Finding Heavily-Weighted Features in Data Streams

Kai Sheng Tai, Vatsal Sharan, Peter Bailis, Gregory Valiant
{kst, vsharan, pbailis, valiant}@cs.stanford.edu
Stanford University

Abstract

We introduce a new sub-linear space data structure—the Weight-Median Sketch—that captures the most heavily weighted features in linear classifiers trained over data streams. This enables memory-limited execution of several statistical analyses over streams, including online feature selection, streaming data explanation, relative deltoid detection, and streaming estimation of pointwise mutual information. In contrast with related sketches that capture the most commonly occurring features (or items) in a data stream, the Weight-Median Sketch captures the features that are most discriminative of one stream (or class) compared to another. The Weight-Median sketch adopts the core data structure used in the Count-Sketch, but, instead of sketching counts, it captures sketched gradient updates to the model parameters. We provide a theoretical analysis of this approach that establishes recovery guarantees in the online learning setting, and demonstrate substantial empirical improvements in accuracy-memory trade-offs over alternatives, including count-based sketches and feature hashing.

1 Introduction

With the rapid growth of streaming data volumes, memory-efficient sketches are an increasingly-important workhorse in analytics tasks including finding frequent items (Charikar et al., 2002; Cormode and Muthukrishnan, 2005b; Metwally et al., 2005; Larsen et al., 2016), estimating quantiles (Greenwald and Khanna, 2001; Luo et al., 2016), and approximating the number of distinct items (Flajolet, 1985). Sketching algorithms compute approximations of these analyses in exchange for significant reductions in memory utilization; therefore, they are well-suited to settings where highly-accurate estimation is not essential, or where practitioners wish to trade off between memory usage and approximation accuracy (Boykin et al., 2014; Yu et al., 2013). These properties make sketches a natural fit for stream processing applications where it is infeasible to store the entire stream in memory.

Simultaneously, a wide range of streaming analytics workloads can be formulated as machine learning problems over streaming data. For example, in streaming data explanation (Bailis et al., 2017; Meliou et al., 2014), analyses seek to differentiate between subpopulations in the data—for example, between an *inlier* class and an *outlier* class, as determined by some metric of interest. In network monitoring, analyses seek to identify significant relative differences between streams of network traffic (Cormode and Muthukrishnan, 2005a) such as in detecting destinations that are substantially more popular on one link versus another. In natural language processing on text streams, several applications require the identification of strongly-associated groups of tokens (Durme and Lall, 2009)—this pertains more broadly to the problem of identifying groups of events which tend to co occur. In the language of machine learning, these tasks can all be framed as instances of streaming *classification* between two or more classes of interest, followed by the *interpretation* of the learned model in order to identify the features that are the most discriminative between the classes.

In streaming classification applications such as the above, we can identify several key desiderata:

Low Memory. An emerging application domain for ML systems is in running training and inference on resource-constrained mobile and Internet of Things (IoT) devices (Kumar et al., 2017; Gupta et al., 2017). A widely-deployed example is the Arduino Uno Rev3 board, which operates with 2KB of onboard SRAM and 32KB of flash memory; predictive models operating in these settings must therefore contend with strict limits on model size. The use of on-device models is not limited to simply performing prediction with a pre-trained model; it is often desirable to support model updates in these memory-limited environments as well. For

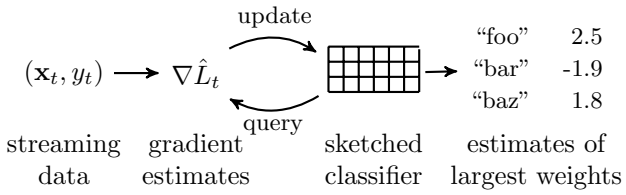

Figure 1: Streaming updates to a sketched classifier with approximations of the most heavily-weighted features.

instance, an on-device model can be updated "in the field" on the basis of input data streams like local sensor readings. Even in the traditional setting of stream processing on commodity servers, memory-constrained techniques can be useful for processing a large number of distinct streams on a single node.

Model Interpretability. In the stream processing applications described earlier, the goal is not solely to achieve high classification accuracy, but also to identify important features in the learned model, where each feature corresponds to items or attributes in the data stream. Learned model weights can themselves be a useful output of the learning process; a model trained to discriminate between two classes can yield actionable information regarding the attributes most characteristic of each subpopulation. While "model interpretability" is a broadly construed term in the literature (Lipton, 2016), in this work we consider the specific task of identifying the largest-magnitude weights in the model; intuitively, this corresponds to identifying which features are most influential towards making model predictions. Beyond the above applications, the identification of influential features in a learned model is relevant to issues in feature selection (Zhang et al., 2016), fairness (Corbett-Davies et al., 2017), model trustworthiness (Ribeiro et al., 2016), and legally-mandated rights to explanation in scenarios involving algorithmic decision-making (Goodman and Flaxman, 2016).

Fast Updates. The high throughput requirements of modern stream analytics necessitate the use of models that support fast inference and update times. For example, network traffic monitoring requires methods that support updates at line rates.

Existing sketch-based methods may seem to be attractive building-blocks for developing streaming classification systems that satisfy these requirements. However, as we show, simple adaptations of existing sketch algorithms can fail to capture influential features in the model. For example, a heavy-hitters algorithm (Cormode and Hadjieleftheriou, 2008) will identify frequently-occurring features, but these features are not necessarily the most *discriminative* features between classes—a simple instance of this failure is when the most frequent features are statistically independent of the class variable. On the other hand, online learning algorithms with sparsity-inducing regularization impose only soft constraints on memory usage, and it is difficult to *a priori* select a regularization parameter to satisfy hard memory constraints without strong assumptions on the data. Thus, existing methods are not fully satisfactory for our target applications.

In response, we develop new small-space algorithms for finding heavily-weighted features in data streams. In this paper, we formalize the problem by introducing the *Heavy-Weights problem* as a generalization of the well-studied *Heavy-Hitters problem*. We introduce a new sketch algorithm for the Heavy-Weights problem—the *Weight-Median Sketch*—that solves this problem for streaming linear classification using space logarithmic in the feature dimension. This sketch can be applied in many of the previously-mentioned stream analytics workloads, including: (i) online feature selection, (ii) streaming data explanation, (iii) detecting large relative differences in data streams (i.e., relative deltoids (Cormode and Muthukrishnan, 2005a)), and (iv) streaming estimation of highly-correlated pairs of items via pointwise mutual information (Durme and Lall, 2009).

The key intuition behind the Weight-Median Sketch is that by sketching the gradient updates to a linear classifier, we can incrementally maintain a compressed version of the true, high-dimensional model that supports efficient approximate recovery of the model weights. In particular, the Weight-Median Sketch maintains a linear projection of the weight vector of the linear classifier, similar to how a Count-Sketch (Charikar et al., 2002) maintains a sketch of a vector of frequency counts. Unlike heavy-hitters sketches that simply increment or decrement counters, the Weight-Median Sketch updates its state using online gradient descent (Hazan

et al., 2016). Since these updates themselves depend on the current weight estimates, a careful analysis is needed to ensure that the estimated weights do not diverge from the true model parameters over the course of multiple online updates. In this paper, we provide theoretical guarantees on the approximation error of these weight estimates. Additionally, in extensive experiments over real data, we show that the Weight-Median Sketch outperforms alternative approaches, including those based on tracking the most frequent features, truncation-based techniques, and feature hashing. For example, on the standard Reuters RCV1 binary classification benchmark, the Weight-Median Sketch recovers the most heavily-weighted features in the model with 4× better approximation error than a baseline using the Space Saving heavy-hitters algorithm and 10× better than a naïve weight truncation baseline, while using the same amount of memory.

To summarize, we make the following contributions in this work:

- We introduce the Weight-Median Sketch, a new sketch for identifying heavily-weighted features in linear classifiers over data streams.
- We provide a theoretical analysis of the Weight-Median Sketch that provides guarantees on the accuracy of the sketch estimates. In particular, we show that for feature dimension d and with success probability $1-\delta$, we can learn a compressed model of dimension $O\left(\epsilon^{-4}\log^3(d/\delta)\right)$ that supports approximate recovery of the optimal weight vector $\mathbf{w}_*$, where the absolute error of each weight is bounded above by $\epsilon\|\mathbf{w}_*\|_1$. For a given input vector $\mathbf{x}$, this structure can be updated in time $O\left(\epsilon^{-2}\log^2(d/\delta)\cdot\text{nnz}(\mathbf{x})\right)$.
- We present optimizations to the basic Weight-Median Sketch—in particular, using an active set of features—to improve both weight recovery and classification accuracy.
- We demonstrate that the Weight-Median Sketch empirically outperforms several alternatives in terms of memory-accuracy trade-offs across a range of real-world datasets.

2 Preliminaries

In this section, we present our problem description as well as several theoretical concepts that we draw upon in our design and analysis of the Weight-Median Sketch.

Conventions and Notation. We denote all vector quantities by bolded lowercase letters, $\mathbf{w}$. The notation w_i or $[\mathbf{w}]_i$ (where appropriate for clarity) denotes the ith element of $\mathbf{w}$. The notation $[n]$ denotes the set $\{1,\ldots,n\}$. We write p-norms as $\|\mathbf{w}\|_p$, where the p-norm of $\mathbf{w}$ is defined as $\left(\sum_{i=1}^d |w_i|^p\right)^{1/p}$.

2.1 The Heavy-Weights Problem

The well-studied ϵ-approximate Heavy-Hitters problem can be defined as follows:

Definition 1. *(Cormode and Hadjieleftheriou, 2008) Given a stream of n items, let n_i denote the count of item i over the stream. Given any $\phi > 0$, the ϵ-approximate Heavy-Hitters problem is to return a set of items S such that for all $i \in S$, $n_i \geq (\phi-\epsilon)n$, and there is no $i \notin S$ such that $n_i \geq \phi n$.*

We formalize the Heavy-Weights problem as the analog of the Heavy-Hitters problem for general optimization problems where a unique solution exists. As is typical in the Heavy-Hitters formulation, we allow for an arbitrary approximation factor $\epsilon > 0$.

Definition 2. *((ϵ,p)-Approximate Heavy-Weights Problem) For any function $L : \mathcal{X} \to \mathbb{R}$, $\mathcal{X} \subseteq \mathbb{R}^d$ for which a unique minimizer exists, define $\mathbf{w}_* = \arg\min_{\mathbf{w}} L(\mathbf{w})$. Given any $\phi > 0$, the (ϵ,p)-approximate Heavy-Weights problem is to return a set $S \subseteq [d]$ such that for all $i \in S$, $|[\mathbf{w}_*]_i| \geq (\phi-\epsilon)\|\mathbf{w}_*\|_p$, and there is no $i \notin S$ such that $|[\mathbf{w}_*]_i| \geq \phi\|\mathbf{w}_*\|_p$.*

The function L represents an objective or cost to be minimized. Note that the familiar Heavy-Hitters problem can be posed as an optimization problem with the following objective: $L(\mathbf{w}) = -\mathbf{w}^T\mathbf{x}$, subject to $\|\mathbf{w}\|_1 \leq n$, where $\mathbf{x}$ is the vector of true counts with $\|\mathbf{x}\|_1 = n$ and $\min_i x_i > 0$. This objective is minimized by $\mathbf{w}_* = \mathbf{x}$. Therefore, the ϵ-approximate Heavy-Hitters problem is a special case of the $(\epsilon,1)$-approximate Heavy-Weights problem.

We can also define the related *Weight Estimation* problem:

Definition 3. *((ϵ, p)-Approximate Weight Estimation Problem) For any function $L : \mathcal{X} \to \mathbb{R}$, $\mathcal{X} \subseteq \mathbb{R}^d$ for which a unique minimizer exists, define $\mathbf{w}_* = \arg\min_{\mathbf{w}} L(\mathbf{w})$. The (ϵ, p)-approximate Weight Estimation problem is to return a value $\hat{w}_i$, for any $i \in [d]$, such that $|\hat{w}_i - [\mathbf{w}_*]_i| \leq \epsilon \|\mathbf{w}_*\|_p$.*

If $\|\mathbf{w}_*\|_p$ is known, an algorithm that solves the Weight Estimation problem also solves the Heavy-Weights problem, since we can simply enumerate all weights $i \in [d]$ and output those for which $\hat{w}_i \geq (\phi - \epsilon)\|\mathbf{w}_*\|_p$. In practice, it is often more convenient to work in terms of retrieving estimates of the top-K weights of $\mathbf{w}_*$ in absolute value; this formulation is equivalent to the ϵ-approximate Heavy-Weights problem for some implicit value of ϕ. Note that if we have an algorithm that solves the (ϵ, p)-approximate Weight Estimation problem and are given an upper bound $\|\mathbf{w}_*\|_p \leq B$, we can retrieve a set—possibly of size $> K$—that is guaranteed to contain the top-K weights using the following procedure: enumerate the weight estimates $\hat{w}_i$ and return those whose range $[\hat{w}_i - \epsilon B, \hat{w}_i + \epsilon B]$ intersects with the top-K set.

2.2 Heavy-Weights in Linear Classifiers

The Heavy-Weights and Weight Estimation problems defined above are extremely broad as they concern general optimization problems. In this paper, we specialize to the class of objectives corresponding to binary classification with ℓ_2-regularized linear models. As we show in our subsequent exploration of specific applications (Sec. 6), this model family is sufficiently expressive to capture a variety of applications in stream processing, including streaming feature selection, data explanation, relative deltoid detection, and pointwise mutual information estimation.

We consider a binary[1] linear classifier parameterized[2] by vector $\mathbf{w} \in \mathbb{R}^d$. Given an input feature vector $\mathbf{x} \in \mathbb{R}^d$, the classifier outputs a prediction $\hat{y} \in \{-1, +1\}$ as:

$$\hat{y} = \text{sign}\left(\mathbf{w}^T \mathbf{x}\right).$$

Given that $T > 0$ examples have been observed in the stream, define the loss L as the following:

$$L(\mathbf{w}) = \frac{1}{T} \sum_{t=1}^{T} \ell\left(y_t \mathbf{w}^T \mathbf{x}_t\right) + \frac{\lambda}{2} \|\mathbf{w}\|_2^2,$$

where $\{(\mathbf{x}_t, y_t)\}_{t=1}^T$ is the set of examples observed in the stream so far, $\ell(\cdot)$ is a convex, differentiable function of its argument and $\lambda > 0$ controls the strength of ℓ_2-regularization. The choice of $\ell(\cdot)$ defines the linear classification model to be used. For example, the logistic loss $\ell(\xi) = \log(1 + \exp(-\xi))$ defines logistic regression, and smoothed versions of the hinge loss $\ell(\xi) = \max\{0, 1 - \xi\}$ define close relatives of linear support vector machines. Our goal is to obtain good estimates of $\mathbf{w}_*$, the optimal weight vector in hindsight over the T observed examples.

We consider the online learning setting where the model weights are updated over a series of rounds $t = 1, 2, \ldots, T$. In each round, we update the model weights via the following process:

1. Receive an input $(\mathbf{x}_t, y_t)$.
2. Incur loss $L_t(\mathbf{w_t}) = \ell\left(y_t \mathbf{w}_t^T \mathbf{x}_t\right) + \frac{\lambda}{2} \|\mathbf{w}_t\|_2^2$.
3. Update state $\mathbf{w}_t$ to $\mathbf{w}_{t+1}$.

The online learning literature describes numerous update rules for selecting the next state $\mathbf{w}_{t+1}$ given the current state $\mathbf{w}_t$. For concreteness, we specify our algorithms to perform updates with online gradient descent (OGD) (Hazan et al., 2016; Chp. 3), which uses the following update rule:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla L_t(\mathbf{w}_t),$$

where $\eta_t > 0$ is the learning rate at step t. OGD enjoys the advantages of simplicity and minimal space requirements (needing to maintain only a representation of the weight vector $\mathbf{w}_t$ and a global scalar learning

[1] In Sec. 8, we describe an extension to the multiclass setting using a standard reduction to binary classification.

