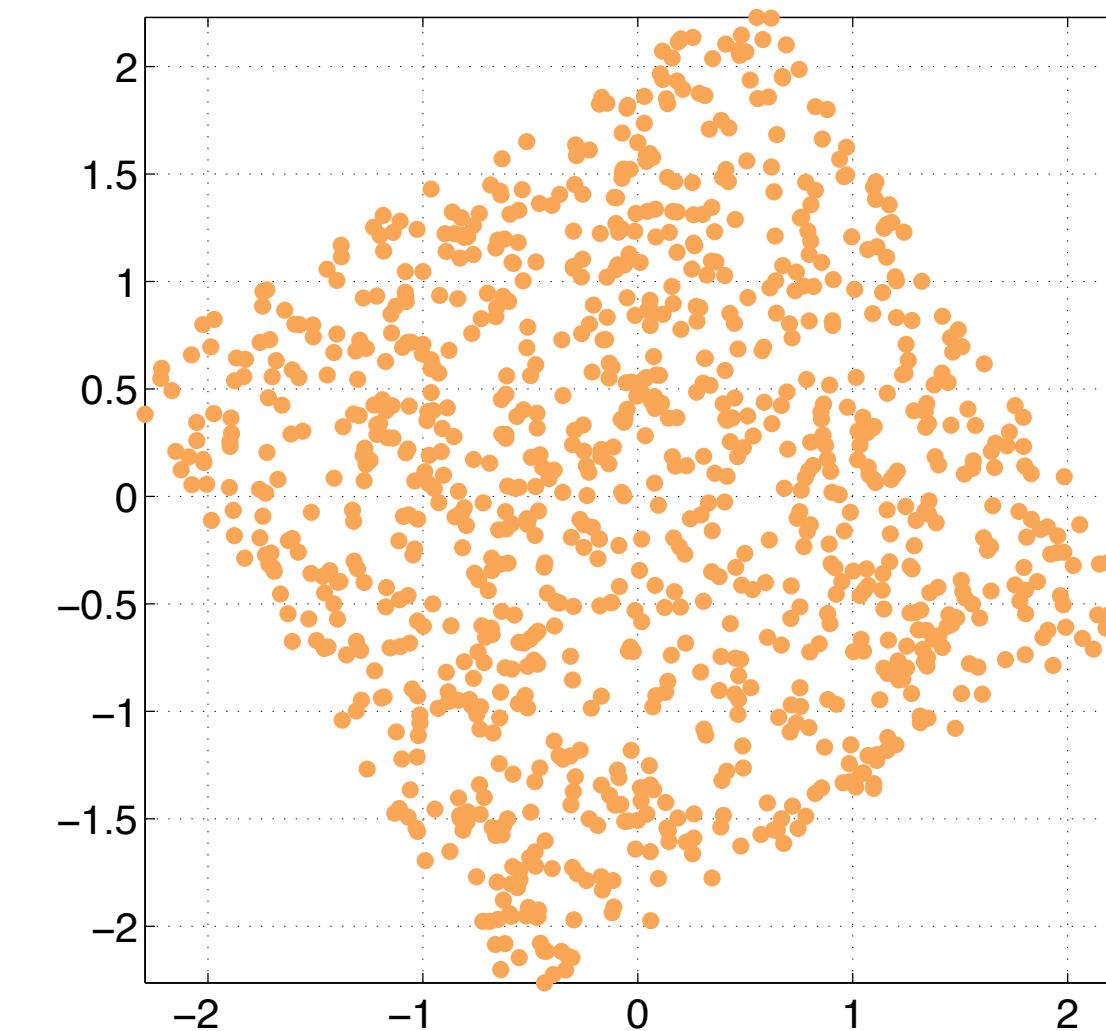
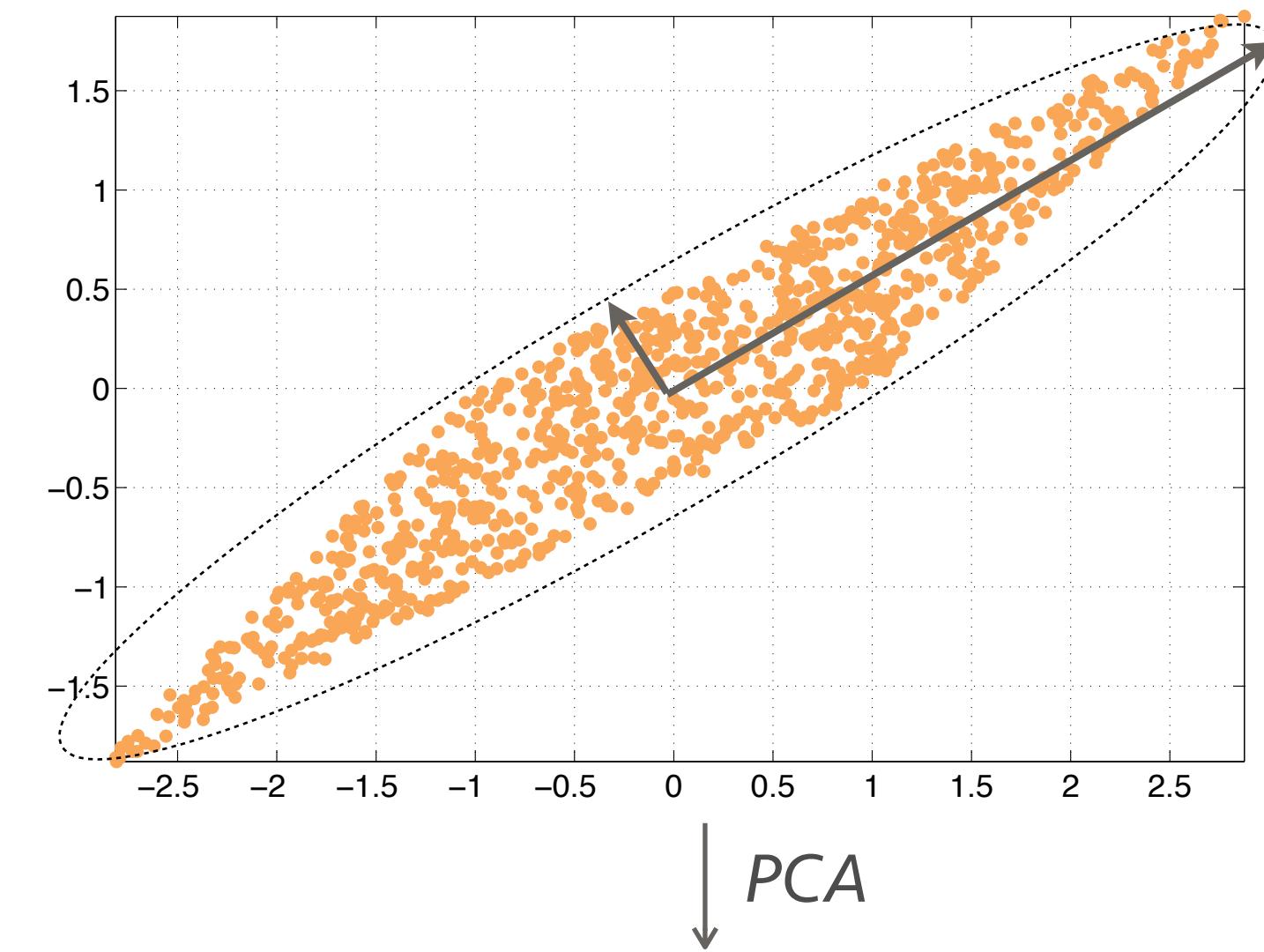
$$y = r_1 + r_2$$

