Features Part II – ICA and NMF
Today’s lecture

• What comes after PCA

• Independent Component Analysis
  • Achieving complete “decorrelation”

• Non-Negative Matrix Factorization
PCA and decorrelation

• Goal of PCA
  • Diagonalize the covariance
    $x^\top \cdot y = E\{xy\} = 0$
    • i.e. Decorrelate the feature weights

• 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:

\[ P(x, y) = P(x)P(y) \]
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 can’t tell anything about \( x \) if we observe \( y \)
Decorrelation and Independence

- Decorrelation does not imply independence!
  - Decorrelation: \( E{xy} = E{x}E{y} \)
  - Independence: \( E{f(x)g(y)} = E{f(x)}E{g(y)} \)

- But independence 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 uncorrelated?

<table>
<thead>
<tr>
<th></th>
<th>$x = -1$</th>
<th>$x = 0$</th>
<th>$x = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y = -1$</td>
<td>0</td>
<td>$\frac{1}{4}$</td>
<td>0</td>
</tr>
<tr>
<td>$y = 0$</td>
<td>$\frac{1}{4}$</td>
<td>0</td>
<td>$\frac{1}{4}$</td>
</tr>
<tr>
<td>$y = 1$</td>
<td>0</td>
<td>$\frac{1}{4}$</td>
<td>0</td>
</tr>
</tbody>
</table>
Decorrelation and Independence

• An example with discrete variables

  • Are they correlated?

  \[
  \begin{array}{ccc}
  y = -1 & x = -1 & x = 0 & x = 1 \\
  0 & \frac{1}{4} & 0 \\
  y = 0 & \frac{1}{4} & 0 & \frac{1}{4} \\
  y = 1 & 0 & \frac{1}{4} & 0 \\
  \end{array}
  \]

  • \(x, y\) are uncorrelated

  \[
  E\{xy\} = E\{x\}E\{y\} = 0
  \]
### Decorrelation and Independence

- An example with discrete variables
- Are they independent?

<table>
<thead>
<tr>
<th></th>
<th>$x = -1$</th>
<th>$x = 0$</th>
<th>$x = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y = -1$</td>
<td>0</td>
<td>$\frac{1}{4}$</td>
<td>0</td>
</tr>
<tr>
<td>$y = 0$</td>
<td>$\frac{1}{4}$</td>
<td>0</td>
<td>$\frac{1}{4}$</td>
</tr>
<tr>
<td>$y = 1$</td>
<td>0</td>
<td>$\frac{1}{4}$</td>
<td>0</td>
</tr>
</tbody>
</table>
Decorrelation and Independence

- An example with discrete variables
  - Are they independent?

<table>
<thead>
<tr>
<th></th>
<th>x == −1</th>
<th>x == 0</th>
<th>x == 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>y == −1</td>
<td>0</td>
<td>1/4</td>
<td>0</td>
</tr>
<tr>
<td>y == 0</td>
<td>1/4</td>
<td>0</td>
<td>1/4</td>
</tr>
<tr>
<td>y == 1</td>
<td>0</td>
<td>1/4</td>
<td>0</td>
</tr>
</tbody>
</table>

- \(x, y\) are not statistically independent

\[
E\{x^2 y^2\} = 0 \neq E\{x^2\}E\{y^2\} = \frac{1}{4}
\]
The signals version

- Decorrelated?

\[ x = \sin(t) \]
\[ y = \sin(2t) \]
The signals version

- Decorrelated?
  \[ x = \sin(t) \]
  \[ y = \sin(2t) \]
- Yes: \( E\{xy\} = 0 \)
  - But I can predict one from the other
  - 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:

\[ y = W \cdot x \]

\[ P(y_i, y_j) = P(y_i)P(y_j), \forall i, j \]
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

\[
\begin{align*}
\Delta W & \propto \left( D - f(y_i) \cdot g(y_j^\top) \right) \cdot W \\
W & = W + \mu \Delta W
\end{align*}
\]

\[
D = \begin{bmatrix}
d_1 & 0 \\
\vdots & \ddots \\
0 & d_n
\end{bmatrix}
\]

\( f(x), g(x) \) can be \( \tanh(x), x^3, \ldots \)
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 (a tensor!)
    $$Q(y)_{i,j,k,l} = \kappa_4(y_i, y_j, y_k, y_l) = E\{y_i y_j y_k y_l\} -$$
    $$E\{y_i y_j\} E\{y_k y_l\} - E\{y_i y_k\} E\{y_j y_l\} - E\{y_i y_l\} E\{y_j y_k\}$$
  - confused yet?
Approach 2

• Conceptually we perform a tensor singular value decomposition

• Comon’s algorithm
  • 1) Do PCA
    • Imposes decorrelation (halfway there)
  • 2) Find unitary transform that minimizes fourth order cross-cumulants
Approach 3

- Information theoretic optimization

  - Minimize mutual information: \( I(y) = \sum H(y_k) - H(y) \)

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

- Iterative rule: \( \Delta W \propto (I - f(y) \cdot y^T) \cdot W \)

  - 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?