[2] In the following, we assume linear classifiers with no bias term for simplicity, but it is straightforward to extend our methods to the nonzero bias case.

rate), but may be suboptimal given the particular structure of the problem. Generalizations to other online learning algorithms are straightforward.

Clearly, if we were to explicitly maintain the weight vector $\mathbf{w}_t$ at each step, both the Heavy-Weights problem and the Weight Estimation problem would be trivial to solve. The challenge is the following: is it possible to maintain, for each time step, a *compact summary* $\mathbf{z}_t \in \mathbb{R}^k$, where $k \ll d$, that supports efficient recovery of the highest-magnitude weights in the model? In the following sections, we will show that in a restricted online setting where the order in which the examples $(\mathbf{x}_t, y_t)$ are observed is non-adversarial, there exists an algorithm using $O\left(\epsilon^{-4}\text{polylog}(d)\right)$ space that solves the $(\epsilon, 1)$-approximate Weight Estimation problem.

2.3 Relevant Background

At its core, our proposed method combines techniques from (1) online learning, (2) sketching, and (3) norm-preserving embeddings via random projections. In the previous section, we gave a brief overview of online learning. Here, we present the relevant background for areas (2) and (3).

Count-Sketch. The Count-Sketch (Charikar et al., 2002) is a linear projection of a vector $\mathbf{w} \in \mathbb{R}^d$ that supports efficient approximate recovery of the entries of $\mathbf{w}$. For a given size k and depth s, the Count-Sketch algorithm maintains a collection of s hash tables, each with width k/s (see Fig. 2). Each $i \in [d]$ is assigned a random bucket $h_j(i)$ in table j along with a random sign $\sigma_j(i)$. Increments to each coordinate i are multiplied by $\sigma_j(i)$ added to the corresponding buckets $h_j(i)$. The estimator for the ith coordinate is the median of the values in the assigned buckets multiplied by the corresponding sign flips. Charikar et al. (2002) showed the following recovery guarantee:

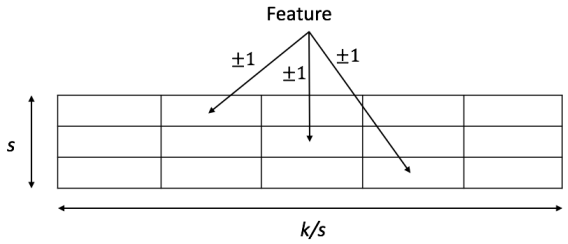

Figure 2: An illustration of the Count-Sketch with depth s and width k/s. Each feature hashes to s locations, multiplied by a random ± 1 sign.

Lemma 1. *(Charikar et al., 2002) Let $\mathbf{v}_{\text{cs}}$ be the Count-Sketch estimate of the vector $\mathbf{v}$. For any vector $\mathbf{v}$, with probability $1 - \delta$, a Count-Sketch matrix with width $\Theta(1/\epsilon^2)$ and depth $\Theta(\log(d/\delta))$ satisfies*

$$\|\mathbf{v} - \mathbf{v}_{\text{cs}}\|_\infty \leq \epsilon \|\mathbf{v}\|_2.$$

This result implies that the Count-Sketch solves the ϵ-approximate Heavy-Hitters problem.

Johnson-Lindenstrauss (JL) property. A random projection matrix is said to have the Johnson-Lindenstrauss (JL) property (Johnson and Lindenstrauss, 1984) if it preserves the norm of a vector with high probability:

Definition 4. *A random matrix $R \in \mathbb{R}^{k \times d}$ has the JL property with error ϵ and failure probability δ if for any given $\mathbf{x} \in \mathbb{R}^d$, we have with probability $1 - \delta$:*

$$\left| \|R\mathbf{x}\|_2 - \|\mathbf{x}\|_2 \right| \leq \epsilon \|\mathbf{x}\|_2.$$

The JL property holds for dense matrices with independent Gaussian or Bernoulli entries (Achlioptas, 2003), and recent work has shown that it applies to certain sparse matrices as well (Kane and Nelson, 2014). Intuitively, JL matrices preserve the geometry of a set of points, and we leverage this key fact to ensure that we can still recover the original solution after projecting to low dimension.

3 Finding Heavily-Weighted Features

In this section, we present the Weight-Median Sketch for identifying heavily-weighted features in linear classifiers. First, for intuition and comparison, we describe a number of simple baseline methods based on maintaining a K-sparse weight vector at each iteration. The first two are based on truncating the weight vector to its largest-magnitude components; the next two are extensions of streaming heavy-hitters algorithms. We subsequently describe our new sketch-based approaches for finding heavily-weighted features.

3.1 Baseline Methods

Simple Truncation. Given a budget of K weights, a natural baseline method is to simply truncate $\mathbf{w}$ after each update to the K entries with highest absolute value, setting all other entries to zero. The simple truncation baseline is similar to the truncated Perceptron algorithm proposed by Hoi et al. (Hoi et al., 2012). We give a pseudocode description in Appendix B.

Probabilistic Truncation. A problem with the simple truncation method is that it may end up "stuck" with a bad set of weights: a "good" index that would have been included in the top-K set by the unconstrained classifier may fail to be included in the feature set under Alg. 3 if its gradient updates are insufficiently large relative to the smallest weight in the set; this results in the weight being repeatedly zeroed-out in each iteration. To remedy this problem, we can instead adopt a *randomized* approach where indices are accepted into the K-sparse set with probability proportional to the magnitude of their weights. Therefore, even if some feature has small but nonzero weight after an update, there is still a positive probability that it is accepted into the feature set. This "probabilistic truncation" algorithm is inspired by weighted reservoir sampling (Efraimidis and Spirakis, 2006). We give the pseudocode in Appendix B.

Count-Min Frequent Features. The Count-Min sketch (Cormode and Muthukrishnan, 2005b) is a commonly-used method for finding frequent items in data streams. This baseline uses a Count-Min sketch to identify the K most frequently-occurring features; the weights for these frequent features are maintained, while the remaining weights are set to 0.

Space Saving Frequent Features. This method is identical to the previous approach except for the use of the Space Saving algorithm (Metwally et al., 2005) in place of the Count-Min sketch for frequent item estimation. The Space Saving algorithm has previously been found to emprically outperform Count-Min in insertion-only settings such as ours (Cormode and Hadjieleftheriou, 2008).

3.2 Weight-Median Sketch

We propose a new algorithm—the *Weight-Median Sketch* (WM-Sketch)—for the problem of identifying heavily-weighted features. The main data structure in the WM-Sketch is identical to that used in the Count-Sketch. The sketch is parameterized by size k, depth s, and width k/s. We initialize the sketch with a size-k array set to zero. For a given depth s, we view this array as being arranged in s rows, each of width k/s (assume that k is a multiple of s). We denote this array as $\mathbf{z}$, and equivalently view it as a vector in $\mathbb{R}^k$.

The high-level idea is that each row of the sketch is a compressed version of the model weight vector $\mathbf{w} \in \mathbb{R}^d$, where each index $i \in [d]$ is mapped to some assigned bucket $j \in [k/s]$. Since $k/s \ll d$, there will be many collisions between these weights; therefore, we maintain s rows—each with different assignments of features to buckets—in order to disambiguate weights.

Hashing Features to Buckets. In order to avoid explicitly storing these mappings, which would require space linear in d, we implement the feature-to-bucket maps using hash functions. For each row $j \in [s]$, we maintain a pair of hash functions, $h_j : [d] \to [k/s]$ and $\sigma_j : [d] \to \{-1, +1\}$. We now explain how these hash functions are used to update and query the sketch.

Updates. Suppose an update $\Delta_i \in \mathbb{R}$ to weight $\mathbf{w}_i$ of the ith feature is given. We apply this update using the following procedure: for each row $j \in [s]$, add the value $\sigma_j(i) \cdot \Delta_i$ to bucket $h_j(i)$ in the row. Let the matrix $A \in \{-1, +1\}^{k \times d}$ denote the map implicitly represented by the hash functions h_j and σ_j. We can then write the update $\tilde{\mathbf{\Delta}}$ to $\mathbf{z}$ as $\tilde{\mathbf{\Delta}} = A\mathbf{\Delta}$.

Algorithm 1: Weight-Median (WM) Sketch

input: size k, depth s, loss function ℓ, ℓ_2-regularization parameter λ, learning rate schedule η_t

initialization

$\mathbf{z} \leftarrow s \times k/s$ array of zeroes

Sample Count-Sketch matrix A

$t \leftarrow 0$

function `Update(`$\mathbf{x}$, y`)`

$\tau \leftarrow \frac{1}{s}\mathbf{z}^T A\mathbf{x}$ ▷ Prediction for $\mathbf{x}$

$\mathbf{z} \leftarrow (1 - \lambda\eta_t)\mathbf{z} - \eta_t y \nabla \ell\left(y\tau\right) A\mathbf{x}$

$t \leftarrow t + 1$

function `Query(`i`)`

return output of Count-Sketch retrieval on $\mathbf{z}$

Instead of being provided the updates Δ, we must compute them as a function of the input example $(\mathbf{x}, y)$ and the sketch state $\mathbf{z}$. Given the loss function ℓ, we define the update to $\mathbf{z}$ as the online gradient descent update for the *sketched* example $(A\mathbf{x}, y)$ where we first make a prediction τ, and then compute the gradient of the loss at this value:

$$\tau = \frac{1}{s}\mathbf{z}^T A\mathbf{x}$$
$$\tilde{\mathbf{\Delta}} = -\eta y \nabla \ell\left(y\tau\right) A\mathbf{x},$$

where $\eta > 0$ is the learning rate.

To build intuition, it is helpful to compare this update to the Count-Sketch update rule (Charikar et al., 2002). In the heavy-hitters setting, the input $\mathbf{x}$ is a one-hot encoding for the item seen in that time step. The update to the Count-Sketch state $\mathbf{z}_{\text{cs}}$ is the following:

$$\tilde{\mathbf{\Delta}}_{\text{cs}} = A\mathbf{x},$$

where A is defined identically as above. Therefore, our update rule is simply the Count-Sketch update scaled by the constant $-\eta y \nabla \ell(y\mathbf{z}^T A\mathbf{x}/s)$. However, an important detail to note is that the Count-Sketch update is *independent* of the sketch state $\mathbf{z}_{\text{cs}}$, whereas the WM-Sketch update does depend on $\mathbf{z}$. This cyclical dependency between the state and state updates is the main challenge in our analysis of the WM-Sketch.

Queries. To obtain an estimate $\hat{w}_i$ of the ith weight, we return the median of the values $\sigma_j(i) \cdot z_{j,h_j(i)}$ for $j \in [s]$. This is identical to the query procedure for the Count-Sketch.

We summarize the update and query procedures for the WM-Sketch in Algorithm 1. In the next section, we show how the sketch size k and depth s parameters can be chosen to satisfy an ϵ approximation guarantee with failure probability δ over the randomness in the sketch matrix.

Efficient Weight Decay. It is important that ℓ_2-regularization can be applied efficiently on the sketch state $\mathbf{z}$ of size k. A naïve implementation that scales each entry in $\mathbf{z}$ by $(1 - \eta_t\lambda)$ in each iteration incurs an update cost of $O(k + s \cdot \text{nnz}(\mathbf{x}))$, which masks the computational gains that can be realized when $\mathbf{x}$ is sparse. Here, we use a standard trick (Shalev-Shwartz et al., 2011): we maintain a global *scale* parameter α that scales the sketch values $\mathbf{z}$. Initially, $\alpha = 1$ and we update $\alpha \leftarrow (1 - \eta_t\lambda)\alpha$ to implement weight decay over the entire feature vector. Our weight estimates are therefore scaled by α: $\hat{w}_i = \text{median}\left\{\alpha\sigma_j(i) \cdot z_{j,h_j(i)} : j \in [s]\right\}$. This optimization reduces the cost of each sketch update from $O(k + s \cdot \text{nnz}(\mathbf{x}))$ to $O(s \cdot \text{nnz}(\mathbf{x}))$.

Hash Functions and Independence. Our analysis of the WM-Sketch requires hash functions that are $O(\log(d/\delta))$-wise independent. In comparison, the Count-Sketch requires 2-independent hashing. While hash functions satisfying this level of independence can be constructed using polynomial hashing (Carter and Wegman, 1977), hashing each input value would require time $O(\log(d/\delta))$, which can be costly when the dimension d is large. Instead of satisfying the full independence requirement, our implementation simply uses fast, 3-wise independent tabulation hashing (P?traşcu and Thorup, 2012). In our experiments (Sec. 5), we did not observe any significant degradation in performance from this choice of hash function, which is consistent with empirical results in other hashing applications that the degree of independence does not significantly impact performance on real-world data (Mitzenmacher and Vadhan, 2008).

3.3 Active-Set Weight-Median Sketch

Algorithm 2: Active-Set Weight-Median (AWM) Sketch

initialization
- $S \leftarrow \{\}$ ▷ Empty heap

function `Update(x, y)`
- $\mathbf{x}_s \leftarrow \{x_i : i \in S\}$ ▷ Features in heap
- $\mathbf{x}_{\text{wm}} \leftarrow \{x_i : i \notin S\}$ ▷ Features in sketch
- $\tau \leftarrow \sum_{i \in S} S[i] \cdot x_i + \frac{1}{s}\mathbf{z}^T A\mathbf{x}_{\text{wm}}$ ▷ Prediction for $\mathbf{x}$
- $S \leftarrow (1 - \lambda\eta_t)S - \eta_t y \nabla\ell(y\tau)\mathbf{x}_s$ ▷ Heap update
- $\mathbf{z} \leftarrow (1 - \lambda\eta_t)\mathbf{z}$ ▷ Apply regularization
- **for** $i \notin S$ **do**
  - ▷ Either update i in sketch or move to heap
  - $\tilde{w} \leftarrow$ `Query(i)` $- \eta_t y x_i \nabla\ell(y\tau)$
  - $i_{\text{min}} \leftarrow \arg\min_j(|S[j]|)$
  - **if** $|\tilde{w}| > |S[i_{\text{min}}]|$ **then**
    - Remove i_{min} from S
    - Add i to S with weight $\tilde{w}$
    - Update i_{min} in sketch with $S[i_{\text{min}}] -$ `Query`(i_{min})
  - **else**
    - Update i in sketch with $\eta_t y x_i \nabla\ell(y\tau)$
- $t \leftarrow t + 1$

We now describe a simple, heuristic extension to the WM-Sketch that significantly improves the recovery accuracy of the sketch in practice. We refer to this variant as the Active-Set Weight-Median Sketch (AWM-Sketch).

To efficiently track the top elements across sketch updates, we can use a min-heap ordered by the absolute value of the estimated weights. This technique is also used alongside heavy-hitters sketches to identify the most frequent items in the stream (Charikar et al., 2002). In the basic WM-Sketch, the heap merely functions as a mechanism to passively maintain the heaviest weights. This baseline scheme can be improved by noting that weights that are already stored in the heap need not be tracked in the sketch; instead, the sketch can be updated lazily only when the weight is evicted from the heap. This heuristic has previously been used in the context of improving count estimates derived from a Count-Min Sketch (Roy et al., 2016). The intuition here is the following: since we are already maintaining a heap of heavy items, we can utilize this structure to reduce error in the sketch as a result of collisions with heavy items.

The heap can be thought of as an "active set" of high-magnitude weights, while the sketch estimates the contribution of the tail of the weight vector. Since the weights in the heap are represented exactly, this active set heuristic should intuitively yield better estimates of the heavily-weighted features in the model.

