Integral and Discrete Transforms in Image Processing

Selected Orthogonal Transforms

David Svoboda

email: svoboda@fi.muni.cz
Centre for Biomedical Image Analysis
Faculty of Informatics, Masaryk University, Brno, CZ

CBIA

August 24, 2022

Outline

Definition:

Let $C \subset \mathbb{R}^{n \times n}$ be a square matrix. Vector $\mathbf{v} \in \mathbb{C}^n$ is an *eigenvector* of $C \Leftrightarrow \exists \lambda \in \mathbb{C} : C\mathbf{v} = \lambda\mathbf{v}$. λ is called an *eigenvalue*.

Properties of eigenvalues:

- $C\mathbf{v} = \lambda\mathbf{v}$ equals to $(C - \lambda E)\mathbf{v} = 0$

  The solution of equation $|C - \lambda E| = 0$ is identical to the search for roots in the polynomial of degree n.
- if C is symmetric then all the eigenvalues are real

Properties of eigenvectors:

- the two eigenvectors corresponding to two different eigenvalues are orthogonal
- if C is symmetric then all the eigenvectors are real and orthogonal

Properties of symmetric matrices:

- if $C \subset \mathbb{R}^{n \times n} : C = C^T$ with its eigenvectors $e_1, e_2, \ldots, e_n$ and eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ then $\exists A \subset \mathbb{R}^{n \times n}$ such that $A^T C A = D$, where

$$D = \begin{bmatrix} \lambda_1 & 0 & 0 & \ldots & 0 \\ 0 & \lambda_2 & 0 & \ldots & 0 \\ 0 & 0 & \lambda_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & \lambda_n \end{bmatrix} \quad \text{and} \quad A = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_n \end{bmatrix}$$

Cross-correlation:

- given two 1D signals $f(i)$ and $g(i)$:

$$(f \star g)(i) = \sum_k f(i+k)g(k)$$

- it is a measure of similarity of two signals

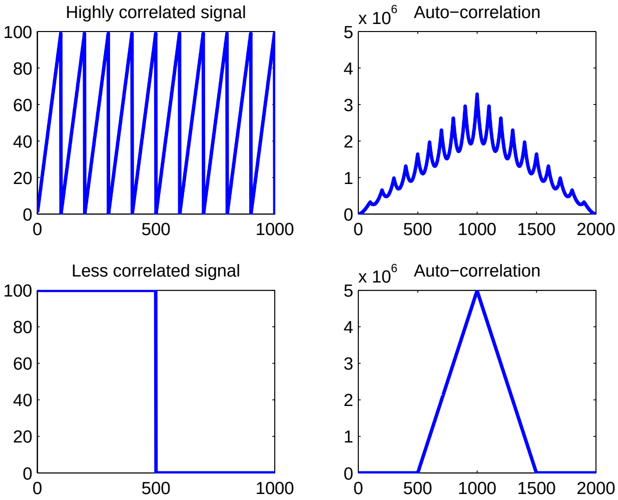
Auto-correlation:

- given 1D signal $f(i)$:

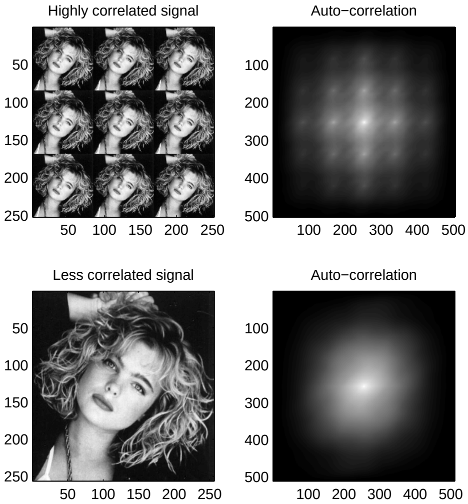
$$(f \star f)(i) = \sum_k f(i+k)f(k)$$

- it is a measure for finding repeating patterns

Auto-correlation example



Auto-correlation example



Highly correlated signals

- contain repeating patterns
- energy (nonzero values) is stretched over the whole space
- this is what we usually have