- This creates a dependent x and y
 - Seen as rotation and stretching of data



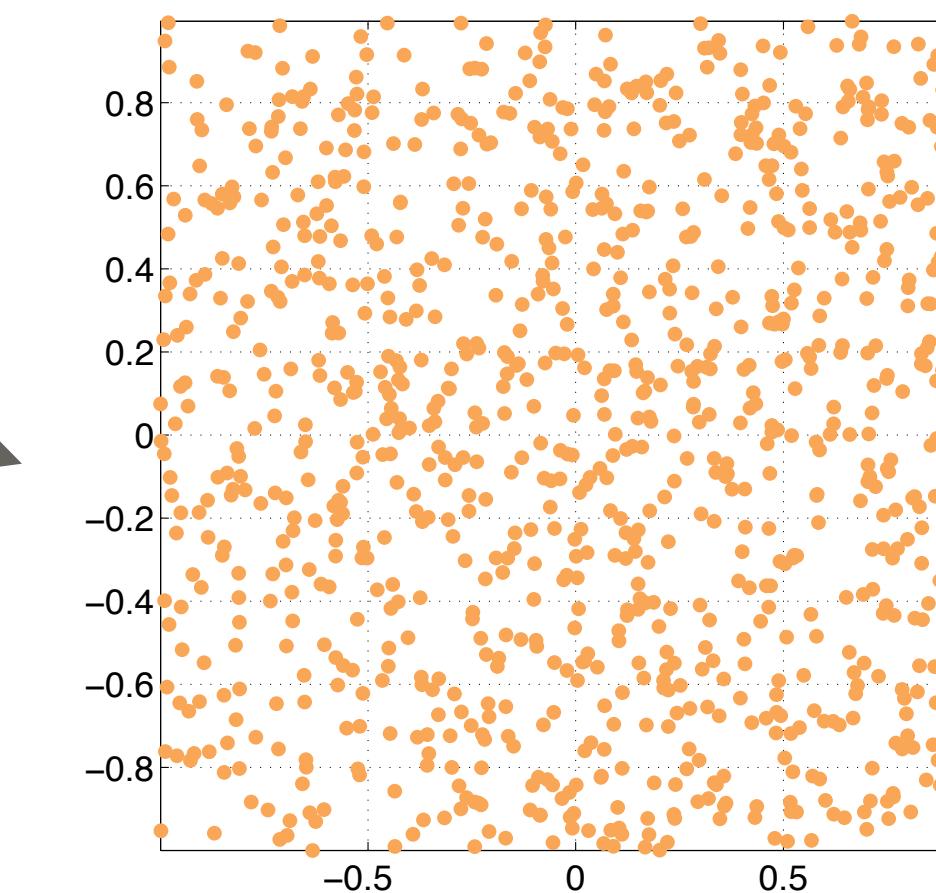
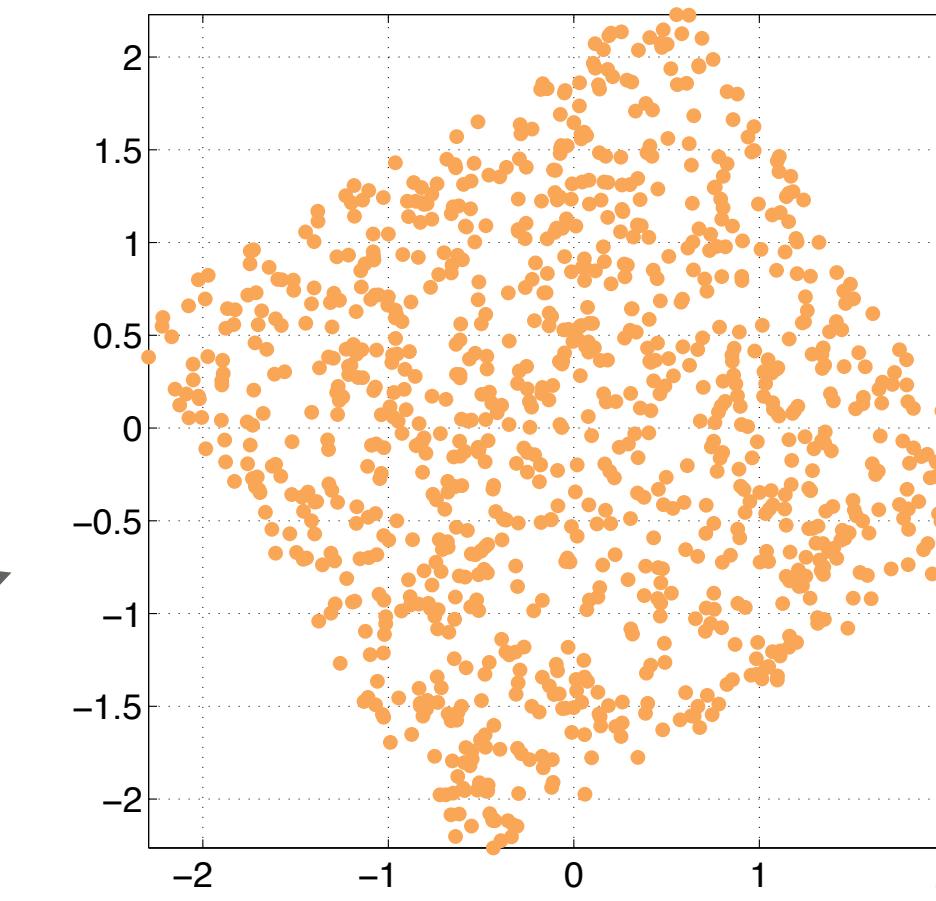
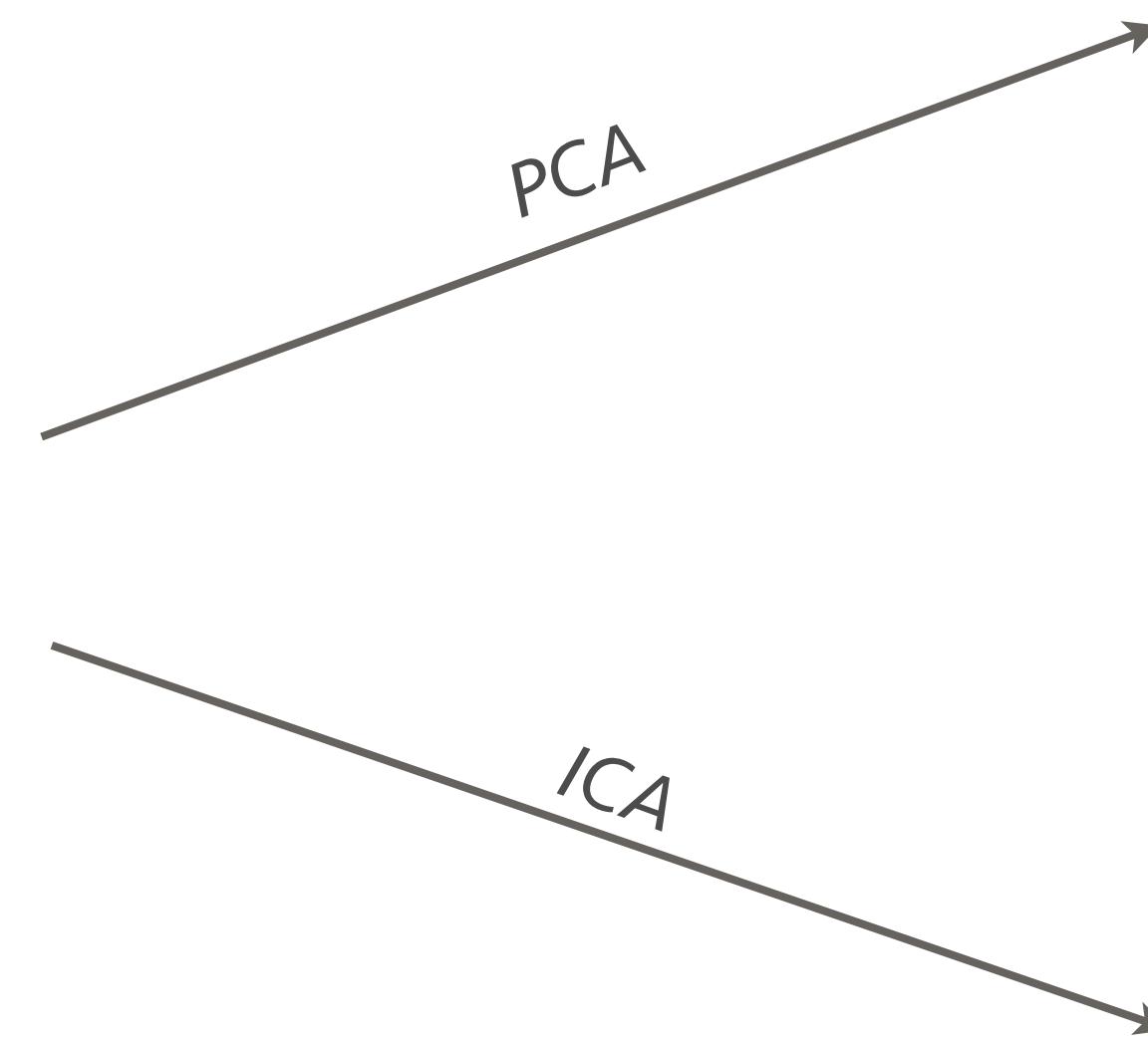
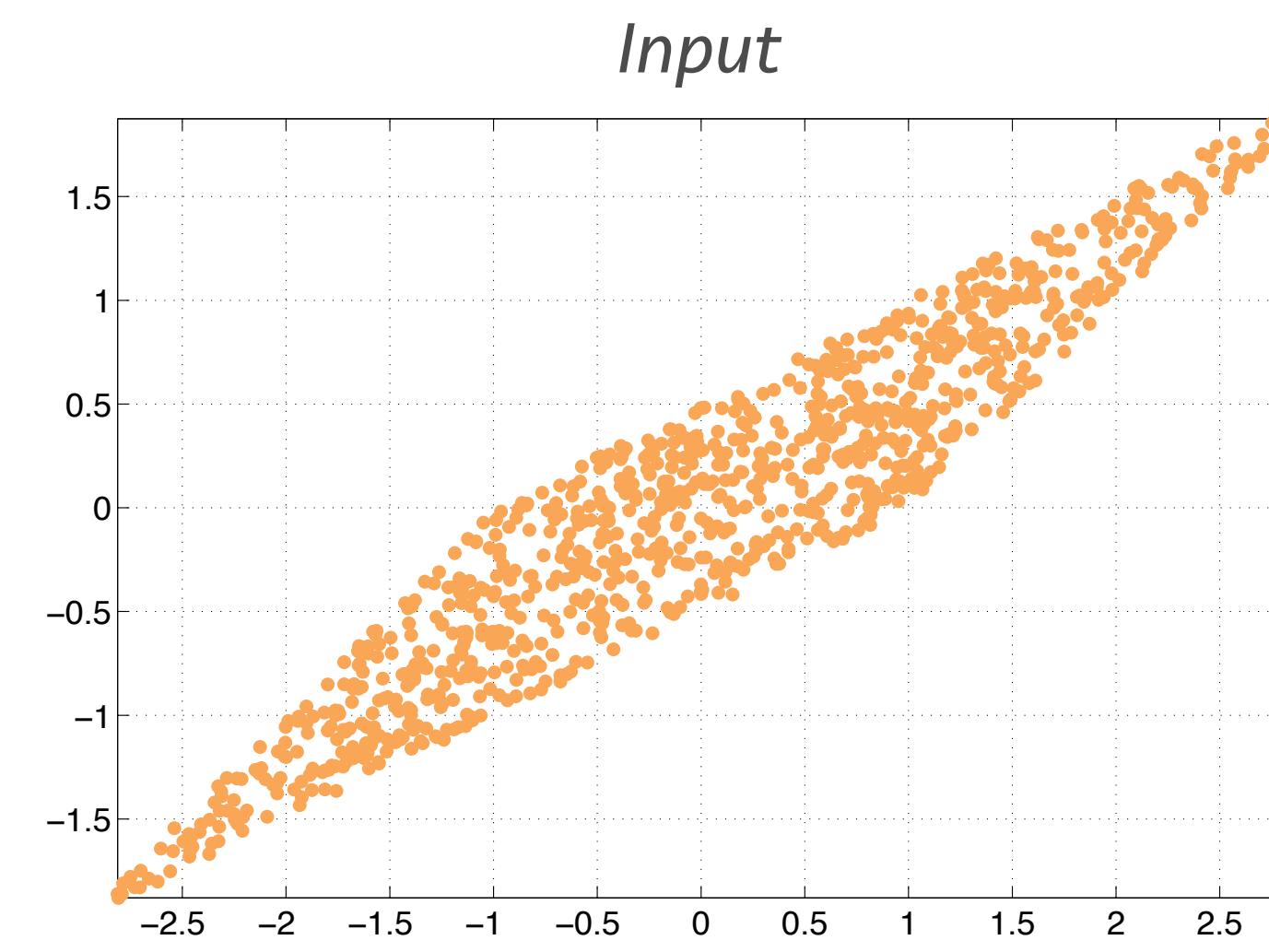
Performing PCA

- PCA will decorrelate
 - Note that PCA relies on the maximal variance directions
- The resulting projection has not produced independence



So what does ICA do?

- ICA output is independent!
 - We essentially recover the original RVs that composed the input



So why are they different?

- PCA operates on input's variance
 - Gaussians are fully characterized by mean and variance
 - Gaussians don't have *higher order statistics*
 - And that's why PCA doesn't do that last rotation step
 - It won't matter with Gaussian data since it's "round"
- More handwavingly
 - Variance $\rightarrow \ell_2$ -norm \rightarrow Gaussian

ICA issues

- Most estimators are approximate
 - The resulting output is not necessarily the correct one
 - And will often vary between runs and algorithms
- There might not be independence in the data
 - ICA returns a maximally independent projection, not an independent one
 - The output might not be what you expected to get!

ICA limitations

- Invariance to output permutations

$$P(y_1, y_2, y_3) = P(y_1)P(y_2)P(y_3) = P(y_2)P(y_1)P(y_3) = \dots$$

- Output order is not guaranteed and can differ through runs
- No ordering of components
 - PCA orders outputs in terms of contributing variance
 - ICA doesn't have an order
 - As a result we can't reduce dimensionality!

Combining PCA and ICA

- If we need to perform dimensionality reduction we precede ICA with PCA
 - 1) Use PCA to reduce dimensionality
 - 2) Use ICA to impose independence
 - Apply ICA on the output of the PCA
- That's ok, since ICA is a generalization of PCA

So what about the features?

- How do ICA and PCA features differ?
- ICA features provide a more compact/sparse projection
 - PCA projection often has statistical dependencies
- Both PCA features and projection are decorrelated
 - There is no fixed structure across the ICA features
 - Just the projection exhibits independence

Analysis vs. synthesis features

- PCA **features** are “bi-directional”
 - Because the eigenvectors are orthogonal

$$\mathbf{z} = \mathbf{W} \cdot \mathbf{x}, \quad \hat{\mathbf{x}} = \mathbf{W}^\top \cdot \mathbf{z}$$

- That does not hold with ICA
 - We have **analysis** features: $\mathbf{z} = \mathbf{W} \cdot \mathbf{x}$
 - And **synthesis** features: $\hat{\mathbf{x}} = \mathbf{W}^+ \cdot \mathbf{z}$

Be careful when combining the two!

- If we want both dimensionality reduction and independence
 - Step 1: Do PCA to reduce the dimensions

$$\mathbf{Z}_P = \mathbf{W}_P \cdot \mathbf{X}, \quad \mathbf{X} \in \mathbb{R}^{M \times N}, \mathbf{W}_P \in \mathbb{R}^{K \times M}, \mathbf{Z}_P \in \mathbb{R}^{K \times N}$$

- Step 2: Do ICA on the PCA weights to produce independence

$$\mathbf{Z}_I = \mathbf{W}_I \cdot \mathbf{Z}_P, \quad \mathbf{W}_I \in \mathbb{R}^{K \times K}, \mathbf{Z}_I \in \mathbb{R}^{K \times N}$$

- Consolidated transforms:

- Analysis features: $\mathbf{Z}_I = (\mathbf{W}_I \cdot \mathbf{W}_P) \cdot \mathbf{X} \Rightarrow \mathbf{W} = \mathbf{W}_I \cdot \mathbf{W}_P, \quad \mathbf{W} \in \mathbb{R}^{K \times M}$

- Synthesis features: $\hat{\mathbf{X}} = (\mathbf{W}_I \cdot \mathbf{W}_P)^+ \cdot \mathbf{Z}_I$

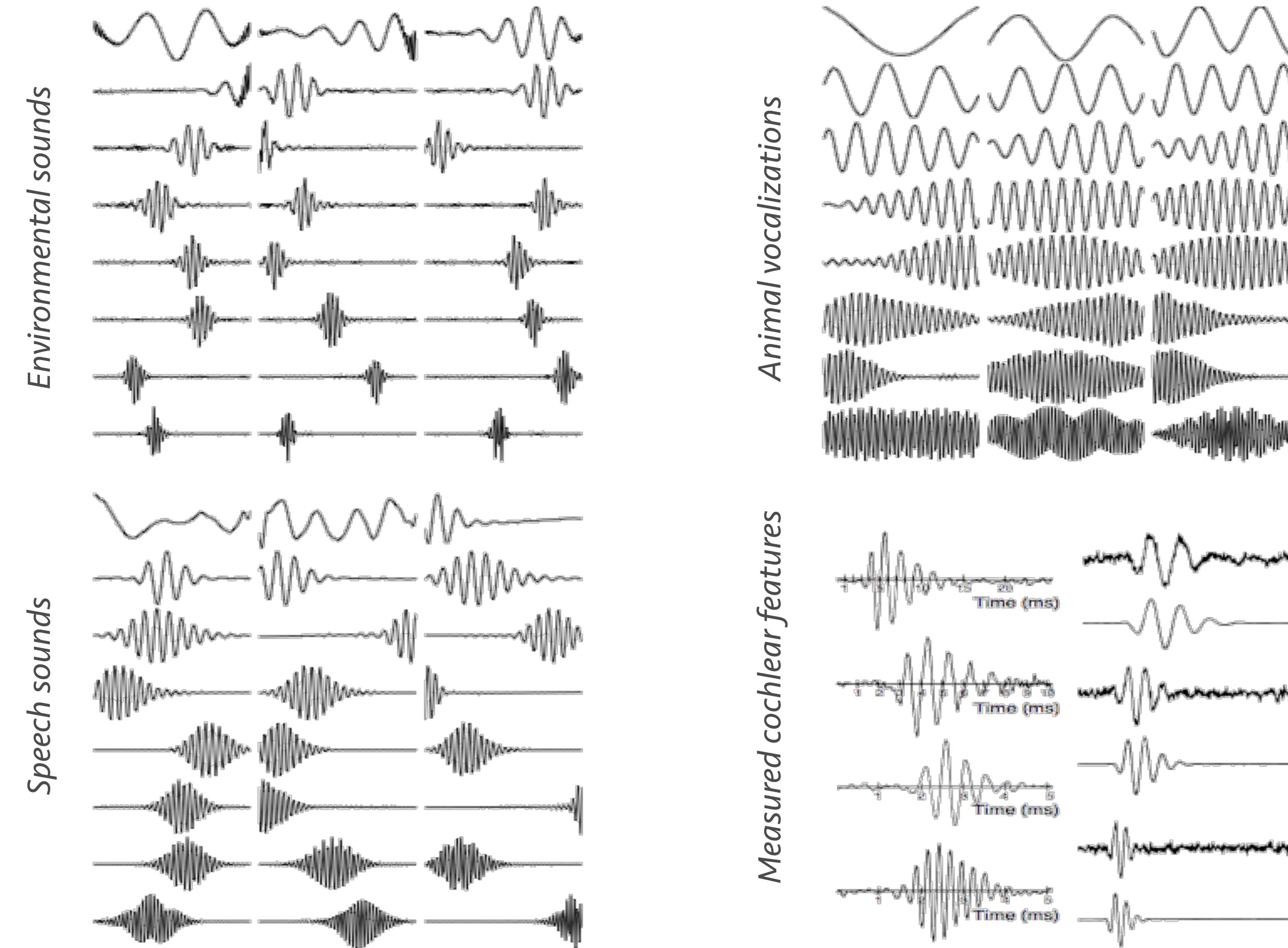
Example features from sounds

- Obtain lots and lots of natural sounds
 - E.g. sounds found in nature, birds, walking on leafs, etc.
- Place short snippets as columns in a large matrix
 - and do PCA and ICA

$$\mathbf{Z} = \mathbf{W} \cdot \begin{bmatrix} x(t) & x(t+1) \\ \vdots & \vdots \\ x(t+N) & x(t+1+N) \end{bmatrix} \dots$$