As a general note, similar coarse-to-fine approximation schemes have been proposed in other online learning settings. A similar scheme for memory-constrained sparse linear regression was analyzed by Steinhardt and Duchi (2015). Their algorithm similarly uses a Count-Sketch for approximating weights, but in a different setting (K-sparse linear regression) and with a different update policy for the active set.

4 Analysis

We derive bounds on the recovery error achieved by the WM-Sketch for given settings of the size k and depth s. The main challenge in our analysis is that the updates to the sketch depend on gradient estimates which in turn depend on the state of the sketch. This reflexive dependence makes it difficult to straightforwardly transplant the standard analysis for the Count-Sketch to our setting. Instead, we turn to ideas drawn from norm-preserving random projections and online convex optimization. In this section, we begin with an analysis of recovery error in the batch setting, where we are given access to a fixed dataset of size T

consisting of the first T examples observed in the stream and are allowed multiple passes over the data. Subsequently, we use this result to show guarantees in a restricted online case where we are only allowed a single pass through the data, but with the assumption that the order of the data is not chosen adversarially.

4.1 Batch Setting

To begin, we briefly outline the main ideas in our analysis. With high probability, we can sample a random projection to dimension $k \ll d$ that satisfies the JL norm preservation property (Definition 4). We use this property to show that for any fixed dataset of size T, optimizing a projected version of the objective yields a solution that is close to the projection of the minimizer of the original, high-dimensional objective. We then use the observation that our JL map can be identified as a Count-Sketch projection; therefore, we can make use of existing error bounds for Count-Sketch estimates to bound the error of our recovered weight estimates.

Let $R \in \mathbb{R}^{k \times d}$ denote the sketching matrix implicitly defined by the hashing construction described in Algorithm 1, scaled by $1/\sqrt{s}$: $R = \frac{1}{\sqrt{s}}A$. This is the hashing-based sparse JL projection proposed by Kane and Nelson (Kane and Nelson, 2014). We consider the following pair of objectives defined over the batch of T observed examples $(\mathbf{x}_t, y_t)$—the first defines the problem in the original space and the second defines the corresponding problem where the learner observes sketched examples $(R\mathbf{x}_t, y_t)$:

$$L(\mathbf{w}) = \frac{1}{T}\sum_{t=1}^{T} \ell\left(y_t \mathbf{w}^T \mathbf{x}_t\right) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2,$$
$$\hat{L}(\mathbf{z}) = \frac{1}{T}\sum_{t=1}^{T} \ell\left(y_t \mathbf{z}^T R\mathbf{x}_t\right) + \frac{\lambda}{2}\|\mathbf{z}\|_2^2,$$

with parameters $\mathbf{w} \in \mathbb{R}^d$ and $\mathbf{z} \in \mathbb{R}^k$.

Suppose we optimized these objectives to obtain solutions $\mathbf{w}_* = \arg\min_{\mathbf{w}} L(\mathbf{w})$ and $\mathbf{z}_* = \arg\min_{\mathbf{z}} \hat{L}(\mathbf{z})$. How then does $\mathbf{w}_*$ relate to $\mathbf{z}_*$ given our choice of sketching matrix R and regularization parameter λ? Intuitively, if we stored all the data observed up to time T and optimized $\mathbf{z}$ over this dataset, we should hope that the optimal solution $\mathbf{z}_*$ is close to $R\mathbf{w}_*$, the sketch of $\mathbf{w}_*$, in order to have any chance of recovering the largest weights of $\mathbf{w}_*$. We show that in this batch setting, $\|\mathbf{z}_* - R\mathbf{w}_*\|_2$ is indeed small; we then use this property to show element-wise error guarantees for the Count-Sketch recovery process. We first define a useful regularity condition on functions:

Definition 5. *A function $f : \mathcal{X} \to \mathbb{R}$ is β-strongly smooth w.r.t. a norm $\|\cdot\|$ if f is everywhere differentiable and if for all $\mathbf{x}, \mathbf{y}$ we have:*

$$f(\mathbf{y}) \leq f(\mathbf{x}) + (\mathbf{y} - \mathbf{x})^T \nabla f(\mathbf{x}) + \frac{\beta}{2}\|\mathbf{y} - \mathbf{x}\|^2.$$

We now state our main result for recovery error in the batch setting:

Theorem 1. *Let the loss function ℓ be β-strongly smooth (w.r.t. $\|\cdot\|_2$) and $\max_t \|\mathbf{x}_t\|_1 = \gamma$. For fixed constants $C_1, C_2 > 0$, let:*

$$k = \left(C_1/\epsilon^4\right)\log^3(d/\delta)\max\left\{1, \beta^2\gamma^4/\lambda^2\right\},$$
$$s = \left(C_2/\epsilon^2\right)\log^2(d/\delta)\max\left\{1, \beta\gamma^2/\lambda\right\}.$$

Let $\mathbf{w}_*$ be the minimizer of the original objective function $L(\mathbf{w})$ and $\mathbf{w}_{\text{est}}$ be the estimate of $\mathbf{w}_*$ returned by performing Count-Sketch recovery on the minimizer $\mathbf{z}_*$ of the projected objective function $\hat{L}(\mathbf{z})$. Then with probability $1 - \delta$ over the choice of R,

$$\|\mathbf{w}_* - \mathbf{w}_{\text{est}}\|_\infty \leq \epsilon\|\mathbf{w}_*\|_1.$$

We defer the full proof to Appendix A.1 but briefly discuss some salient properties of this recovery result. First, the feature dimension d enters only polylogarithmically into the sketch size k and the sparsity (i.e.,

depth) parameter s: this establishes that memory-efficient learning and recovery is possible when in the high-d regime that we are interested in. Moreover, our result is independent of the number of examples T; this is relevant as our applications involve learning over streams and datasets containing possibly millions of examples. For standard loss functions such as the logistic loss and the smoothed hinge loss, we have smoothness parameter $\beta = 1$. Finally, k and s scale inversely with the strength of ℓ_2 regularization: this is intuitive because additional regularization will shrink both $\mathbf{w}_*$ and $\mathbf{z}_*$ towards zero. We observe this inverse relationship between recovery error and ℓ_2 regularization in practice (see Fig. 5). Also, note that the recovery error depends on the maximum ℓ_1-norm γ of the data points $\mathbf{x}_t$, and the bound is most optimistic when γ is small. Across all of the applications we consider in Section 5 and Section 6, the data points are sparse with a small ℓ_1-norm, and hence the bound is meaningful across a number of interesting settings.

The per-parameter recovery error in Theorem 1 is bounded above by a multiple of the ℓ_1-norm of the optimal weights $\mathbf{w}_*$ for the uncompressed problem. This supports the intuition that sparse solutions with small ℓ_1-norm should be more easily recovered. In practice, we can augment the objective with an additional $\|\mathbf{w}\|_1$ (resp. $\|\mathbf{z}\|_1$) term to induce sparsity; this corresponds to elastic net-style composite ℓ_1/ℓ_2 regularization on the parameters of the model (Zou and Hastie, 2005).

4.2 Online Setting

We now provide guarantees for WM-Sketch in the online setting. We make two small modifications to WM-Sketch for the convenience of analysis. First, we assume that the iterate $\mathbf{z}_t$ is projected onto a ℓ_2 ball of radius D at every step. Second, we also assume that we perform the final Count-Sketch recovery on the average $\bar{\mathbf{z}} = \frac{1}{T}\sum_{i=1}^{T} \mathbf{z}_t$ of the weight vectors, instead of on the current iterate $\mathbf{z}_t$. While using this averaged sketch is useful for the analysis, maintaining a duplicate data structure in practice for the purpose of accumulating the average would double the space cost of our method. Therefore, in our implementation of the WM-Sketch, we simply maintain the current iterate $\mathbf{z}_t$. As we show in the next section this approach achieves good performance on real-world datasets, in particular when combined with the active set heuristic.

Our guarantee holds in expectation over uniformly random permutations of $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}$. In other words, we achieve low recovery error on average over all orderings in which the T data points could have been presented. We believe this condition is necessary to avoid worst-case adversarial orderings of the data points—since the WM-Sketch update at any time step depends on the state of the sketch itself, adversarial orderings can potentially lead to high error accumulation.

Theorem 2. *Let the loss function ℓ be β-strongly smooth (w.r.t. $\|\cdot\|_2$), and have its derivative bounded by H. Assume $\|\mathbf{x}_t\|_2 \leq 1, \max_t \|\mathbf{x}_t\|_1 = \gamma$, $\|\mathbf{w}_*\|_2 \leq D_2$ and $\|\mathbf{w}_*\|_1 \leq D_1$. Let G be a bound on the ℓ_2 norm of the gradient at any time step t, in our case $G \leq H(1 + \epsilon\gamma) + \lambda D$. For fixed constants $C_1, C_2, C_3 > 0$, let:*

$$k = \left(C_1/\epsilon^4\right) \log^3(d/\delta) \max\left\{1, \beta^2\gamma^4/\lambda^2\right\},$$
$$s = \left(C_2/\epsilon^2\right) \log^2(d/\delta) \max\left\{1, \beta\gamma^2/\lambda\right\},$$
$$T \geq (C_3/\epsilon^4)\zeta \log^2(d/\delta) \max\{1, \beta\gamma^2/\lambda\},$$

where $\zeta = (1/\lambda^2)(D_2/\|w_*\|_1)^2(G + (1 + \epsilon\gamma)H)^2$. Let $\mathbf{w}_*$ be the minimizer of the original objective function $L(\mathbf{w})$ and $\mathbf{w}_{\text{wm}}$ be the estimate $\mathbf{w}_*$ returned by the WM-Sketch algorithm with averaging and projection on the ℓ_2 ball with radius $D = (D_2 + \epsilon D_1)$. Then with probability $1 - \delta$ over the choice of R,

$$\mathbb{E}[\|\mathbf{w}_* - \mathbf{w}_{\text{wm}}\|_\infty] \leq \epsilon\|\mathbf{w}_*\|_1,$$

where the expectation is taken with respect to uniformly sampling a permutation in which the samples are received.

Again, we defer the full proof to Appendix A.2.

Remark. Let us compare the guarantees for finding *heavy-hitters* in data streams with our guarantees for finding *heavily-weighted features* in data streams. The Count-Sketch uses $\Theta(\epsilon^{-2}\log(d/\delta))$ space to obtain frequency estimates $\mathbf{v}_{\text{cs}}$ with error $\|\mathbf{v} - \mathbf{v}_{\text{cs}}\|_\infty \leq \epsilon\|\mathbf{v}\|_2$, where $\mathbf{v}$ is the true frequency vector (Lemma 6), while the Count-Min Sketch uses $\Theta\left(\epsilon^{-1}\log(d/\delta)\right)$ space for error bounded by $\epsilon\|\mathbf{v}\|_1$ (Cormode and Muthukrishnan, 2005b). In comparison, we use space $O\left(\epsilon^{-4}\log^3(d/\delta)\right)$ to achieve error $\mathbb{E}[\|\mathbf{w}_* - \mathbf{w}_{\text{wm}}\|_\infty] \leq \epsilon\|\mathbf{w}_*\|_1$

Dataset	# Examples	# Features	Space (MB)
Reuters RCV1	6.77×10^5	4.72×10^4	0.4
Malicious URLs	2.40×10^6	3.23×10^6	25.8
KDD Cup Algebra	8.41×10^6	2.02×10^7	161.8
Senate/House Spend.	4.08×10^7	5.14×10^5	4.2
Packet Trace	1.86×10^7	1.26×10^5	1.0
Newswire	2.06×10^9	4.69×10^7	375.2

Table 1: Summary of benchmark datasets with the space cost of representing full weight vectors and feature identifiers using 32-bit values. The first set of three consists of standard binary classification datasets used in Sec. 5; the second set consists of datasets specific to the applications in Sec. 6.

(Theorem 2). Thus, we obtain a guarantee of a similar flavor to bounds for heavy-hitters for this more general framework, but with somewhat worse polynomial dependencies.

5 Evaluation

In this section, we evaluate the Weight-Median Sketch on three standard binary classification datasets. Our goal here is to compare the WM-Sketch and AWM-Sketch against alternative limited-memory methods in terms of (1) recovery error in the estimated top-K weights, (2) classification error rate, and (3) runtime performance. In the next section, we explore specific applications of the WM-Sketch in stream processing tasks.

5.1 Datasets and Experimental Setup

We evaluated our proposed sketches on several standard benchmark datasets as well as in the context of specific streaming applications. Table 1 lists summary statistics for these datasets.

Classification Datasets. We evaluate the recovery error on ℓ_2-regularized online logistic regression trained on three standard binary classification datasets: Reuters RCV1 (Lewis et al., 2004), malicious URL identification (Ma et al., 2009), and the `Algebra` dataset from the KDD Cup 2010 large-scale data mining competition (Stamper et al., 2010; Yu et al., 2010). We use the standard training split for each dataset except for the RCV1 dataset, where we use the larger "test" split as is common in experimental evaluations using this dataset (Golovin et al., 2013).

For each dataset, we make a single pass through the set of examples. Across all our experiments, we use an initial learning rate $\eta_0 = 0.1$ and $\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. We used the following set of space constraints: 2KB, 4KB, 8KB, 16KB and 32KB. For each setting of the space budget and for each method, we evaluate a range of configurations compatible with that space constraint; for example, for evaluating the WM-Sketch, this corresponds to varying the space allocated to the heap and the sketch, as well as trading off between the sketch depth s and the width k/s. For each setting, we run 10 independent trials with distinct random seeds; our plots show medians and the range between the worst and best run.

Memory Cost Model. In our experiments, we control for memory usage and configure each method to satisfy the given space constraints using the following cost model: we charge 4B of memory utilization for each feature identifier, feature weight, and auxiliary weight (e.g., random keys in Algorithm 4 or counts in the Space Saving baseline) used. For example, a simple truncation instance (Algorithm 3) with 128 entries uses 128 identifiers and 128 weights, corresponding to a memory cost of 1024B.

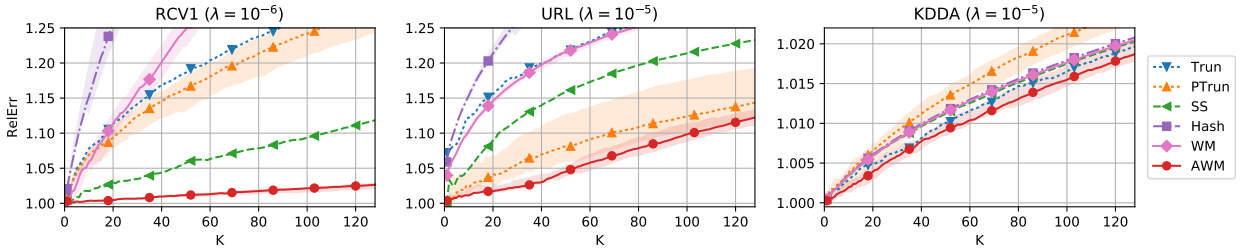

Figure 3: Relative ℓ_2 error of estimated top-K weights vs. true top-K weights for ℓ_2-regularized logistic regression under 8KB memory budget. Shaded area indicates range of errors observed over 10 trials. The AWM-Sketch achieves lower recovery error across all three datasets.

