Sparse Embedding Matrices

October 19th Lecture Notes

Lecturer: Anil Damle

Scribe: Scott Wu (ssw74)

Sparse Embedding Matrices

We can use a *subspace embedding* matrix S to estimate the column space of A within a given epsilon and with high probability. The epsilon restricts the estimated norm of the range of SA to the norm of the range of A. This subspace embedding matrix can be applied to algorithms we've seen previously as a tradeoff between computation and accuracy.

More precisely, we would like to find a matrix $S \in \mathbb{R}^{t \times m}$ such that $\forall y \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$,

Eq 1

$$(1-\varepsilon)\|Ay\|_2 \leq \|SAy\|_2 \leq (1+\varepsilon)\|Ay\|_2$$

Clarkson and Woodruff (2013) describe the use of the *count sketch* matrix (Charikar et. al. 2004) to generate such a subspace embedding matrix. The count sketch matrix has the advantage over other subspace embedding matrices of being able to compute SA in $O\big(\text{nnz}(A)\big)$ time, where $\text{nnz}(A)$ is the number of non-zero entries in A. For comparison, the best previous algorithm requires $O(mn \log t)$ time to find such a matrix S. Since the count sketch matrix is sparse, Clarkson and Woodruff call S a *sparse embedding* matrix.

Construction

To construct the sparse embedding matrix S, we follow the same algorithm for generating a count sketch matrix. For every column in S, we add one non-zero entry in a uniformly chosen random row. We choose the non-zero entry to be either 1 or -1 with probability $\frac{1}{2}$ each.

Let A be an $m \times n$ matrix of rank r. Let S be a $t \times m$ matrix, where $t < m$.

Eq 2

$$\text{For all } j = 1 \ldots m, \qquad S_{h(j),j} = \begin{cases} 1 & \text{with probability } \frac{1}{2} \\ -1 & \text{with probability } \frac{1}{2} \end{cases}$$

The function $h(j)$ defines a hashing function with column j as the input. For our count sketch matrix, we can let $h(j)$ select an integer from 1 to t with uniformly random distribution. Ideally, we would want $h(j)$ to be a perfect hashing function, such that each column hashes to a unique row. In this case, every column has exactly one entry, and every row has at most one entry.

Eq 3

$$\forall j, j', j \neq j' \rightarrow h(j) \neq h(j')$$

Since we choose $t < m$ (and ideally $t \ll m$), there is a high probability that $h(j)$ will be perfectly hashed, or very close to perfectly hashed, because the domain is smaller than the range.

Selection of t

We defined the size of the sparse embedding matrix to be dependent on the number of rows in A and an arbitrary variable t. We want t to be smallest value which satisfies the epsilon inequality in Eq 1.

Clarkson and Woodruff proved that $t = O\left(\text{polylog}\left(\frac{n}{\varepsilon}\right)\right)$ for large matrices or $t = O\left(\text{polylog}\left(\frac{r}{\varepsilon}\right)\right)$ for high rank matrices, where polylog(x) represents some combination of polynomials and logarithms. In this case, these polynomials typically have a degree greater than 2.

Intuitively these bounds make sense as well. As ε decreases, t must increase to meet tighter bounds. As the rank of A increases, t must increase to better represent the column space. As n, the number of columns, or r, the rank, increases, t must increase to satisfy the potentially larger column space.

Proof

Given some fixed vector y, the norm of Sy is within the norm of y with high probability $1 - e^{-n}$.

Eq 4

$$(1-\varepsilon)\|y\|_2 \leq \|Sy\|_2 \leq (1+\varepsilon)\|y\|_2$$

Since we would like for S to preserve the norm for range(A), which may be a large set of vectors y, we will have to sacrifice the property of high probability.

We pick of set of vectors which form an "ε-net" on vectors in range(A). Since this is scale invariant, we can limit these vectors to those of unit length. These vectors cover the range(A).

Eq 5

$$\forall x \in \text{range}(A), \exists y \in \varepsilon\text{-net s.t. } \|x-y\|_2 \leq \varepsilon$$

The size of this net grows exponentially with dimensionality. For example, to achieve this net in a single dimension we would only need two vectors ($+\varepsilon$ and $-\varepsilon$). For every additional dimension, both directions are possible in each dimension, thus increasing exponentially in rank r or n.

