# Efficient Anomaly Detection via Matrix Sketching

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.


**Vatsal Sharan**
Stanford University
vsharan@stanford.edu

**Parikshit Gopalan**
**VMware Research**
**pgopalan@vmware.com**

**Udi Wieder**
**VMware Research**
**uwieder@vmware.com**

**PPT maker:** 謝幸娟 (Hsing-Chuan Hsieh)
connie0915549431@hotmail.com

# 1. Introduction

# 1. Introduction

- **Context**

➢ Anomaly detection (AD) in high-dimensional numeric (streaming) data is a ubiquitous problem in machine learning
  - E.g., parameters regarding the health of machines in a data-center

➢ Popular PCA/subspace based AD is used
  - E.g. of anomaly scores: rank-k leverage scores , rank-k projection distance…

# 1. Introduction

- **Computational challenge**

➢ Dimension of the data matrix $A \in R^{n \times d}$ may be very large
  - E.g. of the data center: $d \approx 10^6$ and $n \gg d$

➢ The quadratic dependence on *d* renders standard approach of AD inefficient in high dimensions

- Note: the standard approach to compute anomaly scores from k PC's in a streaming setting:
  1. Compute covariance matrix $A^T A \in R^{d \times d}$ → space ~ $O(d^2)$ ; time ~ $O(nd^2)$
  2. then compute the top k PC's for anomaly scores

# 1. Introduction

- **Requirement for an algorithm to be efficient**

1. As *n* too large:
   the algorithm must work in a streaming fashion where it only gets a constant number of passes over the dataset stored in memory.

2. As d too large:
   the algorithm should ideally use memory linear (i.e. $O(d)$) or even sublinear in d (e.g. $O(\log(d))$)

# 1. Introduction

- **Purpose**

1. Provide simple and practical algorithms at a significantly lower cost in terms of time and memory. I.e., $O(d^2)$ → $O(d)$ or $O(\log(d))$

   - using popular matrix sketching techniques: $A \in R^{n \times d}$ → $\tilde{A} \in R^{l \times d}$ $(\mathrm{l} \ll \mathrm{n})$ or $\tilde{A} \in R^{n \times l}$ $(\mathrm{l} \ll d)$
   - $\tilde{A}$ preserves some desirable properties of the large matrix $A$

2. Prove that estimated subspace-based anomaly scores computed from $\tilde{A}$ approximate the true anomaly scores

# 2. Notation and Setup

# 2. Notation and Setup

- Data matrix $A \in R^{n \times d}$ s.t.

$$A = \left[a_{(1)}^T; \dots; a_{(n)}^T\right] \, where \; a_{(i)} \in R^d$$
$$= \left[a^{(1)}, \dots, a^{(d)}\right] \, where \; a^{(i)} \in R^n$$

- SVD of A: $A = U \Sigma V^T \qquad where$

$$\Sigma = diag(\sigma_1, \dots, \sigma_d), \sigma_1 \geq \cdots \geq \sigma_d > 0$$
$$V = [v^{(1)}, \dots, v^{(d)}]$$

- Condition number of the top k subspace of A: $\kappa_k = \sigma_1^2 / \sigma_k^2$
- Frobenius norm: $\|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{d} |a_{ij}|^2}$
- Operator norm: $\|A\| = \sigma_1$

# 2. Notation and Setup

- **Subspace based measures of anomalies** for each instance $a_{(i)} \in R^d$

➢ Mahalanobis distance/ leverage score:

$$L(i) = \sum_{j=1}^{d} (a_{(i)}^T v^{(j)} / \sigma_i)^2$$

  - $L(i)$ is highly sensitive to smaller singular values

➢ Rank k leverage score $(k \ll d)$:

$$L^k(i) = \sum_{j=1}^{k} (a_{(i)}^T v^{(j)})^2 / \sigma_j^2$$

  - real world data sets often have most of their signal in the top singular values