Decorrelated signals

- energy (nonzero values) is compacted in one location
- easy to compress
- this is what we wish to have

Covariance

- Covariance exhibits how much two signals f and g (let us assume $|f| = |g| = n$) vary from the mean with respect to each other:

$$cov(f, g) = \frac{\sum_{i=1}^{n} (f_i - \overline{f})(g_i - \overline{g})}{n - 1}$$

- Covariance Matrix – a matrix of covariances between two signals f and g:

$$C = \begin{bmatrix} cov(f, f) & cov(f, g) \\ cov(g, f) & cov(g, g) \end{bmatrix}$$

Matrix C is always real and symmetric.

Notice: Two signals f and g are decorrelated iff $cov(f, g) = cov(g, f) = 0$, i.e. when the matrix C is diagonal.

Motivation

The most common linear transforms

- sinusoidal transforms
  - DFT (discrete Fourier transform)
  - DCT (discrete cosine transform)
  - DST (discrete sine transform)
- rectangular wave transforms
  - Walsh-Hadamard transform
  - Haar transform
- variable basis
  - Discrete wavelet transform
- eigenvector-based transforms
  - Karhunen-Loeve transform

Motivation

Energy compaction

Evaluate DFT over some short signal:

$$\begin{aligned} f &= [1\ 3\ 4\ 2] \\ &\Downarrow \quad /\text{DFT}/ \\ \mathcal{F} &= \frac{1}{\sqrt{4}}[10 \quad (-3-i) \quad 0 \quad (-3+i)] \end{aligned}$$

Measure the energy of the signals:

$$\begin{aligned} E(f) &= \sum f(i)^2 = 1^2 + 3^2 + 4^2 + 2^2 = 30 \\ E(\mathcal{F}) &= \sum \mathcal{F}(i)^2 = 25 + 10/4 + 10/4 = 30 \end{aligned}$$

The first component in f accounts for 3.3% of energy while the first component in $\mathcal{F}$ accounts for 83.3% of energy.

Conclusion: Ability of energy compaction $\approx$ optimality of the transform

Motivation

Why do we need image transforms?

- manipulation with data in another domain might be simpler
  example of use: low-pass, high-pass filtering
- data decorrelation $\approx$ co-variance removal $\approx$ energy compaction
  example of use: image compression

Which transform properties are the most important?

- speed
- simple to implement

Notice: The requirements suggest to use *linear transforms*, i.e. the input signal f is transformed into the signal F by:

$$F = Af,$$

where A is a *transform matrix*.

PCA-based Transform

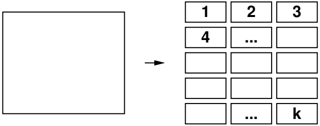
Design of transform matrix A

Task to solve

Given an input discrete signal, design a transformation matrix such that the transformed signal has decorrelated samples (they are mutually independent).

Solution

Given a 2D image, let us break it up into k blocks of n pixels each.



Each block i is characterized with its vector $\mathbf{b}^{(i)}, i = 1, 2, \ldots, k$ where length$(\mathbf{b}^{(i)}) = n$.

input (correlated)	input (mean centered)	expected output (decorrelated)
$\mathbf{b}^{(i)}$	$\mathbf{v}^{(i)} = \mathbf{b}^{(i)} - \overline{\mathbf{b}}$	$\mathbf{w}^{(i)} = A\mathbf{v}^{(i)}, i = 1, 2, \ldots, k$
	$V = \left[\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(k)}\right]$	$W = AV = \left[\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(k)}\right]$
	$C_V = V \cdot V^T$	$C_W = W \cdot W^T$
	C_V ... real, symmetric	C_W ... diagonal

$$C_W = W \cdot W^T = (AV) \cdot (AV)^T = A(V \cdot V^T)A^T = A \cdot C_V \cdot A^T$$