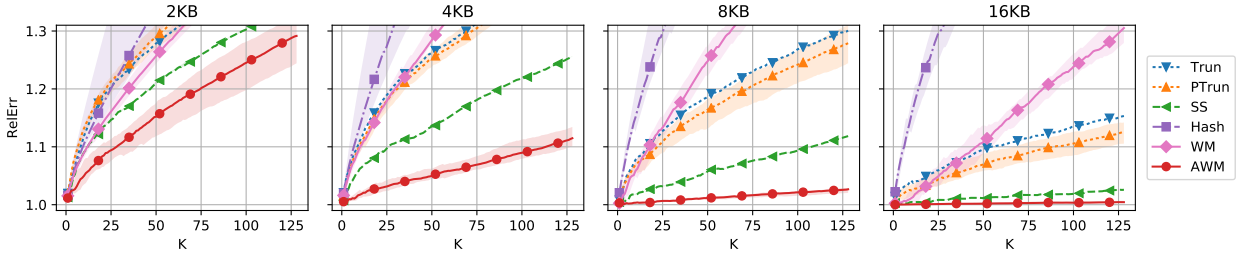

Figure 4: Relative ℓ_2 error of estimated top-K weights on `RCV1` dataset under different memory budgets ($\lambda = 10^{-6}$). Shaded area indicates range of errors observed over 10 trials. The recovery quality of the AWM-Sketch quickly improves with more allocated space.

5.2 Recovery Error Comparison

We measure the accuracy to which our methods are able to recover the top-K weights in the model using the following relative ℓ_2 error metric:

$$\text{RelErr}(\mathbf{w}^K, \mathbf{w}_*) = \frac{\|\mathbf{w}^K - \mathbf{w}_*\|_2}{\|\mathbf{w}_*^K - \mathbf{w}_*\|_2},$$

where $\mathbf{w}^K$ is the K-sparse vector representing the top-K weights returned by a given method, $\mathbf{w}_*$ is the weight vector obtained by the uncompressed model, and $\mathbf{w}_*^K$ is the K-sparse vector representing the true top-K weights in $\mathbf{w}_*$. The relative error metric is therefore bounded below by 1 and quantifies the relative suboptimality of the estimated top-K weights. The best configurations for the WM- and AWM-Sketch on `RCV1` are listed in Table 2; the optimal configurations for the remaining datasets are similar.

We compare our methods across datasets (Fig. 3) and across memory constraints on a single dataset (Fig. 4). For clarity, we omit the Count-Min Frequent Features baseline since we found that the Space Saving baseline achieved consistently better performance. We found that the AWM-Sketch consistently achieved lower recovery error than alternative methods on our benchmark datasets. The Space Saving baseline is competitive on `RCV1` but underperforms the simple Probabilistic Truncation baseline on `URL`: this demonstrates that tracking frequent features can be effective if frequently-occurring features are also highly discriminative, but this property does not hold across all datasets. Standard feature hashing achieves poor recovery error since colliding features cannot be disambiguated.

In Fig. 5, we compare recovery error on `RCV1` across different settings of λ. Higher ℓ_2-regularization results in less recovery error since both the true weights and the sketched weights are closer to 0; however, λ settings that are too high can result in increased classification error.

5.3 Classification Error Rate

We evaluated the classification performance of our models by measuring their online error rate (Blum et al., 1999): for each observed pair $(\mathbf{x}_t, y_t)$, we record whether the prediction $\hat{y}_t$ (made without observing y_t) is

	WM-Sketch			AWM-Sketch		
Budget (KB)	$\|S\|$	width	depth	$\|S\|$	width	depth
2	128	128	2	128	256	1
4	256	256	2	256	512	1
8	128	128	14	512	1024	1
16	128	128	30	1024	2048	1
32	128	256	31	2048	4096	1

Table 2: Sketch configurations with minimal ℓ_2 recovery error on `RCV1` dataset ($|S|$ denotes heap capacity).

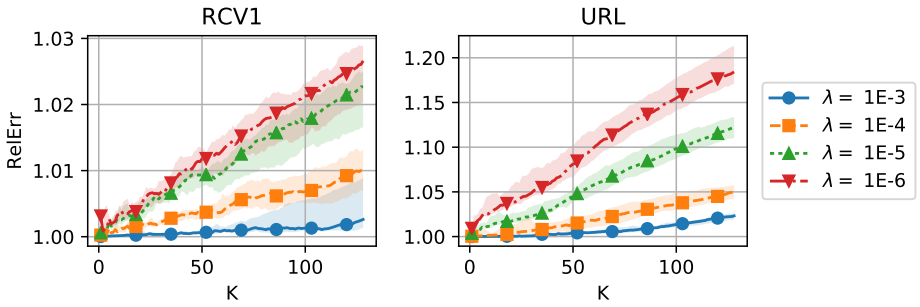

Figure 5: Relative ℓ_2 error of top-K AWM-Sketch estimates with varying regularization parameter λ on `RCV1` and `URL` datasets under 8KB memory budget.

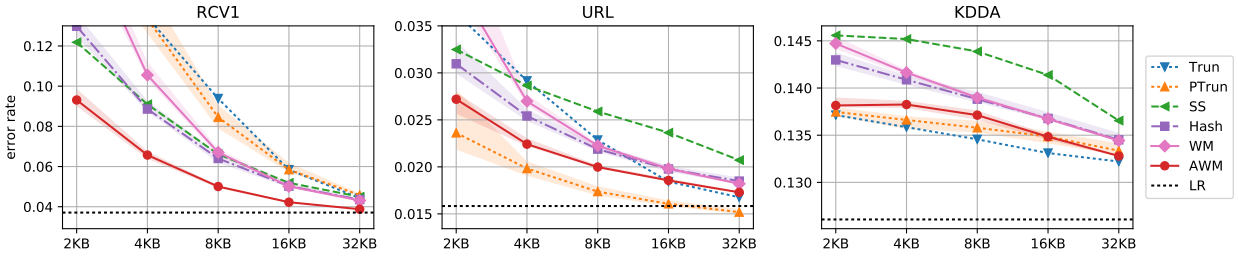

Figure 6: Online classification error rates with ℓ_2-regularized logistic regression under different memory budgets (`Trun` = Simple Truncation, `PTrun` = Probabilistic Truncation, `SS` = Space Saving Frequent, `Hash` = Feature Hashing, `LR` = Logistic Regression without memory constraints). The AWM-Sketch consistently achieves better classification accuracy than methods that track frequent features.

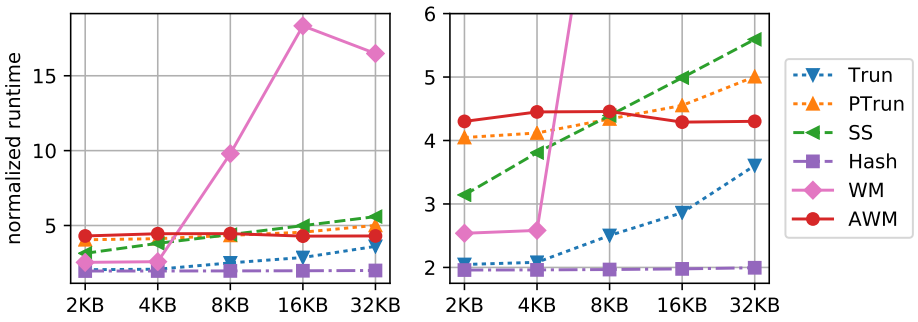

Figure 7: Normalized runtime of each method vs. memory-unconstrained logistic regression on `RCV1` using configurations that minimize recovery error (see Table 2). The right panel is a zoomed-in view of the left panel.

correct before updating the model. The error rate is defined as the cumulative number of mistakes made divided by the number of iterations. Our results here are summarized in Fig. 6. For each dataset, we used the value of λ that achieved the lowest error rate across all our memory-limited methods. For each method and for each memory budget, we chose the configuration that achieved the lowest error rate. For the WM-Sketch, this corresponded to a width of 2^7 or 2^8 with depth scaling proportionally with the memory budget; for the AWM-Sketch, the configuration that uniformly performed best allocated half the space to the active set and the remainder to a depth-1 sketch (i.e., a single hash table without any replication).

We found that across all tested memory constraints, the AWM-Sketch consistently achieved lower error rate than heavy-hitter-based methods. Surprisingly, the AWM-Sketch outperformed feature hashing by a small but consistent margin: 0.5–3.7% on `RCV1`, 0.1–0.4% on `URL`, and 0.2–0.5% on `KDDA`, with larger gains seen at smaller memory budgets. This suggests that the AWM-Sketch benefits from the precise representation of the largest, most-influential weights in the model, and that these gains are sufficient to offset the increased collision rate due to the smaller hash table. The Space Saving baseline exhibited inconsistent performance across the three datasets, demonstrating that tracking the most frequent features is an unreliable heuristic: features that occur frequently are not necessarily the most predictive. We note that higher values of the regularization parameter λ correspond to greater penalization of rarely-occurring features; therefore, we would expect the Space Saving baseline to better approximate the performance of the unconstrained classifier as λ increases.

5.4 Runtime Performance

We evaluated runtime performance relative to a memory unconstrained logistic regression model using the same configurations as those chosen to minimize ℓ_2 recovery error (Table 2). In all our timing experiments, we ran our implementations of the baseline methods, the WM-Sketch, and the AWM-Sketch on Intel Xeon E5-2690 v4 processor with 35MB cache using a single core. The memory-unconstrained logistic regression weights were stored using an array of 32-bit floating point values of size equal to the dimensionality of the feature space, with the highest-weighted features tracked using a min-heap of size $K = 128$; reads and writes to the weight vector therefore required single array accesses. The remaining methods tracked heavy weights alongside 32-bit feature identifiers using a min-heap sized according to the corresponding configuration.

In our experiments, the fastest method was feature hashing, with about a 2x overhead over the baseline. This overhead was due to the additional hashing step needed for each read and write to a feature index. The AWM-Sketch incurred an additional 2x overhead over feature hashing due to more frequent heap maintenance operations.

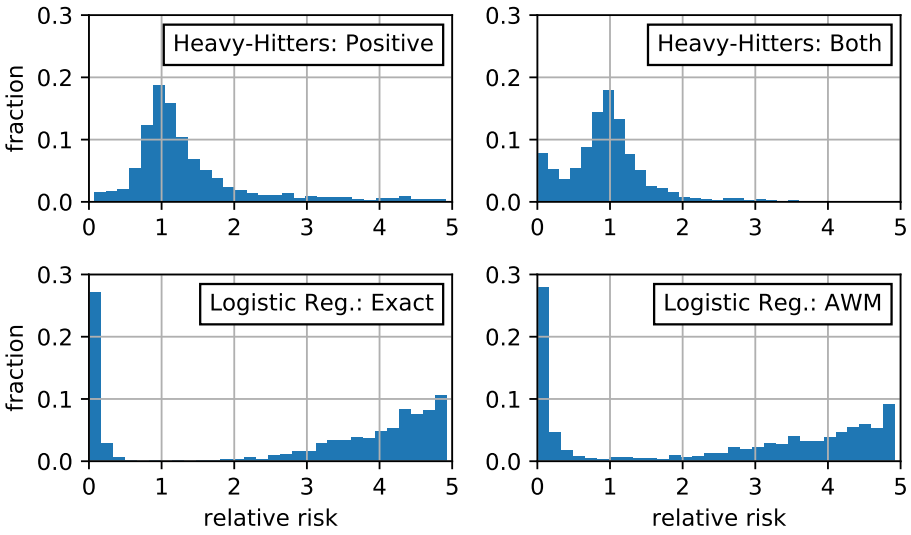

Figure 8: Distribution of relative risks among top-2048 features retrieved by each method. *Top Row*: Heavy-Hitters. *Bottom Row*: Classifier-based methods.

6 Applications

We now show that a variety of tasks in stream processing can be framed as memory-constrained classification. The unifying theme between these applications is that classification is a useful abstraction whenever the use case calls for *discriminating* between streams or between subpopulations of a stream. These distinct classes can be identified by partitioning a single stream into quantiles (Sec. 6.1), comparing separate streams (Sec. 6.2), or even by generating *synthetic* examples to be distinguished from real samples (Sec. 6.3).

6.1 Streaming Explanation

In data analysis workflows, it is often necessary to identify characteristic attributes that are particularly indicative of a given subset of data (Meliou et al., 2014). For example, in order to diagnose the cause of anomalous readings in a sensor network, it is helpful to identify common features of the outlier points such as geographical location or time of day. This use case has motivated the development of methods for finding common properties of outliers found in aggregation queries (Wu and Madden, 2013) and in data streams (Bailis et al., 2017).

This task can be framed as a classification problem: assign positive labels to the outliers and negative labels to the inliers, then train a classifier to discriminate between the two classes. The identification of characteristic attributes is then reduced to the problem of identifying heavily-weighted features in the trained model. In order to identify indicative *conjunctions* of attributes, we can simply augment the feature space to include arbitrary combinations of singleton features.

The *relative risk* or *risk ratio* $r_x = p(y = 1 \mid x = 1)/p(y = 1 \mid x = 0)$ is a statistical measure of the relative occurrence of the positive label $y = 1$ when the feature x is active versus when it is inactive. In the context of stream processing, the relative risk has been used to quantify the degree to which a particular attribute or attribute combination is indicative of a data point being an outlier relative to the overall population (Bailis et al., 2017). Here, we are interested in comparing our classifier-based approach to identifying high-risk features against the approach used in `MacroBase` (Bailis et al., 2017), an existing system for explaining outliers over streams, that identifies candidate attributes using a variant of the Space Saving heavy-hitters algorithm.

Experimental Setup. We used a publicly-available dataset of itemized disbursements by candidates in U.S. House and Senate races from 2010–2016.[3] The outlier points were set to be the set of disbursements

[3]FEC candidate disbursements data: http://classic.fec.gov/data/CandidateDisbursement.do

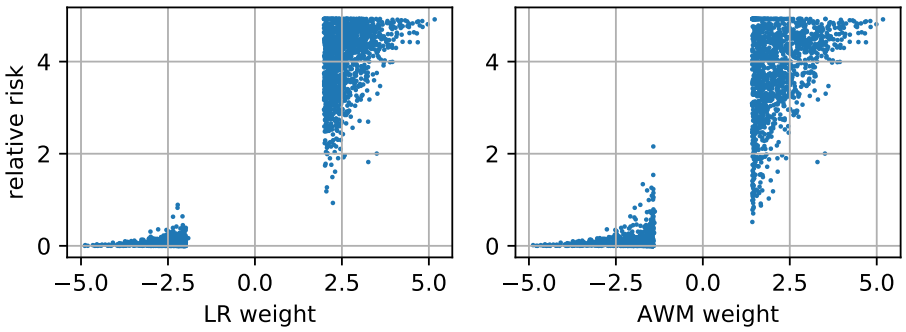

Figure 9: Correlation between top-2048 feature weights and relative risk. *Left*: Memory-unconstrained logistic regression (Pearson correlation 0.95). *Right*: AWM-Sketch (Pearson correlation 0.91).

in the top-20% by dollar amount. For each row of the data, we generated a sequence of 1-sparse feature vectors[4] corresponding to the observed attributes. We set a space budget of 32KB for the AWM-Sketch.

Results. Our results are summarized in Figs. 8 and 9. The former empirically demonstrates that the heuristic of filtering features on the basis of frequency can be suboptimal for a fixed memory budget. This is due to features that are frequent in both the inlier and outlier classes: it is wasteful to maintain counts for these items since they have low relative risk. In Fig. 8, the top row shows the distribution of relative risks among the most frequent items within the positive class (left) and across both classes (right). In contrast, our classifier-based approaches use the allocated space more efficiently by identifying features at the extremes of the relative risk scale.

In Fig. 9, we show that the learned classifier weights are strongly correlated with the relative risk values estimated from true counts. Indeed, logistic regression weights can be interpreted in terms of log odds ratios, a related quantity to relative risk. These results show that the AWM-Sketch is a superior filter compared to Heavy-Hitters approaches for identifying high-risk features.

6.2 Network Monitoring

IP network monitoring is one of the primary application domains for sketches and other small-space summary methods (Venkataraman et al., 2005; Bandi et al., 2007; Yu et al., 2013). Here, we focus on the problem of finding packet-level features (for instance, source/destination IP addresses and prefixes, port numbers, network protocols, and header or payload characteristics) that differ significantly in relative frequency between a pair of network links.

