Math 515
Fall, 2010

Introduction to Kronecker Products

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

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

Note that if A and B are large matrices, then the Kronecker product $A \otimes B$ 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 $A \otimes B$ without actually forming the Kronecker product. Instead we need only do computations with A and B individually.

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

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

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

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

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

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

Then

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

which can be written as

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

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

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

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

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

Then

$$\mathbf{y}_i = BX\mathbf{a}_i\,, \quad i = 1, 2, \ldots, m$$

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

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

Then we have

$$\begin{aligned} Y &= \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_m \end{bmatrix} \\ &= \begin{bmatrix} BX\mathbf{a}_1 & BX\mathbf{a}_2 & \cdots & BX\mathbf{a}_m \end{bmatrix} \\ &= BX \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_m \end{bmatrix} \\ &= BXA^T \end{aligned}$$

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

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

$$\boxed{\begin{gathered} \mathbf{y} = (A \otimes B)\mathbf{x} \quad \Leftrightarrow \quad Y = BXA^T \\ \text{where} \\ X = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \end{bmatrix} \in \mathcal{R}^{q \times n} \quad \text{and} \quad Y = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_m \end{bmatrix} \in \mathcal{R}^{p \times m} \end{gathered}}$$

Remarks on notation and MATLAB computations:

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

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

- 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 $X \in \mathcal{R}^{q\times n}$, then the vector $\mathbf{x} = \texttt{vec}(X)$ 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 $\mathbf{x}$ to 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 A, B, and $\mathbf{x}$ in MATLAB, and we want to compute $\mathbf{y} = (A \otimes B)\mathbf{x}$. This can be done without explicitly forming $A \otimes B$ 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 $\mathbf{y}$. On the other hand, if we explicitly form $A\otimes B$, then the cost to compute $\mathbf{y}$ is $O(mpnq)$. In addition to the computational savings, we save a lot on storage if we don't explicitly construct $A \otimes B$, and instead just store the smaller matrices A and B.

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

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

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

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

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

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

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

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

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

In MATLAB, using the efficient method outlined above to solve $(A \otimes B)\mathbf{x} = \mathbf{y}$ 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`.