On Kronecker Sums

Zouhair Mahboubi

March 20, 2015

1 Introduction

The goal of this short note is to outline how a special case of Kronecker sums can be computed efficiently. In our application, the joint transition rate matrix Q can be expressed as a Kronecker sum of basic transition matrices. Particularly, by using a compact representation of the state and action space, we can eventually show that for any action $a \in \mathcal{A}$ the associated transition matrix takes the form:

$$Q = A \oplus \underbrace{B \oplus B \cdots \oplus B}_{K}$$

In our problem $A, B \in M_{n \times n}$ are sparse matrices with $n \approx 30$. Computing Q for values of $K + 1 < 5$ is tractable on a personal computer, but for values of $K + 1 \geq 5$ we run into memory constraints. However, since Q is ultimately used in a value iteration algorithm, where only one row at a time is needed, we investigate how the i^{th} row of Q can be computed efficiently.

2 Definitions and Notation

In this section, we introduce the notation that we use and define different symbols and operators. We then outline how the Kronecker sum of a repeated matrix can be represented in a compact form.

2.1 Dimensions and Special matrices

- We will denote a matrix A of dimensions $m \times n$ as $A \in M_{m \times n}$ or$A_{m \times n}$.
  - We will omit dimensions of matrices when they are obvious
- If the matrix is square, we will drop one dimension and write $A_n \in M_n$.
- The identity matrix of dimensions $n \times n$ will be denoted by I_n.
- e_i is a versor of Cartesian coordinates with entries $e_i(k) = \delta_{ik}$
- $E^{i,j}_{m \times n}$ will denote the sparse $m \times n$ matrix whose entries are defined by $E^{i,j}(k, l) = e_i e_j^{\intercal} = \delta_{ki}\delta_{lj}$ (i.e. everything is 0 except the (i, j) entry)

2.2 Kronecker Product

Kronecker product between two matrices $X \in M_{m\times n}, Y \in M_{p\times q}$ is written as $X \otimes Y \in M_{mp\times nq}$. Definition can be found in any matrix analysis textbook. Some of the Kronecker product properties we will use later:

$$\begin{aligned} X \otimes (Y \otimes Z) &= (X \otimes Y) \otimes Z \\ X \otimes (Y + Z) &= X \otimes Y + X \otimes Z \\ I_n \otimes I_m &= I_{nm} \end{aligned}$$

Note that in general the Kronecker product is not commutative, i.e.

$$X \otimes Y \neq Y \otimes X$$

However, the product is permutation equivalent, and it can be shown that:

$$X_{m\times n} \otimes Y_{p\times q} = P_{(m,p)} \left(Y \otimes X\right) P^{\intercal}_{(n,p)} \tag{1}$$

Where $P_{(m,p)} \in M_{mp}$ is known as the *perfect shuffle permutation*:

$$P_{(m,p)} = \sum_{i=1}^{m} \sum_{j=1}^{p} \left(E^{i,j}_{m\times p} \otimes (E^{i,j}_{m\times p})^{\intercal}\right) \tag{2}$$

We note one particular case which we will use subsequently,

$$P_{(1,m)} = P_{(m,1)} = I_m$$

2.3 Kronecker Sum

The Kronecker sum is defined for square matrices only and is defined as follows:

$$X_n \oplus Y_m = X_n \otimes I_m + I_n \otimes Y_m \in M_{nm\times nm}$$

The Kronecker sum is also non-commutative. However, we will show in this section how the special case of $C = \underbrace{B \oplus B \cdots \oplus B}_{K}$ can be simplified:

$$\begin{aligned} B_n \oplus B_n &= B_n \otimes I_n + I_n \otimes B_n \\ &= I_n \otimes B_n + P_{(n,n)} \left(I_n \otimes B_n\right) P^{\intercal}_{(n,n)} \end{aligned}$$

Likewise,

$$\begin{aligned} B_n \oplus B_n \oplus B_n &= B_n \oplus \left(B_n \otimes I_n + I_n \otimes B_n\right) \\ &= B_n \otimes I_{n^2} + I_n \otimes \left(B_n \otimes I_n + I_n \otimes B_n\right) \\ &= B_n \otimes I_{n^2} + \left(I_n \otimes B_n\right) \otimes I_n + I_{n^2} \otimes B_n \\ &= I_{n^2} \otimes B_n + P_{(n^2,n)} \left(I_{n^2} \otimes B_n\right) P^{\intercal}_{(n^2,n)} + P_{(n,n^2)} \left(I_{n^2} \otimes B_n\right) P^{\intercal}_{(n,n^2)} \end{aligned}$$

One last time...

