# Efficient Calculation of Interaction Features on Sparse Matrices

Andrew Nystrom

**Abstract**

FILL THIS IN

## 1 Introduction

Introduction Interaction features are a way of capturing correlations between features in a machine learning setting. A feature vector $\vec{x}$ of dimensionality $D$ has second degree interaction features $\{x_i \cdot x_j : i, j \in \{0, 1, ..., D-1\} \wedge i < j\}$, so a $D$ dimensional vector has $\binom{D}{2} = \frac{D^2-D}{2}$ second degree interaction features. A naive approach to calculating these features is to simply iterate through the combinations of the column indices. For a sparse vector, many of the resulting interaction features would be zero, and could therefore be ignored. This work describes a method to efficiently calculate second degree interaction features for a sparse matrix that has time and space complexities that decrease quadratically with the density of the input matrix with respect to the naive approach.

## 2 Approach

Let the list of nonzero columns for a given row $\vec{x}$ be denoted by $N_{zc}$. The nonzero second degree interaction features are simply the products of all combinations of two elements whose columns are in $N_{zc}$. However, to properly place an interaction feature into the correct column, a mapping from the column index pairs of $N_{zc}$ into the columns of the interaction matrix is needed. The mapping is from the space (a, b) where a,b are in $1, 2, ..., D$ onto the space $1, 2, ..., \frac{D^2-D}{2}$. This is isomorphic to mapping the coordinates of the upper triangle of a matrix onto a flat list. The following is a proof by construction for such a mapping.

INSERT JOHN'S PROOF HERE

With this mapping, an algorithm for generating second degree interaction features on a matrix $A$ can be formulated as follows:

SPARSE INTERACTION($A$)

$\text{map}(a, b) = \frac{2Da-a^2+2b-3a-2}{2}$
$N$ = row count of $A$
$D$ = column count of $A$
$B$ = Compressed Sparse Row Matrix of size $N \times \frac{D^2-D}{2}$
**for** *row* in $A$
  $N_{zc}$ = nonzero columns of *row*
  **for** $i = 0$ **to** $|N_{zc}| - 1$
    **for** $j = i + 1$ **to** $|N_{zc}|$
      $k = \text{map}(i, j)$
      $r$ = index of *row*
      $B[r, k] = row[i] \cdot row[j]$

## 3 Complexity Analysis

Assume that A is a matrix with sparsity $0 < d < 1$, $N$ rows, and $D$ columns. Finding interaction features with the proposed algorithm has time and space complexity $O(dND^2)$, whereas a naive approach of using non-sparse matrices and multiplying all column combinations has time and space complexity $O(ND^2)$. The algorithm is therefore an improvement by a factor of the density factor of $A$.

This can represent a large gain in speed and time. For example, the 20 Newsgroups dataset has density $d$ of 0.12 when its unigrams are represented in a vector space model. This means the proposed approach would take less than $\frac{1}{8}$ time and memory.

The real benefit of this method is revealed when the average complexity is analysed. The number of interactions calculated for a given row are $\binom{|N_{zc}|}{2}$. If the matrix has density $d$, then on average, $N_{zc} = Dd$, so the number of interaction features calculated in total is

$$N\binom{dD}{2} = N\frac{(Dd)!}{2!(Dd-2)!}$$

$$= N\frac{(D^2d^2 - Dd)}{2}$$

This means that the average complexity decreases quadratically with the density.

# 4 Future Work

The approach for generating second degree interaction features required a mapping from combinations of two to the space $1, 2, ..., \frac{D^2-D}{2}$, which is isomorphic to a mapping from the indices of an upper triangular matrix to the indices of a flat list of the same size. To generate third degree interaction features, a mapping from combinations of three $(a, b, c)$ to the space $1, 2, ...\frac{D^3-3D^2+2D}{6}$ (which is $\binom{D}{3}$), or the upper 3-simplex of a tensor to a flat list of the same size $\frac{D^3-3D^2+2D}{6}$ would be required. In general, for interaction features of degree k, the upper $k$-simplex of a $k$-dimensional tensor must be mapped to the space $1, 2, ...\frac{D!}{k!(D-k)!}$. A similar approach for finding these mappings could be taken as the one used here for $k = 2$.

Motivation for deriving mapping functions for higher orders of interaction features is that the average complexity of generating degree $k$ interaction features is $N\binom{Dd}{k}$, which decreases polynomially with respect to k compared to generating the features naively.