➢ Rank k projection distance:

$$T^k(i) = \sum_{j=k+1}^{d} (a_{(i)}^T v^{(j)})^2$$

  - to catch anomalies that are far from the principal subspace

# 2. Notation and Setup

- **Subspace based measures of anomalies** for each instance $a_{(i)} \in R^d$

➢Illustration of Rank k  leverage score/projection distance

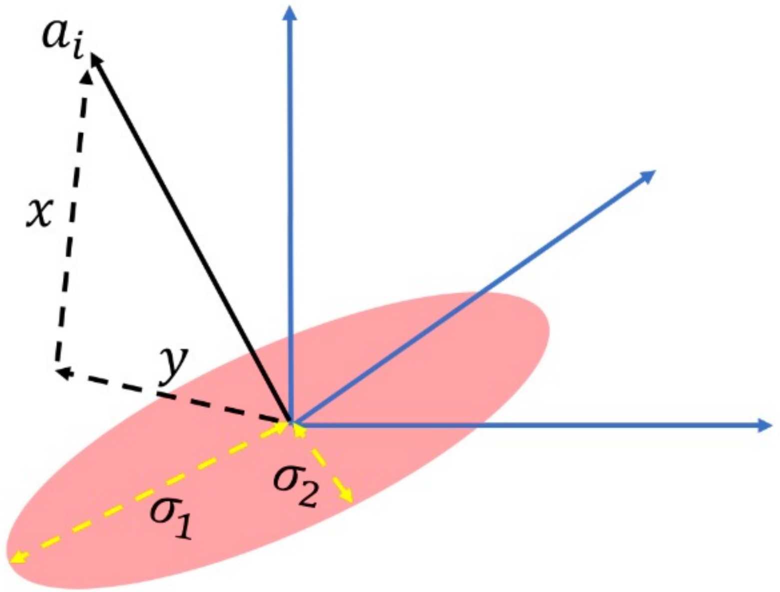


Figure 1: Illustration of subspace based anomaly scores. Here, the data lies mostly in the $k = 2$ dimensional principal subspace shaded in red. For a point $a_{(i)}$ the rank-$k$ projection distance equals $\|x\|_2$, where $x$ is the component of $a_{(i)}$ orthogonal to the principal subspace. The rank-$k$ leverage score measures the *normality* of the projection $y$ onto the principal subspace.

# 2. Notation and Setup

- **Assumptions**

1. Separation assumption
   - In case of degeneracy (i.e., $\sigma_k^2 = \sigma_{k+1}^2$) → $L^k(i), T^k(i)$ misdefined

**Assumption 1.** *We define a matrix* $\mathbf{A}$ *as being* $(k, \Delta)$*-separated if* $\sigma_k^2 - \sigma_{k+1}^2 \geq \Delta\sigma_1^2$. *Our results assume that the data are* $(k, \Delta)$*-separated for* $\Delta > 0$.

2. Approximate low-rank assumption
   - that the top-k principal subspace captures a constant fraction (at least 0.1) of the total variance in the data

**Assumption 2.** *We assume the matrix* $\mathbf{A}$ *is approximately rank-k, i.e.,* $\sum_{i=1}^{k} \sigma_i^2 \geq (1/10) \sum_{i=1}^{d} \sigma_i^2$.

- Assumption 2 captures the setting where the scores $L^k$ and $T^k$ are most meaningful.
- Our algorithms work best on data sets where relatively few principal components explain most of the variance.

# 3. Guarantees for anomaly detection via sketching

- Our main results say that given $\mu > 0$ and a (k, Δ)-separated matrix A $\in R^{n\times d}$ with top singular value $\sigma_1$, any sketch $\tilde{A} \in R^{l\times d}$ satisfying
$$\|\underbrace{A^T A}_{\mathrm{d}\times d} - \tilde{A}^T\tilde{A}\| \leq \mu\sigma_1^2 \tag{2}$$
or a sketch $\tilde{A} \in R^{n\times l}$ satisfying
$$\|\underbrace{AA^T}_{n\times n} - \tilde{A}\tilde{A}^T\| \leq \mu\sigma_1^2 \tag{3}$$
can be used to approximate rank *k* leverage scores and the projection distance from the principal *k*-dimensional subspace.