$$\begin{aligned}
B_n \oplus B_n \oplus B_n \oplus B_n &= B_n \oplus (B_n \otimes I_{n^2} + (I_n \otimes B_n) \otimes I_n + I_{n^2} \otimes B_n) \\
&= B_n \otimes I_{n^3} + I_n \otimes (B_n \otimes I_{n^2} + (I_n \otimes B_n) \otimes I_n + I_{n^2} \otimes B_n) \\
&= B_n \otimes I_{n^3} + (I_n \otimes B_n) \otimes I_{n^2} + (I_{n^2} \otimes B_n) \otimes I_n + I_{n^3} \otimes B_n \\
&= +P_{(n^4,n^0)} (I_{n^3} \otimes B_n) P^{\intercal}_{(n^4,n^0)} \\
&\quad +P_{(n^3,n^1)} (I_{n^3} \otimes B_n) P^{\intercal}_{(n^3,n^1)} \\
&\quad +P_{(n^2,n^2)} (I_{n^3} \otimes B_n) P^{\intercal}_{(n^2,n^2)} \\
&\quad +P_{(n^1,n^3)} (I_{n^3} \otimes B_n) P^{\intercal}_{(n^1,n^3)}
\end{aligned}$$

This can be generalized to [1]:

$$\underbrace{B \oplus B \cdots \oplus B}_{K} = \sum_{u=1}^{K} P_{(n^u,n^{K-u})} (I_{n^{K-1}} \otimes B) P^{\intercal}_{(n^u,n^{K-u})} \tag{3}$$

More generally, the following relationship can be proven:

$$A_1 \oplus A_2 \cdots \oplus A_K = \sum_{u=1}^{K} P_{(n^u,n^{K-u})} (I_{n^{K-1}} \otimes A_i) P^{\intercal}_{(n^u,n^{K-u})} \tag{4}$$

3 Computing Q_i

In this section, we show how the i^{th} row of $Q = A \oplus \underbrace{B \oplus B \cdots \oplus B}_{K}$ can be computed efficiently and give a brief discussion of the required storage space and show how the compact representation introduced is helpful.

3.1 Row Entry of a Kronecker product

First, we point out that the i^{th} row of the Kronecker product $Z = X_{m \times n} \otimes Y_{p \times q}$ is given by:

$$Z_i = X_a \otimes Y_b \mid (b, a) = \text{ind2sub}((p, m), i) \tag{5}$$

Where X_a, Y_b are the a^{th} and b^{th} rows of X and Y respectively, and the ind2sub function returns the indexing tuple (b, a) from the linear index i. We assume here that the implementation uses a column major ordering of matrices. If a row major ordering is used we just need to reverse the order of dimensions and the order of the returned tuple.

[1]Proof by induction left as an exercise to the reader :)

3.2 Row Entry of Q

We have

$$
\begin{aligned}
Q_i &= \left(A \oplus \underbrace{B \oplus B \cdots \oplus B}_{K} \right)_i \\
&= (A \oplus C)_i \\
&= (A \otimes I_{n^K} + I_n \otimes C)_i \\
&= A_a \otimes e_b^\intercal + e_a^\intercal \otimes C_b
\end{aligned}
$$

Where as explained before $(b, a) = \text{ind2sub}((n^K, n), i)$. We also have $e_a \in M_{n\times 1}$ and $e_b \in M_{n^K\times 1}$the Cartesian coordinates versors. And as long as we can compute (and store) the b row of C, it is trivial to construct the i row of Q, and since there are only n non-zero entries in $A_a \otimes e_b^\intercal$, summing it with$e_a^T \otimes C_b$ can be done in $O(n)$.

Finding the b row of C deserves a discussion as it involves some Kronecker algebra. We are interested in finding:

$$
\begin{aligned}
C_b &= e_b^\intercal C = e_b^\intercal \sum_{u=0}^{K-1} P_{(n^u, n^{K-u})} \left(I_{n^{K-1}} \otimes B\right) P_{(n^u, n^{K-u})}^\intercal \\
&= \sum_{u=0}^{K-1} e_b^\intercal P_{(n^u, n^{K-u})} \left(I_{n^{K-1}} \otimes B\right) P_{(n^u, n^{K-u})}^\intercal
\end{aligned}
$$

Let's explain how this can be done for a given u. Let $n^u = p, n^{K-u} = q$ (note that $pq = n^K \forall u$), we are interested in an efficient way to compute:

$$
x = e_b^\intercal P_{(p,q)} \left(I_{n^{K-1}} \otimes B\right) P_{(p,q)}^\intercal \tag{6}
$$

