# Sketching Linear Classifiers Over Data Streams

Kai Sheng Tai

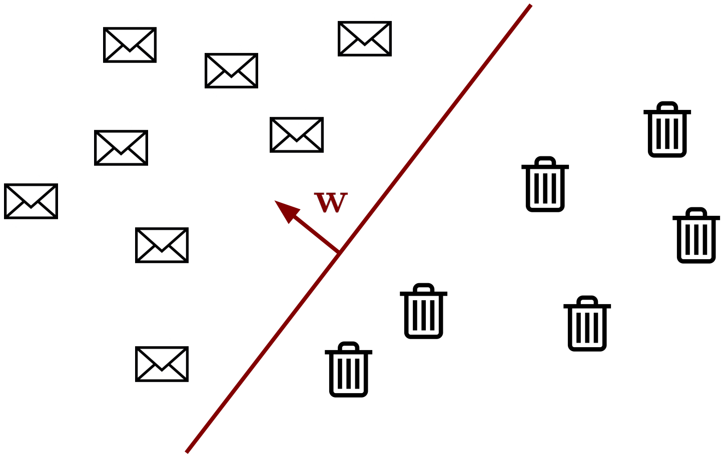
Vatsal Sharan, Peter Bailis, Gregory Valiant
Stanford University

# High-dimensional linear classifiers on streams

**Ubiquitous:** spam detection, ad click prediction, network traffic classification, ...

**Fast:** computationally cheap inference and updates

**Adaptive:** updated online in response to changing data distributions




**Problem: high memory usage**

Lots of features $\Rightarrow$ more expressive classifiers,
BUT more memory needed to store weights

# Example: Traffic classification with limited memory

Network packet

```
Version:    IPv4
Src:   136.0.1.1
Dest:  129.0.1.1
...
```

→

Features

```
 Src[:1] = 136
Dest[:1] = 129
 Src[:2] = 136.0
Dest[:2] = 129.0
...
```

→ **Classifier**

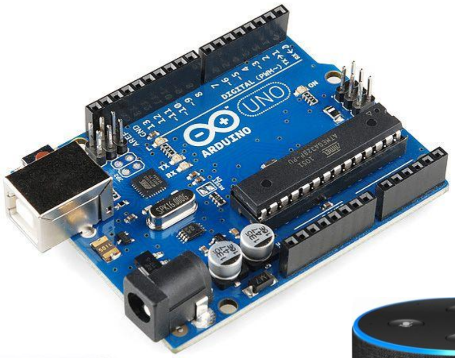
→ Accept

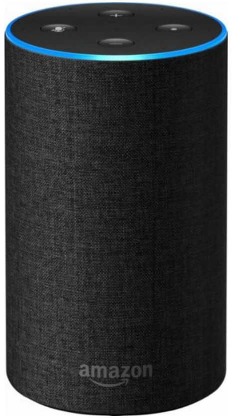
→ Reject (filter)

- Want classifiers that adhere to strict memory budgets (e.g., 1MB)
- But also want accuracy: more features, feature combinations

network switch

# More broadly: Online learning on memory-constrained devices



# Problem: How to restrict memory usage while preserving accuracy?


"I don't think you're getting the point of this exercise."

Proposal 1: Use only most informative features?

- In streaming setting, often don't know feature importance *a priori*
- Feature importance can change over time (e.g., spam classification)

Proposal 2: Use only most frequent features?

- Most frequent ≠ most informative

# Sketches for memory-limited stream processing

Long line of work on memory-efficient *sketches* for stream processing

e.g., identifying the *k* most frequent items in a stream ("heavy hitters" problem)

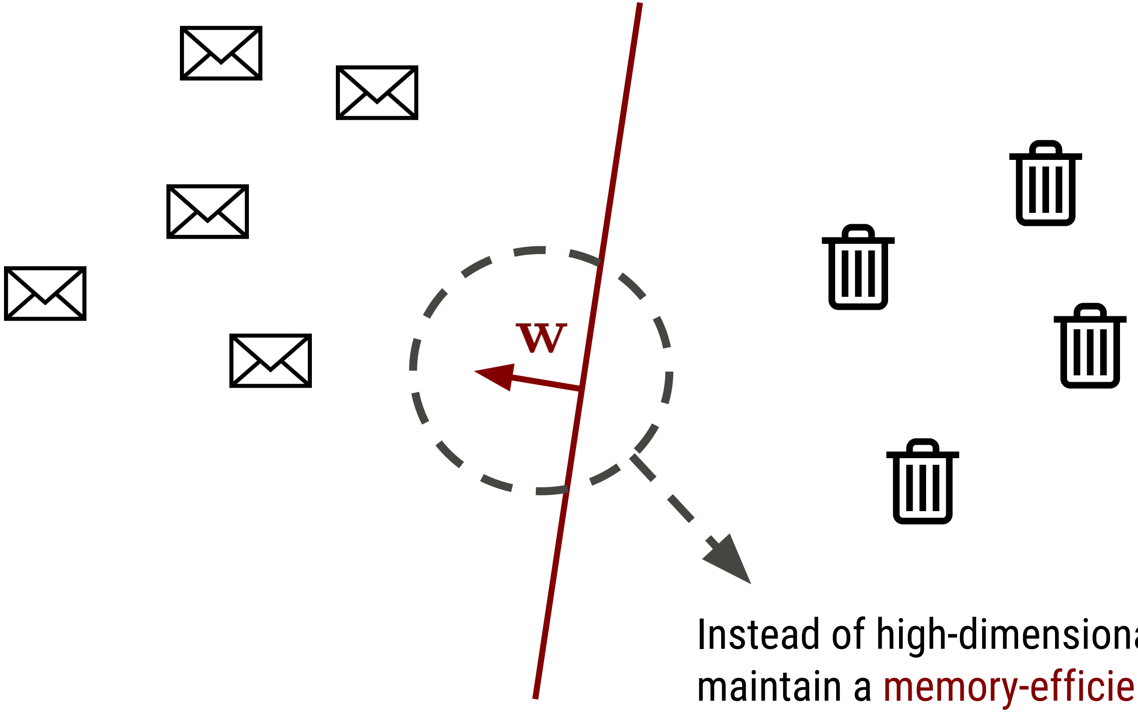
- Count-Sketch [Charikar, Chen & Farach-Colton '02]
- Count-Min Sketch [Cormode & Muthukrishnan '05]