- As usual, no good answer ...
- Algebraic algorithms
  - HSVD, cumulant tensors, etc.
  - Computationally demanding
- Iterative algorithms
  - Non-linear decorrelation, infomax, etc
  - Small, fast, but prone to blowups
- FastICA
  - Fixed-point algorithm
  - Quite robust and reliable
So what does ICA do?

- 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 rotation highlights 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
ICA issues

• Most estimators are approximate
  • The resulting output is not necessarily the correct one

• There might not be independence
  • ICA returns a maximally independent projection, not an independent one
    • Again the output might not be what you expected to get!
ICA limitations

- Invariance to output permutations

\[ P(y_1, y_2, y_3) = P(y_1)P(y_2)P(y_3) = P(y_2)P(y_1)P(y_3) = \ldots \]

  - Output order is not guaranteed and can differ through runs

- No sense of ordering of components
  - PCA orders outputs in terms of 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 “code”
  • PCA “code” can still have statistical dependencies

• PCA features and projection are decorrelated
  • There is no constraint on the ICA features
  • Only the decomposition output is independent
Analysis vs. synthesis features

• One more distinction to make

• PCA features are “bi-directional”
  \[ z = W \cdot x \]
  \[ \hat{x} = W^\top \cdot z \]

• That won’t hold anymore
  • We have analysis features: \[ z = W \cdot x \]
  • And synthesis features: \[ \hat{x} = W^+ \cdot z \]
Be careful when combining the two!

• If we want both dimensionality reduction and independence
  • Step 1: Do PCA to reduce the dimensions
    \[ Z_p = W_p \cdot X, \quad X \in \mathbb{R}^{M \times N}, \quad W_p \in \mathbb{R}^{K \times M}, \quad Z_p \in \mathbb{R}^{K \times N} \]
  • Step 2: Do ICA on the PCA weights to produce independence
    \[ Z_I = W_I \cdot Z_p, \quad W_I \in \mathbb{R}^{K \times K}, \quad Z_I \in \mathbb{R}^{K \times N} \]

• What’s what?
  • Analysis features: \[ Z_I = \left( W_I \cdot W_p \right) \cdot X \Rightarrow W = W_I \cdot W_p, \quad W \in \mathbb{R}^{K \times M} \]
  • Synthesis features: \[ \hat{X} = \left( W_I \cdot W_p \right)^+ \cdot Z_I \]
Example features from sounds

• Obtain lots and lots of natural sounds
  • E.g. sounds found in nature, birds, walking on leaves, etc.

• Place short windows in a large matrix
  • and do PCA and ICA

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

• We know that PCA results in sinusoids
Example features from sounds
Same with images

- ICA components look a lot like the V1 receptive fields!
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 work relaxes the Gaussian assumption and assumes a “heavy-tailed” world
  - This is more like the world we live
  - This was a big revelation in machine learning!
Non-Negative Matrix Factorization

• A 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
Example case

• The Iris data set
  • Each row is a size measurement (i.e. positive)

<table>
<thead>
<tr>
<th>Iris Setosa</th>
<th>Iris Versicolour</th>
<th>Iris Virginica</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sepal length in cm</td>
<td>Sepal width in cm</td>
<td>Petal length in cm</td>
</tr>
<tr>
<td>Petal width in cm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PCA/ICA analysis on iris data

- Both give features that are partly negative
  - What does that mean?

