Outline

Gegeben sei ein Signal in der Form seiner N Abtastwerte als Vektor

$$\mathbf{x} = [x_0, x_1, \ldots, x_{N-1}]^T .$$

Wir "*zerlegen*" das Signal in Kosinus-Schwingungen:

$$y_n = c_n \sum_{j=0}^{N-1} \cos\left(\frac{\pi n (2j+1)}{2N}\right) x_j, \quad n = 0, 1, \ldots, N-1 \text{ mit } c_n = \begin{cases} \frac{1}{\sqrt{N}}, & n = 0 \\ \sqrt{\frac{2}{N}}, & n > 0 \end{cases}$$

Dann ist das (Kosinus-) transformierte Signal der Vektor $\mathbf{y} = [y_0, y_1, \ldots, y_{N-1}]^T$.

In Matrixschreibweise hat man

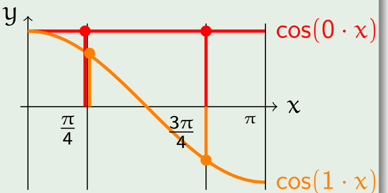
$$\mathbf{y} = \mathbf{C}\mathbf{x},$$

wobei $\mathbf{C}$ die Transformationsmatrix ist mit den Elementen $C_{n,j} = c_n \cos\left(\frac{\pi n (2j+1)}{2N}\right)$.

1D DCT

Example ($N = 2$)

Für $N = 2$ hat man die folgenden vier Kosinus-Schwingungen:



$$C_{n,j} = c_n \cos\left(\frac{\pi n\,(2j+1)}{2N}\right) \quad \text{wobei } n = 0,\, 1,\; j = 0,\, 1.$$

Setzt man noch $c_0 = \dfrac{1}{\sqrt{2}}$ und $c_1 = \sqrt{\dfrac{2}{2}} = 1$ ein, erhält man die Kosinus-Transformationsmatrix

$$\mathbf{C} = \begin{array}{c} \\ n=0 \\ n=1 \end{array} \begin{array}{c} \begin{array}{cc} j=0 & \quad j=1 \end{array} \\ \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ \cos\left(\frac{\pi}{4}\right) & \cos\left(\frac{3\pi}{4}\right) \end{bmatrix} \end{array} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

Die Beziehung $\mathbf{y} = \mathbf{C}\mathbf{x}$ ist eine **lineare Abbildung**; denn es gilt

erstens $\quad \mathbf{C}(\mathbf{x}_1 + \mathbf{x}_2) = \mathbf{C}\mathbf{x}_1 + \mathbf{C}\mathbf{x}_2 \quad$ und

zweitens $\quad \forall \lambda \in \mathbb{R}, \mathbf{C}(\lambda \mathbf{x}) = \lambda \mathbf{C}\mathbf{x}.$

1D DCT (Fort.)

Example ($N = 2$)

Es ist nicht nur eine lineare Abbildung, es ist sogar eine orthogonale Abbildung, d.h. eine Abbildung, welche die Länge der Vektoren nicht ändert. Kontrollieren Sie das, indem Sie den Vektor $\mathbf{x} = [1, 1]^T$ transformieren.

Man kann zeigen, dass die DCT-Matrix $\mathbf{C}$ orthogonal ist, d.h. dass $C^{-1} = C^T$ gilt[a]. Zeigen Sie's für dieses $\mathbf{C}$!!

[a] verwenden Sie dazu die Formel für die Inverse $\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

1D DCT (Fort.)

Example ($N = 3$)

Schreiben Sie die Kosinus-Transformationsmatrix $\mathbf{C}$ für $N = 3$ auf und überprüfen Sie, ob $\mathbf{C}\mathbf{C}^{-1} = \mathbf{C}\mathbf{C}^{T} = \mathbf{1}$.

Orthogonale Matrizen lassen sich also problemlos und schnell invertieren!

1D DCT (Fort.)

Example (N beliebig)

Führen Sie die selbe Rechnung für beliebige N mit Hilfe von python/matlab/octave durch.