Notice:

- the off-diagonal elements of covariance matrix C_V are the covariances of the $\mathbf{v}^{(i)}$ vectors ... $(V \cdot V^T)_{ab} = \sum_{i=1}^{k} v_a^{(i)} v_b^{(i)}$
- the off-diagonal elements of covariance matrix C_W are zero
- the eigenvectors of C_V form the rows of a new matrix A
- C_W is a diagonal matrix formed of the eigenvalues of C_V

PCA-based Transform

Algorithm:

1. collect the input data $\mathbf{b}^{(i)}$
2. form the matrix V
3. calculate the covariance matrix C_V
4. find eigenvectors and eigenvalues of C_V
5. use eigenvectors to form the transform matrix A
6. use A as a transform matrix to original data
7. get transformed decorrelated data: $\mathbf{w}^{(i)} = A(\mathbf{b}^{(i)} - \overline{\mathbf{b}})$

Notice: Red lines $\equiv$ Principal Component Analysis (PCA).

PCA-based Transform

An example

Let us submit the following 2D image to the PCA

2.5	**2.4**	**0.5**	**0.7**
2.2	**2.9**	**1.9**	**2.2**
3.1	**3.0**	**2.3**	**2.7**
2.0	**1.6**	**1.0**	**1.1**
1.5	**1.6**	**1.1**	**0.9**

Notice: Neighbouring pixels have usually similar value.

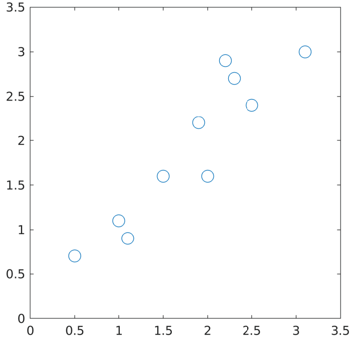
PCA-based Transform

An example

We have chosen $n = 2, k = 10$ and fetched k vectors $\mathbf{b}^{(i)}, i = 1, 2, \ldots, k$ in the following manner:

	X	Y
$\mathbf{b}^{(1)}$	2.5	2.4
$\mathbf{b}^{(2)}$	0.5	0.7
$\mathbf{b}^{(3)}$	2.2	2.9
$\mathbf{b}^{(4)}$	1.9	2.2
$\mathbf{b}^{(5)}$	3.1	3.0
$\mathbf{b}^{(6)}$	2.3	2.7
$\mathbf{b}^{(7)}$	2.0	1.6
$\mathbf{b}^{(8)}$	1.0	1.1
$\mathbf{b}^{(9)}$	1.5	1.6
$\mathbf{b}^{(10)}$	1.1	0.9

Correlation of the neighbouring intensities:



PCA-based Transform

An example

Modify the input datasets:

	X	Y		$X - \overline{X}$	$Y - \overline{Y}$
$\mathbf{b}^{(1)}$	2.5	2.4	$\mathbf{v}^{(1)}$	0.69	0.49
$\mathbf{b}^{(2)}$	0.5	0.7	$\mathbf{v}^{(2)}$	-1.31	-1.21
$\mathbf{b}^{(3)}$	2.2	2.9	$\mathbf{v}^{(3)}$	0.39	0.99
$\mathbf{b}^{(4)}$	1.9	2.2	$\mathbf{v}^{(4)}$	0.09	0.29
$\mathbf{b}^{(5)}$	3.1	3.0	$\mathbf{v}^{(5)}$	1.29	1.09
$\mathbf{b}^{(6)}$	2.3	2.7	$\mathbf{v}^{(6)}$	0.49	0.79
$\mathbf{b}^{(7)}$	2.0	1.6	$\mathbf{v}^{(7)}$	0.19	-0.31
$\mathbf{b}^{(8)}$	1.0	1.1	$\mathbf{v}^{(8)}$	-0.81	-0.81
$\mathbf{b}^{(9)}$	1.5	1.6	$\mathbf{v}^{(9)}$	-0.31	-0.31
$\mathbf{b}^{(10)}$	1.1	0.9	$\mathbf{v}^{(10)}$	-0.71	-1.01