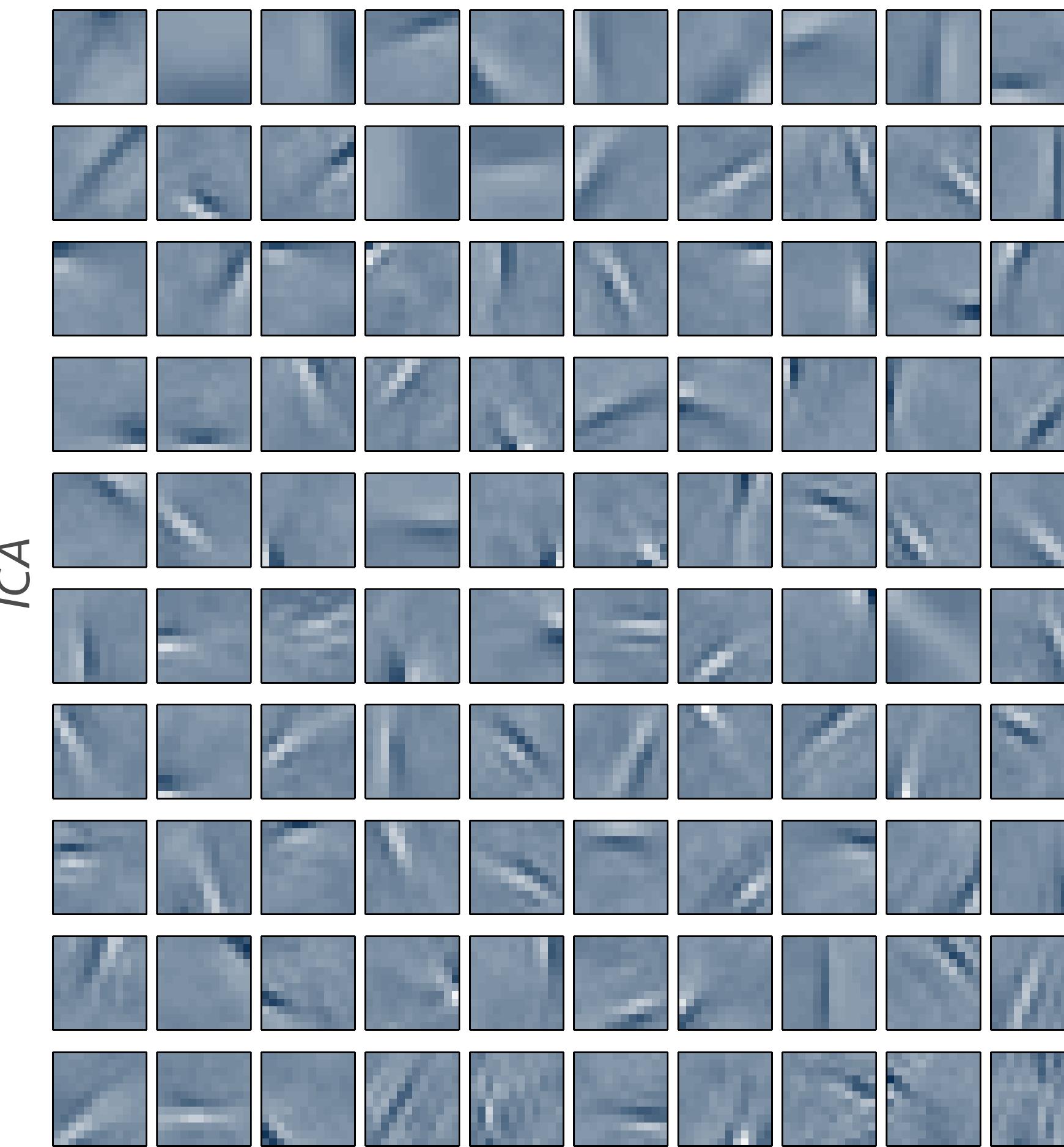
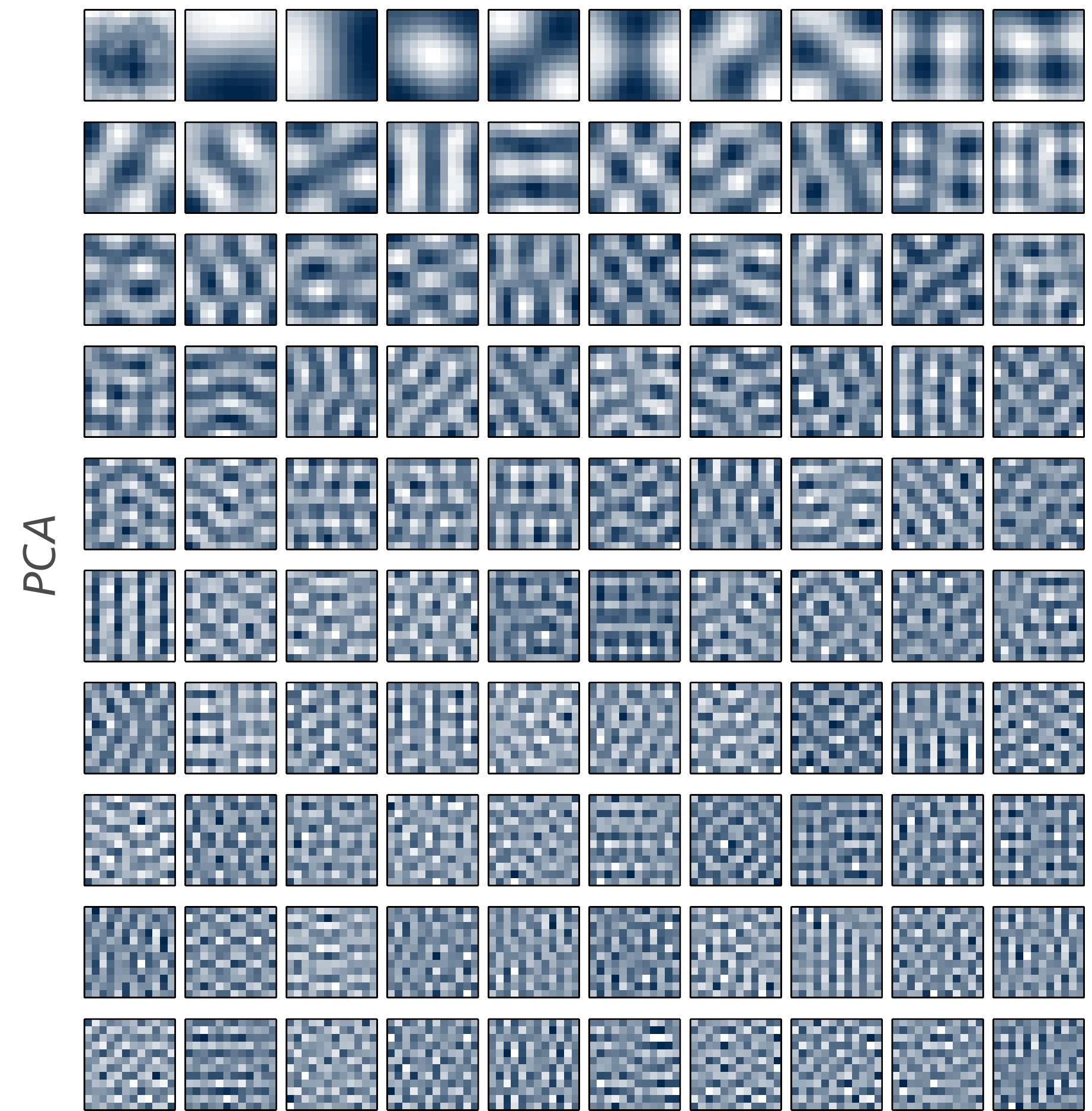
- We know that PCA results in sinusoids

ICA features from varying sounds



Same with images

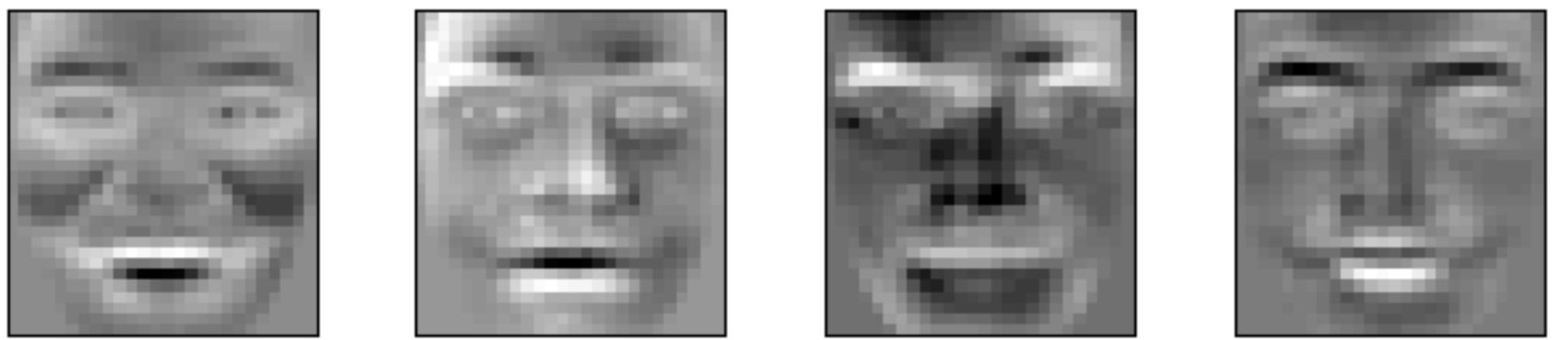
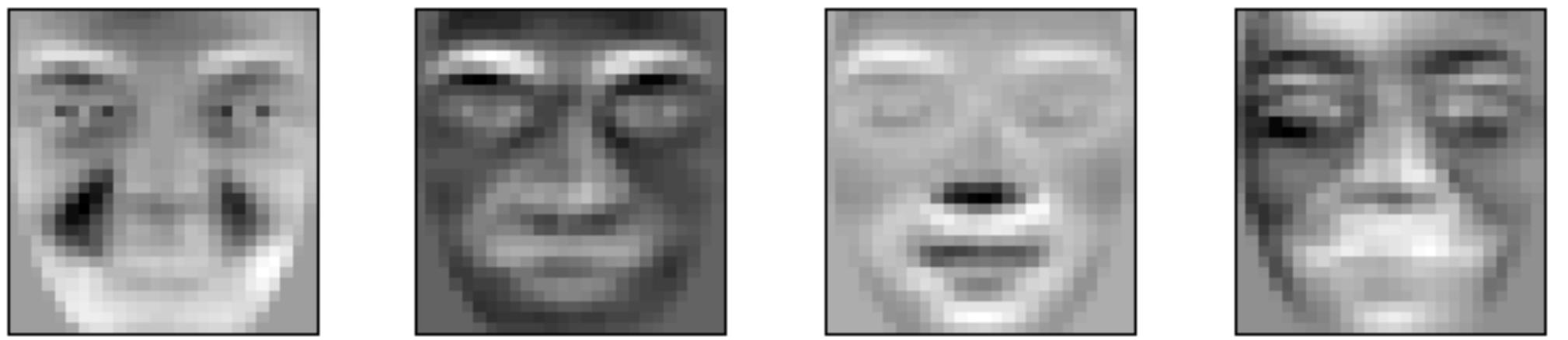
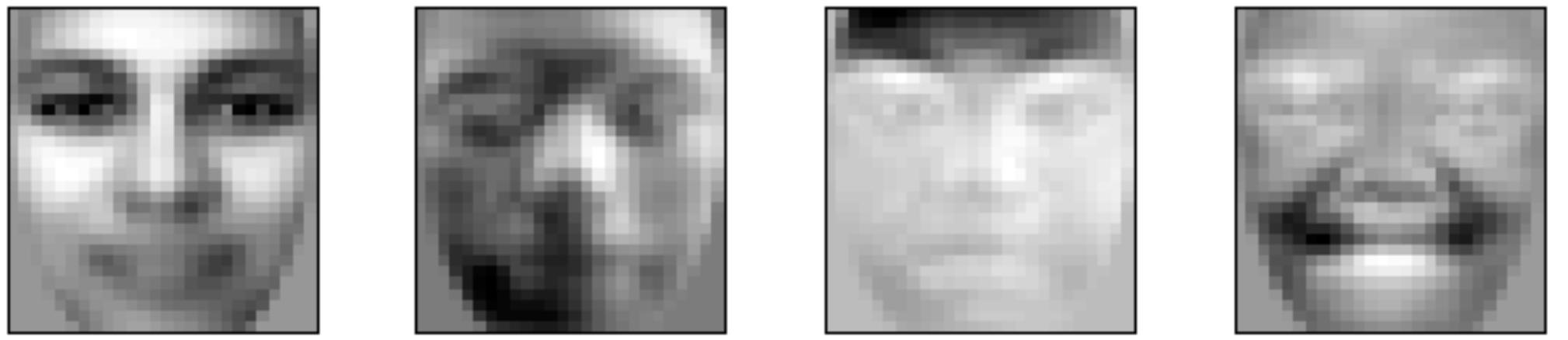
- Do ICA features look familiar?



