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

ABSTRACT

We provide an algorithm for polynomial feature expansion that operates directly on a sparse matrix. For a vector of dimension 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 has long been used in statistics to approximate nonlinear functions Gergonne (1974); Smith (1918). Despite this, we are unaware of any efforts to optimize calculating them. Here we provide an algorithm for calculating polynomial features for a vector of dimension D and density d with time and space complexity $O(d^k D^k)$ where k is the polynomial order, and $0 \leq d \leq 1$ is the fraction of elements that are nonzero. The standard algorithm has time and space complexity $O(D^k)$, so the added factor of d^k represents a significant complexity reduction. The algorithm avoids densification of the vector, i.e. 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

In the naive computation of polynomial features for a vector $\vec{x}$, we create a new feature for each product (with repetition) of k features in $\vec{x}$ (or without repetition, for "interaction features"). This ignores 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, such zero-products are common. If we store vectors in a sparse matrix format, these zero-products need not be computed or stored.

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 *only these columns* for each row to calculate k-degree polynomial features.

While the idea is straightforward, there is yet an unaddressed challenge: 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 address 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. Thus the map has the form

$$(i_0, i_1, \ldots, i_{k-1}) \rightarrowtail\!\!\!\!\!\rightarrow p_{i_0 i_1 \ldots i_{k-1}} \in \{0, 1, \ldots, \binom{D}{k}\} \tag{1}$$

such that $0 \leq i_0 \leq i_1 \leq \cdots \leq i_{k-1} < D$ where $(i_0, i_1, \ldots, i_{k-1})$ are column indicies of a row vector $\vec{x}$ of an $N \times D$ input matrix, and $p_{i_0 i_1 \ldots i_{k-1}}$ is a column index into the polynomial expansion vector for $\vec{x}$ where the product of elements corresponding to indices $i_0, i_1, \ldots, i_{k-1}$ will be stored.

2.1 CONSTRUCTION OF MAPPINGS

For the second degree case, 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

$$\begin{bmatrix} x & 0 & 1 & 3 \\ x & x & 2 & 4 \\ x & x & x & 5 \\ x & x & x & x \end{bmatrix} \tag{2}$$

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

To simplify slightly, we introduce a notation for the nth triangular number,

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

The subscript 2 indicates that these are triangles in two dimensions; we'll use $T_3(n)$ to indicate the nth 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

$$T_2(j-1) \leq e < T_2(j). \tag{4}$$

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

$$f(i, j) = i + T_2(j-1) \tag{5}$$

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

$$= \frac{2i + j^2 - j}{2}. \tag{7}$$

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

$$f_1(i, j) = 1 + f(i-1, j-1) \tag{8}$$

$$= 1 + f(i-1, j-1) \tag{9}$$

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

$$= \frac{2i + j^2 - 3j + 2}{2} \tag{11}$$

2.1.2 POLYNOMIAL FEATURES

For polynommial 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 $0 \leq f(i, j) < \frac{D(D+1)}{2}$, one that follows the pattern indicated by

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

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.,

$$g(i, j) = f(i, j+1) \tag{13}$$

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

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

Alternatively, we can write this as

$$g(i,j) = i + T_2(j), \tag{16}$$

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

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

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

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

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

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

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

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

$$= \frac{n^3 + 3n^2 + 2n}{6}, \tag{21}$$

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

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

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

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

$$\ell(i,j,k) = T_1(i) + T_2(j) + T_3(k), \tag{24}$$

with analogous formulas for higher degree polynomial interactions.

3 COMPLEXITY ANALYSIS

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

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

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

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

$$= O\left(d^k D^k\right) \tag{28}$$

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.