In [147]:
```
import numpy as np
from numpy import linalg as LA
```

In [148]:
```
N = 4
C = np.zeros(shape=(N,N))
# C
```

In [149]:
```
for i in range(N):
    for j in range(N):
        if i == 0:
            c_n = np.sqrt(1/N)
        else:
            c_n = np.sqrt(2/N)
        C[i][j] = c_n * np.cos((2*j+1)*i*np.pi/(2*N))

print(C)
```

```
[[ 0.5         0.5         0.5         0.5       ]
 [ 0.65328148  0.27059805 -0.27059805 -0.65328148]
 [ 0.5        -0.5        -0.5         0.5       ]
 [ 0.27059805 -0.65328148  0.65328148 -0.27059805]]
```

In [153]:
```
np.max(abs(LA.inv(C)-C.T))
```

Out[153]:
```
1.6653345369377348e-16
```

Im allgemeinen Fall, wenn $N \in \mathbb{N}$ beliebig ist, hat man:

$$\mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \sqrt{\frac{2}{N}} \begin{bmatrix} 1/\sqrt{2} & \cdots & 1/\sqrt{2} \\ \cos\frac{\pi}{2N} & \cdots & \cos\frac{\pi(2N-1)}{2N} \\ \vdots & \ddots & \vdots \\ \cos\frac{\pi(N-1)}{2N} & \cdots & \cos\frac{\pi(N-1)(2N-1)}{2N} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} = \mathbf{C}\mathbf{x}.$$

Wie sieht das Matrixelement $C_{n,j}$ aus? Schreiben Sie dann

$$y_n = c_n \sum_{j=0}^{N-1} C_{n,j} x_j, \qquad 0 \leqslant n \leqslant N-1.$$

Programmieren Sie die 1D-DCT in python/ocatve/matlab!

Outline

2D DCT (Fort.)

Wir stellen uns das Bild als eine $M \times N$-Matrix $\mathbf{X}$ vor. Dann wird die 1D-DCT zuerst zeilenweise auf das Bild angewandt:

$$\mathbf{T} = \mathbf{X}\mathbf{C}_N^T \quad \text{wobei} \quad (C_N)_{n,j} = c_{N,n} \cos\left(\frac{\pi n(2j+1)}{2N}\right) \quad \text{mit} \quad c_{N,n} \; = \begin{cases} \frac{1}{\sqrt{N}}, & n = 0 \\ \sqrt{\frac{2}{N}}, & 0 < n < N \end{cases}$$

Hier gilt $0 \leqslant n, j \leqslant N$. Danach spaltenweise auf das Resultat von vorher:

$$\mathbf{Y} = \mathbf{C}_M\mathbf{T} \quad \text{wobei} \quad (C_M)_{m,i} = c_{M,m} \cos\left(\frac{\pi m(2i+1)}{2M}\right) \quad \text{mit} \quad c_{M,m} \; = \begin{cases} \frac{1}{\sqrt{M}}, & m = 0 \\ \sqrt{\frac{2}{M}}, & 0 < m < M \end{cases}$$

Hier gilt $0 \leqslant m, i \leqslant M$. In Matrixschreibweise also

$$\mathbf{Y} = \mathbf{C}_M\mathbf{T} \; = \; \mathbf{C}_M\mathbf{X}\mathbf{C}_N^T$$

wobei $\mathbf{X}$ das Bild und $\mathbf{Y}$ dessen diskrete Kosinustransformierte (DCT) darstellt. Komponentenweise

$$Y_{m,n} = c_{M,m}c_{N,n} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \underbrace{\cos\left(\frac{\pi m(2i+1)}{2M}\right) \cos\left(\frac{\pi m(2j+1)}{2N}\right)}_{\text{Basisfunktionen}} X_{ij} \quad 0 \leqslant m < M, 0 \leqslant n < N.$$

2D DCT — Details

Die Zeile i der $M \times N$-Bildmatrix transformiert ergibt

$$T_{i,n} = c_{N,n} \sum_{j=0}^{N-1} X_{i,j} \cos\left(\frac{\pi n(2j+1)}{2N}\right) \quad \text{was mit} \quad (C_N)_{n,j} = c_{N,n} \cos\left(\frac{\pi n(2j+1)}{2N}\right)$$

so geschrieben werden kann

$$T_{i,n} = \sum_{j=0}^{N-1} X_{i,j} (C_N)_{n,j} \;=\; \sum_{j=0}^{N-1} X_{i,j} (C_N)^T_{j,n} \quad \text{und in Matrixschreibweise so:} \quad \mathbf{T} = \mathbf{X}\mathbf{C}_N^T.$$