The expression for $P_{(p,q)}$ is given by Eq.2, which can be rearranged as follows:

$$
\begin{aligned}
P_{(p,q)} &= \sum_{i=1}^{p}\sum_{j=1}^{q} \left(E_{p\times q}^{i,j} \otimes (E_{p\times q}^{i,j})^\intercal\right) = \sum_{i=1}^{p}\sum_{j=1}^{q} \left(e_i e_j^\intercal \otimes (e_i e_j^\intercal)^\intercal\right) \\
&= \sum_{i=1}^{p}\sum_{j=1}^{q} \left(e_i e_j^\intercal \otimes e_j e_i^\intercal\right) = \sum_{i=1}^{p}\sum_{j=1}^{q} ((e_i \otimes e_j)(e_j \otimes e_i)^\intercal) \\
&= \sum_{i=1}^{p}\sum_{j=1}^{q} (e_k e_l^\intercal) \mid \begin{cases} k = \text{sub2ind}((q,p), j, i) \\ l = \text{sub2ind}((p,q), i, j) \end{cases} \\
&= \sum_{i=1}^{p}\sum_{j=1}^{q} \left(E_{n^K}^{k,l}\right)
\end{aligned} \tag{7}
$$

Let's now compute the first term in x:

$$\begin{aligned} e_b^\intercal P_{(p,q)} &= e_b^\intercal \sum_{i=1}^{p}\sum_{j=1}^{q}(e_k e_l^\intercal) \\ &= \sum_{i=1}^{p}\sum_{j=1}^{q} e_b^\intercal e_k e_l^\intercal = \sum_{i=1}^{p}\sum_{j=1}^{q} \delta_{bk} e_l^\intercal \\ &= e_{b'}^\intercal | \begin{cases} b = k = \mathrm{sub2ind}((q,p),j,i) \\ b' = l = \mathrm{sub2ind}((p,q),i,j) \end{cases} \end{aligned}$$

Where we can solve for $b' = \mathrm{sub2ind}((p,q), \mathrm{reverse}(\mathrm{ind2sub}((q,p),b))...)$. The next step in computing x in Eq.6 is to compute:

$$e_{b'}^\intercal \quad (I_{n^{K-1}} \otimes B) = \quad e_c^\intercal \otimes B_d \mid (d,c) = \mathrm{ind2sub}((n, n^{K-1}), b')$$

Note that $e_c^\intercal \in M_{1\times n^{K-1}}$ and $B_d \in M_{1\times n}$ and therefore the result has size $1 \times n^K$. Also note that this a sparse vector, with only n entries in the $[(c-1)n+1 : cn]$ locations, i.e. $e_c^\intercal \otimes B_d = [\cdots, B_d, \cdots]$ with $\cdots$ being zeros.

The only thing left to construct x in Eq.6 is to multiply by $P_{(p,q)}^\intercal$. But this is just a permutation matrix and the resulting vector has entries at l that were at k, where l, k are given by Eq.7. Since $e_c^\intercal \otimes B_d$ is sparse, we can just iterate over the non zero entries, and move their column storage accordingly, which can be done in $O(n)$.

3.3 Space Usage and Computational Complexity

Although in our application A and B are sparse matrices, we will discuss the difference of storing the entire Q matrix compared to computing the i^{th} entry in terms of space usage for the general case.

We have shown how $C = B \oplus B \cdots \oplus B$ can be written compactly using a set of perfect shuffle matrices and the Kronecker product $X = I_{n^{K-1}} \otimes B_n$. Note that while this product has dimension $n^K \times n^K$ (i.e. n^{2K} entries), even if B is full, X is sparse with only $n^{K-1}n^2 = n^{K+1}$ non-zero entries. However, only n^2 of those entries (the ones defined by B_n) are unique.

If we look back at the expression for C, the b row is constructed by summing K vectors, each of dimension n^K. Each one of those vectors is sparse and has at most n entries. So every time we add a new vector we do at most $O(n)$ operations, and therefore we expect the total number of operations needed to construct C_b to be $O(Kn)$.

If B is full, we might end up with n^K entries in C_b. But even without sparsity, being able to compute the n^K entries on the fly in $O(Kn)$ as opposed to storing the whole n^{2k} entries is worthwhile. Indeed, with $n = 30, K = 5$, using single precision floats, it takes $\approx$ 100MB to store n^K entries while it takes

$\approx 2 \times 10^6 \mathbf{G}$B to store n^{2K} entries. Unfortunately this is still exponential, and a 120GB would only allow us to handle $K = 7$... But this is assuming B is full, hopefully with a sparse$B$ we can handle more if needed (Might spend more time later finding out the required storage...)