- Explanation:
  1. Equation (2): an approximation to the covariance matrix of the row vectors
  2. Equation (3): an approximation to the covariance matrix of the column vectors.

# 3. Guarantees for anomaly detection via sketching

- Next, we show how to design efficient algorithms for finding anomalies in a streaming fashion.

- **Pointwise guarantees from row space approximations**

➢Algorithm 1: random column projection/ row space approximation $(n \rightarrow l)$

---

**Algorithm 1:** Algorithm to approximate anomaly scores using Frequent Directions

---

**Input**: Choice of $k$, sketch size $\ell$ for Frequent Directions [26]
**First Pass:**
  Use Frequent Directions to compute a sketch $\tilde{\mathbf{A}} \in \mathbb{R}^{\ell \times d}$
**SVD:**
  Compute the top $k$ right singular vectors of $\tilde{\mathbf{A}}^T \tilde{\mathbf{A}}$
**Second Pass:** *As each row $a_{(i)}$ streams in,*
  Use estimated right singular vectors to compute leverage scores and projection distances

---

- Sketching techs: Frequent Directions sketch [18], row-sampling [19], random column projection, …

# 3. Guarantees for anomaly detection via sketching

- **Pointwise guarantees from row space approximations**

**Theorem 1.** *Assume that* $\mathbf{A}$ *is* $(k, \Delta)$*-separated. There exists* $\ell = k^2 \cdot \mathrm{poly}(\varepsilon^{-1}, \kappa_k, \Delta)$*, such that the above algorithm computes estimates* $\tilde{T}^k(i)$ *and* $\tilde{L}^k(i)$ *where*

$$|T^k(i) - \tilde{T}^k(i)| \leq \varepsilon \|a_{(i)}\|_2^2,$$

$$|L^k(i) - \tilde{L}^k(i)| \leq \varepsilon k \frac{\|a_{(i)}\|_2^2}{\|\mathbf{A}\|_F^2}.$$

*The algorithm uses memory* $O(d\ell)$ *and has running time* $O(nd\ell)$*.*

$\rightarrow O(d)$ $\rightarrow O(nd)$ ,since $l$ is independent of d

- Thm 1 improves on the trivial $O(d^2)$ bound of memory/ time

# 3. Guarantees for anomaly detection via sketching

- Next we show that substantial savings are unlikely for any algorithm with strong pointwise guarantees:
  there is an $\Omega(d)$ lower bound for any approximation

**Theorem 2.** *Any streaming algorithm which takes a constant number of passes over the data and can compute a* $0.1$ *error additive approximation to the rank-*$k$ *leverage scores or the rank-*$k$ *projection distances for all the rows of a matrix must use* $\Omega(d)$ *working space.*

# 3. Guarantees for anomaly detection via sketching

- **Average-case guarantees from columns space approximations**

➢ Even though the sketch gives column space approximations (i.e. $AA^T$ approximation satisfying Equ. (3))

➢ Our goal is still to compute the row anomaly scores from $\tilde{A}^T\tilde{A}$

- E.g., by random matrix $\boldsymbol{R} \in \boldsymbol{R}^{d\times l}$ s.t. $r_{ij} \overset{i.i.d}{\sim} U\{\pm 1\}$) where $l = O(k/\mu^2)$ [27]

→sketch $\widetilde{\boldsymbol{A}} = \boldsymbol{AR}$ for which returns the anomaly scores

➢ Space consumption

- For $\tilde{A}^T\tilde{A}$ → $O(l^2)$
- For $\boldsymbol{R}$ able to be pseudorandom [28-30] → $O(\log(d))$

# 3. Guarantees for anomaly detection via sketching

- **Average-case guarantees from columns space approximations**
- Algorithm 2: random row projection/ column space approximation