This problem of identifying significant relative differences—also known as *relative deltoids*—was studied by Cormode and Muthukrishnan (Cormode and Muthukrishnan, 2005a). Concretely, the problem is to estimate—for each item i—ratios $\phi(i) = n_1(i)/n_2(i)$ (where n_1, n_2 denote occurrence counts in each stream) and to identify those items i for which this ratio, or its reciprocal, is large. Here, we are interested in identifying differences between traffic streams that are observed *concurrently*; in contrast, the empirical evaluation in (Cormode and Muthukrishnan, 2005a) focused on comparisons between different time periods.

Experimental Setup. We used a subset of an anonymized, publicly-available passive traffic trace dataset recorded at a peering link for a large ISP (cai). The positive class was the stream of outbound source IP addresses and the negative class was the stream of inbound destination IP addresses. We compared against several baseline methods, including ratio estimation using a pair of Count-Min sketches (as in (Cormode and Muthukrishnan, 2005a)). For each method we retrieved the top-2048 features (i.e., IP addresses in this case) and computed the recall against the set of features above the given ratio threshold, where the reference ratios were computed using exact counts.

Results. We found that the AWM-Sketch performed comparably to the memory-unconstrained logistic

[4]We can also generate a single feature vector per row (with sparsity greater than 1), but the learned weights would then correlate more weakly with the relative risk. This is due to the effect of correlations between features.

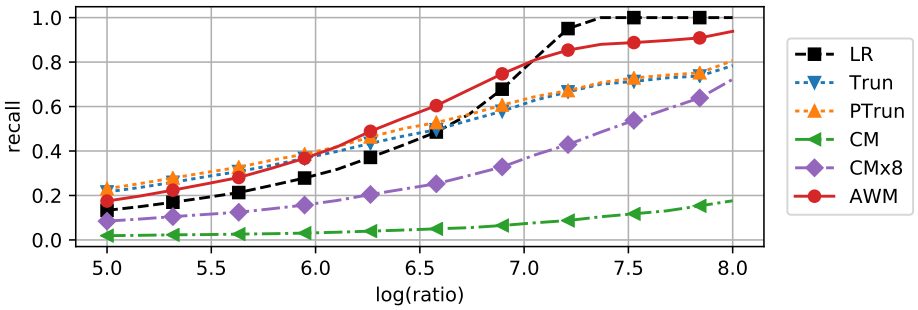

Figure 10: Recall of IP addresses with relative occurrence ratio above the given threshold with 32KB of space. `LR` denotes recall by full memory-unconstrained logistic regressor. `CMx8` denotes Count-Min baseline with 8x memory usage.

regression baseline on this benchmark. We significantly outperformed the paired Count-Min baseline by a factor of over 4× in recall while using the same memory budget, as well as a paired CM baseline that was allocated 8x the memory budget. These results indicate that linear classifiers can be used effectively to identify relative deltoids over pairs of data streams.

6.3 Streaming Pointwise Mutual Information

Pointwise mutual information (PMI), a measure of the statistical correlation between a pair of events, is defined as:

$$\mathrm{PMI}(x, y) = \log \frac{p(x, y)}{p(x)p(y)}.$$

Intuitively, positive values of the PMI indicate events that are positively correlated, negative values indicate events that are negatively correlated, and a PMI of 0 indicates uncorrelated events.

In natural language processing, PMI is a frequently-used measure of word association (Turney and Pantel, 2010). Traditionally, the PMI is estimated using empirical counts of unigrams and bigrams obtained from a text corpus. The key problem with this approach is that the number of bigrams in standard natural language corpora can grow very large; for example, we found ∼47M unique co-occurring pairs of tokens in a small subset of a standard newswire corpus. This combinatorial growth in the feature dimension is further amplified when considering higher-order generalizations of PMI.

More generally, streaming PMI estimation can be used to detect pairs of events whose occurrences are strongly correlated. For example, we can consider a streaming log monitoring use case where correlated events are potentially indicative of cascading failures or trigger events resulting in exceptional behavior in the system. Therefore, we expect that the techniques developed here should be useful beyond standard NLP applications.

Sparse Online PMI Estimation. Streaming PMI estimation using approximate counting has previously been studied (Durme and Lall, 2009); however, this approach has the drawback that memory usage still scales linearly with the number of observed bigrams. Here, we explore streaming PMI estimation from a different perspective: we pose a binary classification problem over the space of bigrams with the property that the model weights asymptotically converge to an estimate of the PMI.[5]

The classification problem is set up as follows: in each iteration t, with probability 0.5 sample a bigram (u, v) from the bigram distribution $p(u, v)$ and set $y_t = +1$; with probability 0.5 sample (u, v) from the unigram product distribution $p(u)p(v)$ and set $y_t = -1$. The input $\mathbf{x}_t$ is the 1-sparse vector where the index corresponding to (u, v) is set to 1. We train a logistic regression model to discriminate between the *true* and *synthetic* samples. If $\lambda = 0$, the model asymptotically converges to the distribution $\hat{p}(y = 1 \mid (u, v)) = f(w_{uv}) = p(u, v)/\left(p(u, v) + p(u)p(v)\right)$ for all pairs (u, v), where f is the logistic function. It follows that

[5]This classification formulation is used in the popular `word2vec` skip-gram method for learning word embeddings (Mikolov et al., 2013); the connection to PMI approximation was first observed by Levy and Goldberg (2014). To our knowledge, we are the first to apply this formulation in the context of sparse PMI estimation.

Pair	PMI	Est.	Pair	PMI
prime minister	6.339	7.609	, the	0.044
los angeles	7.197	7.047	the ,	-0.082
http /	6.734	7.001	the of	0.611
human rights	6.079	6.721	the .	0.057

Table 3: *Left:* Top recovered pairs with PMI computed from true counts and PMI estimated from model weights (2^{16} bins, 1.4MB total memory). *Right:* Most common pairs in corpus.

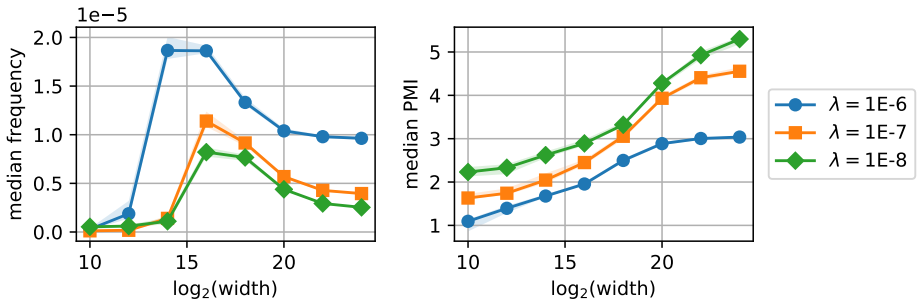

Figure 11: Median frequencies and exact PMIs of retrieved pairs with Active-Set Weight-Median Sketch estimation. Lower λ settings and higher bin counts favor less frequent pairs.

$w_{uv} = \log(p(u,v)/p(u)p(v))$, which is exactly the PMI of (u,v). If $\lambda > 0$, we obtain an estimate that is biased, but with reduced variance in the estimates for rare bigrams.

Experimental Setup. We train on a subset of a standard newswire corpus (Chelba et al., 2013); the subset contains 77.7M tokens, 605K unique unigrams and 47M unique bigrams over a sliding window of size 6. In our implementation, we approximate sampling from the unigram distribution by sampling from a reservoir sample of tokens (May et al., 2017; Kaji and Kobayashi, 2017). We estimated weights using the AWM-Sketch with heap size 1024 and depth 1; the reservoir size was fixed at 4000. We make a single pass through the dataset and generate 5 negative samples for every true sample. Strings were first hashed to 32-bit values using `MurmurHash3`;[6] these identifiers were hashed again to obtain sketch bucket indices.

Results. For width settings up to 2^{16}, our implementation's total memory usage was at most 1.4MB. In this regime, memory usage was dominated by the storage of strings in the heap and the unigram reservoir. For comparison, the standard approach to PMI estimation requires 188MB of space to store exact 32-bit counts for all bigrams, excluding the space required for storing strings or the token indices corresponding to each count. In Table 3, we show sample pairs retrieved by our method; the PMI values estimated from exact counts are well-estimated by the classifier weights. In Fig. 11, we show that at small widths, the high collision rate results in the retrieval of noisy, low-PMI pairs; as the width increases, we retrieve higher-PMI pairs which typically occur with lower frequency. Further, regularization helps discard low-frequency pairs but can result in the model missing out on high-PMI but less-frequent pairs.

7 Related Work

Heavy-Hitters in Data Streams. The Heavy-Hitters problem has been extensively studied in the streaming algorithms literature. Given a sequence of items, the goal is to return the set of all items whose frequency exceeds a specified fraction of the total number of items. Algorithms for finding frequent items can be broadly categorized into counter-based approaches (Manku and Motwani, 2002; Demaine et al., 2002; Karp et al., 2003; Metwally et al., 2005), quantile algorithms (Greenwald and Khanna, 2001; Shrivastava et al., 2004), and sketch-based methods (Charikar et al., 2002; Cormode and Muthukrishnan, 2005b).

[6] https://github.com/aappleby/smhasher/wiki/MurmurHash3

Mirylenka et al. (Mirylenka et al., 2015) develop streaming algorithms for finding *conditional* heavy-hitters, where the goal is to identify items that are frequent in the context of a separate "parent" item. While these methods can be used in related applications to ours, their counter-based algorithms differ significantly from our classification-based approach.

Characterizing Changes in Data Streams. The problem of detecting and characterizing significant absolute and relative differences between data streams has been studied by several authors. Cormode and Muthukrishnan (2005a) proposed a Count-Min-based algorithm for identifying items whose occurrence rates differ significantly; Schweller et al. (2004) propose reversible hashes in this context to avoid storing key information. These approaches focus primarily on identifying differences between time periods, whereas we focus on the case where streams are observed concurrently. In order to explain anomalous traffic flows, Brauckhoff et al. (2012) proposed techniques using histogram-based detectors and association rules. Using histograms to identify changes in feature occurrence is appropriate for detecting large absolute differences; in contrast, we focus on detecting relative differences, a problem which has previously been found to be challenging (Cormode and Muthukrishnan, 2005a).

Resource-Efficient ML. Gupta et al. (2017) and Kumar et al. (2017) explore the use of tree- and k-nearest-neighbor-based classifiers on highly resource-constrained devices. A large body of work studies resource-efficient inference at test time (Xu et al., 2012, 2013; Hsieh et al., 2014). Unlike budget kernel methods (Crammer et al., 2004) that maintain a small set of support vectors, we aim to directly restrict the number of parameters stored. Our work also differs from *model compression* or *distillation*, which aims to imitate a large, expensive model using a smaller one with lower memory and computation costs (Bucilu? et al., 2006; Ba and Caruana, 2014; Hinton et al., 2015).

Sparsity-Inducing Regularization. ℓ_1-regularization is a standard technique for encouraging parameter sparsity in online learning (Langford et al., 2009; Duchi and Singer, 2009; Xiao, 2010; McMahan, 2011). In practice, it is difficult to *a priori* select an ℓ_1 regularization strength in order to satisfy a given sparsity budget; it can therefore be problematic to apply standard online learning methods in settings with hard memory constraints. In this paper, we propose a different approach: we first fix a memory budget and then use the allocated space to approximate a classifier, with the property that our approximation will be better for parameter vectors with small ℓ_1-norm (see Theorem 2). We note that the WM-Sketch and AWM-Sketch are compatible with the use of an ℓ_1-regularizer in addition to the requisite ℓ_2 term.

Feature Hashing. Feature hashing (Shi et al., 2009; Weinberger et al., 2009) is a technique where the classifier is trained on features that have been hashed to a fixed-width table. This approach lowers memory usage by reducing the dimension of the feature space and by obviating the need for storing feature identifiers, but at the cost of model interpretability.

Compressed Learning. Calderbank et al. (Calderbank et al.) introduced *compressed learning*, where a classifier is trained on data obtained via compressive sensing (Candes and Tao, 2006; Donoho, 2006). The authors focus on classification performance in the compressed domain and do not consider the problem of recovering weights in the original space.

8 Discussion

Active Set vs. Multiple Hashing. In the basic WM-Sketch, multiple hashing is needed in order to disambiguate features that collide in a heavy bucket; we should expect that features with truly high weight should correspond to large values in the majority of buckets that they hash to. The active set approach uses a different mechanism for disambiguation. Suppose that all the features that hash to a heavy bucket are added to the active set; we should expect that the weights for those features that were erroneously added will eventually decay (due to ℓ_2-regularization) to the point that they are evicted from the active set. Simultaneously, the truly high-weight features are retained in the active set. The AWM-Sketch can therefore be interpreted as a variant of feature hashing where the highest-weighted features are not hashed.

The Cost of Interpretability. We initially motivated the development of the WM-Sketch with the dual goals of memory-efficient learning and model interpretability. A natural question to ask is: what is the cost of interpretability? What do we sacrifice in classification accuracy when we allocate memory to storing feature

identifiers relative to feature hashing, which maintains only feature weights? A surprising finding in our evaluation on standard binary classification datasets was that the AWM-Sketch achieved uniformly better classification accuracy compared to feature hashing. We hypothesize that the observed gains are due to reduced collisions with heavily-weighted features. This result suggests that in some cases, we can essentially gain model interpretability *for free*.

Per-Feature Learning Rates. In previous work on online learning applications, practitioners have found that the per-feature learning rates can significantly improve classification performance (McMahan et al., 2013). An open question is whether variable learning rate across features is worth the associated memory cost in the streaming setting.

Multiclass Classification. The WM-Sketch be extended to the multiclass setting using the following simple extension. Given M output classes, for each row in the sketch, hash each feature using M independent hash functions, thus maintaining M weights for each feature. In order to predict the output distribution, evaluate the inner product of the feature vector with each set of weights and apply a softmax function to the result. For large M, for instance in language modeling applications, this procedure can be computationally expensive since update time scales linearly with M. In this regime, we can apply noise contrastive estimation (Gutmann and Hyvärinen, 2010)—a standard reduction to binary classification—to learn the model parameters.

Further Extensions. The WM-Sketch is likely amenable to further extensions that can improve update time and further reduce memory usage. Our method is equivalent to online gradient descent using random projections of input features, and gradient updates can be performed asynchronously with relaxed cache coherence requirements between cores (Recht et al., 2011). Additionally, our methods are orthogonal to reduced-precision techniques like randomized rounding (Raghavan and Tompson, 1987; Golovin et al., 2013) and approximate counting (Flajolet, 1985); these methods can be used in combination to realize further memory savings.

9 Conclusions

In this paper, we introduced the Weight-Median Sketch for the problem of identifying heavily-weighted features in linear classifiers over streaming data. We showed theoretical guarantees for our method, drawing on techniques from online learning and norm-preserving random projections. In our empirical evaluation, we showed that the Active Set extension to the basic Weight-Median Sketch method is highly effective, achieving superior weight recovery and competitive classification error compared to baseline methods across several standard binary classification benchmarks. Finally, we explored promising applications of our methods by framing existing stream processing tasks as classification problems. We believe this machine learning perspective on sketch-based stream processing may prove to be a fruitful direction for future research in advanced streaming analytics.

References

The caida ucsd anonymized passive oc48 internet traces dataset. http://www.caida.org/data/passive/passive_oc48_dataset.xml.

Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.

Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

Peter Bailis, Edward Gan, Samuel Madden, Deepak Narayanan, Kexin Rong, and Sahaana Suri. Macrobase: Prioritizing attention in fast data. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 541–556. ACM, 2017.

Nagender Bandi, Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Fast data stream algorithms using associative memories. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 247–256. ACM, 2007.

