

# EFFICIENT CALCULATION OF POLYNOMIAL FEATURES ON SPARSE MATRICES

**Andrew Nystrom**

Savvysherpa Inc.  
6200 Shingle Creek Pkwy  
Suite 400  
Minneapolis, MN 55430, USA  
awnystrom@gmail.com

**John Hughes**

Department of Computer Science  
Brown University  
Providence, RI  
jfh@cs.brown.edu

## ABSTRACT

We provide an algorithm for polynomial feature expansion that can operate directly on sparse a matrix. For a vector of dimensionality  $D$  and density  $d$ , the algorithm has time and space complexity  $O(d^k D^k)$  where  $k$  is the polynomial order.

## 1 INTRODUCTION

Polynomial feature expansion is a tool long used in statistics for approximating nonlinear functions (see Gergonne (1974); Smith (1918)). While their use is widespread, the authors of this work are unaware of any improvements made to their calculation efficiency. In this work we provide an algorithm for calculating polynomial features for a vector of dimensionality  $D$  and density  $d$  with time and space complexity  $O(d^k D^k)$  where  $k$  is the polynomial order. The density of a vector is the fraction of elements that are nonzero, so  $0 \leq d \leq 1$ . The standard algorithm has time and space complexity  $O(D^k)$ , so the added factor of  $d^k$  represents a significant reduction in complexity. The algorithm does not require the densification of the vector, e.i. the vector remains in compressed sparse row form, so the space complexity is also  $O(d^k D^k)$  as opposed to  $O(D^k)$ .

## 2 ALGORITHM

The traditional method of calculating polynomial features for a vector  $\vec{x}$  involves augmenting a feature for the product of each combination of size  $k$  with repetition (without repetition for interaction features) of features in  $\vec{x}$ . This method does not exploit data sparsity and will yield a product of zero any time one of the features involved in the product is zero. In a sparse matrix, products resulting in zero will be common. If the vectors are stored in a sparse matrix format, the products resulting in zero are unnecessary to compute since zero is the default value in a sparse matrix format.

The main idea behind our algorithm is to leverage sparsity by only computing products that do not involve zeros. In a compressed sparse row matrix, the columns containing nonzero data are the only columns that are stored. We can therefore iterate over products of combinations with repetition of order  $k$  of these columns for each row to calculate  $k$ -degree polynomial features.

While the idea is straightforward, there is yet one unaddressed caveat; Given a multiset of column indices whose corresponding nonzero components were multiplied to produce a polynomial feature, where in the augmented polynomial vector does the result of the product belong? To adress this, we give a bijective mapping from the set of possible column index combinations with repetition of order  $k$  onto the column index space of the polynomial feature matrix. More precisely, this the mapping is of the following form:

$$(i_0, i_1, \dots, i_{k-1}) \mapsto p_{i_0 i_1 \dots i_{k-1}} \in \{0, 1, \dots, \binom{D}{k}\} \forall (i_0, i_1, \dots, i_{k-1}) \quad (1)$$

such that  $0 \leq i_0 \leq i_1 \leq \dots \leq i_{k-1} < D$  where  $(i_0, i_1, \dots, i_{k-1})$  are column indicies of a row vector  $\vec{x}$  of an  $N \times D$  input matrix,  $p_{i_0 i_1 \dots i_{k-1}}$  is a column index into the polynomial expansion

vector for  $\vec{x}$  where the product of elements corresponding to indices  $i_0, i_1, \dots, i_{k-1}$  will be stored, and  $\binom{D}{k}$  is the number of combinations with repetition of size  $k$  drawn from  $D$  objects. In general,  $\binom{n}{k} = \binom{n+k-1}{n-1} = \binom{n+k-1}{k}$ , as given by Stanley (1986).

### 2.1 CONSTRUCTION OF MAPPINGS

We seek a map from matrix indices  $(i, j)$  (with  $i < j$  and  $0 \leq i < D$ ) to numbers  $f(i, j)$  with  $0 \leq f(i, j) < \frac{D(D-1)}{2}$ , one that follows the pattern indicated by

$$\begin{bmatrix} x & 0 & 1 & 3 \\ x & x & 2 & 4 \\ x & x & x & 5 \\ x & x & x & x \end{bmatrix} \quad (2)$$

where the entry in row  $i$ , column  $j$ , displays the value  $f(i, j)$ .

To simplify slightly, we introduce a notation for the  $n$ th triangular number,

$$T_2(n) = \frac{n(n+1)}{2} \quad (3)$$

The subscript 2 indicates that these are triangles in two dimensions; we'll use  $T_3(n)$  to indicate the  $n$ th tetrahedral number, and so on for higher dimensions.

Observe that in Equation 2, each entry in column  $j$  (for  $j > 0$ ) lies in the range

$$T_2(j-1) \leq e < T_2(j). \quad (4)$$

And in fact, the entry in the  $i$ th row of that column is just  $i + T_2(j-1)$ . Thus we have

$$f(i, j) = i + T_2(j-1) \quad (5)$$

$$= i + \frac{(j-1)j}{2} \quad (6)$$

$$= \frac{2i + j^2 - j}{2}. \quad (7)$$

For instance, in column  $j = 2$  in our example (the *third* column), the entries range from 1 to 2, while  $T_2(j-1) = T_2(1) = 1$  and  $T_2(j) = T_2(2) = 3$ , and the entry in column  $j = 2$ , row  $i = 1$  is  $i + T_2(j-1) = 1 + 1 = 2$ .