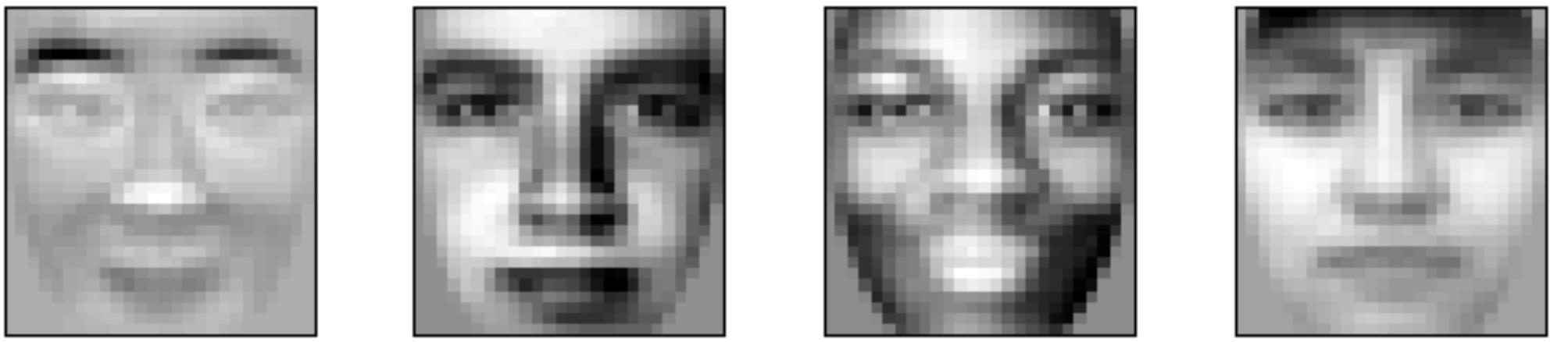
Kinda look like the V1 filters!

What about faces?

Eigenfaces



ICA-faces



One lesson learned from ICA

- PCA assumes a Gaussian world
 - For a multivariate Gaussian input it does indeed return independent outputs
 - >2nd order Gaussian cumulants are already zero
- ICA does not have the Gaussian assumption and thus is more appropriate for real data
 - This is more like the world we live
 - This was a big revelation in machine learning!

Non-negative Matrix Factorization

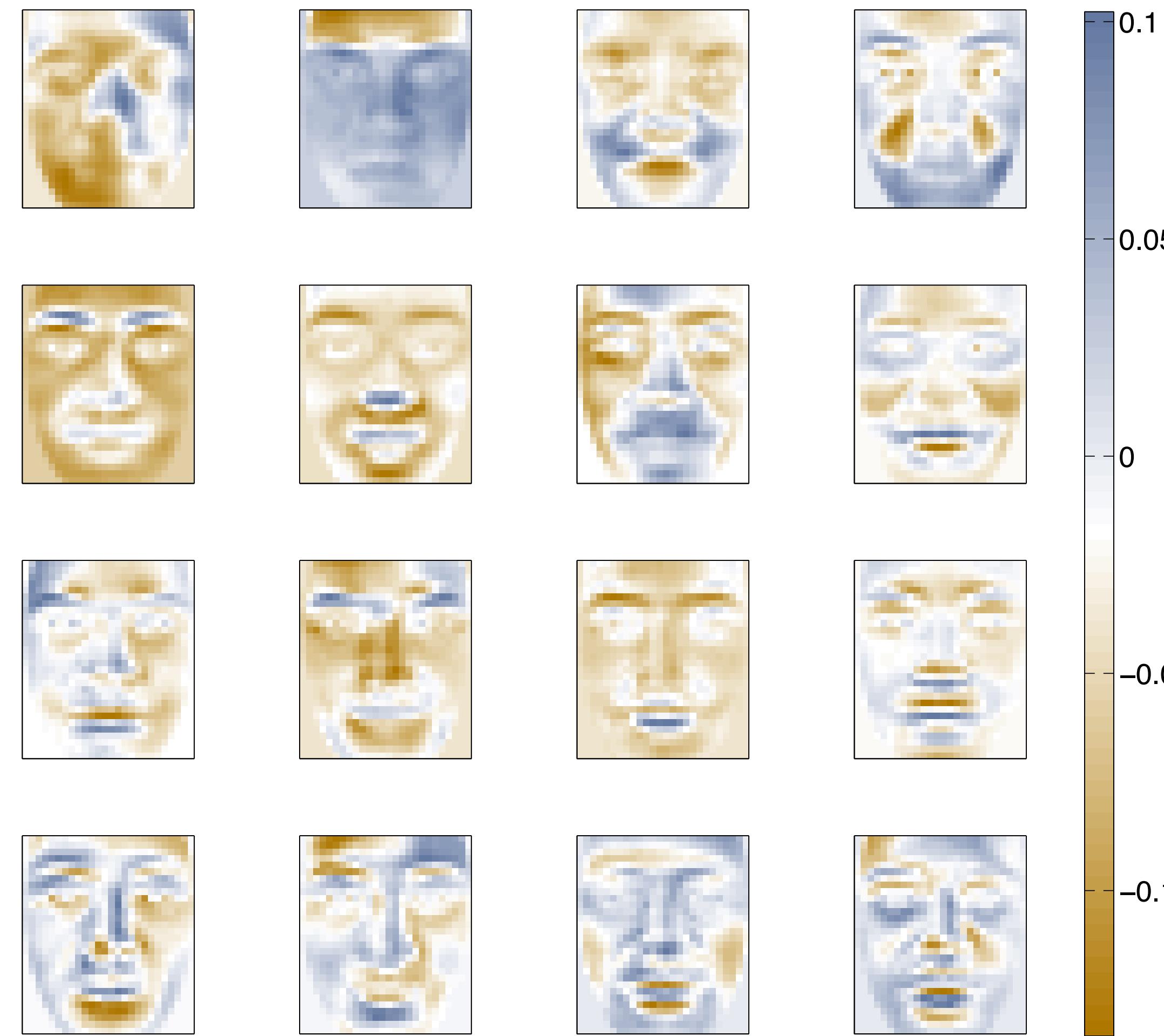
- A (relatively) recent algorithm (Lee & Seung 1999) closely related to components analyses
- Has one magical property
 - It always gives you what you want!
- Has one annoying property
 - Nobody knows quite why!!!

Non-negative data

- We often deal with “non-negative data”
 - Pixels, energies, compositions, counts, etc
- Non-negative data need special treatment
 - Negative-valued features can contradict reality

For example

- Partially negative images as bases – why??



Obtaining non-negative features

- Define the factorization problem

$$\mathbf{X} \approx \mathbf{W} \cdot \mathbf{H}$$

$$\mathbf{X} \in \mathbb{R}^{M \times N, \geq 0}, \mathbf{W} \in \mathbb{R}^{M \times R, \geq 0}, \mathbf{H} \in \mathbb{R}^{R \times N, \geq 0}$$

- This is similar to the PCA/ICA setup
 - \mathbf{W} contains the synthesis features, \mathbf{H} their activations
 - R defines the low-rank dimensionality
- How do we solve this one?
 - One known, two unknowns, ugh ...

Solving for the factorization

- We need to estimate two factors
 - Alternate their estimation
- Example algorithm
 - Start with random \mathbf{W}
 - estimate an \mathbf{H} given \mathbf{W}
 - estimate a new \mathbf{W} given \mathbf{H}
 - repeat until convergence

Solving for one factor

- The problem is simpler
 - Only one unknown

$$\min_{\mathbf{W} \text{ or } \mathbf{H}} \sum_{i,j} \left| \mathbf{X} - \mathbf{W} \cdot \mathbf{H} \right|^2$$

$$\mathbf{X} \in \mathbb{R}^{M \times N, \geq 0}, \mathbf{W} \in \mathbb{R}^{M \times R, \geq 0}, \mathbf{H} \in \mathbb{R}^{R \times N, \geq 0}$$

- Imposing non-negativity
 - Non-negative least squares (slow)
 - Constrained optimization (slow)
 - Do least-squares and clip the negative numbers (fast!)

A simple NMF algorithm

- Start with random \mathbf{W}
 - estimate new \mathbf{H} given \mathbf{W} :
$$\mathbf{H} = \mathbf{W}^+ \cdot \mathbf{X}$$
$$\mathbf{H} = \max(\mathbf{H}, 0)$$
 - estimate new \mathbf{W} given \mathbf{H} :
$$\mathbf{W} = \mathbf{X} \cdot \mathbf{H}^+$$
$$\mathbf{W} = \max(\mathbf{W}, 0)$$
 - repeat until convergence

A problem

- We don't want to use pseudoinverses
 - No guarantee of non-negative outputs
 - They imply least-squares minimization
 - Least squares optimization implies Gaussian assumptions
 - Non-negative data are not Gaussian!
- We instead define a special KL-like distance

$$\min_{\mathbf{W}, \mathbf{H}} \left[\sum_{i,j} \mathbf{X}_{i,j} \log \frac{\mathbf{X}_{i,j}}{(\mathbf{W} \cdot \mathbf{H})_{i,j}} - \mathbf{X}_{i,j} + (\mathbf{W} \cdot \mathbf{H})_{i,j} \right]$$

