2D-Signal Transforms and The Separability Property

Based on project presentation by Evgeny Tenetov
SIPC (236327), Winter 2014-2015
Background: Discrete Transform of 1D Signals

Let $x$ be a $n \times 1$ vector (1-dimensional signal).
Let $x$ be a $n \times 1$ vector (1-dimensional signal).

- An arbitrary 1-dimensional transform is $y = Tx$, where $T$ is $n \times n$ matrix, hence $y$ is the transform-domain representation of the signal formed as a $n \times 1$ vector.
Background: Discrete Transform of 1D Signals

Let $x$ be a $n \times 1$ vector (1-dimensional signal).

- An arbitrary 1-dimensional transform is $y = Tx$, where $T$ is $n \times n$ matrix, hence $y$ is the transform-domain representation of the signal formed as a $n \times 1$ vector.

- It takes $O(n^2)$ operations (addition/multiplication) to calculate the transform. Since we calculate $n$ coefficients, each by multiplying $n$-length row of $T$ with the $n$-length signal vector.
Transforming 2D Signals

Let $X$ be an $n \times n$ matrix (2-dimensional signal / image).
Transforming 2D Signals

Let $X$ be an $n \times n$ matrix (2-dimensional signal / image).

- An arbitrary 2-dimensional transform is $y = Tx$, where $T$ is $n^2 \times n^2$ matrix, and $x$ is a one-dimensional vector, of size $n^2 \times 1$, formed by stacking $X$'s columns.

It takes $O(n^4)$ operations (addition/multiplication) to calculate the transform.

Since we calculate $n^2$ coefficients, each by multiplying $n^2$-length row of $T$ with the $n^2$-length signal vector.

The Problem

For large $n$ the operation becomes extremely computationally demanding.
Transforming 2D Signals

Let $X$ be an $n \times n$ matrix (2-dimensional signal / image).

- An arbitrary 2-dimensional transform is $y = Tx$, where $T$ is $n^2 \times n^2$ matrix, and $x$ is a one-dimensional vector, of size $n^2 \times 1$, formed by stacking $X$’s columns.

- It takes $O(n^4)$ operations (addition/multiplication) to calculate the transform. Since we calculate $n^2$ coefficients, each by multiplying $n^2$-length row of $T$ with the $n^2$-length signal vector.
Transforming 2D Signals

Let $X$ be an $n \times n$ matrix (2-dimensional signal / image).

- An arbitrary 2-dimensional transform is $y =Tx$, where $T$ is $n^2 \times n^2$ matrix, and $x$ is a one-dimensional vector, of size $n^2 \times 1$, formed by stacking $X$’s columns.
- It takes $O(n^4)$ operations (addition/multiplication) to calculate the transform. Since we calculate $n^2$ coefficients, each by multiplying $n^2$-length row of $T$ with the $n^2$-length signal vector.

The Problem

For large $n$ the operation becomes extremely computationally demanding.
The Kronecker Product

Let $A, B$ be $m_1 \times m_2$ and $n_1 \times n_2$ matrices, respectively. Then the **Kronecker Product** (or Tensor product) of $A$ and $B$ is an $n_1 m_1 \times n_2 m_2$ matrix defined as

$$A \otimes B = \begin{bmatrix}
a_{1,1}B & \cdots & a_{1,m_2}B \\
\vdots & & \vdots \\
a_{m_1,1}B & \cdots & a_{m_1,m_2}B
\end{bmatrix}$$
The Kronecker Product

Let $A, B$ be $m_1 \times m_2$ and $n_1 \times n_2$ matrices, respectively. Then the **Kronecker Product** (or Tensor product) of $A$ and $B$ is an $n_1 m_1 \times n_2 m_2$ matrix defined as

$$
A \otimes B = \begin{bmatrix}
    a_{1,1}B & \cdots & a_{1,m_2}B \\
    \vdots & \ddots & \vdots \\
    a_{m_1,1}B & \cdots & a_{m_1,m_2}B
\end{bmatrix}
$$