**Algorithm 2:** Algorithm to approximate anomaly scores using random projection

**Input**: Choice of $k$, random projection matrix $\mathbf{R} \in \mathbb{R}^{d \times \ell}$

**Initialization**

Set covariance $\tilde{\mathbf{A}}^T \tilde{\mathbf{A}} \leftarrow 0$

**First Pass:** *As each row $a_{(i)}$ streams in,*

Project by $\mathbf{R}$ to get $\mathbf{R}^T a_{(i)}$

Update covariance $\tilde{\mathbf{A}}^T \tilde{\mathbf{A}} \leftarrow \tilde{\mathbf{A}}^T \tilde{\mathbf{A}} + (\mathbf{R}^T a_{(i)})(\mathbf{R}^T a_{(i)})^T$

**SVD:**

Compute the top $k$ right singular vectors of $\tilde{\mathbf{A}}^T \tilde{\mathbf{A}}$

**Second Pass:** *As each row $a_{(i)}$ streams in,*

Project by $\mathbf{R}$ to get $\mathbf{R}^T a_{(i)}$

For each projected row, use the estimated right singular vectors to compute the leverage scores and projection distances

# 3. Guarantees for anomaly detection via sketching

- **Average-case guarantees from columns space approximations**

**Theorem 3.** *For $\varepsilon$ sufficiently small, there exists $\ell = k^3 \cdot \mathrm{poly}(\varepsilon^{-1}, \Delta)$ such that the algorithm above produces estimates $\tilde{L}^k(i)$ and $\tilde{T}^k(i)$ in the second pass, such that with high probabilty,*

$$\sum_{i=1}^{n} |T^k(i) - \tilde{T}^k(i)| \le \varepsilon \|\mathbf{A}\|_{\mathrm{F}}^2,$$

$$\sum_{i=1}^{n} |L^k(i) - \tilde{L}^k(i)| \le \varepsilon \sum_{i=1}^{n} L^k(i).$$

*The algorithm uses space $O(\ell^2 + \log(d)\log(k))$ and has running time $O(nd\ell)$.*

- i.e., on average random projections can preserve leverage scores and distances from the principal subspace
- $l$ = poly(k, $\varepsilon^{-1}$, Δ), indep. of both n and d.

# 4. Experimental evaluation

- **Goal**

1. to test whether our algorithms give comparable results to exact anomaly score computation based on full SVD.

2. to determine how large the parameter $l$ (determining the size of the sketch) needs to be to get close to the exact scores

Note: The experiment is in backend mode (i.e. not online mode)

# 4. Experimental evaluation

- **Data:** (1) p53 mutants [32], (2) Dorothea [33], (3) RCV1 [34]
  - ➢All available from the UCI Machine Learning Repository,

Table 1: Running times for computing rank-$k$ projection distance. Speedups between $2\times$ and $6\times$.

| Dataset | Size ($n \times d$) | $k$ | $\ell$ | SVD | Column Projection | Row Projection |
|---|---|---|---|---|---|---|
| p53 mutants | $16772 \times 5409$ | 20 | 200 | 29.2s | 6.88s | 7.5s |
| Dorothea | $1950 \times 100000$ | 20 | 200 | 17.7s | 9.91s | 2.58s |
| RCV1 | $80442 \times 47236$ | 50 | 500 | 39.6s | 17.5s | 20.8s |

- **Ground truth:** to decide anomalies
  - We compute the rand k anomaly scores using a full SVD, and then label the η fraction of points with the highest anomaly scores to be outliers.
    1. k chosen by examining the explained variance of the datatset: typically between (10, 125)
    2. η chosen by examining the histogram of the anomaly score; typically between (0.01, 0.1)

# 4. Experimental evaluation

- **Experimental design**
  - 3 datasets  x 2 algorithms x 2 anomaly scores
  - Parameters *(depending on dataset)* :
    1. *k: 3 settings*
    2. *l: 10 settings ranging (2k, 20k)*
  - ➢ ∴ 3*2*2*3*10=360 combinations
  - ➢ iterate for 5 runs for each combination

- **Measuring accuracy**
  1. After computing anomaly scores for each dataset, we then declare the points with the top *η’* fraction of scores to be anomalies (without knowing η)
  2. Then compute the F1 score
     - ➢ Note: we choose the value of *η’* which maximizes the F1 score
  3. Report the average F1 score over 5 runs

