Computing the Hypergeometric Function of a Matrix Argument

Plamen Koev
Department of Mathematics
M.I.T.

Joint work with Alan Edelman

Random Matrix Theory Short Course, Mathfest 2004
Providence, Rhode Island, August 9-10, 2004

The Hypergeometric Function

- **Univariate**

$$ {}_pF_q(a_1, \ldots, a_p; b_1, \ldots, b_q; x) \equiv \sum_{k=0}^{\infty} \frac{(a_1)_k \ldots (a_p)_k}{k!(b_1)_k \ldots (b_q)_k} \cdot x^k, $$

  where $(a)_k = a(a+1)\ldots(a+k-1)$.

- **Hard problem to approximate. Slow convergence**

The Hypergeometric Function II

- **Univariate**

$$ {}_pF_q(a_1, \ldots, a_p; b_1, \ldots, b_q; x) \equiv \sum_{k=0}^{\infty} \frac{(a_1)_k \ldots (a_p)_k}{k!(b_1)_k \ldots (b_q)_k} \cdot x^k, $$

  where $(a)_k = a(a+1)\ldots(a+k-1)$.

  – Hard problem to approximate. Slow convergence

- **Of a Matrix Argument X**

$$ {}_pF_q^{\alpha}(a_1, \ldots, a_p; b_1, \ldots, b_q; X) \equiv \sum_{k=0}^{\infty} \sum_{\kappa \vdash k} \frac{(a_1)_\kappa \ldots (a_p)_\kappa}{k!(b_1)_\kappa \ldots (b_q)_\kappa} \cdot C_\kappa^{\alpha}(X), $$

  – $(a)_\kappa \equiv \prod_{(i,j)\in\kappa} \left(a - \frac{i-1}{\alpha} + j - 1 \right)$ — Pochhammer symbol

  – $C_\kappa^{\alpha}(X) = C_\kappa^{\alpha}(x_1, \ldots, x_n)$ — "C" Jack Function
  ($x_1, \ldots, x_n$ are the eigenvalues of X)

An Application—p.d.f. of $\lambda_{\min}$ of a β-Laguerre matrix

Definition: $L = BB^T$, where

$$B = \begin{bmatrix} \chi_{2a} & & & \\ \chi_{\beta(n-1)} & \chi_{2a-\beta} & & \\ & \ddots & \ddots & \\ & & \chi_{\beta} & \chi_{2a-\beta(n-1)} \end{bmatrix},$$

and $k = a - \frac{\beta}{2}(n-1) - 1$ is a nonnegative integer.

The p.d.f. of $\lambda_{\min}$ is

$$f(x) = x^{kn} \cdot e^{-\frac{nx}{2}} \cdot \ {}_2F_0^{2/\beta}(-k, \beta\tfrac{n}{2} + 1; ; -\tfrac{2}{x} I_{n-1}).$$

Partitions

$$\kappa = (\kappa_1, \kappa_2, \ldots) \vdash k$$

Means

$$k = \kappa_1 + \kappa_2 + \ldots \quad \text{and} \quad \kappa_1 \geq \kappa_2 \geq \ldots$$

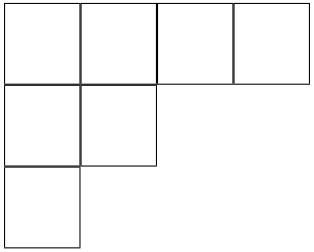
For example:

$$\begin{aligned}
1 &= 1 \\
2 &= 2 \\
&= 1+1 \\
3 &= 3 \\
&= 2+1 \\
&= 1+1+1 \\
4 &= 4 \\
&= 3+1 \\
&= 2+2 \\
&= 2+1+1 \\
&= 1+1+1+1
\end{aligned}$$

Fact: $\#\{\kappa | \kappa \vdash m\} \sim e^{\sqrt{m}}$

Partitions and Young Diagrams

The partition $(4, 2, 1)$ represented by the Young Diagram



Approximate ${}_p F_q^\alpha$ by Truncating – I

- Univariate

$$ {}_pF_q(a_{1:p}; b_{1:q}; x) = \sum_{k=0}^{\infty} \frac{(a_1)_k \dots (a_p)_k}{k!(b_1)_k \dots (b_q)_k} \cdot x^k $$

- Of a matrix argument

$$ {}_p F_q^\alpha(a_{1:p}; b_{1:q}; X) = \sum_{k=0}^{\infty} \sum_{\kappa \vdash k} \frac{(a_1)_\kappa \dots (a_p)_\kappa}{k!(b_1)_\kappa \dots (b_q)_\kappa} \cdot C_\kappa^\alpha(X), $$