PCA-based Transform

An example

Evaluate all the covariances $C_V(X, X)$, $C_V(X, Y)$, $C_V(Y, X)$, and $C_V(Y, Y)$ and form the covariance matrix C_V:

$$C_V = V \cdot V^T = \begin{bmatrix} 0.617 & 0.615 \\ 0.615 & 0.717 \end{bmatrix}$$

Since the covariance matrix is square, we can calculate the eigenvectors and eigenvalues for this matrix:

- eigenvalues

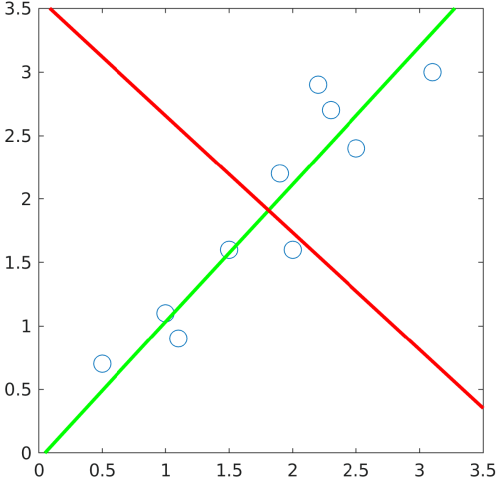
$$C_W = W \cdot W^T = \begin{pmatrix} 0.049 & 0 \\ 0 & 1.284 \end{pmatrix}$$

- eigenvectors

$$A = \begin{pmatrix} -0.735 & 0.678 \\ -0.678 & -0.735 \end{pmatrix}$$

PCA-based Transform

An example



PCA-based Transform

An example

The following statements are equal:

- the eigenvalue λ_j is the highest one
- the eigenvector e_j is more dominant
- the energy (the majority in information) is gathered in element $\mathbf{w}_j^{(i)}, i = 1, 2, \ldots, k$
- the element $\mathbf{w}_j^{(i)}, i = 1, 2, \ldots, k$ has the greatest variance

The following statements are also equal:

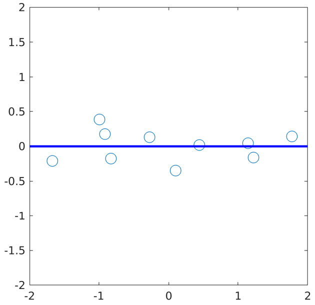
- the eigenvalue λ_l is the lowest one
- the eigenvector e_l is practically useless

PCA-based Transform

An example

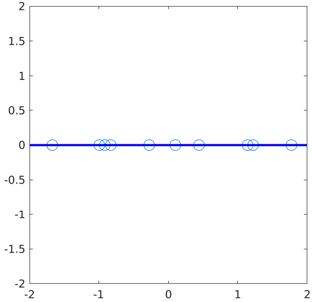
Transformed (decorrelated) data

	X'	Y'
$\mathbf{w}^{(1)}$	-0.827	-0.175
$\mathbf{w}^{(2)}$	1.777	0.142
$\mathbf{w}^{(3)}$	-0.992	0.384
$\mathbf{w}^{(4)}$	-0.274	0.130
$\mathbf{w}^{(5)}$	-1.675	-0.209
$\mathbf{w}^{(6)}$	-0.912	0.175
$\mathbf{w}^{(7)}$	0.099	-0.349
$\mathbf{w}^{(8)}$	1.144	0.046
$\mathbf{w}^{(9)}$	0.438	0.017
$\mathbf{w}^{(10)}$	1.223	-0.162