# 4. Experimental evaluation

- **Performance of dataset:** p53 mutants

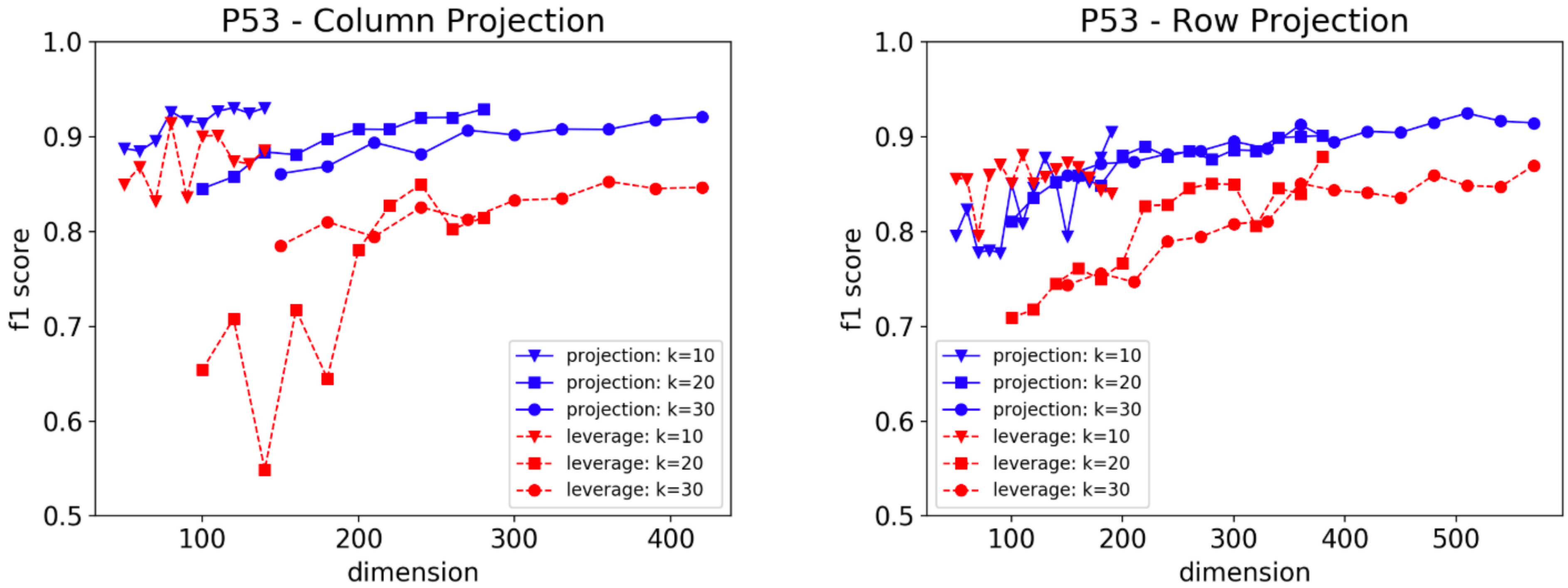


Figure 2: Results for P53 Mutants. We get $F_1$ score $> 0.8$ with $> 10\times$ space savings.