Avrim Blum, Adam Kalai, and John Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 203–208. ACM, 1999.

Oscar Boykin, Sam Ritchie, Ian O'Connell, and Jimmy Lin. Summingbird: A framework for integrating batch and online mapreduce computations. *Proceedings of the VLDB Endowment*, 7(13):1441–1451, 2014.

Daniela Brauckhoff, Xenofontas Dimitropoulos, Arno Wagner, and Kavè Salamatian. Anomaly extraction in backbone networks using association rules. *IEEE/ACM Transactions on Networking (TON)*, 20(6): 1788–1799, 2012.

Cristian Bucilu?, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006.

Robert Calderbank, Sina Jafarpour, and Robert Schapire. Compressed learning: Universal sparse dimensionality reduction and learning in the measurement domain.

Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on information theory*, 52(12):5406–5425, 2006.

J Lawrence Carter and Mark N Wegman. Universal classes of hash functions. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 106–112. ACM, 1977.

Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Automata, languages and programming*, pages 784–784, 2002.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. Algorithmic decision making and the cost of fairness. *arXiv preprint arXiv:1701.08230*, 2017.

Graham Cormode and Marios Hadjieleftheriou. Finding frequent items in data streams. *Proceedings of the VLDB Endowment*, 1(2):1530–1541, 2008.

Graham Cormode and S Muthukrishnan. What's new: Finding significant differences in network data streams. *IEEE/ACM Transactions on Networking (TON)*, 13(6):1219–1232, 2005a.

Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005b.

Koby Crammer, Jaz Kandola, and Yoram Singer. Online classification on a budget. In *Advances in neural information processing systems*, pages 225–232, 2004.

Erik D Demaine, Alejandro López-Ortiz, and J Ian Munro. Frequency estimation of internet packet streams with limited space. In *European Symposium on Algorithms*, pages 348–360. Springer, 2002.

David L Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10(Dec):2899–2934, 2009.

Benjamin V Durme and Ashwin Lall. Streaming pointwise mutual information. In *Advances in Neural Information Processing Systems*, pages 1892–1900, 2009.

Pavlos S Efraimidis and Paul G Spirakis. Weighted random sampling with a reservoir. *Information Processing Letters*, 97(5):181–185, 2006.

Philippe Flajolet. Approximate counting: a detailed analysis. *BIT Numerical Mathematics*, 25(1):113–134, 1985.

Daniel Golovin, D Sculley, Brendan McMahan, and Michael Young. Large-scale learning with less ram via randomization. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 325–333, 2013.

Bryce Goodman and Seth Flaxman. Eu regulations on algorithmic decision-making and a ?right to explanation? In *ICML workshop on human interpretability in machine learning (WHI 2016), New York, NY. http://arxiv. org/abs/1606.08813 v1*, 2016.

Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. In *ACM SIGMOD Record*, volume 30, pages 58–66. ACM, 2001.

Chirag Gupta, Arun Sai Suggala, Ankit Goyal, Harsha Vardhan Simhadri, Bhargavi Paranjape, Ashish Kumar, Saurabh Goyal, Raghavendra Udupa, Manik Varma, and Prateek Jain. Protonn: Compressed and accurate knn for resource-scarce devices. In *International Conference on Machine Learning*, pages 1331–1340, 2017.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Steven CH Hoi, Jialei Wang, Peilin Zhao, and Rong Jin. Online feature selection for mining big data. In *Proceedings of the 1st international workshop on big data, streams and heterogeneous source mining: Algorithms, systems, programming models and applications*, pages 93–100. ACM, 2012.

Cho-Jui Hsieh, Si Si, and Inderjit S Dhillon. Fast prediction for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 3689–3697, 2014.

William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Nobuhiro Kaji and Hayato Kobayashi. Incremental skip-gram model with negative sampling. *arXiv preprint arXiv:1704.03956*, 2017.

Daniel M Kane and Jelani Nelson. Sparser Johnson-Lindenstrauss transforms. *Journal of the ACM (JACM)*, 61(1):4, 2014.

Richard M Karp, Scott Shenker, and Christos H Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems (TODS)*, 28(1):51–55, 2003.

Ashish Kumar, Saurabh Goyal, and Manik Varma. Resource-efficient machine learning in 2 kb ram for the internet of things. In *International Conference on Machine Learning*, pages 1935–1944, 2017.

John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10(Mar):777–801, 2009.

Kasper Green Larsen, Jelani Nelson, Huy L Nguyên, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 61–70. IEEE, 2016.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

Ge Luo, Lu Wang, Ke Yi, and Graham Cormode. Quantiles over data streams: experimental comparisons, new analyses, and further improvements. *The VLDB Journal*, 25(4):449–472, 2016.

Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 681–688. ACM, 2009.

Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 346–357. VLDB Endowment, 2002.

Chandler May, Kevin Duh, Benjamin Van Durme, and Ashwin Lall. Streaming word embeddings with the space-saving algorithm. *arXiv preprint arXiv:1704.07463*, 2017.

Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 525–533, 2011.

H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

Alexandra Meliou, Sudeepa Roy, and Dan Suciu. Causality and explanations in databases. In *VLDB*, 2014.

Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *International Conference on Database Theory*, pages 398–412. Springer, 2005.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Katsiaryna Mirylenka, Graham Cormode, Themis Palpanas, and Divesh Srivastava. Conditional heavy hitters: detecting interesting correlations in data streams. *The VLDB Journal*, 24(3):395–414, 2015.

Michael Mitzenmacher and Salil Vadhan. Why simple hash functions work: exploiting the entropy in a data stream. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 746–755. Society for Industrial and Applied Mathematics, 2008.

Mihai P?traşcu and Mikkel Thorup. The power of simple tabulation hashing. *Journal of the ACM (JACM)*, 59(3):14, 2012.

Prabhakar Raghavan and Clark D Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM, 2016.

Pratanu Roy, Arijit Khan, and Gustavo Alonso. Augmented sketch: Faster and more accurate stream processing. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1449–1463. ACM, 2016.

Robert Schweller, Ashish Gupta, Elliot Parsons, and Yan Chen. Reversible sketches for efficient and accurate change detection over network data streams. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 207–212. ACM, 2004.

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

Ohad Shamir. Without-replacement sampling for stochastic gradient methods. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 46–54. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6245-without-replacement-sampling-for-stochastic-gradient-methods.pdf.

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alexander L Strehl, Alex J Smola, and SVN Vishwanathan. Hash kernels. In *International Conference on Artificial Intelligence and Statistics*, pages 496–503, 2009.

Nisheeth Shrivastava, Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. Medians and beyond: new aggregation techniques for sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 239–249. ACM, 2004.

J. Stamper, A. Niculescu-Mizil, S. Ritter, G.J. Gordon, and K.R. Koedinger. Algebra i 2008-2009. challenge data set from kdd cup 2010 educational data mining challenge., 2010. Find it at http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp.

Jacob Steinhardt and John Duchi. Minimax rates for memory-bounded sparse linear regression. In *Conference on Learning Theory*, pages 1564–1587, 2015.

Peter D Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

Shoba Venkataraman, Dawn Song, Phillip B Gibbons, and Avrim Blum. New streaming algorithms for fast detection of superspreaders. *Department of Electrical and Computing Engineering*, page 6, 2005.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.

Eugene Wu and Samuel Madden. Scorpion: Explaining away outliers in aggregate queries. *Proceedings of the VLDB Endowment*, 6(8):553–564, 2013.

Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11(Oct):2543–2596, 2010.

Zhixiang Xu, Kilian Weinberger, and Olivier Chapelle. The greedy miser: Learning under test-time budgets. *arXiv preprint arXiv:1206.6451*, 2012.

Zhixiang Xu, Matt Kusner, Kilian Weinberger, and Minmin Chen. Cost-sensitive tree of classifiers. In *International Conference on Machine Learning*, pages 133–141, 2013.

Tianbao Yang, Lijun Zhang, Rong Jin, and Shenghuo Zhu. Theory of dual-sparse regularized randomized reduction. In *International Conference on Machine Learning*, pages 305–314, 2015.

Hsiang-Fu Yu, Hung-Yi Lo, Hsun-Ping Hsieh, Jing-Kai Lou, Todd G McKenzie, Jung-Wei Chou, Po-Han Chung, Chia-Hua Ho, Chun-Fu Chang, Yin-Hsuan Wei, et al. Feature engineering and classifier ensemble for kdd cup 2010. In *KDD Cup*, 2010.

Minlan Yu, Lavanya Jose, and Rui Miao. Software defined traffic measurement with opensketch. In *NSDI*, volume 13, pages 29–42, 2013.

Ce Zhang, Arun Kumar, and Christopher Ré. Materialization optimizations for feature selection workloads. *ACM Transactions on Database Systems (TODS)*, 41(1):2, 2016.

Lijun Zhang, Mehrdad Mahdavi, Rong Jin, Tianbao Yang, and Shenghuo Zhu. Random projections for classification: A recovery approach. *IEEE Transactions on Information Theory*, 60(11):7300–7316, 2014.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 928–936, 2003.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

A Proofs

A.1 Proof of Theorem 1

We will use the duals of $L(\mathbf{w})$ and $\hat{L}(\mathbf{z})$ to show that $\mathbf{z}_*$ is close to $R\mathbf{w}_*$. Some other works such as Zhang et al. (2014) and Yang et al. (2015) have also attempted to analyze random projections via the dual, and this has the advantage that the dual variables are often easier to compare, as they at least have the same dimensionality. Note that $R\mathbf{w}_*$ is essentially the count-sketch projection of $\mathbf{w}_*$, hence showing that $\mathbf{z}_*$ is close to $R\mathbf{w}_*$ will allow us to show that doing count-sketch recovery using $\mathbf{z}_*$ is comparable to doing count-sketch recovery from the projection of $\mathbf{w}_*$ itself, and hence give us the desired error bounds. We first derive the dual forms of the objective function $L(\mathbf{w})$, the dual of $\hat{L}(\mathbf{z})$ can be derived analogously. Let $u_i = y_i \mathbf{w}^T \mathbf{x}_i$. Then we can write the primal as:

$$\begin{aligned} \min_{\mathbf{u},\mathbf{w}} \quad & \frac{1}{T}\sum_{i=1}^{T} \ell(u_i) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2, \\ \text{subject to} \quad & u_i = y_i \mathbf{w}^T \mathbf{x}_i, \ \forall\, i \le T. \end{aligned}$$

Define $\tilde{X}_i = y_i \mathbf{x}_i$, i.e. the ith data point $\mathbf{x}_i$ times its label. Let $\tilde{X} \in \mathbb{R}^{d \times T}$ be the matrix of data points such that the ith column is $\tilde{X}_i$. Let $K = \tilde{X}^T \tilde{X}$ be the kernel matrix corresponding to the original data points. Taking the Lagrangian, and minimizing with respect to the primal variables $\mathbf{z}$ and $\mathbf{w}$ gives us the following dual objective function in terms of the dual variable α:

$$J(\alpha) = \frac{1}{T}\sum_i \ell^*(\alpha_i) + \frac{1}{2\lambda T^2}\alpha^T K \alpha,$$

where ℓ^* is the Fenchel conjugate of ℓ. Also, if α_* is the minimizer of $J(\alpha)$, then the minimizer $\mathbf{w}_*$ of $L(\mathbf{w})$ is given by $\mathbf{w}_* = -\frac{1}{\lambda T}\tilde{X}\alpha_*$. Similarly, let $K = \tilde{X}^T R^T R \tilde{X}$ be the kernel matrix corresponding to the projected data points. We can write down the dual $\hat{L}(\alpha)$ of the projected primal objective function $\hat{J}(\mathbf{w})$ in terms of the dual variable $\hat{\alpha}$ as follows:

$$\hat{J}(\hat{\alpha}) = \frac{1}{T}\sum_i \ell^*(\hat{\alpha}_i) + \frac{1}{2\lambda T^2}\hat{\alpha}^T \hat{K} \hat{\alpha}.$$

As before, if $\hat{\alpha}_*$ is the minimizer of $\hat{J}(\hat{\alpha})$, then the minimizer $\mathbf{z}_*$ of $\hat{L}(\mathbf{z})$ is given by $\hat{\mathbf{w}}_* = -\frac{1}{\lambda T} R\tilde{X}\hat{\alpha}_*$. We will first express the distance between $\mathbf{z}_*$ and $R\mathbf{w}_*$ in terms of the distance between the dual variables. We can write:

$$\begin{aligned} \|\mathbf{z}_* - R\mathbf{w}_*\|_2^2 &= \frac{1}{\lambda^2 T^2}\|R\tilde{X}\hat{\alpha}_* - R\tilde{X}\alpha_*\|_2^2 \\ &= \frac{1}{\lambda^2 T^2}(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*). \end{aligned} \tag{1}$$

Hence our goal will be to upper bound $(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*)$. Define $\mathbf{\Delta} = \frac{1}{\lambda T}(\hat{K} - K)\alpha_*$. We will show that $(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*)$ can be upper bounded in terms of $\mathbf{\Delta}$ as follows.

Lemma 2.

$$\begin{aligned}\frac{1}{\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*) &\leq \frac{1}{\lambda T^2}(\alpha_* - \hat{\alpha}_*)^T (\hat{K} - K)\alpha_* \\ &\leq \frac{1}{T}\|\hat{\alpha}_* - \alpha_*\|_1 \|\mathbf{\Delta}\|_\infty.\end{aligned}$$

Proof. We first claim that,

$$\hat{J}(\alpha_*) \geq \hat{J}(\hat{\alpha}_*) + \frac{1}{2\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*). \tag{2}$$

Note that $\hat{J}(\alpha_*) \geq \hat{J}(\hat{\alpha}_*)$ just because $\hat{\alpha}_*$ is the minimizer of $\hat{J}(\hat{\alpha})$, hence Eq. 2 is essentially giving an improvement over this simple bound. In order to prove Eq. 2, define $F(\alpha) = \frac{1}{T}\sum_i \ell^*(\alpha_i)$. As $F(\alpha)$ is a convex function (because $\ell^*(x)$ is convex in x), from the definition of convexity,

$$F(\alpha_*) \geq F(\hat{\alpha}_*) + \langle \nabla F(\hat{\alpha}_*), \alpha_* - \hat{\alpha}_* \rangle. \tag{3}$$

It is easy to verify that,

$$\begin{aligned}\frac{1}{2\lambda T^2}\alpha_*^T \hat{K}\alpha_* = \frac{1}{2\lambda T^2}\hat{\alpha}_*^T \hat{K}\hat{\alpha}_* + \frac{1}{\lambda T^2}(\alpha_* - \hat{\alpha}_*)^T \hat{K}\hat{\alpha}_* \\ + \frac{1}{2\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*).\end{aligned} \tag{4}$$

Adding Eq. 3 and Eq. 4, we get,

$$\begin{aligned}\hat{J}(\alpha_*) &\geq \hat{J}(\hat{\alpha}_*) + \langle \nabla F(\hat{\alpha}_*), \alpha_* - \hat{\alpha}_* \rangle + \frac{1}{\lambda T^2}(\alpha_* - \hat{\alpha}_*)^T \hat{K}\hat{\alpha}_* \\ &\quad + \frac{1}{2\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*) \\ &= \hat{J}(\hat{\alpha}_*) + \langle \nabla \hat{J}(\hat{\alpha}_*), \alpha_* - \hat{\alpha}_* \rangle + \frac{1}{2\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*) \\ &= \hat{J}(\hat{\alpha}_*) + \frac{1}{2\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*),\end{aligned}$$

which verifies Eq. 2. We can derive a similar bound for $\hat{J}(\hat{\alpha}_*)$,

$$\hat{J}(\hat{\alpha}_*) \geq \hat{J}(\alpha_*) + (\hat{\alpha}_* - \alpha_*)^T \nabla \hat{J}(\alpha_*) + \frac{1}{2\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*). \tag{5}$$