**Remarks:**

- We have

$$
(A \otimes B)_{1+(i-1)n_1+(j-1),1+(k-1)n_2+(l-1)} = a_{i,k}b_{j,l}
$$

$$
1 \leq i \leq m_1, \ 1 \leq j \leq n_1, \ 1 \leq k \leq m_2, \ 1 \leq l \leq n_2
$$

- It takes $n_1 m_1 n_2 m_2$ operations to calculate $A \otimes B$. 
Examples

Example 1

$$\begin{bmatrix} 1 & 3 \\ 6 & 5 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 3 & 3 \\ 2 & 3 & 6 & 9 \\ 6 & 6 & 5 & 5 \\ 12 & 18 & 10 & 15 \end{bmatrix}$$
Examples

Example 1

\[
\begin{bmatrix} 1 & 3 \\ 6 & 5 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 1 \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix}^3 & 1 \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} \\ 6 \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix}^5 & 6 \begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 3 & 3 \\ 2 & 3 & 6 & 9 \\ 6 & 6 & 5 & 5 \\ 12 & 18 & 10 & 15 \end{bmatrix}
\]

Example 2 - Hadamard matrices

\[
H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}
\]

\[
H_{2^n} = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{2^{n-1}} & H_{2^{n-1}} \\ H_{2^{n-1}} & -H_{2^{n-1}} \end{bmatrix} = H_2 \otimes H_{2^{n-1}}, \quad n \geq 2
\]
Example 3 - Outer product

In case $u$, $v$ are column vectors, $u \otimes v^t = uv^t$ is also called outer product.

$$u \otimes v^t = uv^t = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 \end{bmatrix}$$
Example 3 - Outer product
In case \( u, v \) are column vectors, \( u \otimes v^t = uv^t \) is also called outer product.

\[
\begin{align*}
  u \otimes v^t = uv^t &= \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \\
  &= \begin{bmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 \end{bmatrix}
\end{align*}
\]

Properties of the Kronecker product

1. \( (A + B) \otimes C = A \otimes C + B \otimes C \)
2. \( (A \otimes B) \otimes C = A \otimes (B \otimes C) \)
3. \( \alpha(A \otimes B) = (\alpha A) \otimes B = A \otimes (\alpha B) \)
4. \( (A \otimes B)^t = A^t \otimes B^t \)
5. \( (A \otimes B)^* = A^* \otimes B^* \) (conjugate transpose)
6. \( (A \otimes B)(C \otimes D) = (AC) \otimes (BD) \)
7. \( (A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \)
Let us consider the $n \times n$ signal $X$. A **separable** image-transform can be applied on the 2D-signal by $y = AXB^t$, where $A$ and $B$ are 1D transformations that operate separately on the signal columns and rows, respectively.
Separable Image Transforms

Let us consider the $n \times n$ signal $X$. A **separable** image-transform can be applied on the 2D-signal by $y = AXB^t$, where $A$ and $B$ are 1D transformations that operate separately on the signal columns and rows, respectively.

- The calculation of $AXB^t$ takes $O(n^3)$ operations, instead of $O(n^4)$ operations, as in the general image transform case.
Separable Image Transforms

Let us consider the $n \times n$ signal $X$. A **separable** image-transform can be applied on the 2D-signal by $y = AXB^t$, where $A$ and $B$ are 1D transformations that operate separately on the signal columns and rows, respectively.