# 4. Experimental evaluation

- **Performance of dataset:** Dorothea

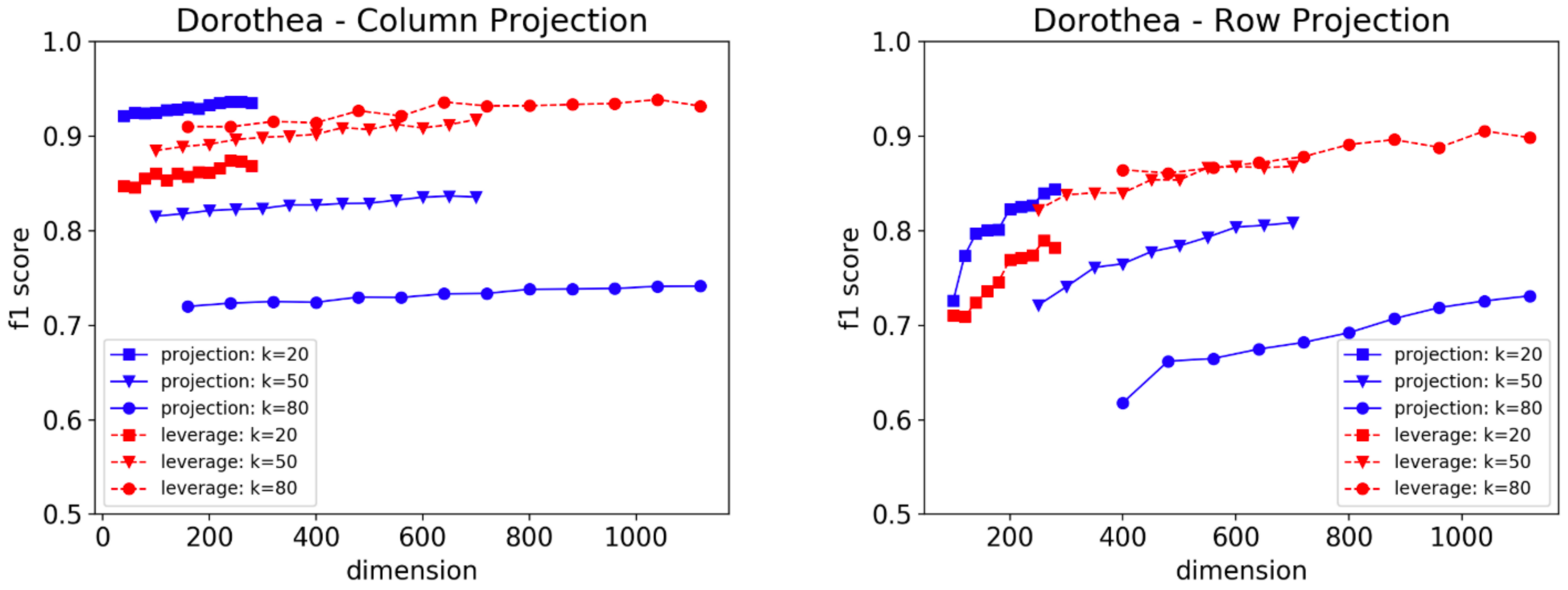


Figure 3: Results for the Dorothea dataset. Column projections give more accurate approximations, but they use more space.