As $J(\alpha)$ is minimized by α_* and $J(\alpha)$ is convex,

$$\begin{aligned}&(\hat{\alpha}_* - \alpha_*)^T \nabla J(\alpha_*) \geq 0 \\ \implies &(\hat{\alpha}_* - \alpha_*)^T (\nabla F(\alpha_*) + \frac{1}{\lambda T^2} K\alpha_*) \geq 0 \\ \implies &(\hat{\alpha}_* - \alpha_*)^T (\nabla \hat{J}(\alpha_*) + \frac{1}{\lambda T^2}(K - \hat{K})\alpha_*) \geq 0 \\ \implies &(\hat{\alpha}_* - \alpha_*)^T \nabla \hat{J}(\alpha_*) \geq \frac{1}{\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T (\hat{K} - K)\alpha_*.\end{aligned} \tag{6}$$

Using Eq. 6 in Eq. 5, we get,

$$\begin{aligned}\hat{J}(\hat{\alpha}_*) - \hat{J}(\alpha_*) \geq \frac{1}{\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T (\hat{K} - K)\alpha_* \\ + \frac{1}{2\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*).\end{aligned}$$

By Eq. 2, $\hat{J}(\hat{\alpha}_*) - \hat{J}(\alpha_*) \leq -\frac{1}{2\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*)$. Therefore we can write,

$$\begin{aligned} -\frac{1}{\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*) &\geq \frac{1}{\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T (\hat{K} - K)\alpha_* \\ \implies \frac{1}{\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T \hat{K}(\hat{\alpha}_* - \alpha_*) &\leq \frac{1}{\lambda T^2}(\alpha_* - \hat{\alpha}_*)^T (\hat{K} - K)\alpha_*. \end{aligned}$$

To finish, by Holder's inequality,

$$\frac{1}{\lambda T^2}(\alpha_* - \hat{\alpha}_*)^T (\hat{K} - K)\alpha_* \leq \frac{1}{\lambda T^2} \|\alpha_* - \hat{\alpha}_*)\|_1 \left\|(\hat{K} - K)\right\|_\infty \alpha_* = \frac{1}{T}\|\hat{\alpha}_* - \alpha_*\|_1 \|\mathbf{\Delta}\|_\infty$$

□

Our next step is to upper bound $\|\hat{\alpha}_* - \alpha_*\|_1$ in terms of $\|\mathbf{\Delta}\|_\infty$ using the β-strongly smooth property of $\ell(z)$.

Lemma 3.

$$\|\hat{\alpha}_* - \alpha_*\|_1 \leq 2T\beta\|\mathbf{\Delta}\|_\infty$$

Proof. We first claim that $J(w)$ is $1/(T\beta)$-strongly convex. Note that as $\ell(u_i)$ is β-strongly smooth, therefore, $\ell^*(\alpha_i)$ is $1/\beta$-strongly convex. Therefore for any $\mathbf{u}$ and $\mathbf{v}$,

$$\begin{aligned} \ell^*(u_i) &\geq \ell^*(v_i) + (u_i - v_i)\nabla\ell^*(v_i) + \frac{(u_i - v_i)^2}{2\beta} \\ \implies \frac{1}{T}\sum_i \ell^*(u_i) &\geq \frac{1}{T}\sum_i \ell^*(v_i) + \frac{1}{T}\langle \nabla\ell^*(\mathbf{v}), \mathbf{u} - \mathbf{v}\rangle \\ &\quad + \frac{\|\mathbf{u} - \mathbf{v}\|_2^2}{2T\beta}. \end{aligned}$$

Therefore $F(\alpha) = \frac{1}{T}\sum_i \ell^*(\alpha_i)$ is $1/(T\beta)$ strongly convex. Note that $\frac{1}{2\lambda T^2}\alpha^T K\alpha$ is a convex function of α as $K = \tilde{X}^T\tilde{X}$ is a positive semidefinite matrix. Therefore $J(\alpha) = F(\alpha) + \frac{1}{2\lambda T^2}\alpha^T K\alpha$ is $1/(T\beta)$ strongly convex. It follows from the same reasoning that $\hat{J}(\hat{\alpha})$ is also $1/(T\beta)$-strongly convex.

By the definition of strong convexity,

$$\hat{J}(\hat{\alpha}_*) \geq \hat{J}(\alpha_*) + (\hat{\alpha}_* - \alpha_*)^T \nabla\hat{J}(\alpha_*) + \frac{1}{2T\beta}\|\hat{\alpha}_* - \alpha_*\|_2^2.$$

As $\hat{\alpha}_*$ is the minimizer of $\hat{J}(\alpha)$, $\hat{J}(\hat{\alpha}_*) \leq \hat{J}(\alpha_*)$. Therefore,

$$\begin{aligned} (\hat{\alpha}_* - \alpha_*)^T \nabla\hat{J}(\alpha_*) + \frac{1}{2T\beta}\|\hat{\alpha}_* - \alpha_*\|_2^2 &\leq \hat{J}(\hat{\alpha}_*) - \hat{J}(\alpha_*) \leq 0 \\ \implies \frac{1}{2T\beta}\|\hat{\alpha}_* - \alpha_*\|_2^2 &\leq -(\hat{\alpha}_* - \alpha_*)^T \nabla\hat{J}(\alpha_*). \end{aligned}$$

Using Eq. 6, we can rewrite this as,

$$\begin{aligned} \frac{1}{2T\beta}\|\hat{\alpha}_* - \alpha_*\|_2^2 &\leq -\frac{1}{\lambda T^2}(\hat{\alpha}_* - \alpha_*)^T (\hat{K} - K)\alpha_* \\ &\leq \frac{1}{T}\|\hat{\alpha}_* - \alpha_*\|_1 \|\mathbf{\Delta}\|_\infty \\ \implies \|\hat{\alpha}_* - \alpha_*\|_2^2 &\leq 2\beta\|\hat{\alpha}_* - \alpha_*\|_1 \|\mathbf{\Delta}\|_\infty. \end{aligned}$$

By Cauchy-Schwartz,

$$\begin{aligned} \|\hat{\alpha}_* - \alpha_*\|_1^2 &\leq T\|\hat{\alpha}_* - \alpha_*\|_2^2 \\ &\leq 2T\beta\|\hat{\alpha}_* - \alpha_*\|_1 \|\mathbf{\Delta}\|_\infty \\ \implies \|\hat{\alpha}_* - \alpha_*\|_1 &\leq 2T\beta\|\mathbf{\Delta}\|_\infty. \end{aligned}$$

□

We now bound $\|\mathbf{\Delta}\|_\infty$. The result relies on the JL property of the projection matrix R (recall Definition 4). If R is a JL matrix with error ϵ and failure probability δ/d^2, then with failure probability δ, for all coordinate basis vectors $\{\mathbf{e}_1, \ldots, \mathbf{e}_d\}$,

$$\|R\mathbf{e}_i\|_2 = 1 \pm \epsilon,\ \forall\ i, \quad |\langle R\mathbf{e}_i, R\mathbf{e}_j\rangle| \leq \epsilon,\ \forall\ i \neq j \tag{7}$$

The first bound follows directly from the JL property, to derive the bound on the inner product, we rewrite the inner product as follows,

$$4\langle R\mathbf{e}_i, R\mathbf{e}_j\rangle = \|R\mathbf{e}_i + R\mathbf{e}_j\|_2^2 - \|R\mathbf{e}_i - R\mathbf{e}_j\|_2^2.$$

The bound then follows by using the JL property for $\|R\mathbf{e}_i + R\mathbf{e}_j\|_2^2$ and $\|R\mathbf{e}_i - R\mathbf{e}_j\|_2^2$. As there are d^2 pairs, if the inner product is preserved for each pair with failure probability δ/d^2, then by a union bound it is preserved simultaneously for all of them with failure probability δ. Condition 7 is useful because it implies that R approximately preserves the inner products and lengths of sparse vectors, as states in the following Lemma—

Lemma 4. *If condition 7 is satisfied, then for any two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$,*

$$|\mathbf{v}_1^T\mathbf{v}_2 - (R\mathbf{v}_1)^T(R\mathbf{v}_2)| \leq 2\epsilon \|\mathbf{v}_1\|_1 \|\mathbf{v}_2\|_1 .$$

Proof. To verify, write $\mathbf{v}_1 = \sum_{i=1}^d \psi_i \mathbf{e}_i$ and $\mathbf{v}_2 = \sum_{i=1}^d \phi_i \mathbf{e}_i$. Then,

$$\begin{aligned}
(R\mathbf{v}_1)^T(R\mathbf{v}_2) &= \Big(\sum_{i=1}^d \psi_i (R\mathbf{e}_i)^T\Big)\Big(\sum_{j=1}^d \phi_j (R\mathbf{e}_j)\Big) \\
&= \sum_{i=1}^d \psi_i\phi_i \|R\mathbf{e}_i\|_2^2 + \sum_{i\neq j} \psi_i\phi_j (R\mathbf{e}_i)^T(R\mathbf{e}_j) \\
&\leq (1+\epsilon)\sum_{i=1}^d \psi_i\phi_i + \sum_{i\neq j} |\psi_i||\phi_j||(R\mathbf{e}_i)^T(R\mathbf{e}_j)| \\
&\leq (1+\epsilon)\mathbf{v}_1^T\mathbf{v}_2 + \epsilon \sum_{i\neq j} |\psi_i||\phi_j| \quad \text{(using condition 7)} \\
&\leq \mathbf{v}_1^T\mathbf{v}_2 + \epsilon \|\mathbf{v}_1\|_2 \|\mathbf{v}_2\|_2 + \epsilon \sum_i |\psi_i| \sum_j |\phi_j| \\
&\leq \mathbf{v}_1^T\mathbf{v}_2 + 2\epsilon \|\mathbf{v}_1\|_1 \|\mathbf{v}_2\|_1 ,
\end{aligned}$$

where the last line is due to $\sum_i |\psi_i| = \|\mathbf{v}_1\|_1$ and $\|\mathbf{v}_1\|_2 \leq \|\mathbf{v}_1\|_1$. By a similar argument, we can show that $(R\mathbf{v}_1)^T(R\mathbf{v}_2) - \mathbf{v}_1^T\mathbf{v}_2 \geq 2\epsilon \|\mathbf{v}_1\|_1 \|\mathbf{v}_2\|_1$. Hence for any two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$,

$$|\mathbf{v}_1^T\mathbf{v}_2 - (R\mathbf{v}_1)^T(R\mathbf{v}_2)| \leq 2\epsilon \|\mathbf{v}_1\|_1 \|\mathbf{v}_2\|_1 .$$

□

We can now bound $\|\mathbf{\Delta}\|_\infty$ using Lemma 4.

Lemma 5. *If R satisfies condition 7 then,*

$$\|\mathbf{\Delta}\|_\infty \leq 2\gamma\epsilon\|\mathbf{w}_*\|_1,$$

where $\gamma = \max_i \|\mathbf{x}_i\|_1$.

Proof. We first rewrite $\mathbf{\Delta}$ as follows,

$$\begin{aligned}
\mathbf{\Delta} &= \frac{1}{\lambda T}(\tilde{X}^T R^T R\tilde{X} - \tilde{X}^T\tilde{X})\alpha_* = \frac{1}{\lambda T}\tilde{X}^T(R^T R - I)\tilde{X}\alpha_* \\
&= \tilde{X}^T(I - R^T R)\mathbf{w}_*,
\end{aligned}$$

using the relation that $\mathbf{w}_* = -\frac{1}{\lambda T}\tilde{X}\alpha_*$. Therefore,

$$\|\mathbf{\Delta}\|_\infty \leq \max_i |\mathbf{x}_i^T(I - R^T R)\mathbf{w}_*| = \max_i |\mathbf{x}_i^T \mathbf{w}_* - (R\mathbf{x}_i)^T(R\mathbf{w}_*)|$$

Using Lemma 4, it follows that,

$$\|\mathbf{\Delta}\|_\infty \leq \max_i |\mathbf{x}_i^T \mathbf{w}_* - (Rx_i)^T(R\mathbf{w}_*)| \leq 2\epsilon\gamma\|\mathbf{w}_*\|_1.$$

□

We will now combine Lemma 2, 3 and 5. By Eq. 1 and Lemma 2,

$$\|\mathbf{z}_* - R\mathbf{w}_*\|_2^2 \leq \frac{1}{\lambda T}\|\hat{\alpha}_* - \alpha_*\|_1 \|\mathbf{\Delta}\|_\infty.$$

Combining this with Lemma 3,

$$\|\mathbf{z}_* - R\mathbf{w}_*\|_2^2 \leq \frac{2\beta}{\lambda}\|\mathbf{\Delta}\|_\infty^2.$$

If R is a JL matrix with error ϵ and failure probability δ/d^2, then by Lemma 5, with failure probability δ,

$$\|\mathbf{z}_* - R\mathbf{w}_*\|_2 \leq \gamma\epsilon\sqrt{\frac{8\beta}{\lambda}}\|\mathbf{w}_*\|_1. \tag{8}$$

By Kane and Nelson (2014), the random projection matrix R satisfies the JL property with error θ and failure probability δ'/d^2 for $k \geq C\log(d/\delta')/\theta^2$, where C is a fixed constant. Using Eq. 8, with failure probability δ',

$$\|\mathbf{z}_* - R\mathbf{w}_*\|_2 \leq 8\gamma\theta\sqrt{\frac{8\beta}{\lambda}}\|\mathbf{w}_*\|_1. \tag{9}$$

Recall that $R\sqrt{s}$ is a count-sketch matrix with width C_1/θ and depth $s = C_2\log(d/\delta')/\theta$, where C_1 and C_2 are fixed constants. Let $\mathbf{w}_{\text{proj}}$ be the projection of $\mathbf{w}_*$ with the count-sketch matrix $\tilde{R}$, hence $\mathbf{w}_{\text{proj}} = \sqrt{s}R\mathbf{w}_*$. Let $\mathbf{z}_{\text{proj}} = \sqrt{s}\mathbf{z}_*$. By Eq. 9, with failure probability δ',

$$\|\mathbf{z}_{\text{proj}} - \mathbf{w}_{\text{proj}}\|_2 \leq \sqrt{\frac{8\beta\gamma^2\theta\log(d/\delta')}{\lambda}}\|\mathbf{w}_*\|_1.$$

Let $\mathbf{w}_{\text{cs}}$ be the count-sketch estimate of $\mathbf{w}_*$ derived from $\mathbf{w}_{\text{proj}}$, and $\mathbf{w}_{\text{est}}$ be the count-sketch estimate of $\mathbf{w}_*$ derived from $\mathbf{z}_{\text{proj}}$. Recall that the count-sketch estimate of a vector is the median of the estimates of all the locations to which the vector hashes. As the difference between the median of any two vectors is at most the ℓ_∞-norm of their difference,

$$\|\mathbf{w}_{\text{est}} - \mathbf{w}_{\text{cs}}\|_\infty \leq \|\mathbf{z}_{\text{proj}} - \mathbf{w}_{\text{proj}}\|_\infty.$$

Therefore with failure probability δ',

$$\begin{aligned}\|\mathbf{w}_{\text{est}} - \mathbf{w}_{\text{cs}}\|_\infty \leq \|\mathbf{z}_{\text{proj}} - \mathbf{w}_{\text{proj}}\|_\infty &\leq \|\mathbf{z}_{\text{proj}} - \mathbf{w}_{\text{proj}}\|_2 \\ &\leq \sqrt{\frac{8\beta\gamma^2\theta\log(d/\delta')}{\lambda}}\|\mathbf{w}_*\|_1.\end{aligned} \tag{10}$$

We now use Lemma 6 to bound the error for Count-Sketch recovery.