- The calculation of $AXB^t$ takes $O(n^3)$ operations, instead of $O(n^4)$ operations, as in the general image transform case.
- The 1D equivalent transform is obtained via $T = A \otimes B$, and can be applied on the vector (column-stack) form of $X$ (in a complexity of $O(n^4)$).
Separable Unitary Image Transforms

For two unitary transforms $A$ and $B$, the transform $A \otimes B$ is also unitary:

$$(A \otimes B)(A \otimes B)^* = (A \otimes B)(A^* \otimes B^*)$$

$$= AA^* \otimes BB^* = I \otimes I = I$$
Separable Unitary Image Transforms

For two unitary transforms $A$ and $B$, the transform $A \otimes B$ is also unitary:

\[(A \otimes B)(A \otimes B)^* = (A \otimes B)(A^* \otimes B^*) = AA^* \otimes BB^* = I \otimes I = I\]

Hence,

- 2D unitary transforms can be constructed from two unitary 1D transforms.
- The inverse transform of $Y = AXB^t$ is given by
  \[(A^* \otimes B^*)y, \quad A^*Y(B^*)^t = A^*Y\overline{B}\]
  where $\overline{B}$ is the complex-conjugate of $B$. 
Example - 2-dimensional DFT

Let $F$ be the one-dimensional DFT transform:

$$F_{mn} = e^{-2\pi i mn/N}, \quad 1 \leq m, n \leq N$$

Then, the separable two-dimensional DFT is

$$F \otimes F, \quad FXF^t = FXF$$

and the corresponding inverse transform is

$$F^* \otimes F^*, \quad F^* Y \overline{F} = \overline{FYF}$$
Example - 2-dimensional DFT

Let $F$ be the one-dimensional DFT transform:

$$F_{mn} = e^{-\frac{2\pi i mn}{N}}, \quad 1 \leq m, n \leq N$$

Then, the separable two-dimensional DFT is

$$F \otimes F, \quad FXF^t = FXF$$

and the corresponding inverse transform is

$$F^* \otimes F^*, \quad F^* Y \overline{F} = \overline{F} Y \overline{F}$$

Alternatively, the above transformation of $X$ can be expressed as

$$Y_{kl} = \sum_{n_1=1}^{N} \sum_{n_2=1}^{N} e^{-\frac{2\pi i k n_1}{N}} e^{-\frac{2\pi i n_2}{N}} X_{n_1,n_2}$$

$$X^{\text{inverse}}_{kl} = \sum_{n_1=1}^{N} \sum_{n_2=1}^{N} e^{\frac{2\pi i k n_1}{N}} e^{\frac{2\pi i n_2}{N}} Y_{n_1,n_2}$$
Separable Basis

The new orthonormal basis associated with the separable transformation \( T(X) = AXB^t \) is

\[
\{ a_i^* \otimes (b_j^*)^t = a_i^*(b_j^*)^t \mid i, j = 1, \ldots, N \}
\]

where \( a_i^* \), \( b_j^* \) denote the \( i \)-th and \( j \)-th columns of \( A^* \) and \( B^* \) respectively. These matrices are called basis images.

This follows from

\[
\left( A^* E_{\alpha\beta} (B^*)^t \right)_{ij} = \sum_{1 \leq k, m \leq n} a_{ik}^* b_{jm}^* (E_{\alpha\beta})_{km} = a_{i\alpha}^* b_{j\beta}^* \\
T_{\text{inverse}}(E_{\alpha\beta}) = A^* E_{\alpha\beta} (B^*)^t = a_{\alpha}^* (b_{\beta}^*)^t \quad (1 \leq \alpha, \beta \leq N)
\]
Separable Basis

Each $N \times N$ matrix $X$ can be written as

$$X = \sum_{1 \leq i, j \leq N} Y_{ij} (a_i^* \otimes (b_j^*)^t) \quad Y_{ij} = \langle X, a_i^* \otimes (b_j^*)^t \rangle, \quad 1 \leq i, j \leq N$$

where $\langle \ , \ \rangle$ is the matrix inner product

$$\langle F, G \rangle = \sum_{1 \leq i, j \leq N} F_{ij} \overline{G_{ij}}$$
Example - 8 × 8 DFT Basis Elements
Other Computational aspects