Approximate ${}_pF_q^\alpha$ by Truncating – II

- Univariate

$$ {}_pF_q(a_{1:p}; b_{1:q}; x) \approx \sum_{k=0}^{m} \frac{(a_1)_k \dots (a_p)_k}{k!(b_1)_k \dots (b_q)_k} \cdot x^k $$

- Of a matrix argument

$$ {}_pF_q^\alpha(a_{1:p}; b_{1:q}; X) \approx \sum_{k=0}^{m} \sum_{\kappa \vdash k} \frac{(a_1)_\kappa \dots (a_p)_\kappa}{k!(b_1)_\kappa \dots (b_q)_\kappa} \cdot C_\kappa^\alpha(X), $$

Complexity of Approximating ${}_pF_q^\alpha$ – I

- Univariate

$$ {}_pF_q(a_{1:p}; b_{1:q}; x) \approx \sum_{k=0}^{m} \frac{(a_1)_k \dots (a_p)_k}{k!(b_1)_k \dots (b_q)_k} \cdot x^k $$

— m terms.

- Of a matrix argument

$$ {}_pF_q^\alpha(a_{1:p}; b_{1:q}; X) \approx \sum_{k=0}^{m} \sum_{\kappa \vdash k} \frac{(a_1)_\kappa \dots (a_p)_\kappa}{k!(b_1)_\kappa \dots (b_q)_\kappa} \cdot C_\kappa^\alpha(X), $$

$\sim e^{\sqrt{m}} n^m$ terms.

Because $\#\{|\kappa| \le m\} \approx e^{\sqrt{m}}$ and $C_\kappa^\alpha(x_{1:n})$ has $\sim n^{|\kappa|}$ terms.

Complexity of Approximating ${}_pF_q^\alpha$ – II

- Univariate

$$ {}_pF_q(a_{1:p}; b_{1:q}; x) \approx \sum_{k=0}^{m} \frac{(a_1)_k \ldots (a_p)_k}{k!(b_1)_k \ldots (b_q)_k} \cdot x^k $$

— m terms.

- Of a matrix argument

$$ {}_pF_q^\alpha(a_{1:p}; b_{1:q}; X) \approx \sum_{k=0}^{m} \sum_{\kappa \vdash k} \frac{(a_1)_\kappa \ldots (a_p)_\kappa}{k!(b_1)_\kappa \ldots (b_q)_\kappa} \cdot C_\kappa^\alpha(X), $$

$\sim e^{\sqrt{m}} n^m$ terms.

Because $\#\{|\kappa| \le m\} \approx e^{\sqrt{m}}$ and $C_\kappa^\alpha(X)$ has $\sim n^{|\kappa|}$ terms

(X is n-by-n).

- In contrast, new algorithm cost bounded by

$$ O(e^m \cdot n) $$

Idea: Dynamic Programming – I

$$ {}_pF_q^{\alpha}(a_{1:p}; b_{1:q}; X) \approx \sum_{k=0}^{m} \sum_{\kappa \vdash k} \underbrace{\frac{(a_1)_\kappa \dots (a_p)_\kappa}{k!(b_1)_\kappa \dots (b_q)_\kappa}}_{Q_\kappa} \cdot C_\kappa^\alpha(X) $$

Store every computed Q_κ and C_κ^α and only update.

If

$$ \begin{aligned} \kappa &= (\kappa_1, \kappa_2, \dots, \kappa_i, \dots) \\ \nu &= (\kappa_1, \kappa_2, \dots, \kappa_i - 1, \dots) \end{aligned} $$

Then

$$ \frac{Q_\kappa}{Q_\nu} = \frac{\prod_{j=1}^{p} (a_j + c)}{\prod_{j=1}^{q} (b_j + c)}, \quad \text{where} \quad c = -\frac{i-1}{\alpha} + \kappa_i - 1; $$

Cost: $2(p+q)$ instead of $|\kappa|(p+q)$.

Idea: Dynamic Programming – II

For the Jack function, look at the Schur function first ($\alpha = 1$).

$$\begin{aligned} s_{(1)}(x_{1:n}) &= x_1 + \ldots + x_n \\ s_{(2)}(x_{1:n}) &= \sum_{i \le j} x_i x_j \\ s_{(2,1)}(x_{1:n}) &= \sum_{i \ne j} x_i^2 x_j + 2 \sum_{i<j<k} x_i x_j x_k \\ s_{(1,1,1)}(x_{1:n}) &= \sum_{i<j<k} x_i x_j x_k \end{aligned}$$

In general:

$$\begin{aligned} s_\kappa(x_{1:n}) &= \sum_T x^T \\ C_\kappa^\alpha(x_{1:n}) &= \sum_T f_T(\alpha) x^T, \end{aligned}$$

where

- T ranges over all SSYT of shape κ
- $f_T(\alpha)$ is a rational function of α

Dynamic Programming for the Schur Function – I

$$\begin{aligned} s_{(1,1)}(x_{1:n}) &= \sum_{i<j} x_i x_j \\ &= x_1 x_2 + (x_1 + x_2) x_3 + \ldots + (x_1 + \ldots + x_{n-1}) x_n \end{aligned}$$

Algorithm:

1. **Precompute** $s_{(1)}(x_{1:i}) = \sum_{j=1}^{i} x_i$
2. $s_{(1,1)}(x_{1:n}) = s_{(1)}(x_{1:1}) x_2 + s_{(1)}(x_{1:2}) x_3 + \ldots + s_{(1)}(x_{1:n-1}) x_n$

Cost $O(n)$ instead of $O(n^2)$.