- **PCA features**
- **ICA features**
Same with eigenfaces

- “Negative” images as bases – why??
Obtaining non-negative features

- Define the factorization problem
  \[ X \approx W \cdot H \]
  \[ X \in \mathbb{R}^{M\times N, \geq 0}, \quad W \in \mathbb{R}^{M\times R, \geq 0}, \quad H \in \mathbb{R}^{R\times N, \geq 0} \]
  - This is similar to the PCA/ICA setup
    - \( R \) 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 $W$
  - estimate an $H$ given $W$
  - estimate a new $W$ given $H$
  - repeat until convergence
Solving for one factor

- The problem is simpler
  - Only one unknown

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

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

- 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 $W$
  • estimate new $H$ given $W$: $H = W^+ \cdot X$
    $H = \max(H, 0)$
  • estimate new $W$ given $H$: $W = X \cdot H^+$
    $W = \max(W, 0)$
• repeat until convergence
Conceptual problem

- We don’t want to use pseudoinverses
  - They imply least-squares minimization
    - Least squares imply Gaussian data
      - We don’t have Gaussian data ...

- We define a special distance
  - A variant of KL divergence

\[
\min_{w, h} \left[ \sum_{i, j} x_{i, j} \log \frac{x_{i, j}}{(w \cdot h)_{i, j}} - x_{i, j} + (w \cdot h)_{i, j} \right]
\]
Multiplicative updates

- Using some optimization magic we get:

\[
W_{i,j} = W_{i,j} \sum_k \frac{X_{i,k}}{(W \cdot H)_{i,k}} H_{j,k}
\]

\[
H_{j,k} = H_{j,k} \sum_i W_{i,j} \frac{X_{i,k}}{(W \cdot H)_{i,k}}
\]

- Significantly faster operations
  - Just matrix and scalar multiplications
    - No inversions
An example

- Start with input $X$
- NMF will decompose as $X \approx W \cdot H$
- The columns of $W$ will contain “vertical” information about $X$
- The rows of $H$ will contain “horizontal” information about $X$
Back to the iris data

- NMF on iris provides interpretable results
  - We see the structure
  - The features are meaningful as sizes

- PCA/ICA features
  - Not so useful
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 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
  - Immense 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, 80 × 60 pixel frames
  - Data will be 4800 × 126

- Using different analyses
  - PCA, ICA, NMF
  - Compare features and weights
PCA Results

- Nothing special about the visual components
- They are orthogonal pictures
  - Does this mean anything? (not really ...)
  - Some segmentation between constant vs. moving parts
- Some highlighting of the action in the weights
ICA Results

- Much more interesting visual components
- They are independent
  - Unrelated elements (l/r hands, background) are now highlighted
  - We have a decomposition by parts
- Component weights are now describing the scene
NMF Results

- A different take on the visual components
- We don’t know how they relate, but ...
  - They describe some of the possible states of the video
  - Perhaps a more semantically meaningful representation
- Component weights are as vague as with PCA (because we have more components than we need)
If we use the right dimensions

- The results look exactly as we would want them!
Audio Visual Components?

• We can even take in both audio and video data and try to find structure.
• Sometimes there is a very strong correlation between auditory and visual elements.
• We should be able to discover that automatically.
What does the data look like?

57,600 pixel dimensions

257 audio dimensions

Time

0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5

x 10^4

56
Audio/Visual PCA components

Audio component 1

Audio component 2

Audio component 3

Audio component 4

Audio component 5

Audio component 6

Video component 1

Video component 2

Video component 3

Video component 4

Video component 5

Video component 6

Weights

Time

Audio/Visual PCA components
Audio/Visual ICA components

Audio component 1
Audio component 2
Audio component 3
Audio component 4
Audio component 5
Audio component 6

Video component 1
Video component 2
Video component 3
Video component 4
Video component 5
Video component 6

Weights

Time

Audio/Visual ICA components – FALL 2015

Machine Learning for Signal Processing
Audio/Visual NMF components

Audio component 1
Audio component 2
Audio component 3
Audio component 4
Audio component 5
Audio component 6

Video component 1
Video component 2
Video component 3
Video component 4
Video component 5
Video component 6

Weights

Time

C1
C2
C3
C4
C5
C6
Audio/Visual NMF components

Audio component 1

Video component 1

Audio component 2

Video component 2

Audio component 3

Video component 3

Audio component 4

Video component 4

Audio component 5

Video component 5

Weights

Time

C1
C2
C3
C4
C5
PCA, ICA or NMF?

- Depends on what you want to do
  - PCA does a fantastic job in dimensionality reduction
  - ICA provides a clean 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

• Last on features for a while

• Non-linear methods
  • What do do when your data looks really strange

• Manifolds and embedding
  • Finding latent structure in high dimensions
Reading

- Textbook sections 6.5-6.6
- Independent Component Analysis (optional)
- Natural stimuli statistics (optional)
- Non-negative Matrix Factorization (optional)