Can we adapt existing sketching algorithms for use in *memory-limited streaming classification*?

**Yes — our contribution.** Weight-Median Sketch: a new sketch for linear classifiers

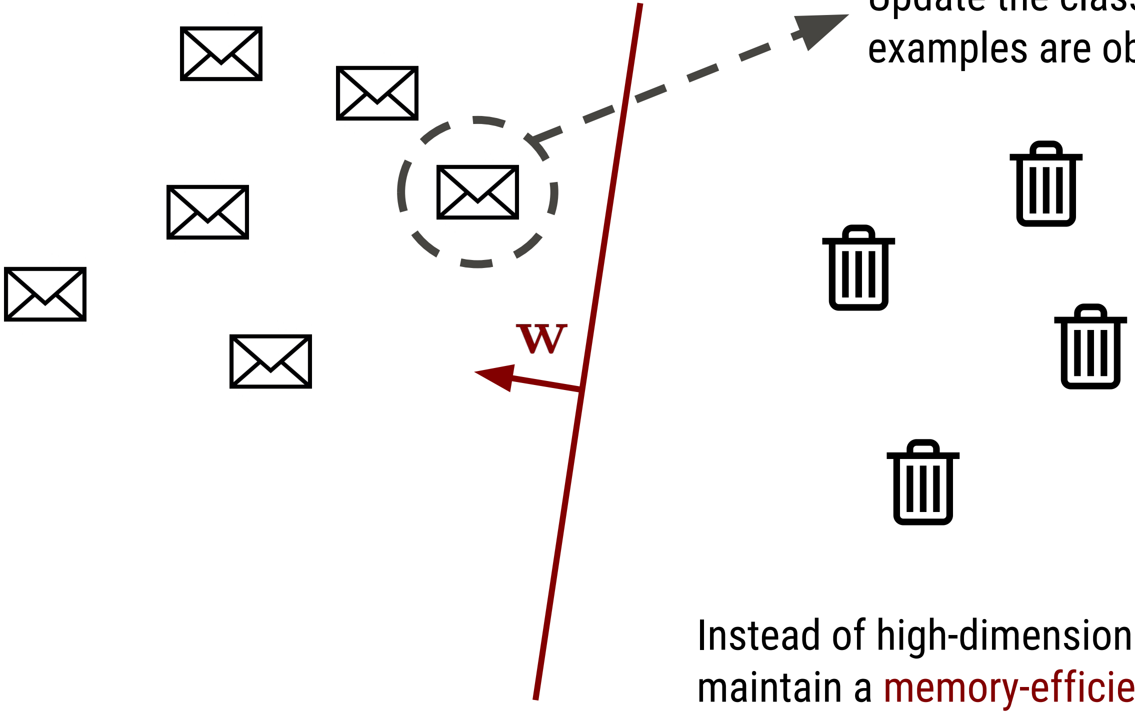
Main idea: most frequent items $\longrightarrow$ highest-magnitude weights

# This work: Sketched linear classifiers with online updates




Instead of high-dimensional classifier, maintain a memory-efficient sketch

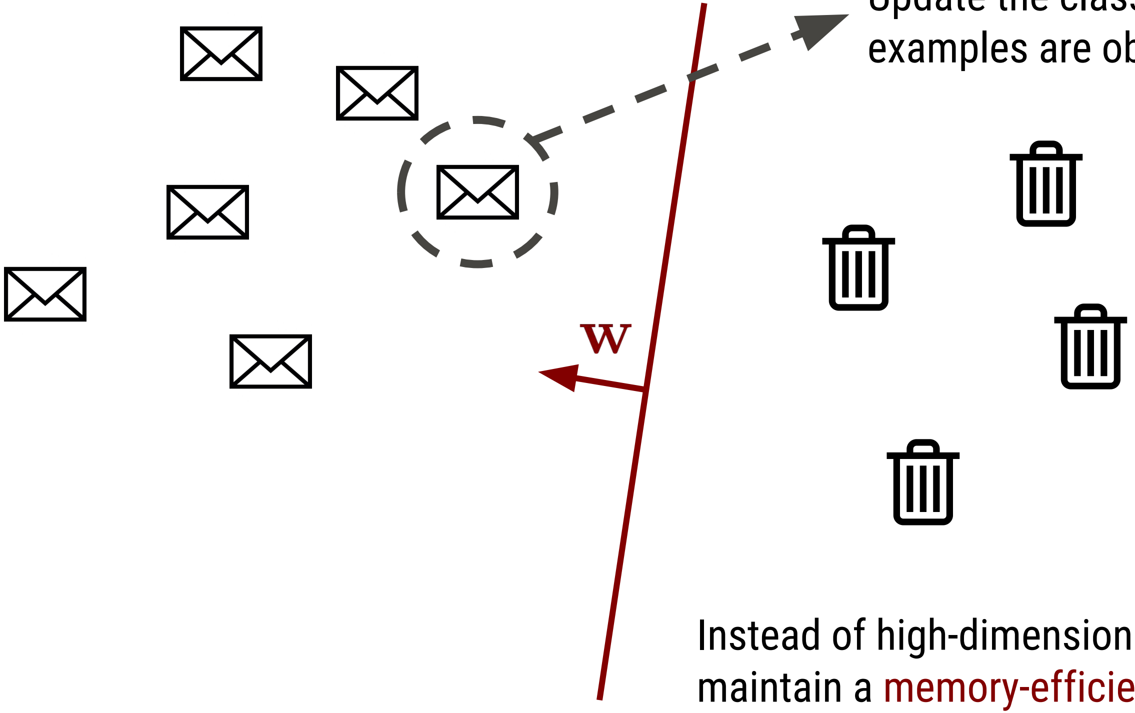
# This work: Sketched linear classifiers with online updates