Jetzt transformiert man die Spalte n von $\mathbf{T}$ und erhält

$$Y_{m,n} = c_{M,m} \sum_{i=0}^{M-1} T_{i,n} \cos\left(\frac{\pi m(2i+1)}{2M}\right) \quad \text{was mit} \quad (C_M)_{m,i} = c_{M,m} \cos\left(\frac{\pi m(2i+1)}{2M}\right)$$

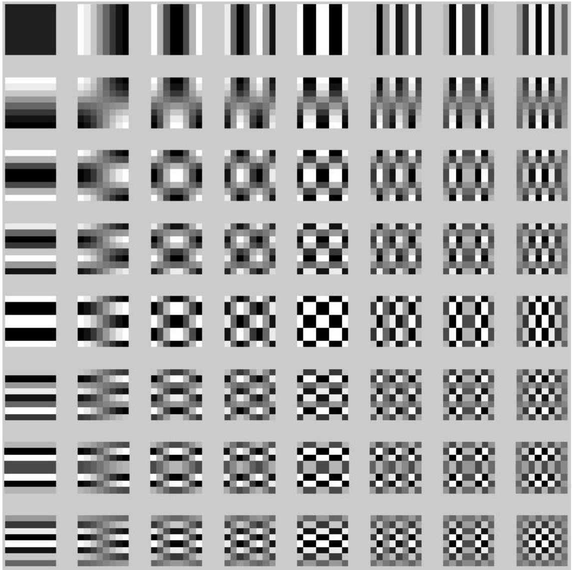
so geschrieben werden kann

$$Y_{m,n} = \sum_{i=0}^{M-1} T_{i,n} (C_M)_{m,i} \;=\; \sum_{i=0}^{M-1} (C_M)_{m,i} T_{i,n} \quad \text{und mit Matrizen so:} \quad \mathbf{Y} = \mathbf{C}_M \mathbf{T} = \mathbf{C}_M \mathbf{X} \mathbf{C}_N^T.$$

2D DCT (Fort.)

Jedes Bild von 8×8 Pixeln lässt sich eindeutig als Linearkombination dieser 8×8 Basisfunktionen der 2D-DCT darstellen. Sie können mit MATLAB wie folgt generiert werden:

```
D8=my_dct(8);
for k=1:8
 for l=1:8
  X = zeros(8);
  X(k,l) = 1;
  kl = 8*(k-1)+l;
  subplot(8, 8, kl)
  Y = D8'*X*D8;
  I = pixeldup(Y, 8)
  imshow(I, [])
 end
end
```



Erstellen Sie den entsprechenden python code!

Verwende `np.repeat(np.repeat(Y,8,axis=0),8,axis=1)` anstelle von `pixeldup`!

Eine Anwendung

Example (Kompression eines Bildes mit DCT)

Das Bild wird in ein Graustufenbild umgewandelt und darauf die DCT angewendet. Dann wird der Logarithmus der Beträge der DCT-Matrix graphisch dargestellt.

```
RGB = imread('autumn.tif');
I = rgb2gray(RGB);
J = dct2(I);
imshow(log(abs(J)),[]), colormap(jet(64)), colorbar
```

Anschliessend werden alle Elemente der DCT kleiner als 20 (Threshold) Null gesetzt. Dann wird die inverse DCT ausgeführt.

```
J(abs(J) < 20) = 0;
K = idct2(J);
subplot (1,2,1), imshow(I)
subplot (1,2,2), imshow(K,[0 255])
```

Vergleichen Sie das Originalbild und das komprimierte Bild! Wo sind die Unterschiede? Welche Grösse (in kB) haben die beiden Bilder?

Eine Anwendung

Example (Kompression eines Bildes mit DCT – Fort.)

Vergleichen Sie auch die Grösse (in kB) der Bilder! Verwenden Sie statt 20 andere Werte (z.B. 5, 10 oder 30). Versuchen Sie das Entdeckte zu verstehen!