Computational Methods for Inverse Problems and Applications in Image Processing

Introduction to Kronecker Products

If $\boldsymbol{A}$ is an $m \times n$ matrix and $\boldsymbol{B}$ is a $p \times q$ matrix, then the *Kronecker product* of $\boldsymbol{A}$ and $\boldsymbol{B}$ is the $mp \times nq$ matrix

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2n}\boldsymbol{B} \\ \vdots & \vdots & & \vdots \\ a_{m1}\boldsymbol{B} & a_{m2}\boldsymbol{B} & \cdots & a_{mn}\boldsymbol{B} \end{bmatrix}$$

Note that if $\boldsymbol{A}$ and $\boldsymbol{B}$ are large matrices, then the Kronecker product $\boldsymbol{A} \otimes \boldsymbol{B}$ will be huge. MATLAB has a built-in function `kron` that can be used as

```
K = kron(A, B);
```

However, you will quickly run out of memory if you try this for matrices that are 50×50 or larger. Fortunately we can exploit the block structure of Kronecker products to do many computations involving $\boldsymbol{A} \otimes \boldsymbol{B}$ without actually forming the Kronecker product. Instead we need only do computations with $\boldsymbol{A}$ and $\boldsymbol{B}$ individually.

To begin, we state some very simple properties of Kronecker products, which should not be difficult to verify:

- $(\boldsymbol{A} \otimes \boldsymbol{B})^T = \boldsymbol{A}^T \otimes \boldsymbol{B}^T$
- If $\boldsymbol{A}$ and $\boldsymbol{B}$ are square and nonsingular, then $(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1}$
- $(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) = \boldsymbol{AC} \otimes \boldsymbol{BD}$

The next property we want to consider involves the matrix-vector multiplication

$$\boldsymbol{y} = (\boldsymbol{A} \otimes \boldsymbol{B})\boldsymbol{x},$$

where $\boldsymbol{A} \in \mathcal{R}^{m \times n}$ and $\boldsymbol{B} \in \mathcal{R}^{p \times q}$. Thus $\boldsymbol{A} \otimes \boldsymbol{B} \in \mathcal{R}^{mp \times nq}$, $\boldsymbol{x} \in \mathcal{R}^{nq}$, and $\boldsymbol{y} \in \mathcal{R}^{mp}$. Our goal is to exploit the block structure of the Kronecker product matrix to compute $\boldsymbol{y}$ without explicitly forming $(\boldsymbol{A} \otimes \boldsymbol{B})$. To see how this can be done, first partition the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ as

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix}, \quad \boldsymbol{x}_i \in \mathcal{R}^q \quad \text{and} \quad \boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_m \end{bmatrix}, \quad \boldsymbol{y}_i \in \mathcal{R}^p$$

Then

$$\boldsymbol{y} = (\boldsymbol{A} \otimes \boldsymbol{B})\boldsymbol{x} = \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2n}\boldsymbol{B} \\ \vdots & \vdots & & \vdots \\ a_{m1}\boldsymbol{B} & a_{m2}\boldsymbol{B} & \cdots & a_{mn}\boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix}$$

which can be written as

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_m \end{bmatrix} = \begin{bmatrix} a_{11}\boldsymbol{B}\boldsymbol{x}_1 + a_{12}\boldsymbol{B}\boldsymbol{x}_2 + \cdots + a_{1n}\boldsymbol{B}\boldsymbol{x}_n \\ a_{21}\boldsymbol{B}\boldsymbol{x}_1 + a_{22}\boldsymbol{B}\boldsymbol{x}_2 + \cdots + a_{2n}\boldsymbol{B}\boldsymbol{x}_n \\ \vdots \\ a_{m1}\boldsymbol{B}\boldsymbol{x}_1 + a_{m2}\boldsymbol{B}\boldsymbol{x}_2 + \cdots + a_{mn}\boldsymbol{B}\boldsymbol{x}_n \end{bmatrix}$$

Now notice that each $\boldsymbol{y}_i$ has the form

$$\begin{aligned} \boldsymbol{y}_i &= a_{i1}\boldsymbol{B}\boldsymbol{x}_1 + a_{i2}\boldsymbol{B}\boldsymbol{x}_2 + \cdots + a_{in}\boldsymbol{B}\boldsymbol{x}_n \\ &= \begin{bmatrix} \boldsymbol{B}\boldsymbol{x}_1 & \boldsymbol{B}\boldsymbol{x}_2 & \cdots & \boldsymbol{B}\boldsymbol{x}_n \end{bmatrix} \begin{bmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{in} \end{bmatrix} \\ &= \boldsymbol{B} \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{bmatrix} \begin{bmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{in} \end{bmatrix} \end{aligned}$$

Now define $\boldsymbol{a}_i^T = \begin{bmatrix} a_{i1} & a_{i2} & \cdots & a_{in} \end{bmatrix} = i$th row of $\boldsymbol{A}$, and define $\boldsymbol{X}$ to be a matrix with ith column $\boldsymbol{x}_i$; that is,

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{bmatrix}$$

Then

$$\boldsymbol{y}_i = \boldsymbol{B}\boldsymbol{X}\boldsymbol{a}_i\,, \quad i = 1, 2, \ldots, m$$

Analogous to the matrix $\boldsymbol{X}$, define a matrix $\boldsymbol{Y}$ with columns $\boldsymbol{y}_i$:

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_m \end{bmatrix}$$

Then we have

$$\begin{aligned} \boldsymbol{Y} &= \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_m \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{B}\boldsymbol{X}\boldsymbol{a}_1 & \boldsymbol{B}\boldsymbol{X}\boldsymbol{a}_2 & \cdots & \boldsymbol{B}\boldsymbol{X}\boldsymbol{a}_m \end{bmatrix} \\ &= \boldsymbol{B}\boldsymbol{X} \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_m \end{bmatrix} \\ &= \boldsymbol{B}\boldsymbol{X}\boldsymbol{A}^T \end{aligned}$$

In the last step above, recall that $\boldsymbol{a}_i^T$ is the ith row of $\boldsymbol{A}$, so $\boldsymbol{a}_i$ is the ith column of $\boldsymbol{A}^T$.