Update the classifier as new examples are observed

Instead of high-dimensional classifier, maintain a memory-efficient sketch

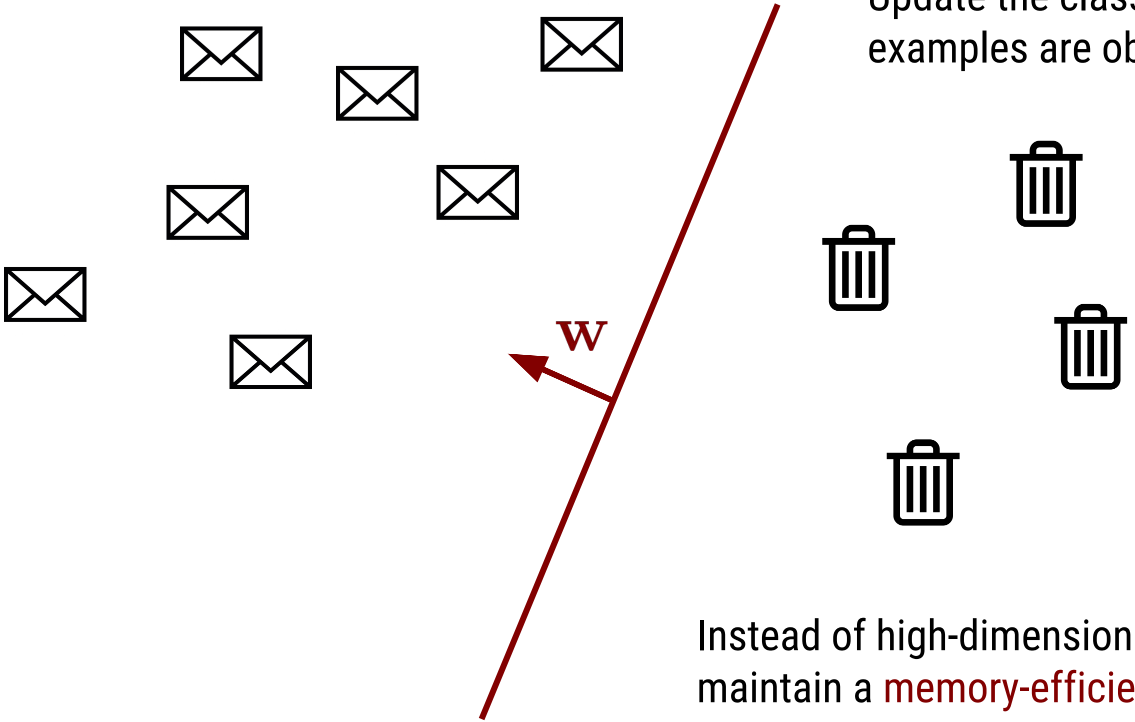
# This work: Sketched linear classifiers with online updates




Update the classifier as new examples are observed

Instead of high-dimensional classifier, maintain a memory-efficient sketch

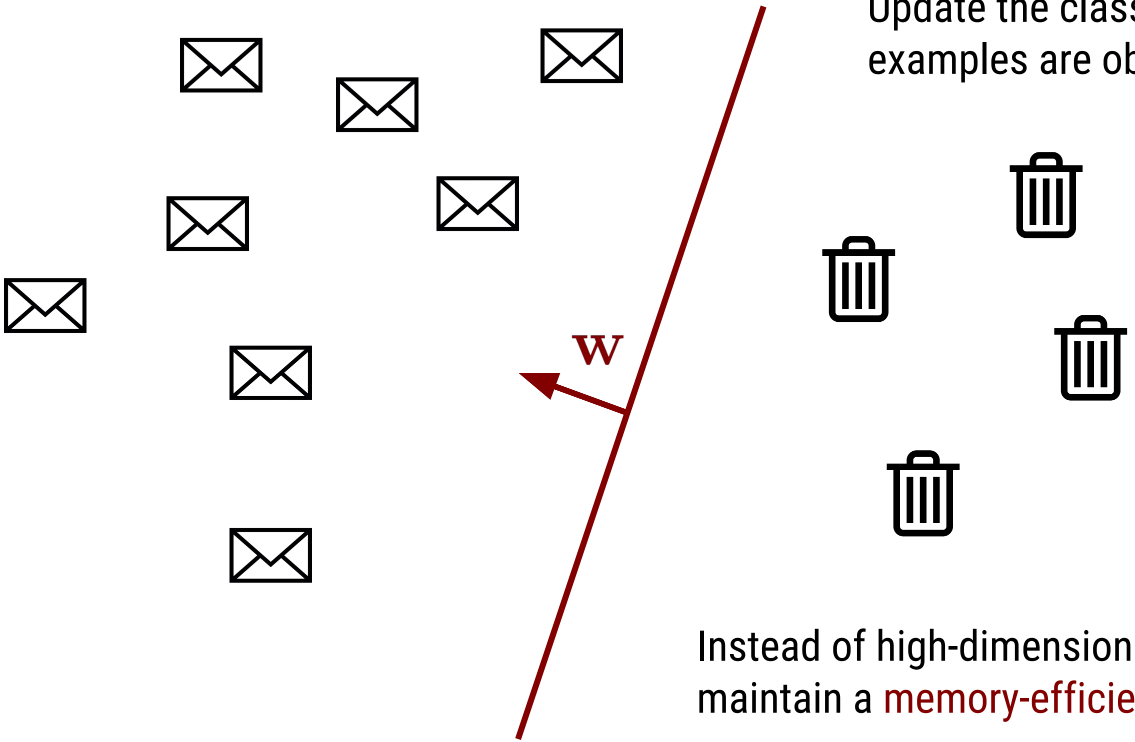
# This work: Sketched linear classifiers with online updates