Clear less important component

	X'	Y'
$\mathbf{w}^{(1)}$	-0.827	0.000
$\mathbf{w}^{(2)}$	1.777	0.000
$\mathbf{w}^{(3)}$	-0.992	0.000
$\mathbf{w}^{(4)}$	-0.274	0.000
$\mathbf{w}^{(5)}$	-1.675	0.000
$\mathbf{w}^{(6)}$	-0.912	0.000
$\mathbf{w}^{(7)}$	0.099	0.000
$\mathbf{w}^{(8)}$	1.144	0.000
$\mathbf{w}^{(9)}$	0.438	0.000
$\mathbf{w}^{(10)}$	1.223	0.000



PCA-based Transform

An example

Transformed back with A^{-1}

	oldX	*oldY*	*newX*	*newY*
$\mathbf{b}^{(1)}$	2.5	2.4	2.42	2.47
$\mathbf{b}^{(2)}$	0.5	0.7	0.50	0.71
$\mathbf{b}^{(3)}$	2.2	2.9	2.54	2.58
$\mathbf{b}^{(4)}$	1.9	2.2	2.01	2.09
$\mathbf{b}^{(5)}$	3.1	3.0	3.04	3.05
$\mathbf{b}^{(6)}$	2.3	2.7	2.48	2.53
$\mathbf{b}^{(7)}$	2.0	1.6	1.74	1.84
$\mathbf{b}^{(8)}$	1.0	1.1	0.97	1.13
$\mathbf{b}^{(9)}$	1.5	1.6	1.49	1.61
$\mathbf{b}^{(10)}$	1.1	0.9	0.91	1.08

PCA-based Transform

Properties:

- optimal decorrelation method (see the example)
- too heavy for evaluation (searching for eigenvalues and eigenvectors)
- matrix A is data dependent (cannot be precomputed)
- all the eigenvectors must be kept for inverse transform

Conclusion: rather theoretical method

Let $f = [f(0) \quad f(1) \ \dots \ f(N-1)]$ be a discrete signal.

Let us modify it as follows:

$$f(0) \quad f(1) \quad f(2) \quad \dots \quad f(N-1)$$

$\Downarrow$ mirror

$$f(N-1) \ \dots \ f(1)\ f(0) \qquad f(0)\ f(1)\ f(2)\ \dots\ f(N-1)$$

$\Downarrow$ insert zeros

$$f(N-1)\ 0\ \dots\ 0\ f(1)\ 0\ f(0) \qquad 0 \qquad f(0)\ 0\ f(1)\ 0\ f(2)\ \dots\ 0\ \dots\ f(N-1)\ 0$$

$\Downarrow$ DFT domain is periodical

$$0 \qquad f(0)\ 0\ f(1)\ 0\ f(2)\ \dots\ 0\ \dots\ f(N-1)\ 0\ f(N-1)\ 0\ \dots\ 0\ f(1)\ 0\ f(0)$$

$\Downarrow$ name substitution

$$c(0) \qquad c(1) \quad c(2) \quad c(3) \quad c(4) \quad \dots \quad c(4N-1)$$

The relationship between signal 'c' and 'f':

$$
\begin{aligned}
c(2n) &= 0 \quad \text{iff} \quad 0 \leq n < N \\
c(2n+1) &= f(n) \quad \text{iff} \quad 0 \leq n < N \\
c(4N-n) &= c(n) \quad \text{iff} \quad 0 < n < 2N
\end{aligned}
$$

The basic properties of signal 'c':