A union bound over this set states that if we have a given probability for a fixed y, then the probability over all y must be less than or equal to the sum of each individual y.

Finally, for a sufficiently fine net, linearity implies that Eq 4 holds for all y in range(A). This means that as long as the vectors that weren't chosen for the ε-net are close to the something in the net, that vector will also satisfy the inequality.

Analysis

The construction of S is oblivious to A, which means A does not even have to be accessed to construct S. It also means that it can usually be applied generically to any computation. Bad sparse embeddings could arise if the non-zero entries mix important information in A, or if a certain row is entirely skipped (no $h(j)$ hashes to that row).

Since we really care about the range of A (and how S works with the range of A), we only need an orthonormal basis of the range of A to study A. In particular, this could be the left singular values of the SVD of A (U in $U\Sigma V^T$). We want to prevent S from mixing norms of U (leverage scores of A) that are large, because those embed the bulk of the information in A.

One key result from Clarkson and Woodruff's paper is identifying how many rows of A can have large leverage scores. For any $\alpha \leq 1$, there is a fixed set H that depends upon range(A) of size $\frac{n}{\alpha}$

such that for any unit vector y in range(A), H contains all the indices where y is larger than $\sqrt{\alpha}$ in magnitude. This turns out to also be the set of large leverage scores of A.

Eq 6

$$\|y\|_2 > \sqrt{\alpha}$$

If t is sufficiently large, then with high probability, no two distinct indices $j, j' \in [1, m]$ such that they hash to the same row.

Eq 7

$$t \geq k|H|^2 \rightarrow \forall j, j' \in [1, m], j \neq j', h(j) \neq h(j')$$

This implies that with high probability, the rows with large leverage scores are perfectly hashed.

Comparison to the Johnson Lindenstrauss Transform

The Fast Johnson Lindenstrauss Transform (FJLT) transform also generates a subspace embedding matrix. Like the sparse embedding matrix, the FJLT is oblivious to A, and chooses t independent of m. The t chosen by the FJLT transform is proportional to $\frac{n}{\varepsilon^2}$, which is generally smaller than that of the sparse embedding matrix. However, the FJLT produces a dense matrix, and requires O$(mn \log t)$ time to compute SA, whereas the sparse embedding matrix only require $O(\text{nnz}(A))$.

Application to LSRN

One direct application of the sparse embedding matrix is to over constrained least squares problems. One drawback of using LSRN, is that it requires a matrix product, which can be expensive for dense matrices. Instead of solving min$\|Ax - b\|_2$, we can solve min$\|S(Ax - b)\|_2$. That is, if x^* solves min$\|S(Ax^* - b)\|_2$, then min$\|Ax^* - b\|_2$ is within ε of the actual solution.

Using the sparse embedding, we are solving an n dimensional space problem in a t dimensional space. Solutions in the t dimensional space generalize to the n dimensional space, especially if A has low rank.

Clarkson and Woodruff propose solving this problem in $O\left(\text{nnz}(A) + O\left(\frac{d^3}{\varepsilon^{-2}} \log^7\left(\frac{d}{\varepsilon}\right)\right)\right)$ time and with probability $\frac{2}{3}$ by computing SA and Sb in $O\big(\text{nnz}(A)\big)$ time, and then using the FJLT to solve the remaining LS problem. The authors also propose an iterative solution by preconditioning the matrix and applying Krylov or CG methods in $O\left(\text{nnz}(A) \log\left(\frac{n}{\varepsilon}\right) + r^3 \log^2 r + r^2 \log\left(\frac{1}{\varepsilon}\right)\right)$ time.

References

Kenneth L. Clarkson and David P. Woodruff (2012). Low Rank Approximation and Regression in Input Sparsity Time. CoRR, https://arxiv.org/abs/1207.6365.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. Theor. Comput. Sci., 312(1):3–15, 2004. https://genfaculty.rutgers.edu/uploads/25/FrequentStream.pdf.

Nir Ailon and Bernard Chazelle (2009). The Fast Johnson-Lindenstrauss Transformation and Approximate Nearest Neighbors. https://www.cs.princeton.edu/~chazelle/pubs/FJLT-sicomp09.pdf.