Update the classifier as new examples are observed

Instead of high-dimensional classifier, maintain a memory-efficient sketch

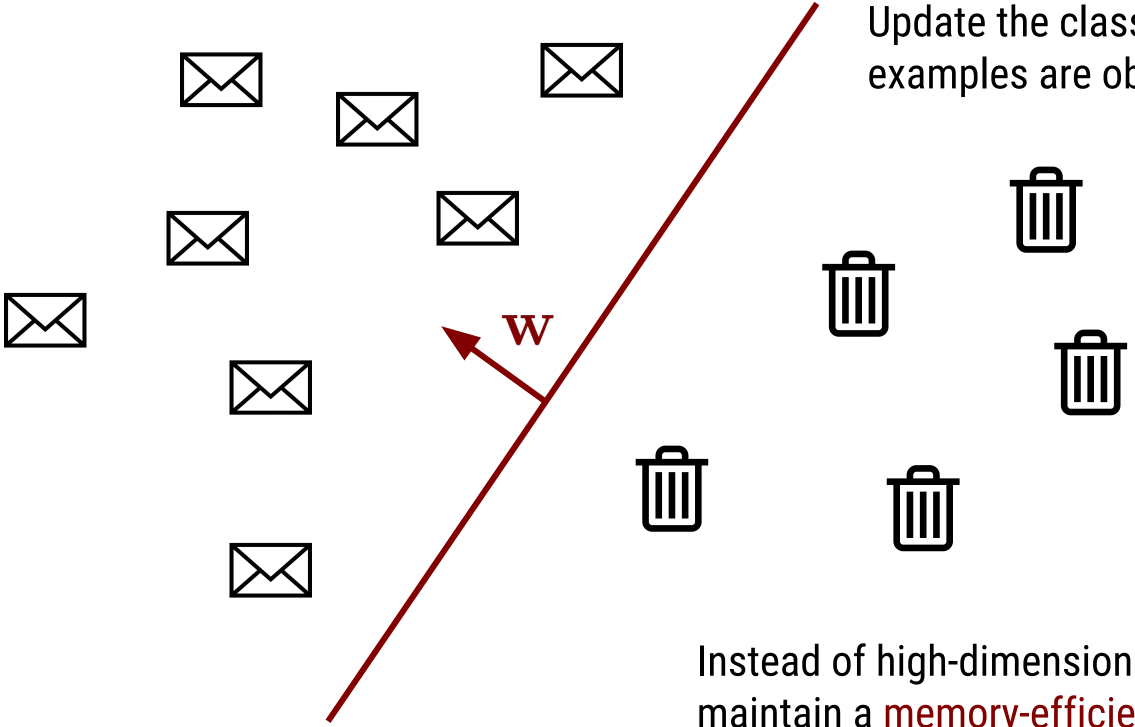
# This work: Sketched linear classifiers with online updates




Update the classifier as new examples are observed

Instead of high-dimensional classifier, maintain a memory-efficient sketch

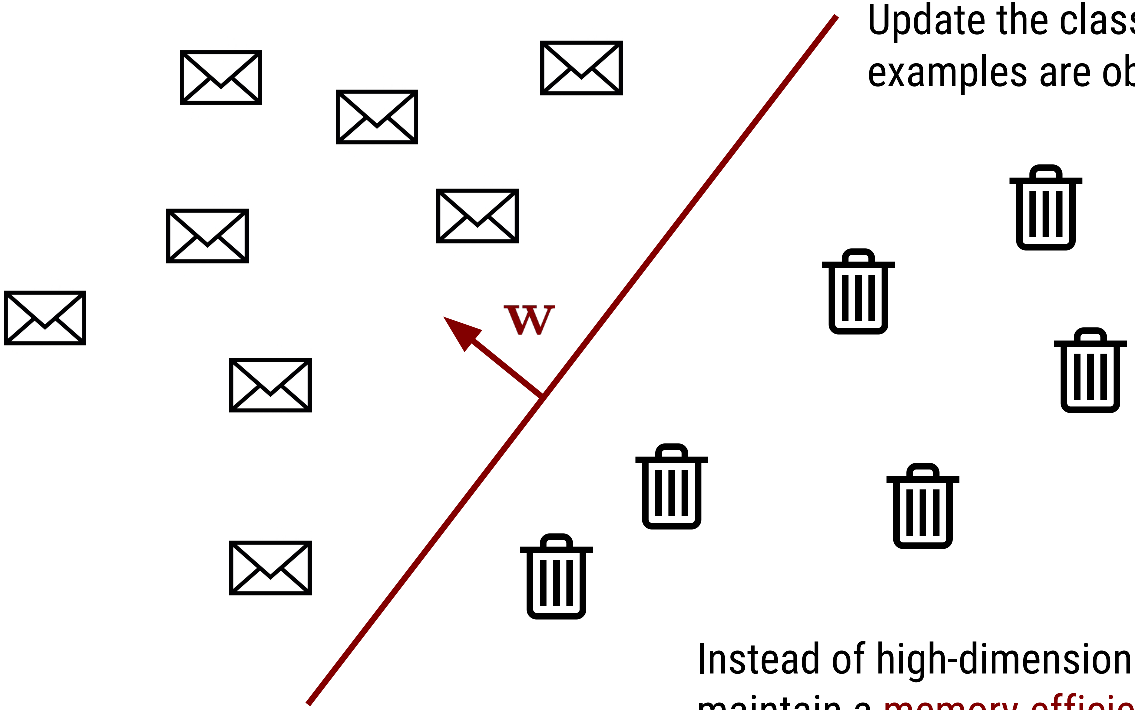
# This work: Sketched linear classifiers with online updates