- c is even (ready for DFT working over real even data)
- $|c| = 4N$

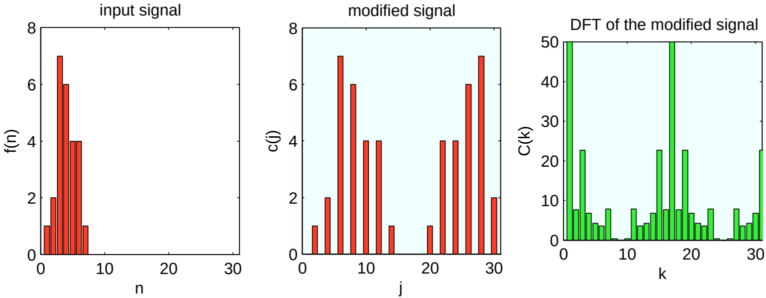
Let us apply DFT to c:

$$\begin{aligned}
\mathcal{C}(k) &= \sum_{j=0}^{4N-1} c(j) e^{-\frac{2\pi ijk}{4N}} = /\text{symmetry}/ = \sum_{j=0}^{2N-1} c(j) \left[e^{-\frac{2\pi ijk}{4N}} + e^{-\frac{2\pi i(4N-j)k}{4N}} \right] \\
&= \sum_{j=0}^{2N-1} c(j) \left[e^{-\frac{2\pi ijk}{4N}} + e^{\frac{2\pi ijk}{4N}} \right] /\text{Euler-Moivre eq.}/ \\
&= \sum_{j=0}^{2N-1} c(j) \left[\left(\cos \frac{2\pi jk}{4N} - i \sin \frac{2\pi jk}{4N} \right) + \left(\cos \frac{2\pi jk}{4N} + i \sin \frac{2\pi jk}{4N} \right) \right] \\
&= \sum_{j=0}^{2N-1} c(j) \left[2 \cos \frac{2\pi jk}{4N} \right] \\
&= \dots
\end{aligned}$$

$$
\begin{aligned}
\mathcal{C}(k) &= \sum_{j=0}^{2N-1} c(j)\left[2\cos\frac{2\pi jk}{4N}\right] \quad \text{/separating odd and even items/} \\
&= \sum_{l=0}^{N-1} 2c(2l)\left[\cos\frac{2\pi 2lk}{4N}\right] \quad \text{/j=2l, } l \in \mathbb{N}\text{/} \\
&\quad + \sum_{l=0}^{N-1} 2c(2l+1)\left[\cos\frac{2\pi(2l+1)k}{4N}\right] \quad \text{/j=2l+1, } l \in \mathbb{N}\text{/} \\
&= \sum_{l=0}^{N-1} 2f(l)\left[\cos\frac{(2l+1)\pi k}{2N}\right]
\end{aligned}
$$

Signal $\mathcal{C}(k)$ is:

- even, because 'c' is even
- twice replicated, because 'c' is stretched by zeros



Notice: Only the coefficients $\mathcal{C}(k), k = \{0, 1, 2, \ldots, N-1\}$ are used.

This version of DCT is also known as DCT-II.

Forward 1D-DCT:

$$\mathcal{C}(k) = \alpha(k) \sum_{l=0}^{N-1} f(l) \left[\cos \frac{(2l+1)\pi k}{2N} \right], \quad k = \{0, 1, 2, \ldots, N-1\}$$

where

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & \text{iff} \quad k = 0 \\ \sqrt{\frac{2}{N}} & \text{iff} \quad k \neq 0 \end{cases}$$

Inverse 1D-DCT:

$$f(l) = \sum_{k=0}^{N-1} \alpha(k) \mathcal{C}(k) \left[\cos \frac{(2l+1)\pi k}{2N} \right], \quad l = \{0, 1, 2, \ldots, N-1\}$$

Basic properties:

- basis formed of sampled cosine waves only
- no complex numbers

Fast Discrete Cosine Transform (F-DCT)

Derivation

Let us recombine the input signal:

$$
\begin{aligned}
y(l) &= f(2l) \\
y(N-1-l) &= f(2l+1) \quad (l = 0, \ldots, N/2-1)
\end{aligned}
$$