- Matrices which are Kronecker product of smaller matrices, can be efficiently multiplied using the property

\[(A \otimes B)(C \otimes D) = (AC) \otimes (BD)\]

The left hand side takes \(O((N^2)^3) + O(N^4) = O(N^6)\) operations, whereas the right hand side takes \(O(N^4) + O(N^3) = O(N^4)\) operations.
Matrices which are Kronecker product of smaller matrices, can be efficiently multiplied using the property

\[(A \otimes B)(C \otimes D) = (AC) \otimes (BD)\]

The left hand side takes \(O((N^2)^3) + O(N^4) = O(N^6)\) operations, whereas the right hand side takes \(O(N^4) + O(N^3) = O(N^4)\) operations.

If \(A\) and \(B\) are diagonalizable

\[U^{-1}AU = D_1, \quad V^{-1}BV = D_2\]

then so is \(A \otimes B\) :

\[(U \otimes V)^{-1}(A \otimes B)(U \otimes V) = (U^{-1} \otimes V^{-1})(A \otimes B)(U \otimes V) = (U^{-1}AU \otimes V^{-1}BV) = D_1 \otimes D_2 \quad (\text{diagonal})\]
Usually, the structure of the transform can be exploited for efficient computation. For example, the Hadamard transform

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad H_{2n} = H_2 \otimes H_{2n-1} = H_2 \otimes \ldots \otimes H_2, \quad n \geq 2$$

can be calculated in less operations than $O(N^2), \quad N = 2^n$.

$$H_{2n} \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{2n-1} & H_{2n-1} \\ H_{2n-1} & -H_{2n-1} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{2n-1}x + H_{2n-1}y \\ H_{2n-1}x - H_{2n-1}y \end{bmatrix}$$

($x, y \in \mathbb{R}^{2^{n-1}}$)

The recursion formula is $T(N) = 2T(N/2) + N$. Hence the complexity is $O(N \log N)$. 
Usually, the structure of the transform can be exploited for efficient computation. For example the Hadamard transform

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad H_{2^n} = H_2 \otimes H_{2^{n-1}} = \underbrace{H_2 \otimes \ldots \otimes H_2}_{n}, \quad n \geq 2$$

can be calculated in less operations than $O(N^2)$, $N = 2^n$.

$$H_{2^n} \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{2^{n-1}} & H_{2^{n-1}} \\ H_{2^{n-1}} & -H_{2^{n-1}} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{2^{n-1}}x + H_{2^{n-1}}y \\ H_{2^{n-1}}x - H_{2^{n-1}}y \end{bmatrix}$$

$(x, y \in \mathbb{R}^{2^{n-1}})$

The recursion formula is $T(N) = 2T(N/2) + N$. Hence the complexity is $O(N \log N)$.

This example can be generalized to an arbitrary transform of the form $A_1 \otimes \ldots \otimes A_n$. 

The DFT, DCT and Walsh-Hadamard transforms can be calculated in $O(N \log N)$ operations. Such transforms are called **Fast transforms**.
The DFT, DCT and Walsh-Hadamard transforms can be calculated in $O(N \log N)$ operations. Such transform are called Fast transforms.

In fact, the Fourier transform became prevalent since the discovery of the fast Fourier transform (FFT) algorithm in the mid-1960s, which made it practical to calculate Fourier transforms.
The DFT, DCT and Walsh-Hadamard transforms can be calculated in $O(N \log N)$ operations. Such transforms are called **Fast transforms**.

In fact, the Fourier transform became prevalent since the discovery of the fast Fourier transform (FFT) algorithm in the mid-1960s, which made it practical to calculate Fourier transforms.

Hence, each of the corresponding separable two-dimensional transforms $AXA^t = A(AX^t)^t$ can be implemented in $O(N^2 \log N)$ operations.
Run-Time Demonstration

Execution time of separable transform vs general transform.
8 × 8 Walsh-Hadamard - Basis Elements
8 × 8 DCT - Basis Elements