Update the classifier as new examples are observed

Instead of high-dimensional classifier, maintain a memory-efficient sketch

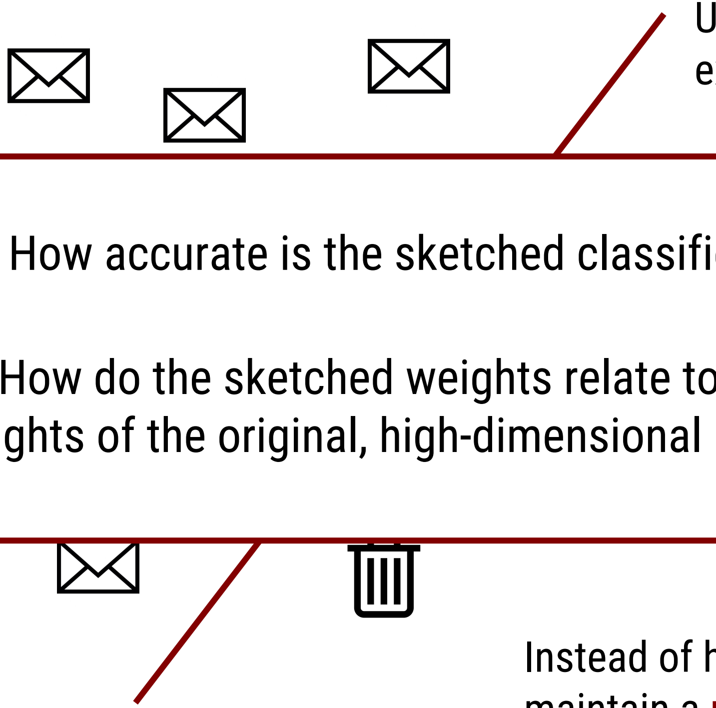
# This work: Sketched linear classifiers with online updates




Update the classifier as new examples are observed

Instead of high-dimensional classifier, maintain a memory-efficient sketch

# This work: Sketched linear classifiers with online updates



Update the classifier as new examples are observed

How accurate is the sketched classifier?

How do the sketched weights relate to the weights of the original, high-dimensional model?

Instead of high-dimensional classifier, maintain a memory-efficient sketch

# Related Work

## Streaming Algorithms

| | |
|---|---|
| Finding frequent items in data streams | [Charikar et al. ‘02, Cormode & Muthukrishnan ‘05, etc.] |
| Identifying differences between streams | [Schweller et al. ‘04, Cormode & Muthukrishnan ‘05, etc.] |

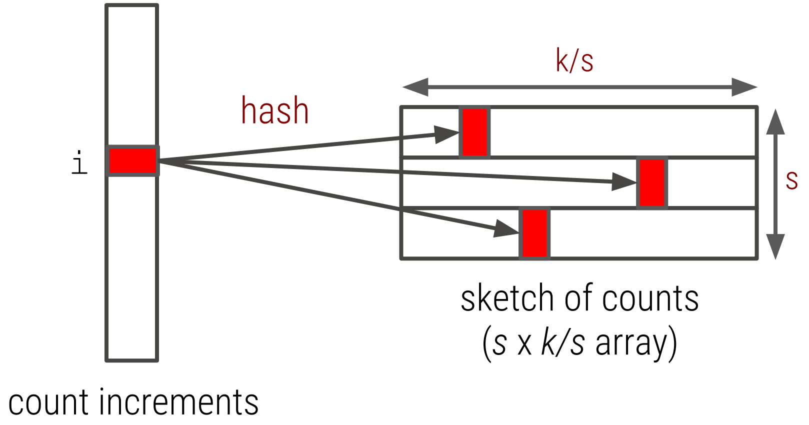
## Machine Learning

| | |
|---|---|
| Resource-constrained learning | [Konecny et al. ‘15, Gupta et al. ‘17, Kumar et al. ‘17] |
| Sparsity-inducing regularization | [Tibshirani ‘96 & many others] |
| Learning compressed classifiers (e.g., feature hashing) | [Shi et al. ‘09, Weinberger et al. ‘09, Calderbank et al. ‘09] |