Multiplicative updates

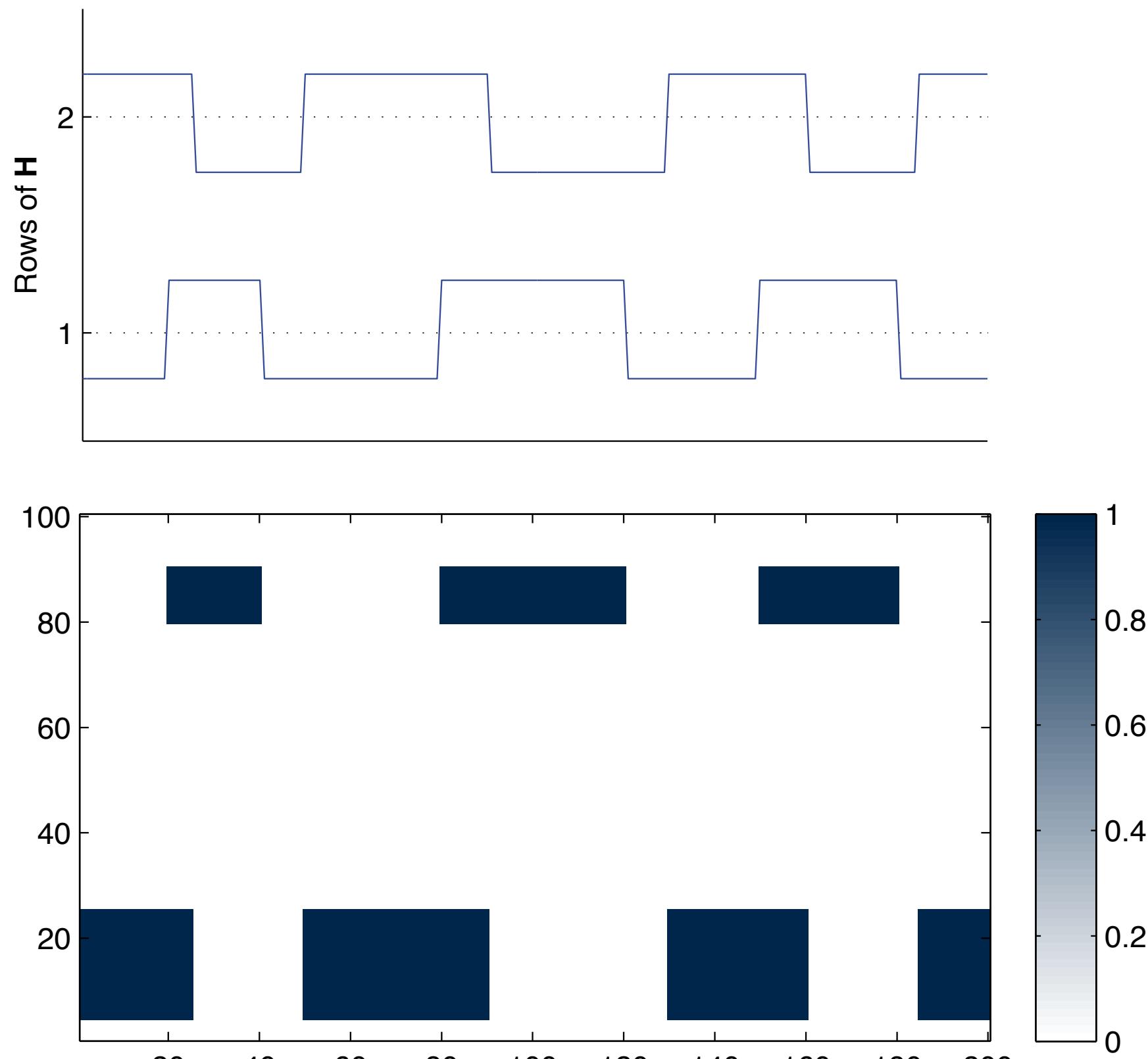
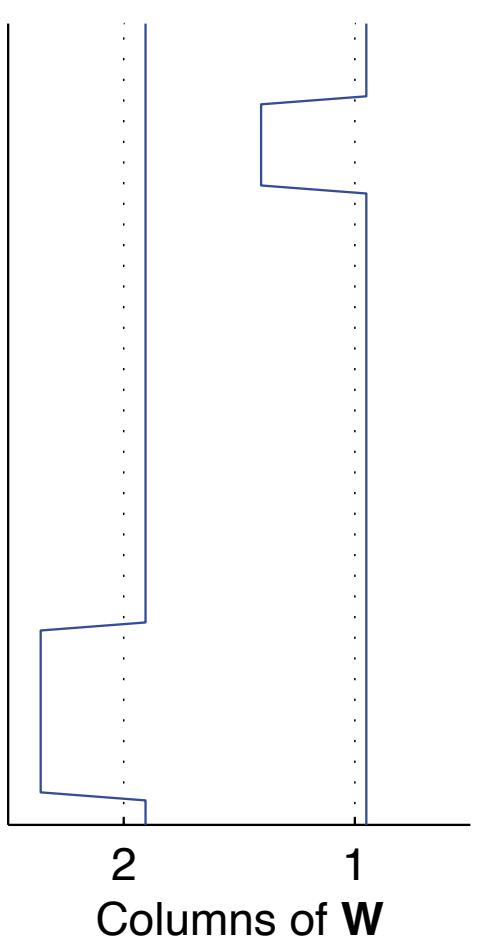
- Using some optimization magic we get:

$$\mathbf{W}_{i,j} = \mathbf{W}_{i,j} \frac{\sum_k \frac{\mathbf{X}_{i,k}}{(\mathbf{W} \cdot \mathbf{H})_{i,k}} \mathbf{H}_{j,k}}{\sum_k \mathbf{H}_{j,k}}, \quad \mathbf{H}_{j,k} = \mathbf{H}_{j,k} \frac{\sum_i \mathbf{W}_{i,j} \frac{\mathbf{X}_{i,k}}{(\mathbf{W} \cdot \mathbf{H})_{i,k}}}{\sum_k \mathbf{W}_{k,j}}$$

- Significantly faster operations
 - No more computationally costly matrix inversions

A toy example

- Start with input \mathbf{X}
- NMF will decompose as $\mathbf{X} \approx \mathbf{W} \cdot \mathbf{H}$
- The columns of \mathbf{W} will contain “vertical” information about \mathbf{X}
- The rows of \mathbf{H} will contain “horizontal” information about \mathbf{X}

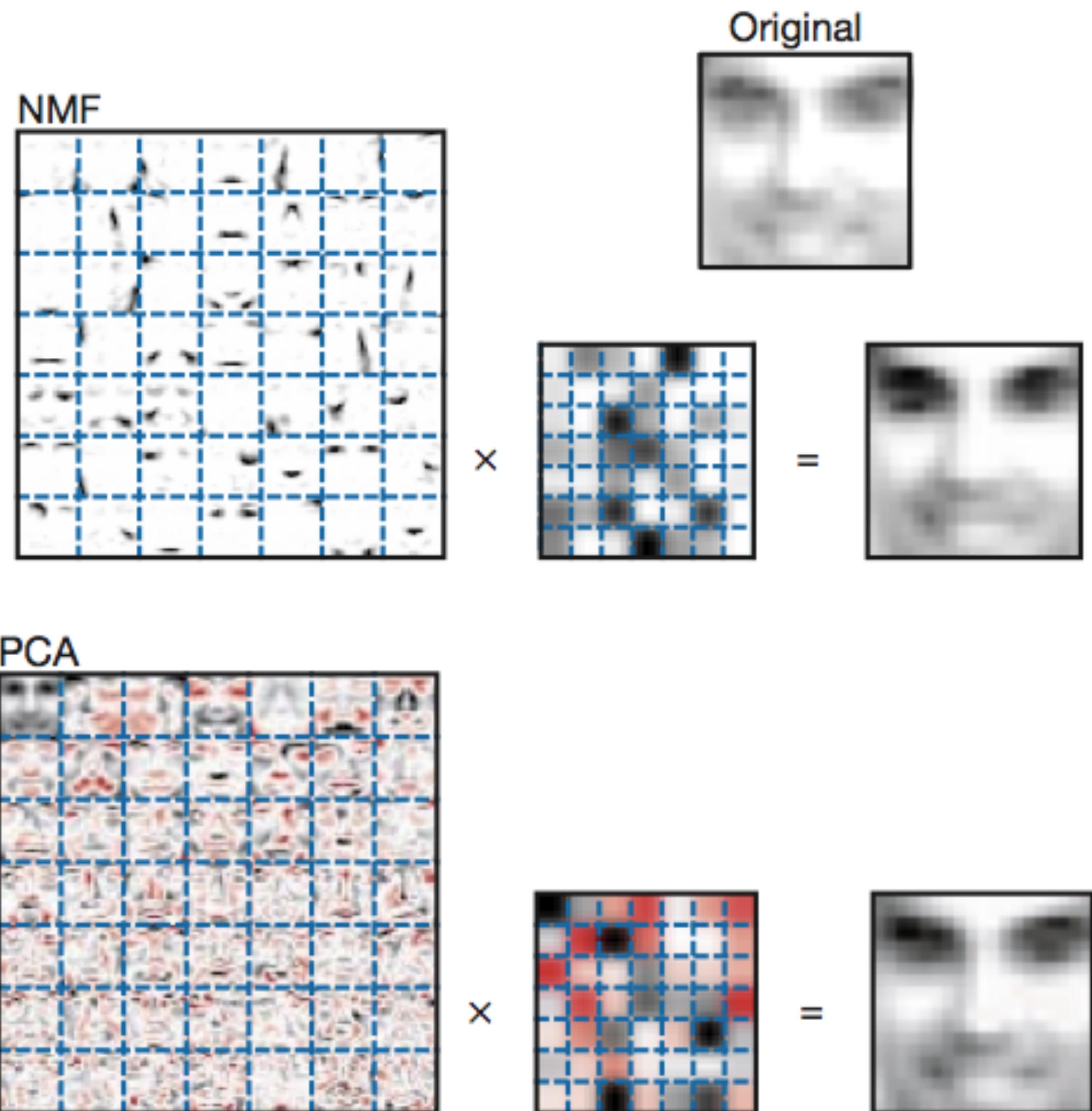


Decomposition by parts

- NMF does “additive decompositions”
 - Explains data in terms of things you add
- This correlates with how we think
 - Scenes are made out of adding objects
 - We never have “negative” object presence

Example on faces

- Both PCA and NMF describe the data to a good degree
 - Eigenfaces are not interpretable though (very abstract notions)
 - NMF-faces find parts that are additive (noses, eyes, etc.)
- NMF is a better way to explain structured data

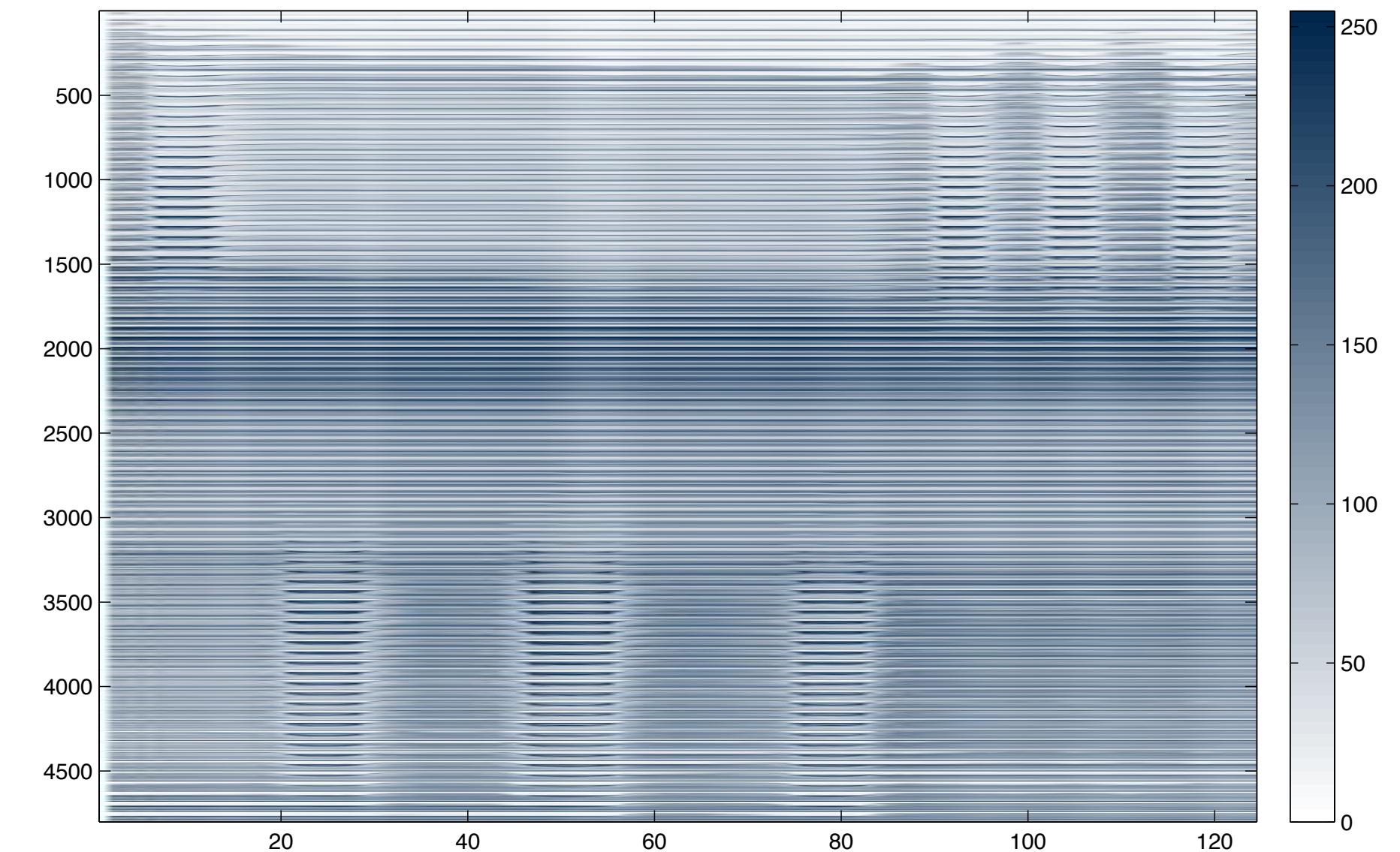


Component analyses on movies

- Movies are fun data for component analyses
 - Very-high dimensionality
 - Too much data to train on, we need a more compact form
 - PCA/NMF can do that!
 - Scenes are composed out of elements
 - We want to discover these elements to better analyze the input
 - ICA/NMF can do that!
 - There are visual data and audio data
 - Both exhibit their own structure, often they interrelate
 - All techniques help there!