To summarize, we have the following property of Kronecker products:

$$\boldsymbol{y} = (\boldsymbol{A} \otimes \boldsymbol{B})\boldsymbol{x} \quad \Leftrightarrow \quad \boldsymbol{Y} = \boldsymbol{B}\boldsymbol{X}\boldsymbol{A}^T$$

where

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{bmatrix} \in \mathcal{R}^{q \times n} \quad \text{and} \quad \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_m \end{bmatrix} \in \mathcal{R}^{p \times m}$$

Remarks on notation and MATLAB computations:

- If $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{bmatrix}$ is an array with columns $\boldsymbol{x}_i$, then in many books you will see the notation $\boldsymbol{x} = \mathtt{vec}(\boldsymbol{X})$ used to denote a vector obtained by stacking the columns of $\boldsymbol{X}$ on top of each other. That is,

$$\boldsymbol{x} = \mathtt{vec}(\boldsymbol{X}) = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix}$$

- MATLAB does not have a `vec` function, but it does have a built-in function called `reshape` that can be used for this purpose. Specifically, if $\boldsymbol{X} \in \mathcal{R}^{q \times n}$, then the vector $\boldsymbol{x} = \mathtt{vec}(\boldsymbol{X})$ can be obtained with the MATLAB statement:

```
x = reshape(X, q*n, 1);
```

  A short cut to using the reshape command is to use the colon operator. That is,

```
x = X(:);
```

  does the same thing as `x = reshape(X, q*n, 1)`. To go the other way, from $\boldsymbol{x}$ to $\boldsymbol{X}$, you can again use the `reshape` function:

```
X = reshape(x, q, n);
```

  For more information, see the MATLAB doc page for `reshape`.

- Thus, suppose we are given $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{x}$ in MATLAB, and we want to compute $\boldsymbol{y} = (\boldsymbol{A} \otimes \boldsymbol{B})\boldsymbol{x}$. This can be done without explicitly forming $\boldsymbol{A} \otimes \boldsymbol{B}$ as follows:

```
[m, n] = size(A);
[p, q] - size(B);
X = reshape(x, q, n);
Y = B*X*A';
y = reshape(Y, m*p, 1);
```

  Note that this computation requires $O(npq+qnm)$ arithmetic operations (FLOPS) to compute $\boldsymbol{y}$. On the other hand, if we explicitly form $\boldsymbol{A} \otimes \boldsymbol{B}$, then the cost to compute $\boldsymbol{y}$ is $O(mpnq)$. In addition to the computational savings, we save a lot on storage if we don't explicitly construct $\boldsymbol{A} \otimes \boldsymbol{B}$, and instead just store the smaller matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

- As with matrix-vector multiplication, we can efficiently solve linear systems

$$(\boldsymbol{A} \otimes \boldsymbol{B})\boldsymbol{x} = \boldsymbol{y}$$

  using properties of Kronecker products. Specifically, assume $\boldsymbol{A}$ and $\boldsymbol{B} \in \mathcal{R}^{n \times n}$ are both nonsingular. Then using properties of Kronecker products we know

$$\boldsymbol{x} = (\boldsymbol{A} \otimes \boldsymbol{B})^{-1}\boldsymbol{y} = (\boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1})\boldsymbol{y}$$

From the matrix-vector multiplication property this is equivalent to computing:

$$\boldsymbol{X} = \boldsymbol{B}^{-1}\boldsymbol{Y}\boldsymbol{A}^{-T}, \quad \boldsymbol{x} = \texttt{vec}(\boldsymbol{X}), \quad \boldsymbol{y} = \texttt{vec}(\boldsymbol{Y})$$

Generally we never explicitly form the inverse of a matrix, and instead should read "inverse" for meaning "solve". That is:

- Computing $\boldsymbol{Z} = \boldsymbol{B}^{-1}\boldsymbol{Y}$ is equivalent to the problem: Given $\boldsymbol{B}$ and $\boldsymbol{Y}$, solve the matrix equation $\boldsymbol{B}\boldsymbol{Z} = \boldsymbol{Y}$ for $\boldsymbol{Z}$.
- Computing $\boldsymbol{X} = \boldsymbol{Z}\boldsymbol{A}^{-T}$ is equivalent to the problem: Given $\boldsymbol{A}$ and $\boldsymbol{Z}$, solve the matrix equation $\boldsymbol{X}\boldsymbol{A}^T = \boldsymbol{Z}$ for $\boldsymbol{X}$. (Note that this is also equivalent to solving the matrix equation $\boldsymbol{A}\boldsymbol{X}^T = \boldsymbol{Z}^T$ for $\boldsymbol{X}$.)

The cost of doing this with matrix factorization methods (e.g., $\boldsymbol{P}\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$) is only $O(n^3)$. On the other hand, if we explicitly form $\boldsymbol{A} \otimes \boldsymbol{B}$ and then solve the system, the cost will be $O(n^6)$.

In MATLAB, using the efficient method outlined above to solve $(\boldsymbol{A} \otimes \boldsymbol{B})\boldsymbol{x} = \boldsymbol{y}$ can be implemented as:

```
n = size(A,1);
Y = reshape(y, n, n);
X = B \ Y / A';
x = reshape(X, n*n, 1);
```

If you're not sure how the back-slash \ and the forward-slash / operators work, then you should read the MATLAB doc pages: `doc slash`.