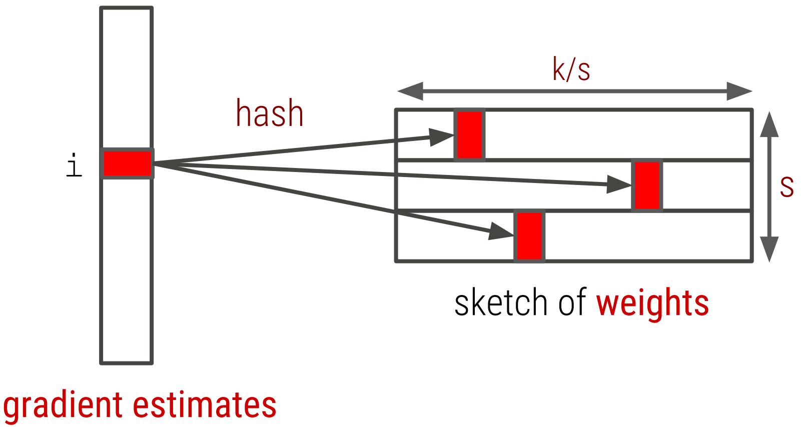
👉 1. algorithm
2. evaluation
3. applications

# The WM-Sketch: an extension of the Count-Sketch

## Count-Sketch update




## WM-Sketch update




Count-Sketch: maintain a low-dimensional sketch of **counts**

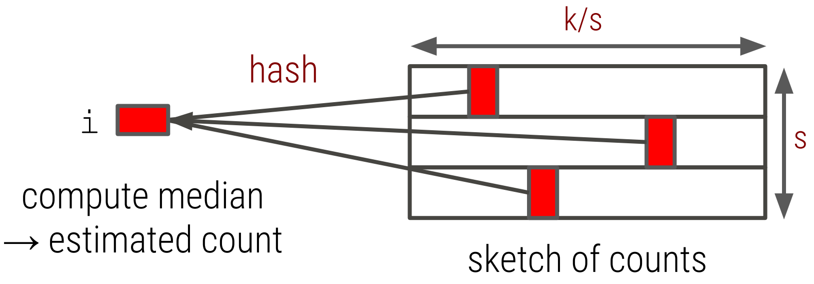
Update: hash each index $i$ to $s$ buckets, apply additive update

WM-Sketch: maintain a low-dimensional sketch of **weights**

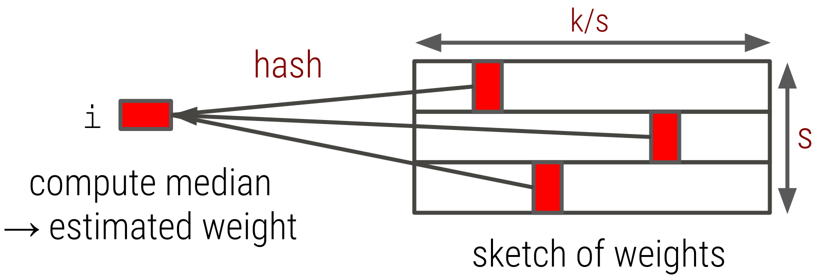
Update: **gradient descent** on sketched weights

# The WM-Sketch: an extension of the Count-Sketch

Count-Sketch query




WM-Sketch query




**Same query procedure**

Count-Sketch → low-error estimates of largest counts

WM-Sketch → low-error estimates of largest-magnitude weights

(note: standard *feature hashing* does **not** support weight recovery)

# WM-Sketch Analysis: Guarantees on weight approximation error

We compare the optimal weights for the **original** data, $w_*$

(i.e., the minimizer of the empirical loss)

to those recovered from the optimal **sketched** weights, $w_{est}$

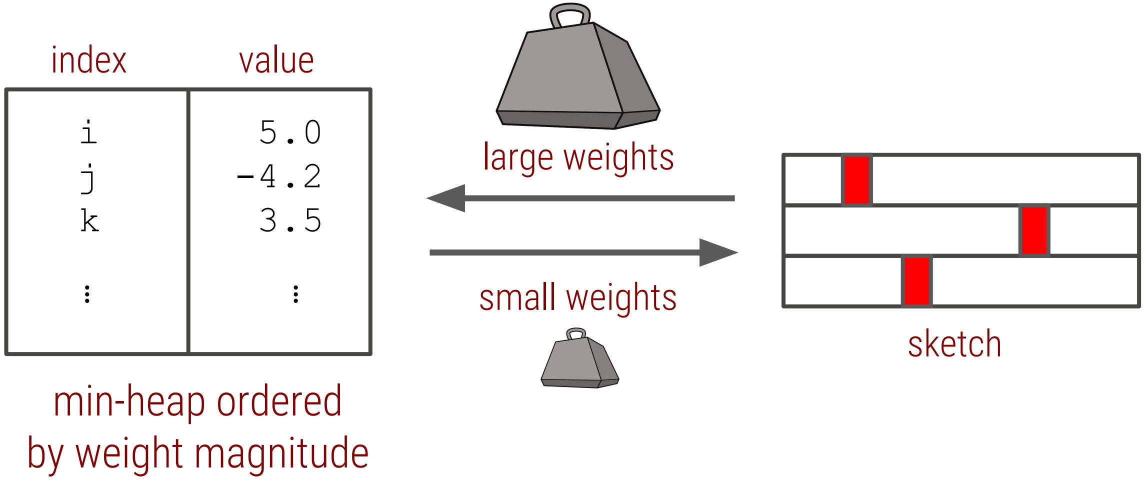
Theorem (informal)