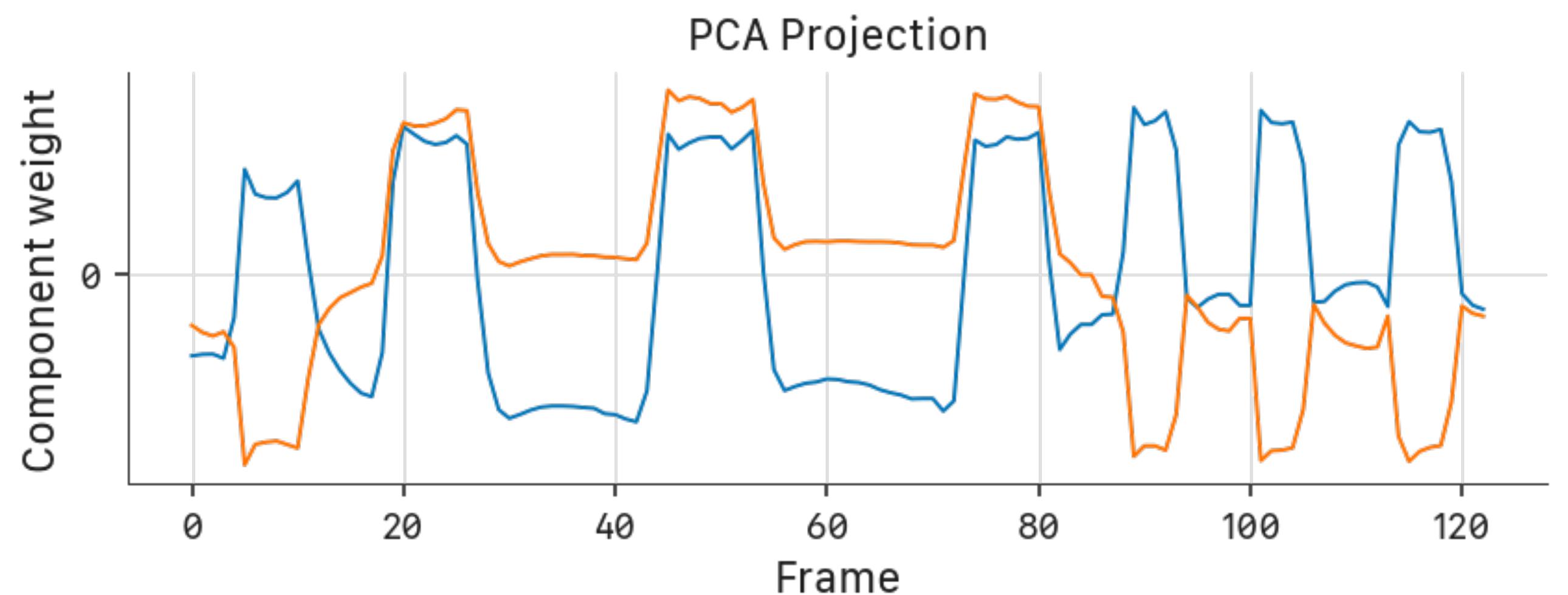
A Video Example

- The movie is a series of frames
 - Each frame is a data point
 - 126 frames of 80×60 pixels
 - Data will be 4800×126 samples
- Lets try various techniques
 - PCA, ICA, NMF
 - Compare features and weights



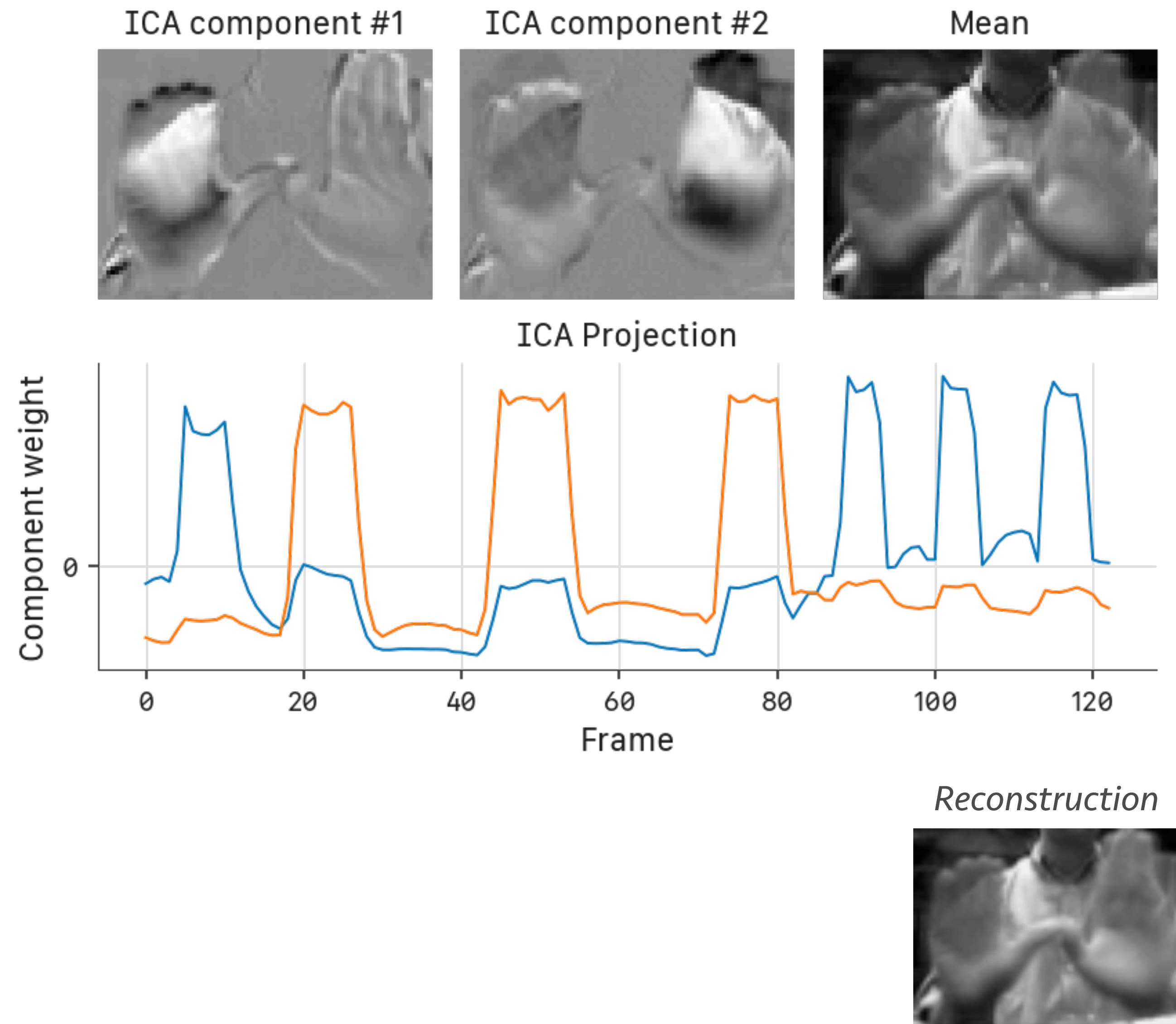
PCA Results

- Nothing special about the visual components
- They are orthogonal pictures
 - Does this mean anything? (no!)
 - Some segmentation between constant vs. moving parts
- Some representation of the movement in the weights



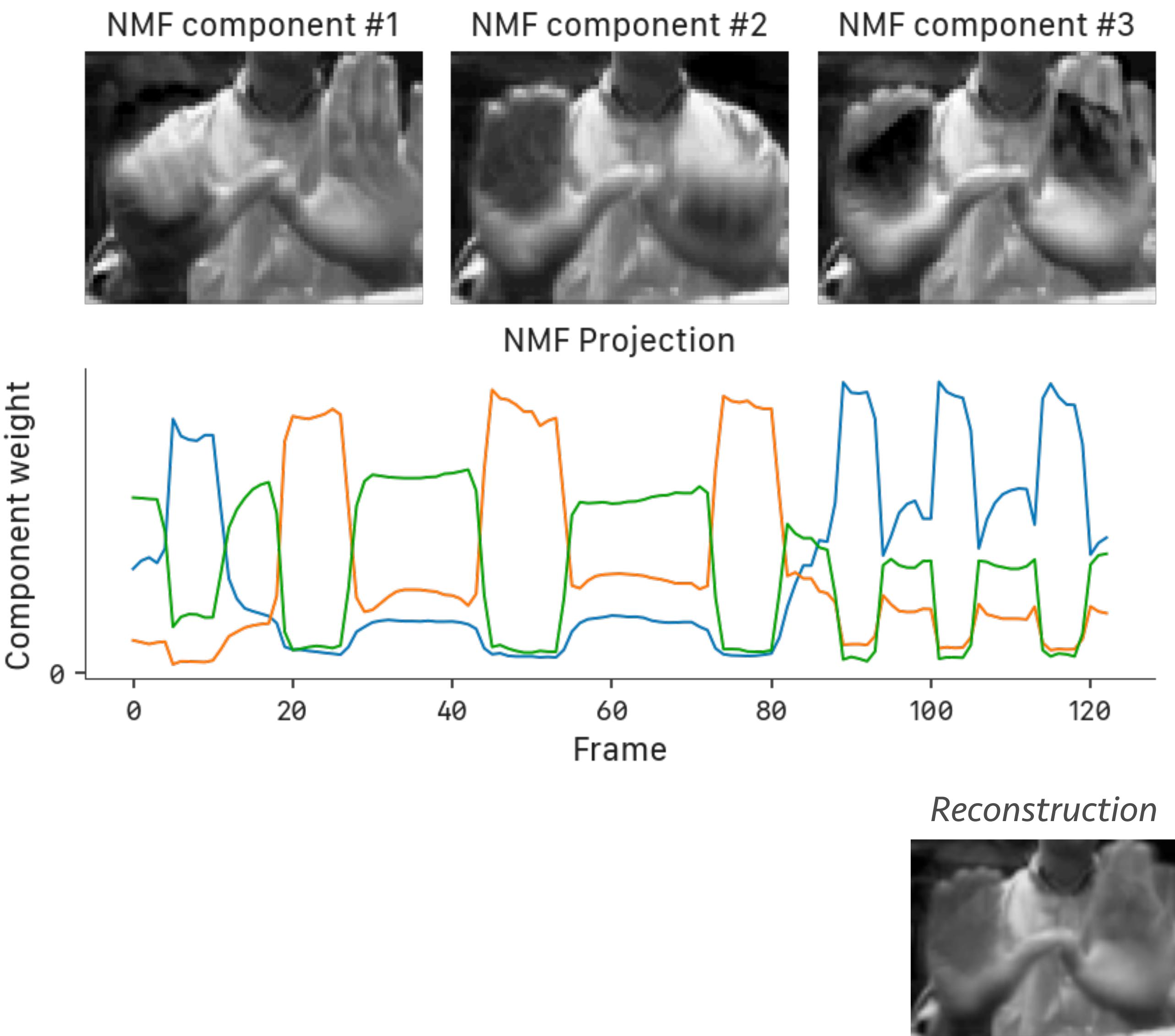
ICA Results

- Somewhat better
- More independence
 - The two actions are split
 - A decomposition by parts
- The component weights roughly describe actions



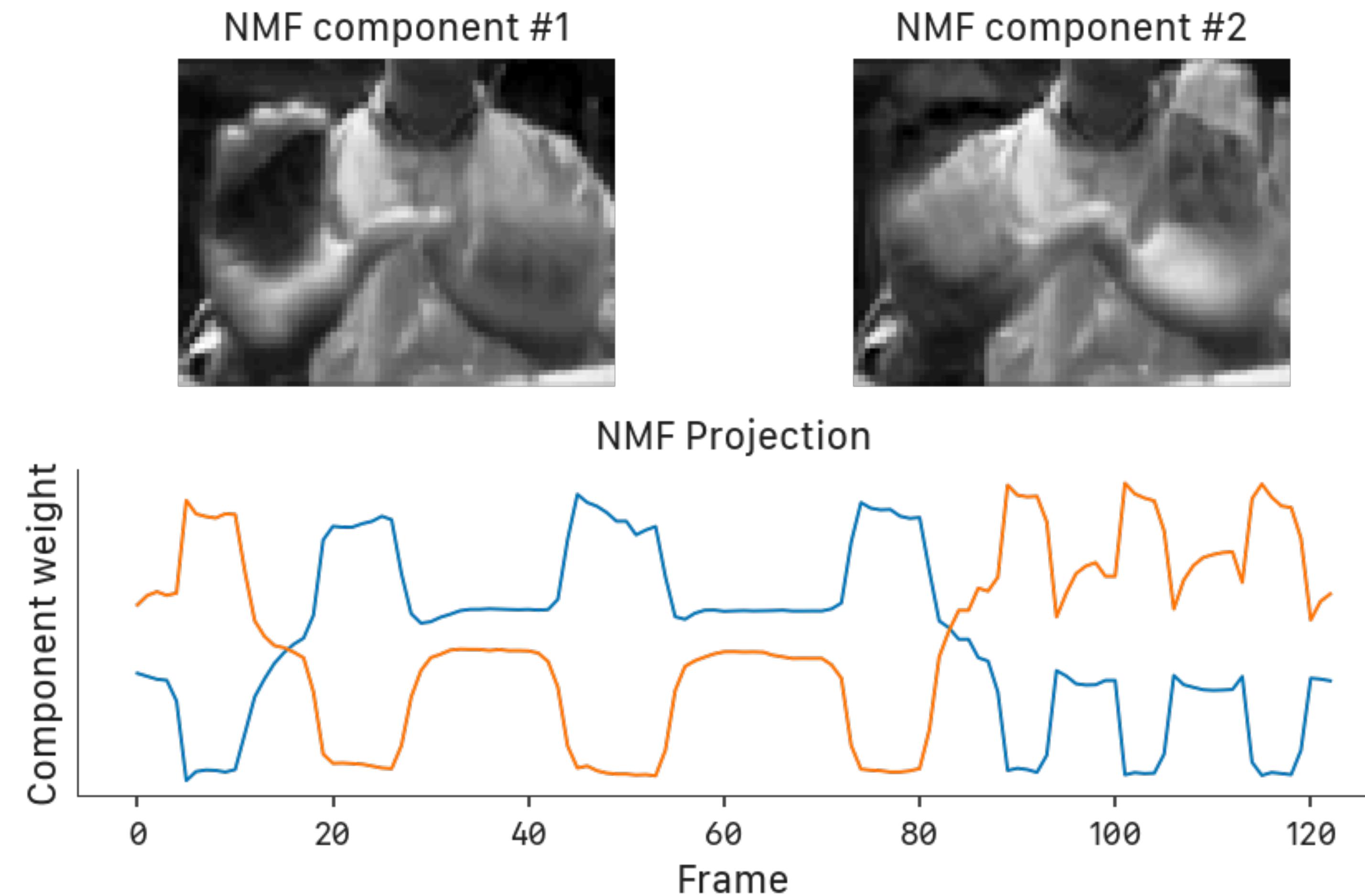
NMF Results

- A different take on the scene
- Somewhat interpretable
 - They describe some of the possible states of the video
 - Perhaps a more semantically meaningful representation
- There's some redundancy
 - We use too many dimensions!