#### 2.1.1 OTHER INDICES

With one-based indexing in both the domain and codomain, the formula above becomes

$$f_1(i, j) = 1 + f(i-1, j-1) \quad (8)$$

$$= 1 + f(i-1, j-1) \quad (9)$$

$$= \frac{2 + 2(i-1) + (j-1)^2 - (j-1)}{2} \quad (10)$$

$$= \frac{2i + j^2 - 3j + 2}{2} \quad (11)$$

#### 2.1.2 POLYNOMIAL FEATURES

For polynomial features, we seek a map from matrix indices  $(i, j)$  (with  $i \leq j$  and  $0 \leq i < D$ ) to numbers  $g(i, j)$  with  $0 \leq g(i, j) < \frac{D(D+1)}{2}$ , one that follows the pattern indicated by

$$\begin{bmatrix} 0 & 1 & 3 & 6 \\ x & 2 & 4 & 7 \\ x & x & 5 & 8 \\ x & x & x & 9 \end{bmatrix} \quad (12)$$

i.e., essentially the same task as before, except that the diagonal is included. One can regard all but the last column of entries in Equation 12 as corresponding to the entries in Equation 2, but shifted to the left. Thus the formula for  $g(i, j)$  is simply the formula for  $f$ , shifted by 1, i.e.,

$$g(i, j) = f(i, j + 1) \quad (13)$$

$$= \frac{2i + (j + 1)^2 - (j + 1)}{2} \quad (14)$$

$$= \frac{2i + j^2 + j + 1}{2}. \quad (15)$$

Alternatively, we can write this as

$$g(i, j) = i + T_2(j), \quad (16)$$

and get the same result.

### 2.1.3 HIGHER DIMENSIONS

To handle three-way interactions, we need to map triples of indices in a 3-index array to a flat list, and similarly for higher-order interactions.

For three indices,  $i, j, k$ , with  $i < j < k$  and  $0 \leq i, j, k < D$ , we have a similar recurrence. Calling the mapping  $h$ , we have

$$h(i, j, k) = i + T_2(j - 1) + T_3(k - 2); \quad (17)$$

if we define  $T_1(i) = i$ , then this has the very regular form

$$h(i, j, k) = T_1(i) + T_2(j - 1) + T_3(k - 2); \quad (18)$$

and from this, the generalization to higher dimensions is straightforward. The formulas for “higher triangular numbers”, i.e., those defined by

$$T_k(n) = \sum_{i=1}^n T_{k-1}(n) \quad (19)$$

for  $k > 1$  can be determined inductively. For  $k = 3$ , the result is

$$T_3(n) = \sum_{i=1}^n T_2(n) \quad (20)$$

$$= \frac{n^3 + 3n^2 + 2n}{6}, \quad (21)$$

so that the formula for 3-way interactions, with zero-based indexing, becomes

$$h(i, j, k) = 1 + (i - 1) + \frac{(j - 1)j}{2} + \quad (22)$$

$$\frac{(k - 2)^3 + 3(k - 2)^2 + 2(k - 2)}{6}. \quad (23)$$

### 2.1.4 HIGHER-DIMENSION POLYNOMIAL FEATURES

For the case where we include the diagonal in higher dimensions, we must shift  $j$  by 1,  $k$  by 2, and so on, and the formula becomes

$$\ell(i, j, k) = T_1(i) + T_2(j) + T_3(k), \quad (24)$$

with analogous formulas for higher degree polynomial interactions.

### 3 ANALYSIS

#### 3.1 ANALYTICAL

Calculating  $k$ -degree polynomial features via our method for a vector of dimensionality  $D$  and density  $d$  requires  $dD$  choose  $k$  (with repetition) products. The big O of the algorithm is therefore given by

$$O\left(\binom{dD+k-1}{k}\right) = O\left(\frac{(dD+k-1)!}{k!(dD-1)!}\right) \quad (25)$$

$$= O\left(\frac{(dD+k-1)(dD+k-2)\dots(dD)}{k!}\right) \quad (26)$$

$$= O((dD+k-1)(dD+k-2)\dots(dD)) \text{ for } k \ll dD \quad (27)$$

$$= O(d^k D^k) \quad (28)$$

#### 3.2 EMPIRICAL

### 4 CONCLUSION

#### REFERENCES

- JD Gergonne. The application of the method of least squares to the interpolation of sequences. *Historia Mathematica*, 1(4):439–447, 1974.
- Kirstine Smith. On the standard deviations of adjusted and interpolated values of an observed polynomial function and its constants and the guidance they give towards a proper choice of the distribution of observations. *Biometrika*, 12(1/2):1–85, 1918.
- Richard P Stanley. *What Is Enumerative Combinatorics?* Springer, 1986.