Let *d* be the dimension of the data. With probability $1 - \delta$, the *maximum* entrywise approximation error is $\leq \epsilon \|\mathbf{w}_*\|_1$ for sketch size $\mathrm{O}(\epsilon^{-4} \log^3(d/\delta))$

only need sketch dimension much smaller than *d*

good approximation of high-magnitude weights

# Important optimization in practice: Store large weights in a heap



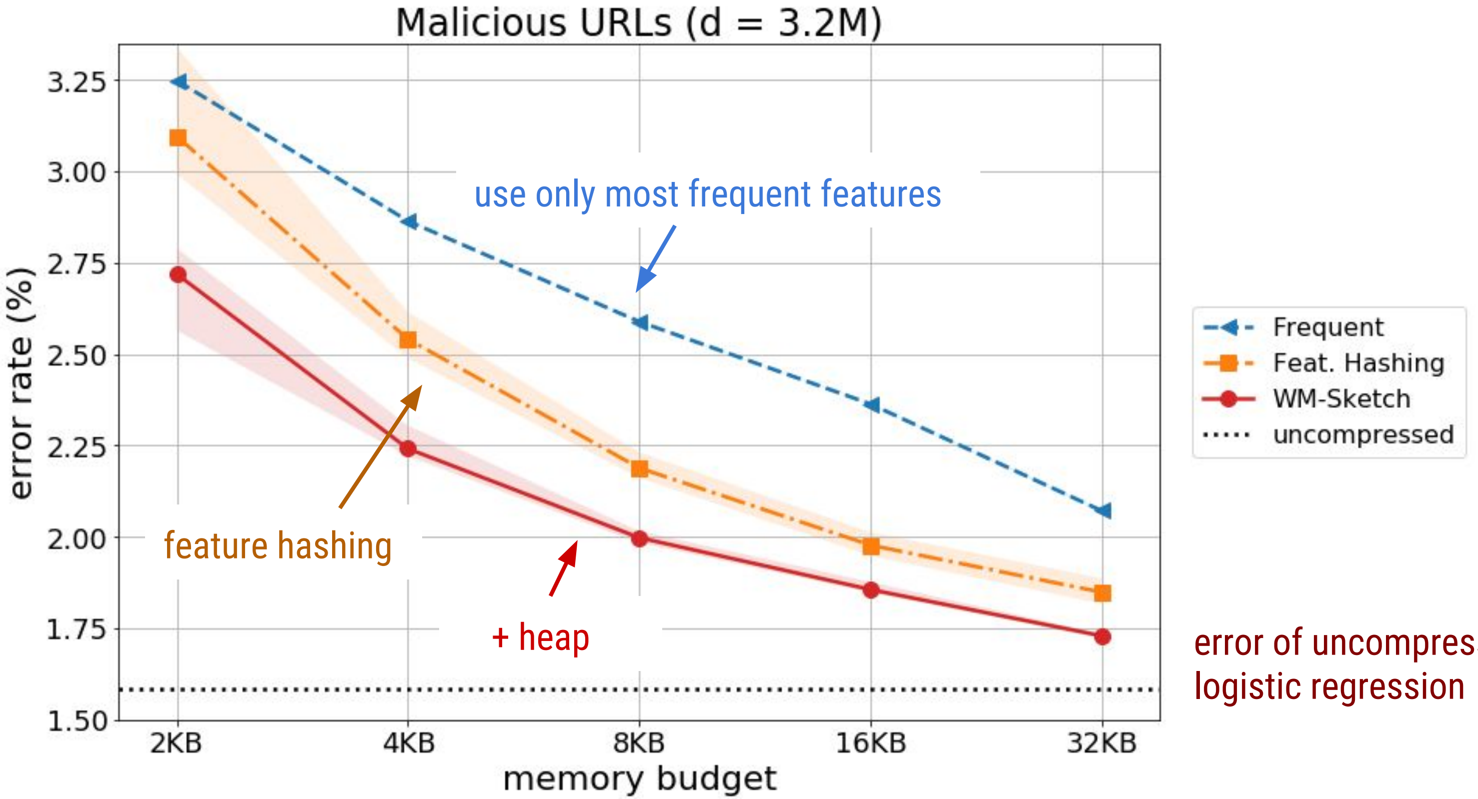

Anytime queries for estimated top-*k* weights

Reduces “bad” collisions with large weights in sketch

Significantly improves classification accuracy
and weight recovery accuracy in practice