If we use fewer dimensions

- NMF gives us the two video states!

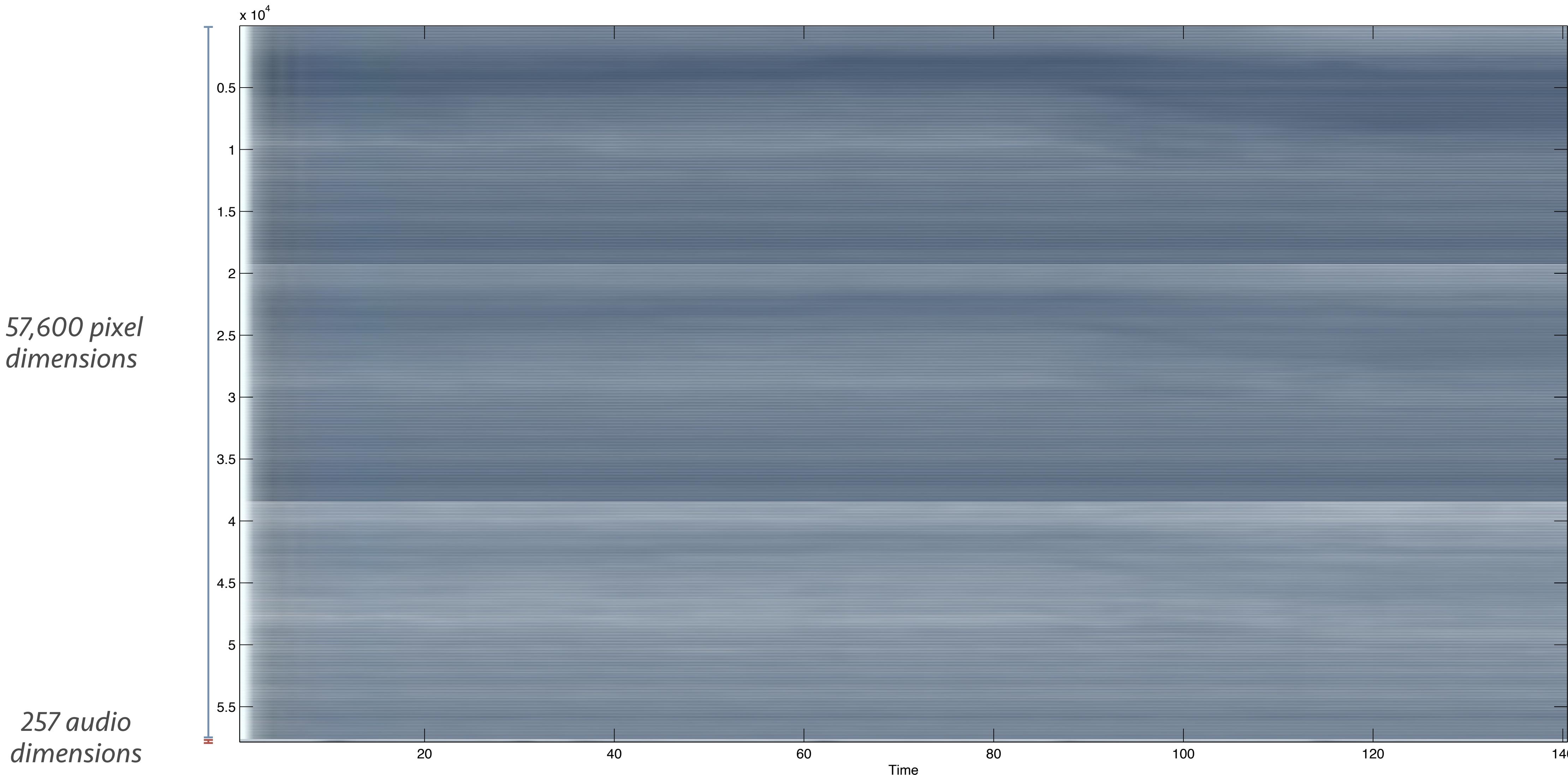


Audio/Visual components?

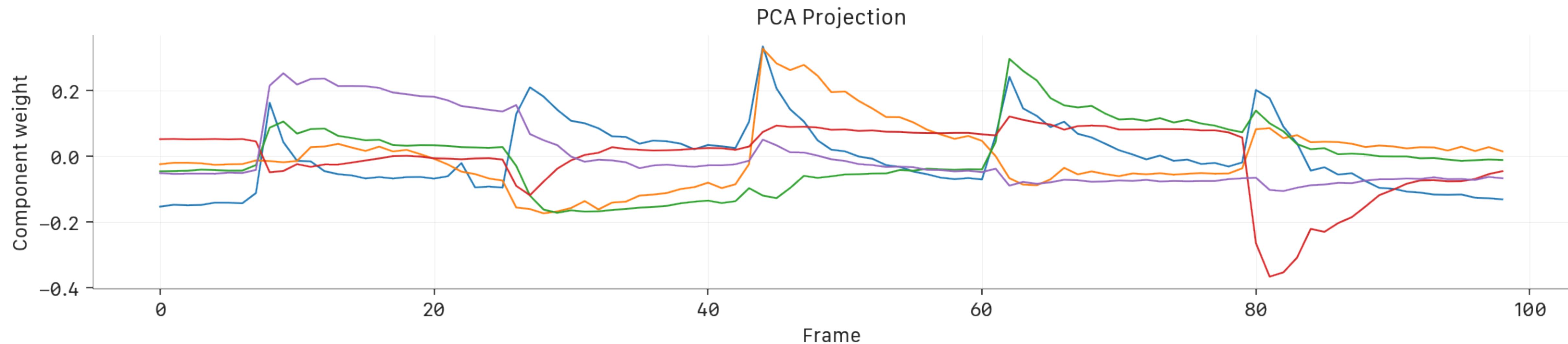
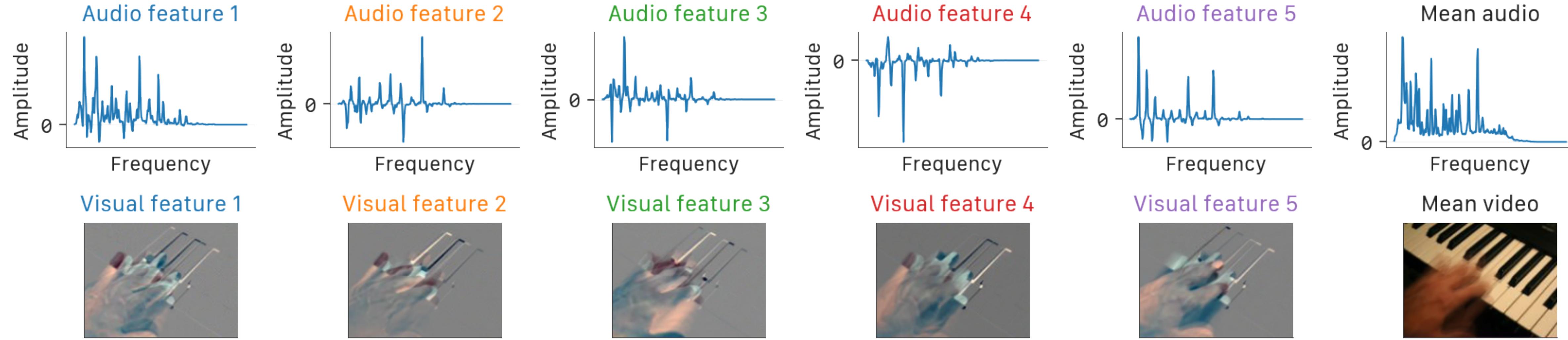
- We can also simultaneously process audio and video data
- Can we find a/v dependencies?
- What is an a/v feature?



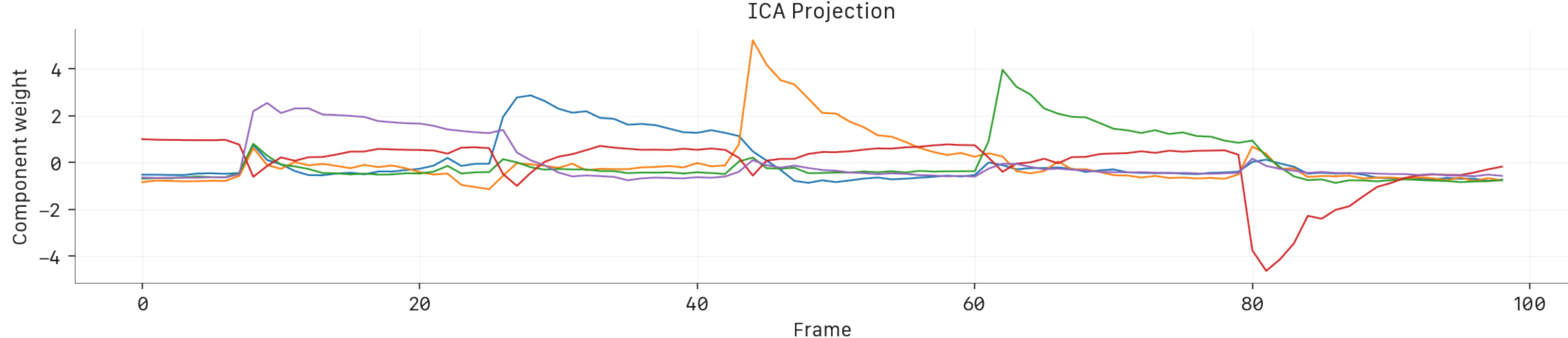
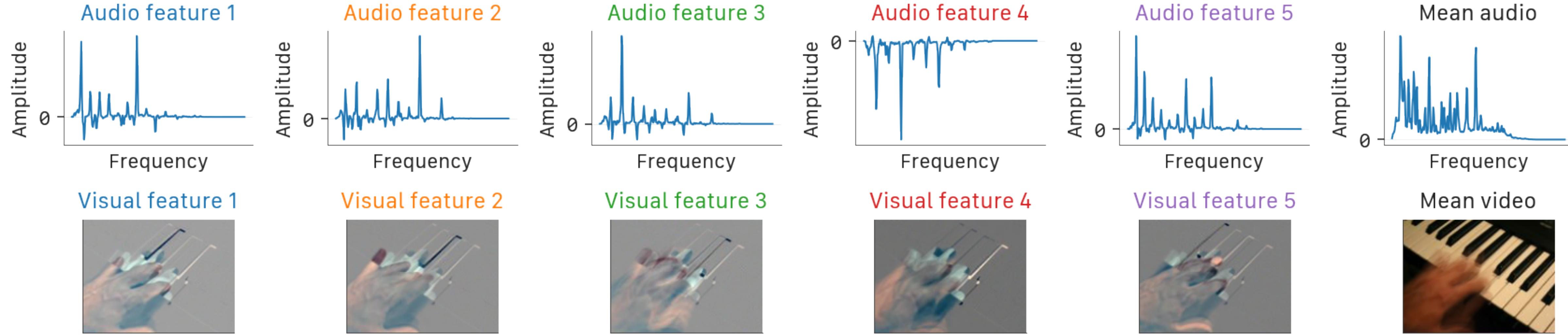
What does the data look like?