Lemma 6. *(Charikar et al., 2002) Let $\mathbf{w}_{\text{cs}}$ be the Count-Sketch estimate of the vector w. For any vector w, with probability $1-\delta$, a Count-Sketch matrix with width $\Theta(1/\epsilon^2)$ and depth $\Theta(\log(d/\delta))$ satisfies,*

$$\|\mathbf{w} - \mathbf{w}_{\text{cs}}\|_\infty \leq \epsilon\|\mathbf{w}\|_2.$$

Hence using Lemma 6 for the matrix $\sqrt{s}R$, with failure probability δ',

$$\|\mathbf{w}_* - \mathbf{w}_{\text{cs}}\|_\infty \le \sqrt{\theta}\|\mathbf{w}_*\|_2.$$

Now using the triangle inequality and Eq. 10, with failure probability $2\delta'$ (due to a union bound),

$$\begin{aligned}\|\mathbf{w}_* - \mathbf{w}_{\text{est}}\|_\infty &\le \|\mathbf{w}_* - \mathbf{w}_{\text{est}}\|_\infty + \|\mathbf{w}_{\text{est}} - \mathbf{w}_{\text{cs}}\|_\infty \\ &\le \sqrt{\theta}\|\mathbf{w}_*\|_2 + \sqrt{\frac{8\beta\gamma^2\theta\log(d/\delta')}{\lambda}}\|\mathbf{w}_*\|_1 \\ &\le \Big(\sqrt{\theta} + \sqrt{\frac{8\beta\gamma^2\theta\log(d/\delta')}{\lambda}}\Big)\|\mathbf{w}_*\|_1.\end{aligned}$$

Therefore choosing $\theta = \min\{1, \lambda/(4\beta\gamma^2\log(d/\delta'))\}\epsilon^2/4$, with failure probability $2\delta'$,

$$\|\mathbf{w}_* - \mathbf{w}_{\text{est}}\|_\infty \le \epsilon\|\mathbf{w}_*\|_1.$$

Choosing $\delta' = \delta/2$, for fixed constants C', C'' and

$$\begin{aligned}k &= (C'/\epsilon^4)\log^3(d/\delta)\max\{1, \beta^2\gamma^4/\lambda^2\}, \\ s &= (C''/\epsilon^2)\log^2(d/\delta)\max\{1, \beta\gamma^2/\lambda\},\end{aligned}$$

with probability $1-\delta$,

$$\|\mathbf{w}_* - \mathbf{w}_{\text{est}}\|_\infty \le \epsilon\|\mathbf{w}_*\|_1.$$

A.2 Proof of Theorem 2

Let $f_t(\mathbf{z})$ be the loss function corresponding to the data point chosen in the tth time step—

$$f_t(\mathbf{z}) = \ell\left(y_t\mathbf{z}^T R\mathbf{x}_t\right) + \frac{\lambda}{2}\|\mathbf{z}\|_2^2. \tag{11}$$

Let $\mathbf{z}_t$ be the weight vector at the tth time step for online updates on the projected problem. Let $\bar{\mathbf{z}} = \frac{1}{T}\sum_{i=1}^T \hat{\mathbf{z}}_i$ be the average of the weight vectors for all the T time steps. We claim that $\bar{\mathbf{z}}$ is close to $\mathbf{z}_*$, the optimizer of $\hat{L}(\mathbf{z})$, using Corollary 1 of Shamir (2016). In order to apply the result we first need to define a few parameters of the function $\hat{L}(\mathbf{z})$. Note that $\hat{L}(\mathbf{z})$ is λ strongly-convex (as $\hat{L}(\mathbf{z}) - \frac{\lambda}{2}\|\mathbf{z}\|_2^2$ is convex. As the derivative of ℓ is bounded above by H, therefore ℓ is H-Lipschitz. We assume $\|R\mathbf{x}_i\|_2 \le B, \|\mathbf{z}_*\|_2 \le D$ and $\max_t \|\nabla f_t(\mathbf{w})\|_2 \le G$. We will bound B, D and G in the end. We now apply Corollary 1 of Shamir (2016), with the notation adapted for our setting.

Lemma 7. *(Shamir, 2016) Consider any loss function $\hat{L}(\mathbf{z}) = \sum_{i=1}^T f_t(\mathbf{z})$, where $f_t(\mathbf{z})$ is defined in Eq. 11. For any H-Lipchitz ℓ_i, $\|R\mathbf{x}_i\|_2 \le B$, $\|\mathbf{z}_t\|_2 \le D$, and some fixed constant C, over the randomness in the order in which the samples are received:*

$$\mathbb{E}\Big[\frac{1}{T}\sum_{t=1}^T \hat{L}(\mathbf{z}_t) - \hat{L}(\mathbf{z}_*)\Big] \le \frac{C(R_T/\sqrt{T} + BDH)}{\sqrt{T}},$$

where R_T is the regret of online gradient descent with respect to the batch optimizer $\mathbf{z}_*$, defined as $R_T = \sum_{t=1}^T [f_t(\hat{\mathbf{z}}) - f_t(\mathbf{z}_*)]$.

By standard regret bounds on online gradient descent (see Zinkevich (2003)), $R_T \le GD\sqrt{T}$. Therefore,

$$\mathbb{E}\Big[\frac{1}{T}\sum_{t=1}^T \hat{L}(\mathbf{z}_t) - \hat{L}(\mathbf{z}_*)\Big] \le \frac{CD(G + BH)}{\sqrt{T}}.$$

Note that by Jensen's inequality,

$$\mathbb{E}[\hat{L}(\mathbf{z})] \leq \mathbb{E}\Big[\frac{1}{T}\sum_{t=1}^{T}\hat{L}(\mathbf{z}_t)\Big]$$
$$\implies \mathbb{E}\Big[\hat{L}(\bar{\mathbf{z}}) - \hat{L}(\mathbf{z}_*)\Big] \leq \frac{CD(G+BH)}{\sqrt{T}}. \tag{12}$$

We will now bound the expected distance between $\bar{\mathbf{z}}$ and $\mathbf{z}_*$ using Eq. 12 and the strong convexity of $\hat{L}(\mathbf{w})$. As $\hat{L}(\mathbf{w})$ is λ strong-convex and $\nabla\hat{L}(\mathbf{z}_*) = 0$, we can write:

$$\hat{L}(\mathbf{z}_*) + (\lambda/2)\|\bar{\mathbf{z}} - \mathbf{z}_*\|_2^2 \leq \hat{L}(\bar{\mathbf{z}})$$
$$\implies \|\bar{\mathbf{z}} - \mathbf{z}_*\|_2^2 \leq (\lambda/2)[\hat{L}(\bar{\mathbf{z}}) - \hat{L}(\mathbf{z}_*)]$$
$$\implies \mathbb{E}\Big[\|\bar{\mathbf{z}} - \mathbf{z}_*\|_2^2\Big] \leq (2/\lambda)\Big[\mathbb{E}[\hat{L}(\bar{\mathbf{z}})] - \hat{L}(\mathbf{z}_*)\Big].$$

Using Eq. 12 and then Jensen's inequality,

$$\mathbb{E}\Big[\|\bar{\mathbf{z}} - \mathbf{z}_*\|_2\Big] \leq \frac{2CD(G+BH)}{\lambda\sqrt{T}}. \tag{13}$$

Let $\bar{\mathbf{z}}_{\text{proj}} = \sqrt{s}\bar{\mathbf{z}}$. Let $\mathbf{z}_{\text{wm}}$ is the Count-Sketch estimate of $\mathbf{w}_*$ derived from $\bar{\mathbf{z}}_{\text{proj}}$. Recall from the proof of Theorem 1 that $\mathbf{z}_{\text{proj}} = \sqrt{s}\mathbf{z}$ and $\mathbf{w}_{\text{est}}$ is the Count-Sketch estimate of $\mathbf{w}_*$ derived from $\mathbf{z}_{\text{proj}}$. As in the proof of Theorem 1, we note that the difference between the medians of any two vectors is at most the ℓ_∞ norm of the difference of the vectors, and hence we can write,

$$\|\mathbf{w}_{\text{est}} - \mathbf{z}_{\text{wm}}\|_\infty \leq \|\mathbf{z}_{\text{proj}} - \bar{\mathbf{z}}_{\text{proj}}\|_\infty \leq \|\mathbf{z}_{\text{proj}} - \bar{\mathbf{z}}_{\text{proj}}\|_2$$
$$= \sqrt{s}\|\mathbf{z}_* - \bar{\mathbf{z}}\|_2.$$

Therefore, using Eq. 13,

$$\mathbb{E}[\|\mathbf{w}_{\text{est}} - \mathbf{z}_{\text{wm}}\|_\infty] \leq \frac{2CD(G+BH)}{\lambda}\sqrt{\frac{s}{T}}. \tag{14}$$

By the triangle inequality,

$$\|\mathbf{w}_* - \mathbf{z}_{\text{wm}}\|_\infty \leq \|\mathbf{w}_* - \mathbf{w}_{\text{est}}\|_\infty + \|\mathbf{w}_{\text{est}} - \mathbf{z}_{\text{wm}}\|_\infty$$
$$\implies \mathbb{E}\Big[\,\|\mathbf{w}_* - \mathbf{z}_{\text{wm}}\|_\infty\,\Big] \leq \mathbb{E}\Big[\,\|\mathbf{w}_* - \mathbf{w}_{\text{est}}\|_\infty\,\Big] + \mathbb{E}\Big[\,\|\mathbf{w}_{\text{est}} - \mathbf{z}_{\text{wm}}\|_\infty\,\Big]$$

By Theorem 1, for fixed constants C_1, C_2 and

$$k = (C_1/\epsilon^4)\log^3(d/\delta)\max\{1, \beta^2\gamma^4/\lambda^2\},$$
$$s = (C_2/\epsilon^2)\log^2(d/\delta)\max\{1, \beta\gamma^2/\lambda\},$$
$$\|\mathbf{w}_* - \mathbf{w}_{\text{est}}\|_\infty \leq \epsilon\|\mathbf{w}_*\|_1,$$

with probability $1-\delta$. Therefore, for fixed constants C_1' and C_2' and probability $1-\delta$,

$$\mathbb{E}\Big[\,\|\mathbf{w}_* - \mathbf{z}_{\text{wm}}\|_\infty\,\Big] \leq \frac{\epsilon}{2}\|\mathbf{w}_*\|_1$$
$$+\sqrt{\frac{4C_2'(GD+BDH)^2\log^2(d/\delta)\max\{1, LR^2/\lambda\}}{\lambda^2\epsilon^2 T}}.$$

Therefore for

$$T \geq (C_3'/(\epsilon^4\lambda^2))(D/\|\mathbf{w}_*\|_1)^2(G+BH)^2\log^2(d/\delta)\max\{1, LR^2/\lambda\},$$
$$\mathbb{E}\Big[\,\|\mathbf{w}_* - \mathbf{z}_{\text{wm}}\|_\infty\,\Big] \leq \frac{\epsilon}{2}\|\mathbf{w}_*\|_1 + \frac{\epsilon}{2}\|\mathbf{w}_*\|_2 \leq \epsilon\|\mathbf{w}_*\|_1.$$

We will now bound B, D and G, starting with B. Note that R is a JL matrix which satisfies condition 7 with $\epsilon = \theta$. Using Lemma 4 and the fact that $\|\mathbf{x}_i\|_2 \leq 1$,

$$\|R\mathbf{x}_i\|_2 \leq \sqrt{1+\theta\gamma^2} \implies B \leq 1 + \sqrt{\theta}\gamma \leq 1 + \epsilon\gamma,$$

where for the last bound we use the setting of

$$\theta = \min\{1, \lambda/(4\beta\gamma^2 \log(d/\delta'))\}\epsilon^2/4$$

from the proof of Theorem 1. We next bound $\|\mathbf{z}_*\|_2$. Using Lemma 4,

$$\begin{aligned} &\|\mathbf{z}_* - R\mathbf{w}_*\|_2 \leq 2R\theta\sqrt{\beta/\lambda}\,\|\mathbf{w}_*\|_1 \\ \implies &\|\mathbf{z}_*\|_2 \leq \|R\mathbf{w}_*\|_2 + 2R\theta\sqrt{\beta/\lambda}\,\|\mathbf{w}_*\|_1 . \end{aligned}$$

By Lemma 4, $\|R\mathbf{w}_*\|_2 \leq \sqrt{\|\mathbf{w}_*\|_2^2 + \theta\,\|\mathbf{w}_*\|_1^2} \leq \|\mathbf{w}_*\|_2 + \sqrt{\theta}\,\|\mathbf{w}_*\|_1$. Therefore,

$$\begin{aligned} \|\mathbf{z}_*\|_2 &\leq \|\mathbf{w}_*\|_2 + \sqrt{\theta}\,\|\mathbf{w}_*\|_1 + 2R\theta\sqrt{\beta/\lambda}\,\|\mathbf{w}_*\|_1 \\ &= \|\mathbf{w}_*\|_2 + \left(\sqrt{\theta} + 2R\theta\sqrt{\beta/\lambda}\right)\|\mathbf{w}_*\|_1 . \end{aligned}$$

For our choice of θ,

$$\|\mathbf{z}_*\|_2 \leq \|\mathbf{w}_*\|_2 + \epsilon\,\|\mathbf{w}_*\|_1 \implies D \leq D_2 + \epsilon D_1.$$

This implies that the $(D/\|\mathbf{w}_*\|_1)$ term in our bound for T can be upper bounded by $2D_2/\|\mathbf{w}_*\|_1$, yielding the bound on T stated in Theorem 2. Finally, we need to upper bound $G = \max_t \|\nabla f_t(\mathbf{w})\|_2$. We do this as follows:

$$\begin{aligned} &\nabla f_t(\mathbf{z}) = \ell'(y_t\mathbf{z}_t^T R\mathbf{x_t})A\mathbf{x_t} + \lambda\mathbf{z}_t \\ \implies &\|\nabla f_t(\mathbf{z})\|_2 \leq |\ell'(y_t\mathbf{z}_t^T Rx)|\|Rx\|_2 + \lambda\|\mathbf{z}_t\|_2 \\ &\leq H(1+\epsilon\gamma) + \lambda D. \end{aligned}$$

B Baseline Algorithms

Algorithm 3: Simple Truncation

input: loss function ℓ, budget K, ℓ_2-regularization parameter λ, learning rate schedule η_t

initialization

- $S \leftarrow \{\}$ ▷ Empty heap
- $t \leftarrow 0$

function `Update(x, y)`

- $\tau \leftarrow \sum_{i \in S} S[i] \cdot x_i$ ▷ Make prediction
- $S \leftarrow (1 - \lambda\eta_t)S - \eta_t y x_i \nabla\ell(y\tau)$
- $S \leftarrow$ `Truncate(`S, K`)`
- $t \leftarrow t + 1$

Algorithm 4: Probabilistic Truncation

input: loss function ℓ, budget K, ℓ_2-regularization parameter λ, learning rate schedule η_t

initialization

$S_0 \leftarrow \{\}$, $S' \leftarrow \{\}$

$t \leftarrow 0$

function `Update(`**x**, y`)`

$\tau \leftarrow \sum_{i \in S} S[i] \cdot x_i$ ▷ Make prediction

for $i \in S_t$ **do**

$S_{t+1}[i] \leftarrow (1 - \lambda\eta_t) S_t[i] - \eta_t y x_i \nabla \ell(y\tau)$

if $S_{t+1}[i] \neq 0$ **and** $S_t[i] = 0$ **then**

$r \sim \mathcal{U}(0, 1)$

else

$r \leftarrow S'[i]$

$S'[i] \leftarrow r^{|S_t[i]/S_{t+1}[i]|}$

$S_{t+1} \leftarrow$ `Truncate(`S_{t+1}, S', K`)`

$t \leftarrow t + 1$