Dynamic Programming for the Schur Function – II

$$
\begin{aligned}
s_{(1,1)}(x_{1:n}) &= \sum_{i<j} x_i x_j \\
&= x_1 x_2 + (x_1 + x_2) x_3 + \ldots + (x_1 + \ldots + x_{n-1}) x_n
\end{aligned}
$$

Algorithm:

1. **Precompute** $s_i = \sum_{j=1}^{i} x_i$
2. $s_{(1,1)}(x_{1:n}) = s_{(1)}(x_{1:1}) x_2 + s_{(1)}(x_{1:2}) x_3 + \ldots + s_{(1)}(x_{1:n-1}) x_n$

Generalizes:

$$
s_\kappa(x_{1:n}) = \underbrace{\sum_{\mu \leq \kappa} s_\mu(x_{1:n-1}) \cdot x_n^{|\kappa| - |\mu|}}_{\#\text{ of terms independent of } n}
$$

Cost $O(n)$ instead of $O(n^{|\kappa|})$

Dynamic Programming for the Jack Function

- **Recursive formula for the Schur Function:**

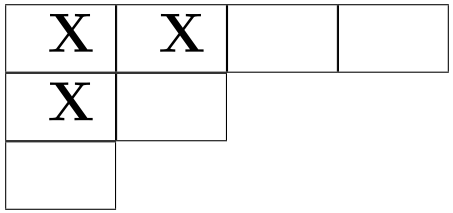
$$s_\kappa(x_{1:n}) = \sum_{\mu \le \kappa} s_\mu(x_{1:n-1}) \cdot x_n^{|\kappa|-|\mu|}$$

- **Recursive formula for the Jack function**

$$C_\kappa^\alpha(x_{1:n}) = \sum_{\mu \le \kappa} C_\mu^\alpha(x_{1:n-1}) \cdot x_n^{|\kappa|-|\mu|} \cdot \gamma_{\kappa\mu},$$

summation over κ/μ–horizontal strip, meaning

$$\mu_1 \ge \kappa_2; \quad \mu_2 \ge \kappa_3; \quad \mu_3 \ge \kappa_4 \ldots$$



Fewer than $M = \#\{|\kappa| \le m\}$ terms above. Need to compute for all $|\kappa| \le m$ and $1, 2, \ldots, n$, so complexity overall bounded by

$$M^2 \cdot n \approx e^m \cdot n$$

How do we store C_{κ}^{α} in memory efficiently?

- **Answer: Linearization of an m-tree.**
- **The partition $\kappa = (\kappa_1, \kappa_2, \ldots, \kappa_p)$, $|\kappa| \leq m$ has at most m children**

  $(\kappa_1, \kappa_2, \ldots, \kappa_p)$

  $(\kappa_1, \kappa_2, \ldots, \kappa_p, 1)$

  $\ldots$

  $(\kappa_1, \kappa_2, \ldots, \kappa_p, \kappa_p)$

  Partitions $|\kappa| \leq 6$:

(1)	(2)	(3)	(4)	(5)	(6)
(1,1)	(2,1), (2,2)	(3,1), (3,2), (3,3)	(4,1), (4,2)	(5,1)	

Easy case: $x_1 = x_2 = \ldots = x_n$

Then (in "J" normalization)

$$J^{\alpha}_{\kappa}(xI_n) = x^{|\kappa|} \prod_{(i,j)\in\kappa} (n - (i-1) + \alpha(j-1)),$$

which implies $J^{\alpha}_{\kappa}(xI_n) = J^{\alpha}_{\nu}(xI_n) \cdot x \cdot (n - (i-1) + \alpha(\kappa_i - 1))$.

No need to store J^{α}_{κ} in this case.

Performance

Convenient to test using the formula

$$ {}_0\boldsymbol{F}_{\boldsymbol{0}}^{\boldsymbol{\alpha}}(\boldsymbol{X}) = \boldsymbol{e}^{\mathbf{tr}(\boldsymbol{X})} $$

Resources

- **Slides, papers, and MATLAB software:**

  `math.mit.edu/~plamen`

- **Paper, this talk also in your folders**