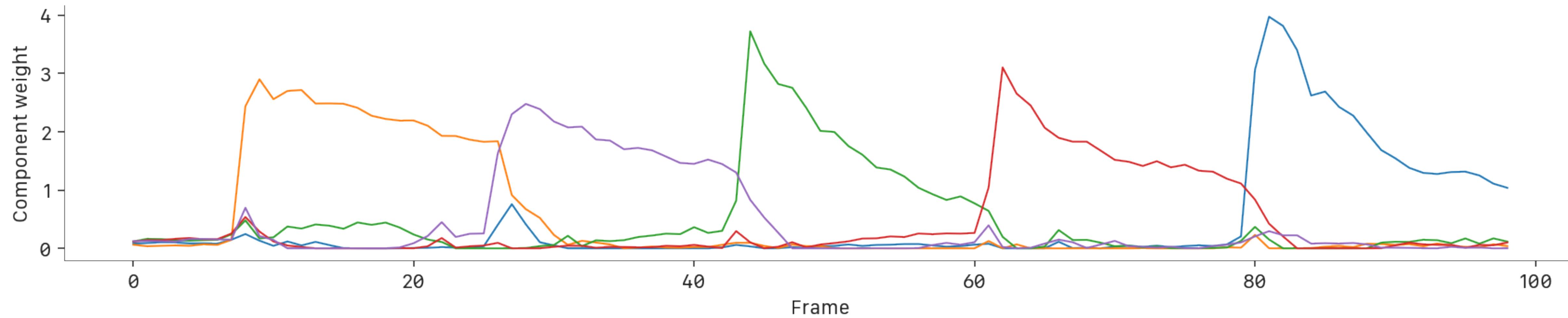
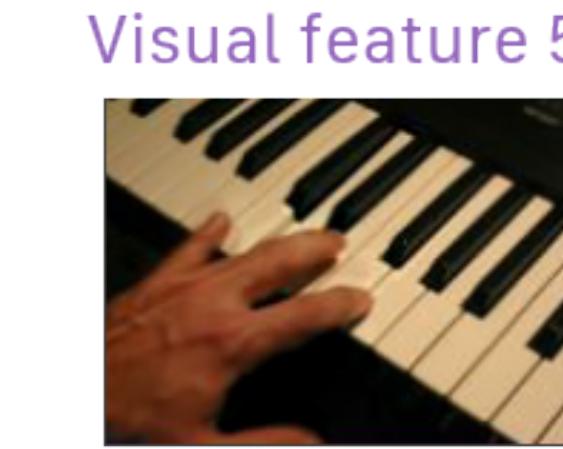
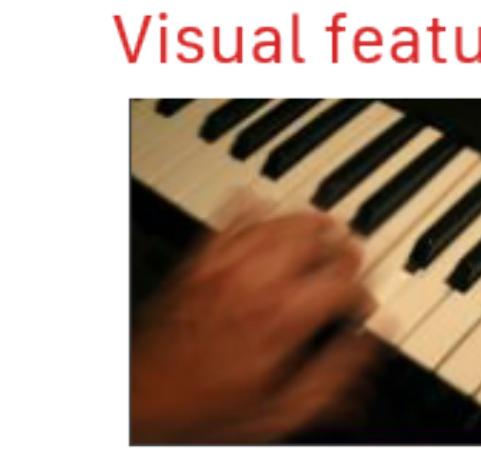
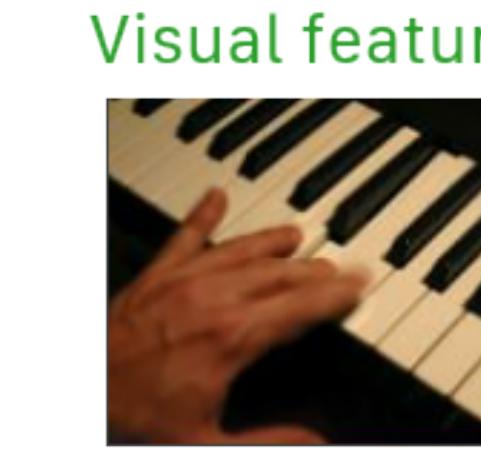
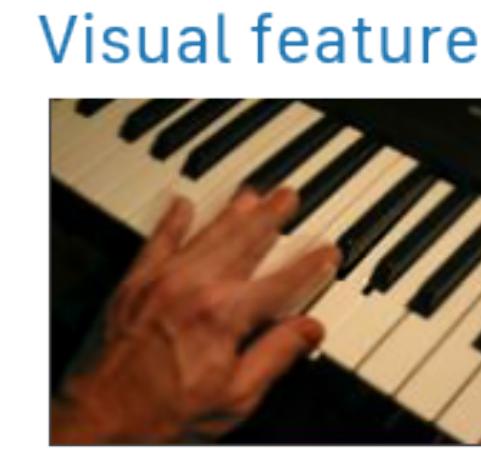
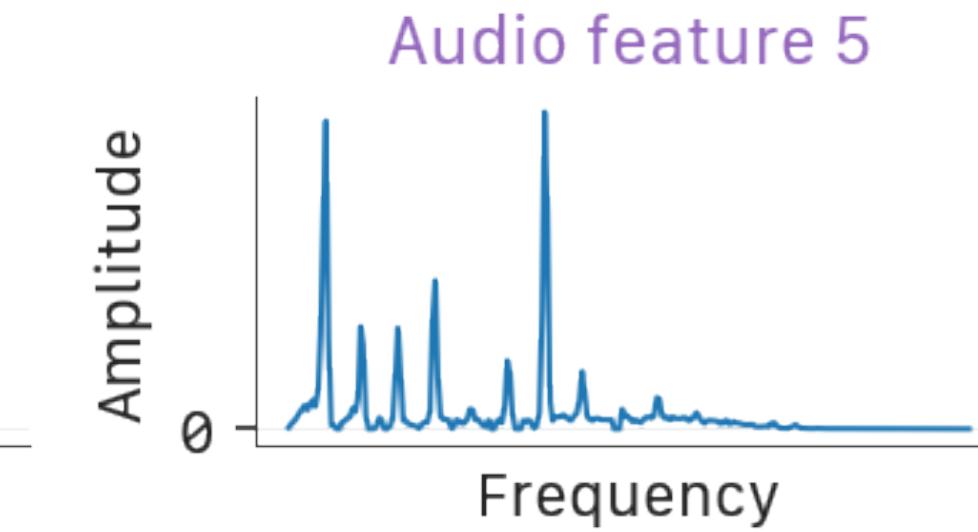
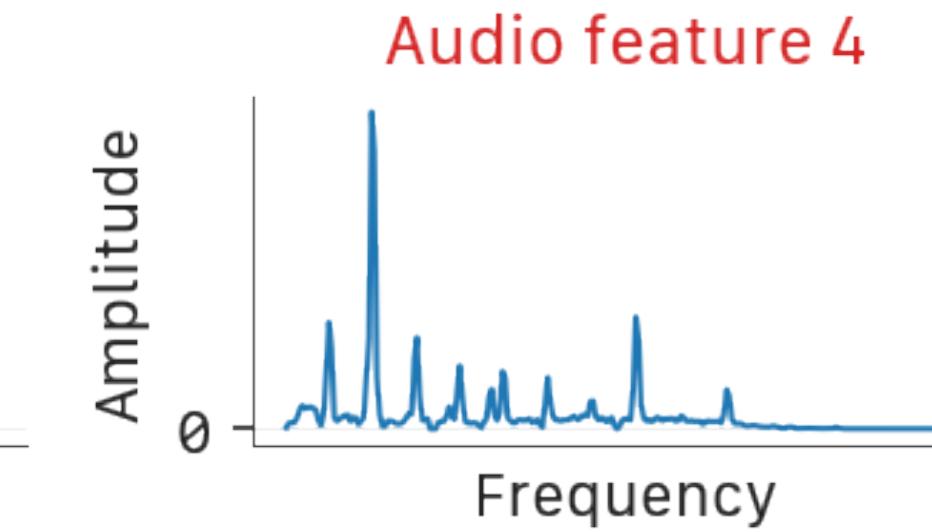
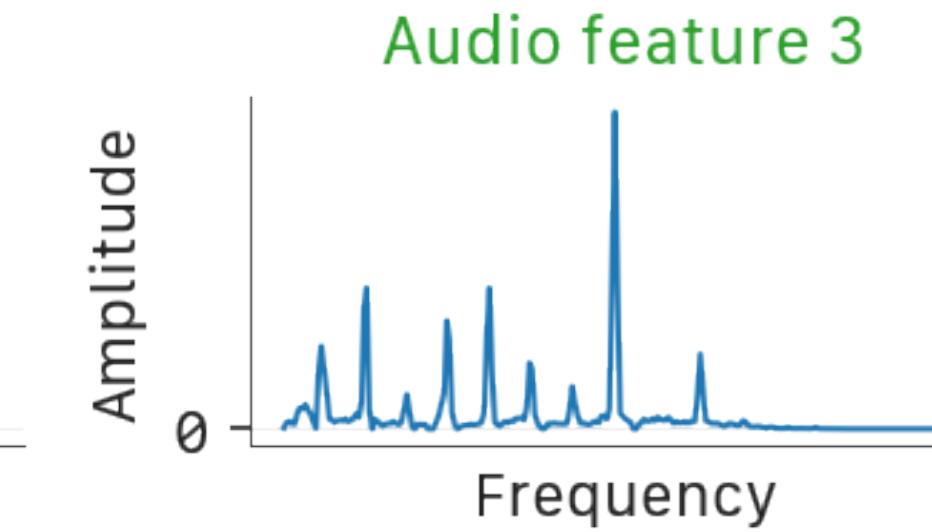
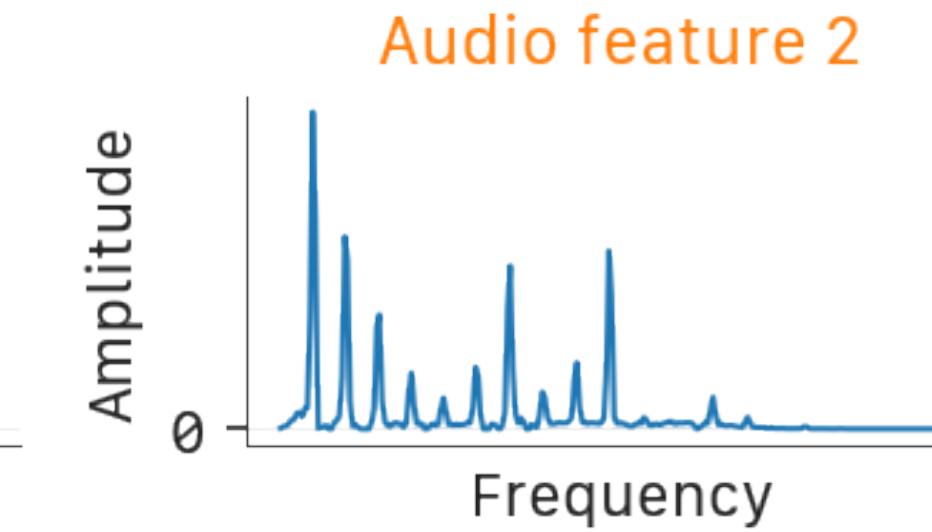
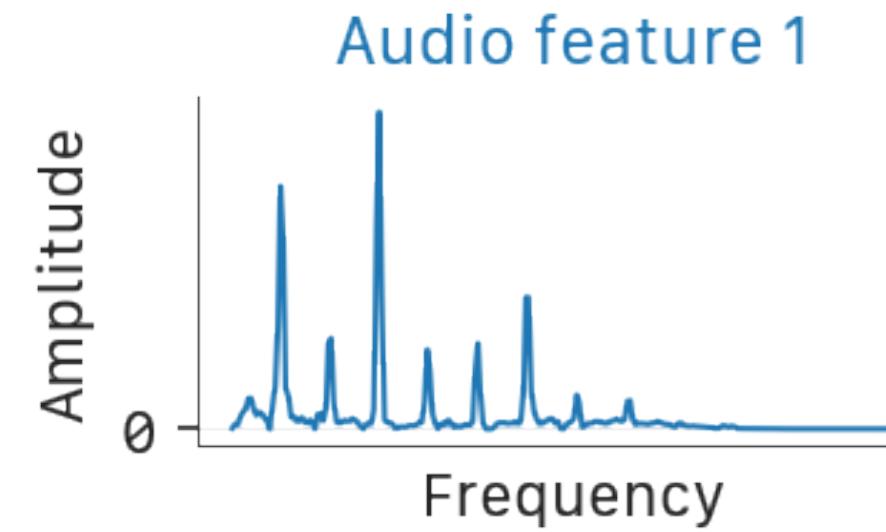
Audio/Visual PCA components



Audio/Visual ICA components



Audio/Visual NMF components



PCA, ICA or NMF?

- Depends on what you want to do
 - PCA does a fantastic job in dimensionality reduction
 - ICA provides a sparser output
 - And is perceptually more relevant
 - NMF provides interpretable outputs
 - But only for non-negative data
- As usual, there is no right answer
 - When in doubt try them all!

Recap

- Independent Component Analysis
 - Obtains maximal independence
 - Does not reduce dimensionality
- Non-Negative Matrix Factorization
 - Best for analysis of non-negative data
 - pixels, energies, count data, etc ...
 - No particular statistical property though

Next lecture

- Non-linear methods
 - What do do when your data has non-linear relations
- Manifolds and embedding
 - Finding latent structure in high dimensions
 - Looking at data topologies and their transforms

Reading

- Textbook sections 6.5-6.6
- Independent Component Analysis (optional)
 - http://www.cis.hut.fi/aapo/papers/IJCNN99_tutorialweb/
- Natural stimuli statistics (optional)
 - <http://redwood.berkeley.edu/bruno/papers/nature-paper.pdf>
 - https://www.cnbc.cmu.edu/~tai/readings/circuit/lewicki_natural-sounds.pdf
- Non-negative Matrix Factorization (optional)
 - <https://www-nature-com.proxy2.library.illinois.edu/articles/44565>