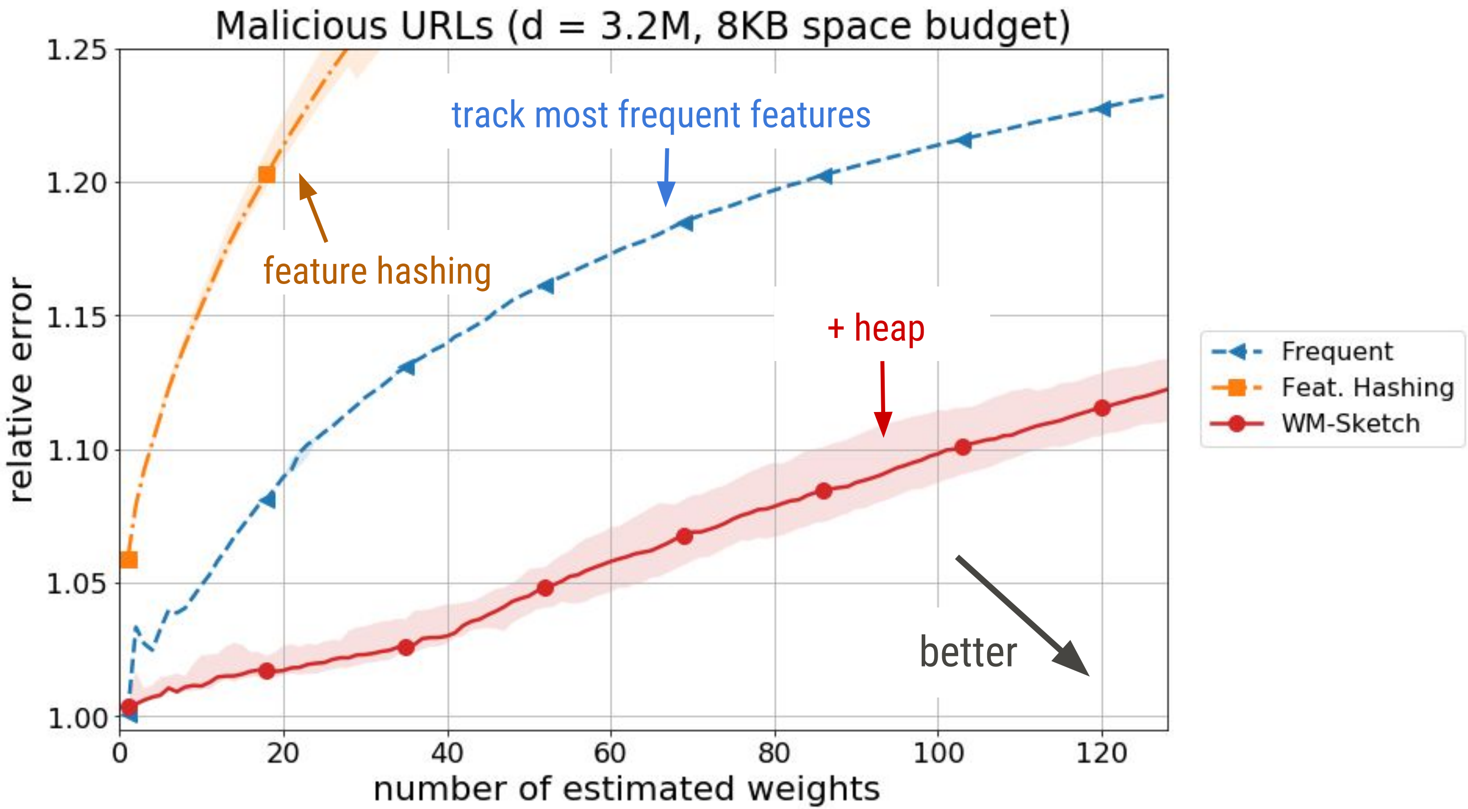
1. algorithm
2. evaluation
   - Classification accuracy
   - Weight recovery accuracy
3. applications

# Classification accuracy: WM-Sketch improves on Feature Hashing




error of uncompressed logistic regression

# Weight recovery: WM-Sketch improves on heavy hitters



1. algorithm
2. evaluation
3. 👉 applications
    - Network monitoring
    - Identifying correlated events

# Network monitoring: what are the largest relative differences?

Network packet

```
Version:    IPv4
Src:   136.0.1.1
Dest:  129.0.1.1
...
```

→

Features

```
 Src[:1] = 136
Dest[:1] = 129
 Src[:2] = 136.0
Dest[:2] = 129.0
...
```

→

**logistic regression with WM-Sketch**

Flow A

Flow B

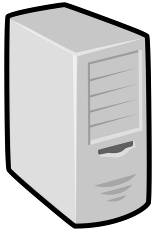
Largest weights → features (e.g., IP prefixes) with largest *relative* differences between flows

Previous work: "relative deltoids" in data streams [CM'05]

Outperforms Count-Min baselines
(even when baselines are given 8x memory budget)

network switch

# Explaining outliers: which features indicate anomalies?



| IP | City | Latency | |
|---|---|---|---|
| 136.0.1.1 | San Francisco | 10ms | label = −1 |
| 161.0.1.1 | New York | 12ms | |
| 129.0.1.1 | Houston | 500ms | label = +1 (e.g. >99th percentile) |
| … | … | … | |

**logistic regression with WM-Sketch**

| Feature | Weight |
|---|---|
| City=Houston | +4.2 |
| City=Austin | +2.0 |
| IP=129.x.x.x | +1.5 |
| … | … |

**Return features most indicative of being an outlier**
(weights can be interpreted as *log-odds ratio*)

**Streaming outlier explanation** (e.g., MacroBase [Bailis et al. '17])

**Outperforms heavy hitter-based methods for identifying "high-risk" features**

# Identifying correlations: which events tend to co-occur?

Text stream →

| Token 1 | Token 2 | Label | |
|---|---|---|---|
| United | States | +1 | real co-occurring events |
| computer | science | +1 | |
| computer | the | −1 | synthetic event pair |
| … | … | | … |

↓

**logistic regression with WM-Sketch**

↓

| Pair | Weight |
|---|---|
| (United, States) | +4.5 |
| (Barack, Obama) | +4.0 |
| (computer, science) | +2.0 |
| … | … |

Features = event pairs

Largest weights → events that are strongly correlated

Exact counter-based approach: 188MB memory
Approximation with WM-Sketch: 1.4MB memory

Approximation ⇒ **>100x less memory usage**

# Takeaways

## Weight-Median Sketch:

- Count-Sketch for linear classification
- Improved feature hashing
- Lightweight, memory-efficient classifiers everywhere


## Stream processing:

- Many tasks can be formulated as classification problems
- Still lots of room for exploration

**Paper:** tinyurl.com/**wmsketch**
**Code:** github.com/stanford-futuredata/**wmsketch**


kaishengtai.github.io
kst@cs.stanford.edu
@kaishengtai