Example: `f = [1 2 3 4 5 6 7 8]` $\rightarrow$ `y = [1 3 5 7 8 6 4 2]`

Applying DCT on signal f we can deduce:

$$
\begin{aligned}
\mathcal{C}(k) &= \alpha(k) \sum_{l=0}^{N-1} f(l) \cos\left(\frac{(2l+1)\pi k}{2N}\right) \quad /f \rightarrow y/ = \cdots = \\
&= \alpha(k) \sum_{l=0}^{N-1} y(l) \cos\left(\frac{(4l+1)\pi k}{2N}\right)
\end{aligned}
$$

Let us apply DFT on signal y:

$$\begin{aligned}
\mathcal{Y}(k) &= \sum_{l=0}^{N-1} y(l) e^{\frac{-2\pi ikl}{N}} \\
&= \sum_{l=0}^{N-1} y(l)\left[\cos\frac{2\pi kl}{N} - i\sin\frac{2\pi kl}{N}\right] \; / \cdot e^{-\frac{\pi ik}{2N}} \; / \\
\text{Real}\left[e^{-\frac{\pi ik}{2N}}\mathcal{Y}(k)\right] &= \sum_{l=0}^{N-1} y(l)\left[\cos\frac{2\pi kl}{N}\cos\frac{k\pi}{2N} - \sin\frac{2\pi kl}{N}\sin\frac{k\pi}{2N}\right] \\
&= \sum_{l=0}^{N-1} y(l)\cos\frac{(4l+1)k\pi}{2N} = \mathcal{C}(k)/\alpha(k) \\
\text{Imag}\left[e^{-\frac{\pi ik}{2N}}\mathcal{Y}(k)\right] &= \text{omitted}
\end{aligned}$$

$$\mathcal{C}(k) = \alpha(k)\text{Real}\left[e^{-\frac{\pi ik}{2N}}\mathcal{Y}(k)\right]$$

Fast Discrete Cosine Transform (F-DCT)

Algorithm

1. recombine input sequence f of length N to get y:

$$\begin{aligned} y(l) &= f(2l) \\ y(N-1-l) &= f(2l+1) \quad (l = 0, \ldots, N/2-1) \end{aligned}$$

2. apply FFT to y:

$$\mathcal{Y} = \mathrm{FFT}(y)$$

3. for each $k = 0, \ldots, N-1$ do:
   1. multiply the k-th Fourier coefficient by factor $e^{-\frac{\pi ik}{2N}}$:

   $$\mathcal{Y}'(k) = e^{-\frac{\pi ik}{2N}} \mathcal{Y}(k)$$

   2. fetch only real part from each Fourier coefficient and normalize the results:

   $$\mathcal{C}(k) = \alpha(k) \mathrm{Real} \left[\mathcal{Y}'(k) \right]$$

2D Discrete Cosine Transform (2D-DCT)

Definition

Forward 2D-DCT:

$$\mathcal{C}(u,v) = \alpha(u)\alpha(v)\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} f(k,l)\left[\cos\frac{(2k+1)\pi u}{2N}\cos\frac{(2l+1)\pi v}{2N}\right]$$

$$\text{where} \quad u,v = \{0,1,2,\ldots,N-1\}$$

Inverse 2D-DCT:

$$f(k,l) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} \alpha(u)\alpha(v)\mathcal{C}(u,v)\left[\cos\frac{(2k+1)\pi u}{2N}\cos\frac{(2l+1)\pi v}{2N}\right]$$

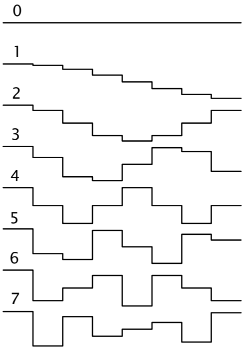
Basic properties:

- 2D-DCT it is simply an extension of 1D-DCT
- separable

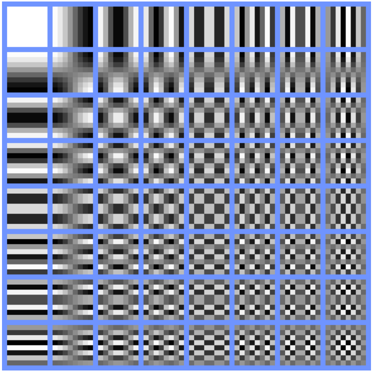
2D Discrete Cosine Transform (2D-DCT)

Basis functions

1D-DCT
8 basis 1D functions
$(N = 8)$



2D-DCT
64 basis 2D functions
$(N \times N = 8 \times 8)$



Discrete Cosine Transform (DCT)

Properties

- Almost as efficient as PCA in term of decorrelation optimality.
- The transform matrix can be easily prepared without having the data.
- Unlike DFT, DCT works with real data.
- As its is derived directly from FFT, there exists fast alternative for DCT.
- Regarding its construction, it works with symmetric data.

Walsh-Hadamard Transform

Jacques Salomon Hadamard (1865 – 1963)

Walsh-Hadamard Transform

Similarly to DFT we can define Hadamard matrix which defines the transform:

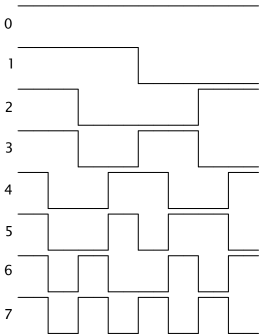
$$
\begin{aligned}
H_m &= \frac{1}{\sqrt{2}} \begin{pmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{pmatrix} \\
H_1 &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\
H_0 &= +1
\end{aligned}
$$

An example:

$$
H_2 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}
$$

Walsh-Hadamard Transform

An 8-samples long Walsh functions:



Walsh-Hadamard Transform

Formal definition:

$$\mathcal{H}(k) = \frac{1}{N} \sum_{n=0}^{N-1} f(n) \prod_{i=0}^{m-1} (-1)^{B(m,i,n,k)}$$

with $k \in \{0, 1, \ldots, N-1\}$, $N = 2^m$, and $B(m, i, n, k) = b_i(n) b_{m-1-i}(k)$, where $b_k(v)$ denotes the k-th bit in the binary representation of a non-negative integer v.

Notice: Similarly to FT we can define inverse (IWHT) or fast (FWHT) Walsh-Hadamard transform.

Walsh-Hadamard Transform

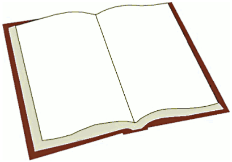
Properties:

- Unlike to DCT the basis functions contain plus and minus ones only.
- So-called *Walsh functions* are used as a basis functions.
- Any two basis functions are orthogonal.
- In real space only (like DCT).
- Worse approximation of PCA than DCT.
- Wide application in digital communications.
- One can implement the WHT on smaller, cheaper hardware.

Notice: Magnitude in WHT is affected by phase shifts in the signal! Typically, orthogonality is broken.

Bibliography

- Gonzalez, R. C., Woods, R. E. Digital image processing / 2nd ed., Upper Saddle River: Prentice Hall, c2002, pages 793, ISBN 0201180758
- Klette R., Zamperoni P. Handbook of Image Processing Operators, Wiley, 1996, ISBN-0471956422
- Salomon D. Data Compression, The Complete Reference, 4th edition, Springer, London, 2007, ISBN-1846286025



You should know the answers . . .

- Explain the difference between correlated and decorrelated signal.
- How does the PCA decorrelate the given signal?
- How do we get the PCA transformation matrix A in practice?
- What is the content of matrix A used in PCA transform?
- Provide your own example (different from the examples presented in the lecture) suitable for PCA transform.
- Explain the relationship between DFT and DCT.
- Describe the F-DCT algorithm and compute F-DCT(`[1 6 6 1]`).
- Analyze the ability to compact the energy for the following transforms: PCA, DFT, DCT, WHT