# 4. Experimental evaluation

- **Performance of dataset:** RCV1

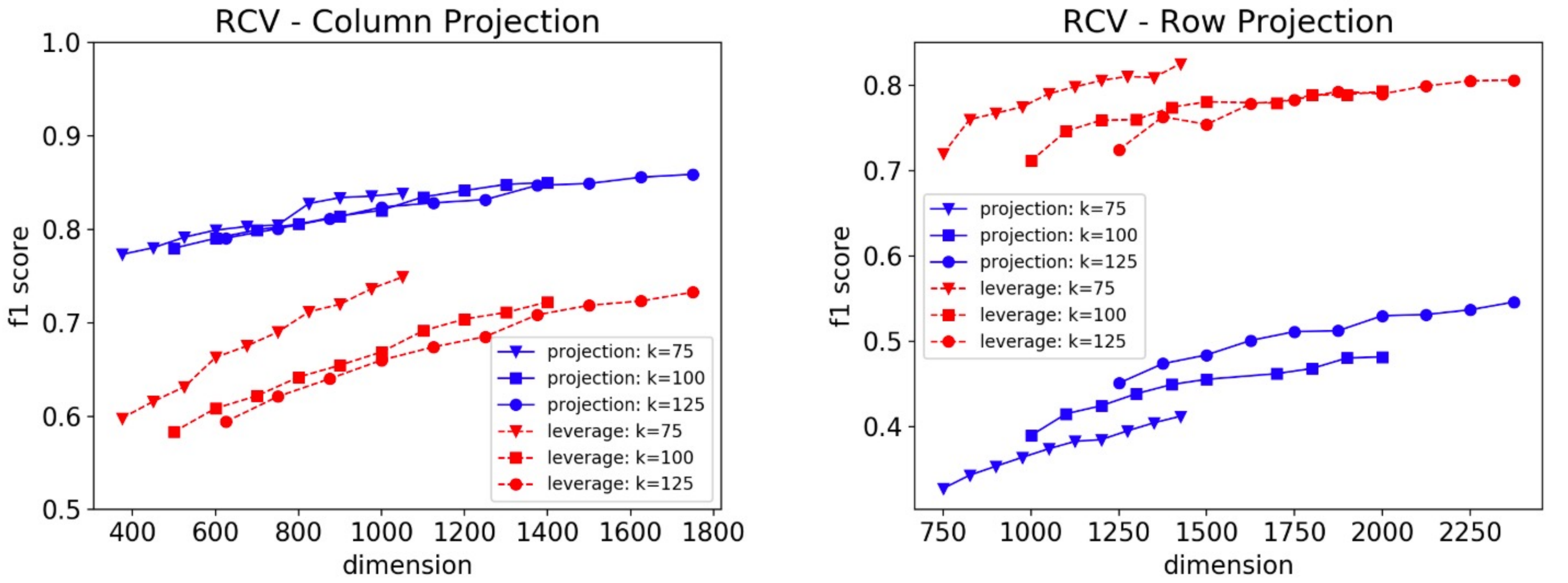


Figure 4: Results for the RCV1 dataset. Our results here are worse than for the other datasets, we hypothesize this is due to this data having less pronounced low-rank structure.

# 4. Experimental evaluation

- Takeaways:

1. Taking $l$ = Ck with a fairly small C ≈ 10 suffices to get F1 scores > 0.75 in most settings
2. Algorithm 1 generally outperforms Algorithm 2 for a given value of $l$

# 5. Conclusion

1. Any sketch $\tilde{A} \in R^{l \times d}$ of A with the property that $\|A^T A - \tilde{A}^T \tilde{A}\|$ is small, can be used to additively approximate the $L^k$ and $T^k$ for each row. We get a streaming algorithm that uses O(d) memory and O(nd) time.
2. Q: Can we get such an additive approximation using memory only $O(d)$?
   1. The answer is no.
   2. Lower bound: must use Ω(d) working space for approximating the outlier scores for every data point

# 5. Conclusion

3. Using random row random projection, we give a streaming algorithm that can preserve the outlier scores for the rows up to small additive error on average
   - and preserve most outliers. The space required by this algorithm is only poly(k)log(d).
4. In our experiments, we found that choosing $l$ to be a small multiple of k was sufficient to get good results.
   1. Our results comparable full-blown SVD using sketches
   2. significantly smaller in memory footprint
   3. faster to compute and easy to implement

# 老師回饋

1. Code on Github?
2. SVD tool package
3. Is their experiment design reasonable?
4. How to apply to online-AD

$$Mahalanobis(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}$$

$$L(\boldsymbol{x}) = (\boldsymbol{y} - \boldsymbol{0})^T \ \boldsymbol{D}^{-1} (\boldsymbol{y} - \boldsymbol{0})$$

$$= \sum_{j=1}^{p} y_{(j)}^2 / \sigma_j^2 \ = \sum_{j=1}^{p} (\boldsymbol{x}^T \boldsymbol{v}^{(j)})^2 / \sigma_j^2$$

$$\boldsymbol{y} = \left[ y_{(1)}, y_{(2)}, \ldots, y_{(p)} \right] = \left[ \boldsymbol{x}^T \boldsymbol{v}^{(1)}, \boldsymbol{x}^T \boldsymbol{v}^{(2)}, \ldots, \boldsymbol{x}^T \boldsymbol{v}^{(p)} \right]$$

$$\boldsymbol{D} = \mathrm{diag}([\sigma_1^2, \sigma_2^2, \ \ldots, \sigma_p^2])$$

$$L^k(\boldsymbol{x}) = \sum_{j=1}^{k} (\boldsymbol{x}^T \boldsymbol{v}^{(j)})^2 / \sigma_j^2$$

$$T^k(i) = \sum_{j=k+1}^{p} (\boldsymbol{x}^T \boldsymbol{v}^{(j)})^2$$